Zachodniopomorski Uniwersytet Technologiczny w Szczecinie
Wydział Informatyki
Katedra Inżynierii Oprogramowania

Agnieszka Kamińska

MODELE STATYSTYCZNE DO OSZACOWANIA WYDAJNOŚCI APLIKACJI GRUBOZIARNISTYCH W STANDARDZIE OPENMP C/C++ ORAZ ICH ZASTOSOWANIE W KOMPILACJI ITERACYJNEJ

ROZPRAWA DOKTORSKA

Promotor:
prof. dr hab. inż. Włodzimierz Bielecki

Zachodniopomorski Uniwersytet Technologiczny w Szczecinie

Szczecin 2014

Składam serdeczne podziękowania memu Promotorowi, Panu prof. dr hab. inż. Włodzimierzowi Bieleckiemu, za inspirację naukową do podjęcia tematu niniejszej pracy oraz cenne wsparcie merytoryczne.

Agnieszka Kamińska

Spis treści

STRESZCZENIE i

ABSTRACT iii

1. WSTĘP 1

1.1. Przedstawienie problemu 1

1.2. Cel i teza pracy 2

1.3. Zawartość pracy 6

2. PRZEGLĄD AKTUALNEGO STANU WIEDZY NA TEMAT PRZETWARZANIA RÓWNOLEGŁEGO 7

2.1. Mierniki oceny wydajności aplikacji równoległych 10

2.2. Podejścia do tworzenia aplikacji równoległych 11

2.3. Możliwości oszacowania wydajności aplikacji równoległych 11

2.4. Podsumowanie 17

3. OPRACOWANIE OGÓLNEGO MODELU STATYSTYCZNEGO ORAZ MODELI SZCZEGÓLNYCH 18

3.1. Zmienne i postać modelu ogólnego 18

3.1.1 Wyznaczanie odcisku danych 33

3.1.2 Wyznaczanie wartości zmiennych niezależnych 48

3.1.2.1. Wyznaczenie wartości zmiennej *X1* 49

3.1.2.2. Wyznaczenie wartości zmiennej *X2* 50

3.1.2.3. Wyznaczenie wartości zmiennej *X3* 51

3.1.2.4. Wyznaczenie wartości zmiennej *X4* 52

3.2. Wyznaczenie wartości parametrów modeli szczególnych 53

3.3. Zakres stosowalności modeli szczególnych 62

3.4. Ocena jakości modelu ogólnego i modeli szczególnych 64

3.5. Stosowanie modelu ogólnego i modeli szczególnych w praktyce 68

3.6. Przykład zastosowania opracowanych modeli szczególnych do oszacowania czasu wykonania pętli niewzorcowej 71

3.7. Prace pokrewne 77

3.8. Podsumowanie 80

4. BADANIA EKSPERYMENTALNE 82

4.1. Sposób przeprowadzenia badań eksperymentalnych 82

4.2. Wyniki badań eksperymentalnych 83

4.2.1 Wyznaczenie wartości parametrów *a1, a2, a3, a4* dla pętli wzorcowych nonInterf oraz matmul 84

4.2.2 Ocena jakości przykładowych modeli szczególnych 90
4.2.2.1. Model szczególny wyznaczony dla pętli wzorcowej nonInterf 92
4.2.2.1.1. Zastosowanie modelu dla pętli niewzorcowej CG_cg_3 92
4.2.2.1.2. Zastosowanie modelu dla pętli niewzorcowej CG_cg_4 96
4.2.2.1.3. Zastosowanie modelu dla pętli niewzorcowej FT_auxfnct_2 100
4.2.2.1.4. Zastosowanie modelu dla pętli niewzorcowej LU_HP_pintgr_11 104
4.2.2.1.5. Zastosowanie modelu dla pętli niewzorcowej MG_mg_3 108
4.2.2.1.6. *Zastosowanie modelu dla pętli niewzorcowej UA_diffuse_2* 112
4.2.2.2. Model szczególny wyznaczony dla pętli wzorcowej matmul 116
4.2.2.2.1. Zastosowanie modelu dla pętli niewzorcowej UA_diffuse_3 116
4.2.2.2.2. Zastosowanie modelu dla pętli niewzorcowej UA_diffuse_4 122
4.2.2.2.3. Zastosowanie modelu dla pętli niewzorcowej UA_transfer_11 129
4.2.2.2.4. Zastosowanie modelu dla pętli niewzorcowej UA_transfer_16 135
4.3. Podsumowanie 141
5. PODSUMOWANIE 142
BIBLIOGRAFIA 153
PUBLIKACJE WŁASNE, ZWIĄZANE Z TEMATYKĄ PRACY 158
SPIS RYSUNKÓW 159
SPIS TABEL 161

STRESZCZENIE

Jednym z podstawowych problemów współczesnej informatyki jest minimalizacja czasu przetwarzania danych. Problem ten ma szczególnie istotne znaczenie w zastosowaniach praktycznych komputerów – gdyż, bez względu na obszar tych zastosowań (nauka, przemysł, transport, życie codzienne, itd.), użytkownicy komputerów oczekują możliwie najszybszej reakcji maszyn na postawione problemy i zadania. Aby sprostać tym oczekiwaniom, opracowano wiele różnych rozwiązań w zakresie sposobu działania sprzętu komputerowego i oprogramowania. W niniejszej pracy, skupiono się na wpisującym się w ten nurt podejściu, które łączy aspekty o charakterze sprzętowym i programistycznym – przetwarzaniu równoległym.

Istotą przetwarzania równoległego jest podział zadania, które ma zostać wykonane, na mniejsze podzadania realizowane równocześnie (czyli, innymi słowy, równolegle), co w efekcie prowadzi do skrócenia czasu wykonywania zadania wyjściowego. Ta immanentna cecha przetwarzania równoległego pociąga za sobą następujące, doniosłe konsekwencje:

- przetwarzanie równoległe może być zastosowane do rozwiązania tylko tych problemów i zadań, które można podzielić na „fragmenty" przewidziane do jednoczesnej realizacji,
- jeżeli, dla danego problemu bądź zadania, można zastosować przetwarzanie równoległe – to istnieje *co najmniej* jeden sposób jego podziału na „fragmenty" przewidziane do jednoczesnej realizacji.

W przypadku, gdy istnieje więcej niż jeden sposób dokonania przedmiotowego podziału, pojawia się pytanie: *Który z możliwych podziałów jest najlepszy, tzn. skutkuje najkrótszym czasem rozwiązywania zrównoleglonego problemu/zadania w docelowym środowisku sprzętowym?*
Poprawnej odpowiedzi na to pytanie nie da się znaleźć na drodze czysto teoretycznej; jest to możliwe wyłącznie na drodze empirycznej, która – jeżeli rozpatrywanych jest wiele różnych podziałów – może być drogą bardzo czasochłonną i przez to, kosztowną w zastosowaniach praktycznych przetwarzania równoległego. Dlatego też, potencjalnym usprawnieniem w tym zakresie byłoby zastosowanie podejścia o charakterze heurystycznym, w celu wyłonienia z możliwych podziałów tych, które mają najlepsze rokowania odnośnie czasu wykonania, i ograniczenie zakresu empirycznej selekcji do tak zawężonego zbioru. Potencjalne korzyści praktyczne z zastosowania takiego podejścia heurystycznego stały się inspiracją dla niniejszej pracy i wyznaczyły jej cel, wpisujący się w nakreśloną powyżej koncepcję, mianowicie: *opracowanie rodziny modeli statystycznych do oszacowania czasu wykonania pewnej, wybranej klasy aplikacji równoległych.*
W niniejszej dysertacji przedstawiono szczegółowo kroki, podjęte celem opracowania wspomnianej rodziny modeli i generalizującego ją, modelu ogólnego. W dysertacji, na tle aktualnego stanu wiedzy na temat przetwarzania równoległego, przedstawiono następujące zagadnienia:

- analizę potencjalnych czynników, które powinny zostać uwzględnione w modelu ogólnym,
- wybór rodzaju i postaci modelu ogólnego oraz uzasadnienie dokonanego wyboru,
- wyznaczenie wartości parametrów modeli szczególnych, będących konkretyzacją modelu ogólnego dla danego środowiska sprzętowo-programowego,
- zakres stosowalności modeli szczególnych,
- ocenę jakości modelu ogólnego i modeli szczególnych,
- stosowanie modelu ogólnego i modeli szczególnych w praktyce.

W dysertacji przedstawiono i szeroko omówiono wyniki badań eksperymentalnych. Celem tych badań było wykazanie, że zaproponowane podejście autorskie pozwala wyłonić z możliwych postaci pewnej klasy programów równoległych te postaci, które mają najlepsze rokowania odnośnie czasu wykonania. Uzyskane wyniki badań potwierdziły prawdziwość weryfikowanej hipotezy badawczej.

Nakreślony powyżej zakres niniejszej pracy mieści się w następujących obszarach informatyki wg klasyfikacji „The 2012 ACM Computing Classification System”:

COMPUTER SYSTEMS ORGANIZATION
Architectures
Parallel architectures

SOFTWARE AND ITS ENGINEERING
Software notations and tools
Compilers

MATHEMATICS OF COMPUTING
Probability and statistics
Statistical paradigms
Regression analysis

COMPUTING METHODOLOGIES
Parallel computing methodologies

COMPUTING METHODOLOGIES
Modeling and simulation
Model development and analysis

STATISTICAL MODELS FOR THE ESTIMATION OF THE PERFORMANCE OF COARSE-GRAINED OPENMP C/C++ PROGRAMS AND THE APPLICATION OF THE MODELS IN ITERATIVE COMPILATION

ABSTRACT

Minimization of data processing duration is one of the fundamental problems of modern computer science. This problem is of particular importance in practical applications of computers – as, irrespective of the application area (science, industry, transport, daily life, etc.) – computer users expect the machines to solve problems and execute tasks as quickly as possible. In order to meet these expectations, many different concepts regarding the operation of computer hardware and software have been elaborated and implemented. The focal point of this thesis is parallel computing – an approach within the trend set by the aforementioned concepts, which combines hardware and software aspects.

The essence of parallel computing is division of the task to be executed into smaller tasks which are then executed in parallel, which in turn results in shortening of the duration of execution of the initial task. This immanent characteristic of parallel computing involves the following, significant consequences:

- Parallel computing may be applied only to these problems and tasks which are divisible into “parts” intended for parallel processing.
- If parallel computing may be applied to a given problem or task, then there is *at least* one way of dividing the problem/task into “parts” intended for parallel processing.

If there exists more than one possible division of the initial problem/task, the inevitable question is: *Which of the possible divisions is the best one, i.e. results in the shortest duration of solving/executing a parallelized problem/task in the target hardware environment?*

The proper answer to this question cannot be found in a purely theoretical way; it can only be found in an empirical way which – if there are many possible divisions to be considered – may turn out to be very time consuming and thus, expensive from the perspective of practical applications of parallel computing. Therefore, a potential improvement in this respect would be to apply a heuristic approach in order to choose from the possible divisions the ones with the shortest expected execution durations and later carry out the empirical selection only for the so limited set of possible divisions. Potential practical advantages expected from applying this kind of heuristic approach have become an inspiration for this thesis and set its purpose, which is consistent with the above specified concept, and is as follows: *elaboration of a family of statistical models for estimation of the execution time of a specific class of parallelized computer programs.*

This dissertation presents in detail the steps taken in order to elaborate the aforementioned family of models and a model generalising this family (hereafter referred to as “general

model"). In the dissertation, the following issues are presented against the state of the art in parallel computing:

- Analysis of potential factors to be taken into account in the general model,
- Selection of the type and form of the general model and justification for the adopted selection,
- Determination of the values of parameters of actual models derived from the general model for a given hardware-software environment,
- Scope of applicability of actual models,
- Assessment of quality of the general model and actual models,
- Applying the general model and actual models in practice.

The results of the carried out experimental research are presented and widely discussed in the dissertation. The purpose of the research was to prove that, by applying the approach proposed by the author, one could select from the possible versions of a specific class of parallelized computer programs the versions with the shortest expected execution durations. The obtained results of the research have proved the correctness of the research hypothesis under examination.

The above presented scope of this thesis is within the following areas of computer science specified in "The 2012 ACM Computing Classification System":

COMPUTER SYSTEMS ORGANIZATION
Architectures
Parallel architectures

SOFTWARE AND ITS ENGINEERING
Software notations and tools
Compilers

MATHEMATICS OF COMPUTING
Probability and statistics
Statistical paradigms
Regression analysis

COMPUTING METHODOLOGIES
Parallel computing methodologies

COMPUTING METHODOLOGIES
Modelling and simulation
Model development and analysis

1. WSTĘP

1.1. Przedstawienie problemu

W zastosowaniach praktycznych komputerów, kluczowe znaczenie ma czas przetwarzania danych. W związku z tym, ogólną tendencją jest dążenie do możliwie największego skrócenia czasu przetwarzania danych. Jednym ze sposobów osiągnięcia tego celu jest zwiększenie prędkości działania procesorów. Z uwagi na ograniczenia wynikające z praw fizyki, nie można zwiększać w nieskończoność prędkości działania pojedynczych procesorów opartych na technologiach półprzewodnikowych. Jedną z alternatyw dla zwiększania prędkości pojedynczych procesorów stało się tworzenie maszyn wieloprocesorowych. W przeciwieństwie do maszyn (komputerów) jednoprocesorowych, maszyny wieloprocesorowe zapewniają pełne sprzętowe wsparcie dla przetwarzania równoległego.

Istotą przetwarzania równoległego jest podział zadania, które ma zostać wykonane, na mniejsze podzadania realizowane równocześnie (czyli, innymi słowy, równolegle), co w efekcie prowadzi do skrócenia czasu wykonywania zadania wyjściowego. Przeciwieństwem przetwarzania równoległego jest przetwarzanie sekwencyjne. W praktyce, przetwarzanie równoległe, realizowane w wieloprocesorowym środowisku sprzętowym, daje możliwość realizacji złożonych i skomplikowanych obliczeniowo zadań w akceptowalnym czasie, nieosiągalnym dla tradycyjnego przetwarzania sekwencyjnego realizowanego w jednoprocesorowym środowisku sprzętowym.

W przetwarzaniu równoległym, najważniejszy jest czas wykonania programu równoległego. W związku z tym, kluczowe kryteria oceny wydajności aplikacji równoległych to:

- czas wykonania programu równoległego,
- relacja czasu wykonania programu równoległego do czasu wykonania równoważnego programu sekwencyjnego,
- relacja pomiędzy czasem wykonania programu równoległego a zasobami sprzętowymi, jakie zostały użyte, aby wykonać program równolegle.

Jednym z podstawowych problemów przetwarzania równoległego jest praktyczna możliwość zastosowania tego podejścia. Jeżeli tylko dany problem bądź zadanie można rozwiązać przy pomocy komputera, to zawsze można to zrobić w sposób sekwencyjny, ale nie zawsze w sposób równoległy, gdyż pewne problemy i zadania z samej swojej natury wykluczają możliwość podziału ich na podzadania realizowane w sposób równoległy.
Jeżeli dany problem bądź zadanie można rozwiązać w sposób równoległy, to istnieją różne, ale semantycznie równoważne, podziały przedmiotowego problemu bądź zadania na podzadania realizowane w sposób równoległy. Wynikające z poszczególnych podziałów czasy rozwiązywania problemu bądź zadania wyjściowego mogą się istotnie różnić między sobą. Oczywiście, do ostatecznego zastosowania (zwłaszcza o charakterze komercyjnym lub naukowym) należałoby wybrać, spośród wszystkich możliwych, ten podział, który gwarantuje najkrótszy czas rozwiązywania problemu bądź zadania wyjściowego. Najprostszym, ale zarazem czasochłonnym, sposobem skutecznego wyboru najlepszego

spośród rozważanych podziałów jest przeprowadzenie kompilacji iteracyjnej. W ramach kompilacji iteracyjnej, dokonuje się empirycznego sprawdzenia czasów wykonania rozważanych podziałów w docelowym środowisku sprzętowym, a następnie do ostatecznego stosowania wybiera się ten z nich, który ma najkrótszy czas wykonania.
Potencjalnym obszarem usprawnień w zakresie kompilacji iteracyjnej jest zastosowanie oszacowań w celu wyłonienia, z grupy możliwych do zastosowania podziałów, tych najlepiej rokujących. Po zastosowaniu oszacowań, jako formy wstępnej selekcji, w środowisku docelowym wystarczyłoby sprawdzić zachowanie wyselekcjonowanej „grupy" – w rezultacie, czas wykonania kompilacji iteracyjnej ulegnie skróceniu.
Podstawowym problemem, który się tu pojawia, jest opracowanie metodyki przeprowadzania przedmiotowych oszacowań.

1.2. Cel i teza pracy

W rozdziale 2.3 niniejszej pracy przedstawiono analizę stosowanych typowo sposobów szacowania czasu wykonania programów równoległych.
Z analizy tej wynika, że w swoim typowym kształcie, żaden z analizowanych sposobów szacowania czasu wykonania programów równoległych nie stanowi metodycznego podejścia pozwalającego na realizację proponowanego usprawnienia kompilacji iteracyjnej, tj. na wstępną selekcję tych postaci (transformacji) kodu źródłowego programu, które w danym środowisku sprzętowym mają najlepsze rokowania odnośnie czasu wykonania. W związku z tym, na potrzeby przedmiotowego usprawnienia kompilacji iteracyjnej proponuje się opracować dedykowaną do tego celu metodykę przeprowadzania wspomnianych oszacowań. Z uwagi na jej przeznaczenie, metodyka ta powinna zapewniać:

1/ możliwość zastosowania dla szerokiego spektrum przypadków spotykanych w praktyce,
2/ uwzględnienie w oszacowaniach specyfiki środowiska sprzętowego, w którym będzie wykonywany program,
3/ możliwość przeprowadzenia oszacowań czasu wykonania programu wyłącznie na podstawie znajomości kodu źródłowego programu i charakterystyki środowiska sprzętowego, w którym program będzie wykonywany.

Uzasadnienie wymagań 1/ ÷ 3/ oraz proponowany sposób pokrycia tych wymagań w metodyce autorskiej:

- Wymaganie 1/

Opracowana metodyka powinna być adekwatna do stosowania dla możliwie największej liczby przypadków, spotykanych w praktyce. Z uwagi na mnogość i różnorodność współczesnych języków programowania oraz metodyk tworzenia oprogramowania, struktura współczesnych programów komputerowych jest bardzo zróżnicowana. Pomimo tego zróżnicowania, większość czasochłonnych operacji – obliczeń wykonywanych w programach – jest realizowana w pętlach. W związku z tym, w programach równoległych dąży się przede

wszystkim do zrównoleglenia pętli. Przy zrównolegleniu, istotną kwestią jest granulacja zrównoleglonej pętli, czyli liczba operacji pomiędzy zdarzeniami komunikacji lub zdarzeniami synchronizacji występującymi w programie.

Można wyróżnić 2 typy granulacji: granulację gruboziarnistą i granulację drobnoziarnistą. Granulacja gruboziarnista występuje wtedy, gdy czas wykonania występujących w programie operacji przetwarzania danych jest dłuższy, niż łączny czas: inicjalizacji tych operacji i transferu danych potrzebnych do wykonania tych operacji. Ten typ granulacji odpowiada strukturze pętli zagnieżdżonej, w której zrównoleglenie jest dokonane w najbardziej zewnętrznej pętli gniazda. Granulacja drobnoziarnista występuje wtedy, gdy czas wykonania występujących w programie operacji przetwarzania danych jest krótszy, niż łączny czas: inicjalizacji tych operacji i transferu danych potrzebnych do wykonania tych operacji. Ten typ granulacji odpowiada strukturze pętli zagnieżdżonej, w której zrównoleglenie jest dokonane w którejś z wewnętrznych pętli gniazda.

Z uwagi na to, że granulacja gruboziarnista jest standardowo stosowana przy zrównolegleniu programów wykonywanych na bardzo popularnych obecnie maszynach wieloprocesorowych z pamięcią dzieloną zakłada się, że obszarem stosowalności zaproponowanej metodyki będą gruboziarniste, równoległe pętle programowe, zrównoleglone w standardzie OpenMP. (Wybór standardu OpenMP wynika z powszechnego stosowania tego standardu.)

- Wymaganie 2/

Współczesne systemy komputerowe cechują się bardzo dużym zróżnicowaniem ze względu na parametry kluczowe dla ich wydajności. Nawet drobna zmiana w newralgicznych parametrach wydajnościowych środowiska sprzętowego może skutkować bardzo dużą zmianą wydajności środowiska sprzętowego, co z kolei rzutuje na czas wykonania programów uruchomionych w tym środowisku. Dlatego też, w opracowywanej metodyce należy uwzględnić newralgiczne parametry wydajnościowe środowiska sprzętowego.

- Wymaganie 3/

Celem proponowanych usprawnień kompilacji iteracyjnej jest skrócenie czasu przeprowadzania kompilacji iteracyjnej. Z uwagi na naturę kompilacji iteracyjnej, cel ten można osiągnąć zastępując czasochłonne wykonywanie różnych postaci programu w docelowym środowisku sprzętowym szybko uzyskiwanymi oszacowaniami ich czasu wykonania, pozwalającymi uszeregować poszczególne postaci programu malejąco wg oczekiwanych czasów wykonania w docelowym środowisku sprzętowym.

W związku z tym, osią zaproponowanej metodyki powinien być model (lub rodzina modeli), pozwalający na obliczeniowe oszacowanie czasu wykonania zrównoleglonego programu (pętli programowej) w docelowym środowisku sprzętowym. Biorąc pod uwagę dużą różnorodność czynników tworzących środowisko sprzętowe oraz złożone i bardzo trudne do

sformalizowania relacje pomiędzy tymi czynnikami, model ten powinien być modelem statystycznym.

W niniejszej pracy, przez „model statystyczny" rozumie się równanie (lub układ równań) opisujące zależności pomiędzy zmiennymi niezależnymi, zmiennymi zależnymi oraz parametrami modelu. Przewiduje się, że wartości parametrów modelu zostaną wyznaczone poprzez analizę regresji przeprowadzoną dla specjalnie w tym celu dobranej próby (prób).

W świetle przedstawionej powyżej dyskusji, celem niniejszej pracy jest:

> *Opracowanie rodziny modeli statystycznych (przewidywanych do zastosowania w optymalizujących kompilatorach zrównoleglających) pozwalających na oszacowanie czasu wykonania aplikacji gruboziarnistych w standardzie OpenMP C/C++ przez komputery wieloprocesorowe z pamięcią dzieloną.*

Aby zrealizować powyższy cel, należy wykazać, że:

> *Istnieje możliwość opracowania rodziny modeli statystycznych do oszacowania czasu wykonania aplikacji gruboziarnistych w standardzie OpenMP C/C++, które pozwolą na osiągnięcie zadawalającej dokładności oszacowania.*

Dokładność oszacowania jest pojęciem jakościowym, określającym zgodność wyniku oszacowania z wartością zmierzoną empirycznie.

W niniejszej pracy, dokładność oszacowania jest uznawana za zadawalającą, jeżeli:

- wielkość uzyskanego błędu oszacowania nie przekracza 100 punktów procentowych (poziom przyjęty arbitralnie, z uwagi na bardzo dużą złożoność modelowanego zjawiska, co przedstawiono szerzej na stronach 18 ÷ 25), oraz
- oszacowania dokonane wg modeli należących do opracowanej rodziny modeli są wystarczające do wskazania tych postaci kodu źródłowego programu, które w danym środowisku sprzętowym mają najlepsze rokowania odnośnie czasu wykonania, a weryfikacja przeprowadzona drogą kompilacji iteracyjnej potwierdza, że wskazania wynikające z oszacowań są zgodne z wynikami uzyskanymi empirycznie. W praktyce oznacza to, że dla danego rozmiaru problemu rozwiązywanego w programie (pętli programowej), kierunek zmian zmierzonych czasów wykonania poszczególnych wersji danego programu (pętli programowej) oraz kierunek zmian odpowiadających im czasów oszacowanych wg modeli należących do opracowanej rodziny modeli jest identyczny.

Aby zrealizować cel pracy i wykazać prawdziwość postawionej hipotezy badawczej, należy zrealizować następujące zadania cząstkowe:

1/ Sformułowanie listy czynników, mających wpływ na czas wykonania gruboziarnistych, równoległych pętli programowych, zrównoleglonych w standardzie OpenMP,

2/ Sformułowanie listy potencjalnych zmiennych objaśniających modelu (będącego protoplastą poszukiwanej rodziny modeli) wyrażających w sposób ilościowy czynniki, o których mowa w zadaniu 1/,

3/ Określenie sposobu konstrukcji modelu,

4/ Określenie możliwych struktur modelu,

5/ Wybór zmiennych objaśniających modelu, spośród potencjalnych zmiennych objaśniających, o których mowa w zadaniu 2/,

6/ Wybór struktury modelu, spośród możliwych struktur modelu, o których mowa w zadaniu 4/,

7/ Opracowanie sposobu oszacowania parametrów modelu,

8/ Opracowanie sposobu oceny jakości modelu,

9/ Opracowanie sposobu wykorzystania modelu w kompilacji iteracyjnej,

10/ Przeprowadzenie badań eksperymentalnych, mających na celu ocenę faktycznych możliwości zastosowania opracowanego modelu (rodziny modeli) w kompilacji iteracyjnej,

11/ Sformułowanie wniosków z przeprowadzonych badań.

Zakres pracy, wyznaczony przez wymienione powyżej zadania cząstkowe, mieści się w następujących obszarach informatyki wg klasyfikacji „The 2012 ACM Computing Classification System” [82]:

COMPUTER SYSTEMS ORGANIZATION
- *Architectures*
  - ***Parallel architectures***

SOFTWARE AND ITS ENGINEERING
- *Software notations and tools*
  - ***Compilers***

MATHEMATICS OF COMPUTING
- *Probability and statistics*
  - *Statistical paradigms*
    - ***Regression analysis***

COMPUTING METHODOLOGIES
- ***Parallel computing methodologies***

COMPUTING METHODOLOGIES

Modeling and simulation

Model development and analysis

1.3. Zawartość pracy

Praca jest podzielona na pięć rozdziałów o następującej zawartości:

Rozdział 1 Wstęp

W rozdziale przedstawiono uzasadnienie wyboru tematu, cel i tezę pracy oraz strukturę pracy.

Rozdział 2 Przegląd aktualnego stanu wiedzy na temat przetwarzania równoległego

W rozdziale przedstawiono następujące zagadnienia:

- mierniki oceny wydajności aplikacji równoległych,
- typowo stosowane podejścia do tworzenia aplikacji równoległych,
- możliwości oszacowania wydajności aplikacji równoległych.

Rozdział 3 Opracowanie ogólnego modelu statystycznego oraz modeli szczególnych

W rozdziale przedstawiono następujące zagadnienia:

- analizę potencjalnych czynników, które powinny zostać uwzględnione w modelu ogólnym,
- wybór rodzaju i postaci modelu ogólnego oraz uzasadnienie dokonanego wyboru,
- wyznaczenie wartości parametrów modeli szczególnych,
- zakres stosowalności modeli szczególnych,
- ocenę jakości modelu ogólnego i modeli szczególnych,
- stosowanie modelu ogólnego i modeli szczególnych w praktyce,
- prace pokrewne.

Rozdział 4 Badania eksperymentalne

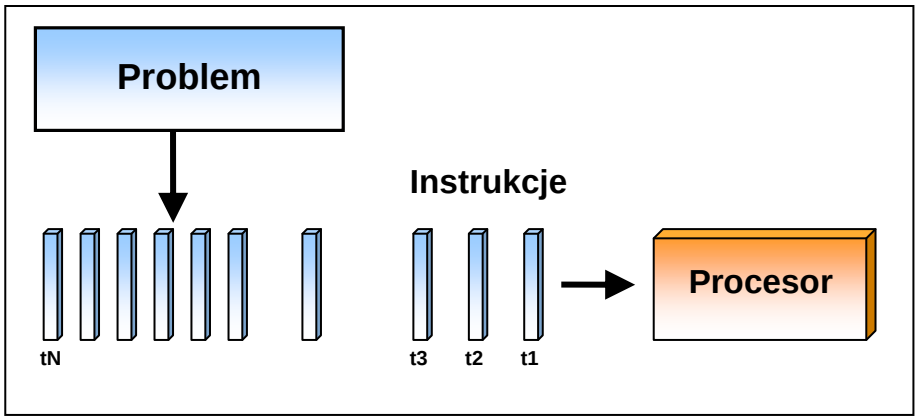
W rozdziale przedstawiono sposób przeprowadzenia badań eksperymentalnych oraz uzyskane wyniki, w ujęciu liczbowym i graficznym.

Rozdział 5 Podsumowanie

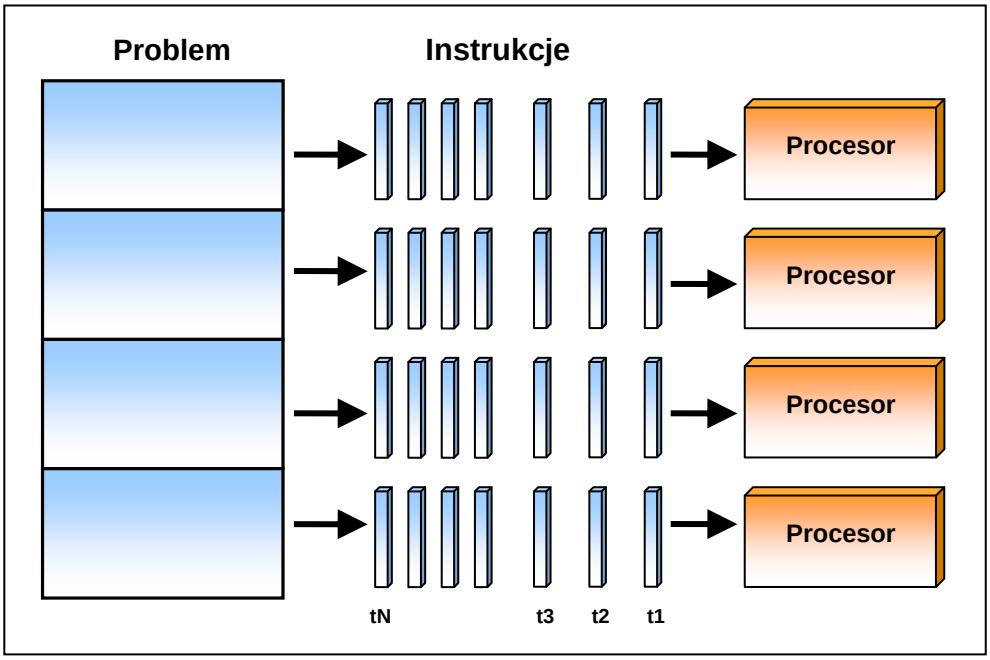
W rozdziale dokonano oceny otrzymanych wyników w odniesieniu do celu i tezy pracy oraz wskazano potencjalne możliwości dalszego wykorzystania podejścia zaproponowanego w pracy.

2. PRZEGLĄD AKTUALNEGO STANU WIEDZY NA TEMAT PRZETWARZANIA RÓWNOLEGŁEGO

Istotą przetwarzania równoległego realizowanego w systemie komputerowym jest podział zadania, które ma zostać wykonane, na mniejsze podzadania realizowane równocześnie, czyli, innymi słowy, równolegle (patrz Rysunek 2). Przeciwieństwem tak rozumianego przetwarzania równoległego jest przetwarzanie sekwencyjne (patrz Rysunek 1).



Rysunek 1 Przetwarzanie sekwencyjne realizowane w systemie komputerowym [7]



Rysunek 2 Przetwarzanie równoległe realizowane w systemie komputerowym [7]

W aspekcie praktycznym oznacza to, że aby wykonać postawione zadanie drogą przetwarzania równoległego realizowanego w systemie komputerowym, muszą być spełnione jednocześnie następujące warunki:

1/ Istnieje możliwość podziału postawionego zadania na równoważne mu semantycznie podzadania, które będą realizowane w sposób równoległy.

2/ Architektura i organizacja systemu komputerowego są adekwatne dla przetwarzania równoległego.

Architektura komputera odnosi się do tych atrybutów systemu, które są widoczne dla programisty. Atrybuty te mają bezpośredni wpływ na logiczne wykonywanie programu[1]. Przykładami atrybutów architektury są: lista rozkazów, liczba bitów wykorzystanych do prezentacji różnych typów danych (np. liczb czy znaków), mechanizmy wejścia-wyjścia oraz metody adresowania pamięci. [77]
Organizacja komputera odnosi się do jednostek operacyjnych i ich połączeń, które stanowią realizację specyfikacji typu architektury. Do atrybutów organizacyjnych należą rozwiązania sprzętowe niewidoczne dla programisty, takie jak: sygnały sterujące, interfejsy między komputerem a urządzeniami peryferyjnymi oraz wykorzystywana technologia pamięci. [77]

Niektóre zadania z samej swojej natury wykluczają możliwość podziału ich na podzadania realizowane w sposób równoległy – dla tych zadań, warunek 1/ nie będzie nigdy spełniony, a więc nie da się ich rozwiązać na drodze przetwarzania równoległego.
Spełnienie warunku 2/ jest tożsame z użyciem systemu komputerowego o architekturze i organizacji równoległej.

Jedna z najczęściej obecnie stosowanych klasyfikacji architektur systemów komputerowych, taksonomia Flynna [77] [86], w oparciu o kryteria:

- liczba jednocześnie przetwarzanych strumieni rozkazów,
- liczba elementów danych, na których można jednocześnie przeprowadzać operacje

wyróżnia następujące klasy maszyn:

- *SISD* (Single Instruction, Single Data Stream) – pojedynczy strumień rozkazów, pojedynczy strumień danych. Do tej klasy należą maszyny jednoprocesorowe, bazujące na architekturze von Neumanna [77]. W niektórych przypadkach, w maszynach SISD mogą pojawić się pewne elementy równoległości, np. przetwarzanie potokowe w procesorze.
- *SIMD* (Single Instruction, Multiple Data Stream) – pojedynczy strumień rozkazów, zwielokrotniony strumień danych. Do tej klasy należą maszyny, w których ta sama operacja jest wykonywana jednocześnie na wielu danych.

[1] W niniejszej pracy, przez „program" rozumie się „program komputerowy" (ang. *computer program*) – czyli, sekwencję symboli opisującą obliczenia/operacje do wykonania przez komputer zgodnie z pewnymi regułami zwanymi językiem programowania [62].

Tak rozumiany program komputerowy można reprezentować dwojako: przy pomocy kodu źródłowego lub przy pomocy kodu maszynowego.

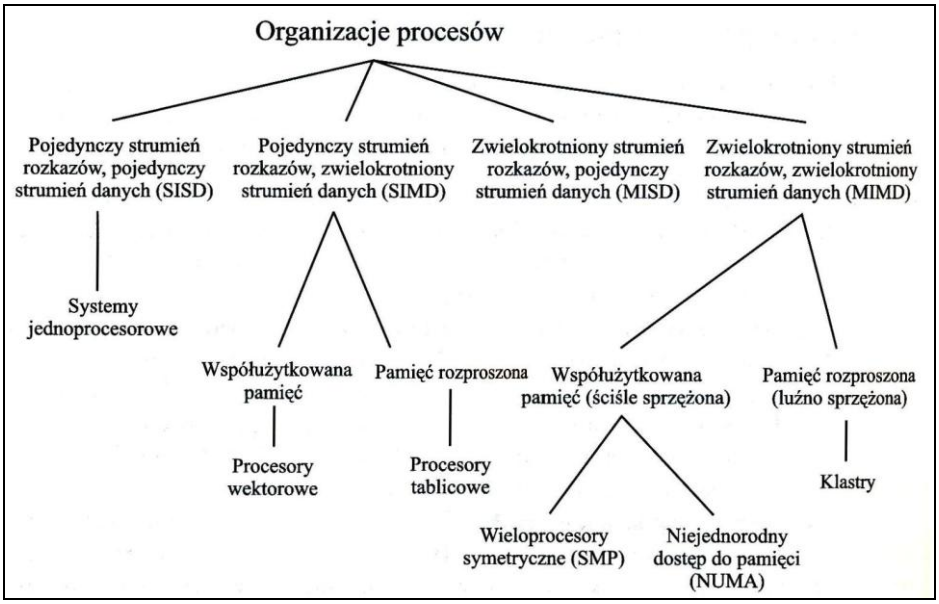
Kod źródłowy jest to reprezentacja programu komputerowego przy pomocy języka (programowania) zrozumiałego dla człowieka. Kod maszynowy (alternatywne nazwy: postać binarna programu, postać wykonywalna programu) jest to reprezentacja programu komputerowego w postaci zrozumiałej dla maszyny, tj. za pomocą ciągu zer i jedynek.

- *MISD* (Multiple Instruction, Single Data Stream) – zwielokrotniony strumień rozkazów, pojedynczy strumień danych. Do tej klasy należą maszyny wykonujące różne operacje na tych samych danych. Taka architektura nie jest stosowana w rozwiązaniach komercyjnych.
- *MIMD* (Multiple Instruction, Multiple Data Stream) – zwielokrotniony strumień rozkazów, zwielokrotniony strumień danych. Jest to najważniejsza klasa komputerów równoległych wg taksonomii Flynna. Do tej klasy należą maszyny, w których poszczególne procesory wykonują jednocześnie różne operacje, na różnych danych, ale w ramach tego samego zadania obliczeniowego.

Maszyny typu SIMD i MIMD można dodatkowo podzielić na:

- maszyny z pamięcią wspólną (współdzieloną, dzieloną, współużytkowaną, globalną), czyli takie, że wszystkie ich procesory mają jedną i tą samą, wspólną przestrzeń adresową,
- maszyny z pamięcią lokalną (rozproszoną), czyli maszyny wieloprocesorowe, w których każdy procesor ma swoją własną pamięć, a także sieci komputerowe.

Taksonomię Flynna przedstawiono na Rysunku 3.



Rysunek 3 Taksonomia Flynna [77]

Cel, który został postawiony w niniejszej pracy, dotyczy komputerów wieloprocesorowych z pamięcią dzieloną – maszyn MIMD z pamięcią dzieloną (wg taksonomii Flynna).

2.1. Mierniki oceny wydajności aplikacji równoległych

W naukach technicznych, wydajność maszyny to wielkość pracy, jaką maszyna ta może typowo wykonać w jednostce czasu, w określonych warunkach pracy. Aby ocenić wydajność maszyny, należy znać jej budowę oraz warunki pracy. Na podstawie znajomości wydajności maszyny można ocenić w jakim czasie maszyna ta wykona konkretne zadanie. Porównując ze sobą dwie maszyny o nieznanej wydajności, za bardziej wydajną przy realizacji konkretnego zadania należy uznać tę z nich, która wykona to zadanie w krótszym czasie. Oznacza to, że miarą, której można użyć do oceny wydajności realizacji zadań jest czas ich wykonania.

Przez analogię do powyższego przykładu, miarą, którą można wykorzystać do oceny wydajności wykonania programu jest czas jego wykonania w środowisku wykonania [41].

Przy ocenie wydajności wykonania aplikacji równoległych, czas wykonania programu powinno się zawsze rozpatrywać w kontekście środowiska sprzętowego, w którym ten czas osiągnięto. Oznacza to, że czas wykonania programu równoległego jest wielkością pierwotną, niezbędną do wyznaczenia wartości wielkości pochodnych: przyspieszenia i efektywności, które dopiero dają możliwość faktycznej oceny wydajności aplikacji równoległych [39].

Przyspieszenie wyraża się wzorem (1):

$$S(P) = \frac{T(1)}{T(P)} \tag{1}$$

gdzie:
$T(1)$ – czas wykonania programu na 1 procesorze
$T(P)$ – czas wykonania programu na P procesorach
$S(P)$ – przyspieszenie osiągnięte w wyniku wykonywania programu na P procesorach

Przyspieszenie wskazuje, ile razy szybciej program zostanie wykonany, gdy stosowane jest przetwarzanie równoległe zamiast przetwarzania sekwencyjnego. Zazwyczaj, $1 \leq S(P) \leq P$. Możliwe są jednak sytuacje, gdy $S(P) < 1$ lub $S(P) > P$ (tzw. przyspieszenie superliniowe). Gdy $S(P) = P$, przyspieszenie jest liniowe.

Efektywność wyraża się wzorem (2):

$$E(P) = \frac{S(P)}{P} = \frac{T(1)}{P \times T(P)} \tag{2}$$

gdzie:
$E(P)$ – efektywność osiągnięta w wyniku wykonywania programu na P procesorach

Efektywność wskazuje, jak osiągnięte przyspieszenie $S(P)$ ma się do liczby procesorów, które zostały użyte, by osiągnąć przedmiotowe przyspieszenie. Zazwyczaj $0 \leq E(P) \leq 1$, przy czym pożądane jest by $E(P) \rightarrow 1$. Jest jednak możliwe, że $E(P) > 1$ (dla przyspieszenia superliniowego).

Przy ocenie wydajności aplikacji równoległych istotne znaczenie ma także lokalność. Wysoka lokalność przekłada się na możliwość pobrania z pamięci szybko dostępnej dla procesora (np. pamięci podręcznej procesora) istotnej części wszystkich danych, które mają być przetworzone w programie. W rezultacie prowadzi to do skrócenia czasu wykonania programu w stosunku do czasu, jaki zajęłoby wykonanie programu gdyby wszystkie potrzebne dane były pobierane z pamięci o dłuższym czasie dostępu (np. pamięci RAM).

2.2. Podejścia do tworzenia aplikacji równoległych

Istnieją 2 alternatywne możliwości tworzenia aplikacji równoległych: „ręczne" tworzenie aplikacji równoległych oraz automatyczne tworzenie aplikacji równoległych. W przypadku ręcznego tworzenia, ciężar napisania „od podstaw" programu równoległego bądź zrównoleglenia istniejącego programu sekwencyjnego spoczywa w całości na programiście. W przypadku automatycznego tworzenia aplikacji równoległych, istniejące programy sekwencyjne są zrównoleglane w sposób automatyczny przez dedykowane do tego celu kompilatory zrównoleglające.

„Ręczne" tworzenie aplikacji równoległych jest czasochłonne i obarczone dużym ryzykiem popełnienia błędów przez programistę. Z tych względów, korzystniejszym rozwiązaniem jest tworzenie ich w sposób automatyczny.

Obecnie, na etapie kompilacji, nie ma możliwości stwierdzenia w sposób czysto teoretyczny, jak dowolny ciąg semantycznie dozwolonych transformacji, użytych do zrównoleglenia danego sekwencyjnego kodu źródłowego, przełoży się na czas wykonania zrównoleglonego programu wynikowego w docelowym środowisku sprzętowym.
Ze względu na występującą w niektórych zastosowaniach praktycznych (np. systemy wbudowane, obliczenia naukowe itp.) konieczność i potrzebę dużej wydajności, stosuje się tzw. kompilację iteracyjną. Kompilacja iteracyjna polega na utworzeniu, w kolejnych krokach (iteracjach) wielu różnych, ale semantycznie identycznych (tutaj: zrównoleglonych), wersji danego programu (tutaj: sekwencyjnego). Każda wersja jest wykonywana w docelowym środowisku sprzętowym. Na podstawie zmierzonych czasów wykonania poszczególnych wersji programu do ostatecznego stosowania wybiera się tę wersję, której czas wykonania jest najkrótszy [16] [36] [50] [51] [52].

2.3. Możliwości oszacowania wydajności aplikacji równoległych

Kompilacja iteracyjna jest podejściem skutecznym, ale zarazem czasochłonnym. Potencjalnym obszarem usprawnień w zakresie kompilacji iteracyjnej jest zastosowanie

oszacowań w celu wyłonienia, z grupy możliwych do zastosowania transformacji, tych najlepiej rokujących. Dzięki zastosowaniu oszacowań jako formy wstępnej selekcji, w środowisku docelowym wystarczy sprawdzić zachowanie wyselekcjonowanej „grupy" – w rezultacie, czas kompilacji iteracyjnej ulegnie skróceniu [16] [36] [50] [51] [52].

Do oszacowania wydajności aplikacji równoległych można wykorzystać 2 alternatywne podejścia: profilowanie oraz modele (do oszacowania) wydajności.
Profilowanie to zbieranie informacji o zachowaniu programu (np. użycie poszczególnych instrukcji, częstotliwość i czas trwania wywołań poszczególnych funkcji) w trakcie jego wykonywania [90], a następnie wykorzystywanie tych informacji do analizy zachowania programu i jego optymalizacji. Ze względu na czasochłonność profilowania, wykorzystanie go do usprawnienia kompilacji iteracyjnej jest nieopłacalne. Dlatego też, jedyną praktycznie sensowną alternatywą dla profilowania jest wykorzystanie modeli wydajności, zorientowanych na oszacowanie czasu wykonania programu.

Model to uproszczony obraz pewnego aspektu rzeczywistości. Model powinien jak najlepiej odwzorowywać te właściwości, które są ważne z punktu widzenia prowadzonej analizy, natomiast powinien pomijać częściowo lub całkowicie te właściwości, które są nieistotne z punktu widzenia prowadzonej analizy.
Dobry model powinien uzasadniać obserwacje empiryczne dotyczące rozpatrywanego obiektu/zjawiska oraz powinien umożliwiać dokonywanie prognoz dotyczących rozpatrywanego obiektu/zjawiska.

Typowo stosowane sposoby modelowania wydajności aplikacji równoległych, oparte na pomiarze czasu wykonania programu, to: prawo Amdahla, ekstrapolacja z obserwacji oraz analiza asymptotyczna [12] [13].

Prawo Amdahla mówi, że maksymalne możliwe przyspieszenie, jakie można uzyskać wykonując program równolegle, jest ograniczone przez składnik sekwencyjny programu, tj. tę część programu, która zawsze musi być wykonana w sposób sekwencyjny [3]. Prawo Amdahla nie pozwala stwierdzić wprost, jaki udział w programie równoległym mają operacje sekwencyjne [13].

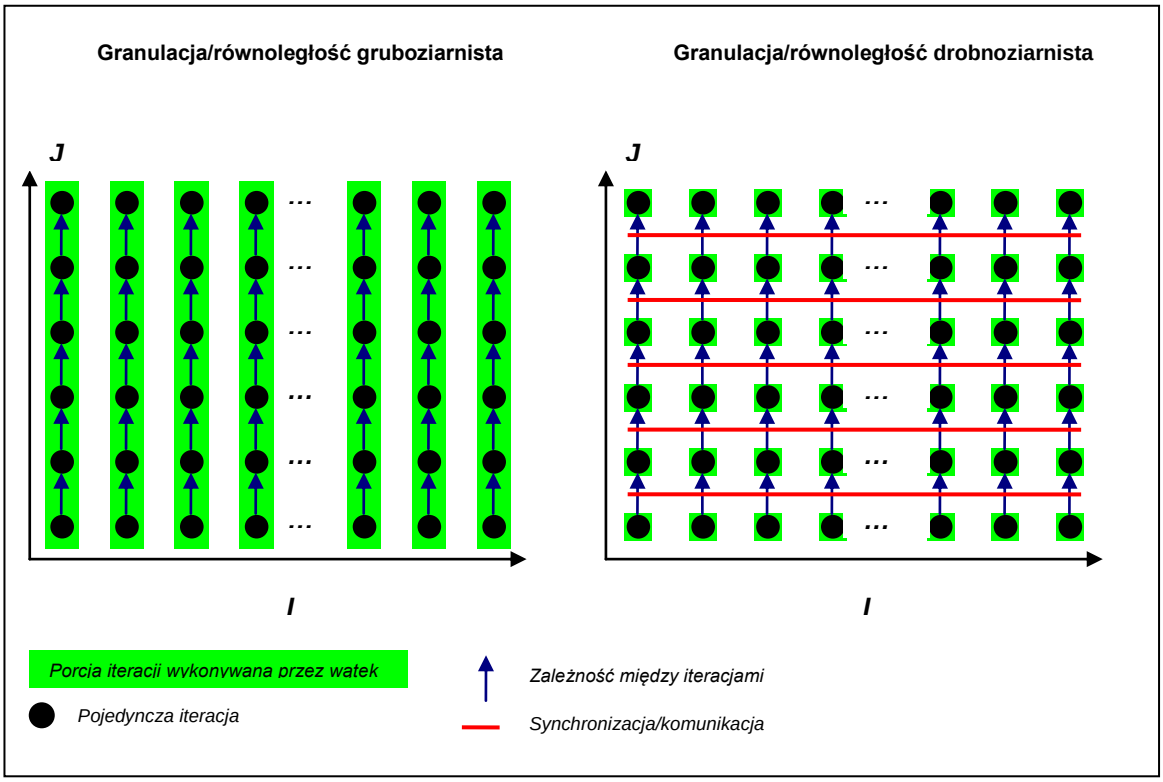
Ekstrapolacja z obserwacji polega na dokonaniu pomiaru wydajności aplikacji w kilku punktach – np. dla kilku różnych rozmiarów problemu, przy liczbie procesorów, na której wykonywany jest program – a następnie wykorzystaniu tych pomiarów do przewidywania zachowania aplikacji w innych punktach [76]. Taki sposób działania w ogóle nie bierze pod uwagę specyfiki algorytmu wykonywanego w programie, przez co ekstrapolacja może być obarczona bardzo dużym błędem [13].

Analiza asymptotyczna pozwala stwierdzić jak czas wykonania programu zmienia się w zależności od rozmiaru problemu, gdy rozmiar problemu i liczba procesorów użytych do wykonania programu są duże. Analiza asymptotyczna nie dostarcza jednak informacji o czasie wykonania programu dla mniejszych rozmiarów problemu i innej liczby procesorów [13].

Omówione powyżej sposoby oszacowania czasu wykonania (a na tej podstawie, wydajności) aplikacji równoległych nie są adekwatne do wstępnej selekcji tych postaci (transformacji) kodu źródłowego programu, które w danym środowisku sprzętowym mają najlepsze rokowania odnośnie czasu wykonania.
Dlatego też w celu usprawnienia kompilacji iteracyjnej poprzez przeprowadzanie takiej właśnie wstępnej selekcji, należałoby opracować zupełnie nowy i dedykowany wyłącznie do tego celu, model do oszacowania czasu wykonania programu równoległego. Aby tak opracowany model był adekwatny dla jak najszerszego spektrum możliwych przypadków, powinien mieć charakter modelu ogólnego z parametrami. Poprzez nadanie parametrom przedmiotowego modelu ogólnego odpowiednich wartości liczbowych – czyli, utworzenie modeli szczególnych – można byłoby dostosować go do konkretnych potrzeb.

Kluczowym problemem jest określenie zakresu stosowalności modelu ogólnego – nie jest bowiem możliwe, aby opracować uniwersalny model ogólny odpowiedni dla wszystkich możliwych przypadków.
Większość czasochłonnych operacji – obliczeń wykonywanych w programach – jest realizowana w pętlach programowych[2]. Dlatego też w programach równoległych dąży się przede wszystkim do zrównoleglenia pętli. Przy zrównolegleniu, istotną kwestią jest granulacja (a w konsekwencji, typ równoległości) zrównoleglonej pętli [21].
Granulacja to liczba operacji pomiędzy zdarzeniami komunikacji lub zdarzeniami synchronizacji występującymi w programie. Można wyróżnić 2 typy granulacji (a w konsekwencji, równoległości): granulację (równoległość) gruboziarnistą i granulację (równoległość) drobnoziarnistą.
Granulacja gruboziarnista [46] występuje wtedy, gdy czas wykonania występujących w programie operacji przetwarzania danych jest dłuższy, niż łączny czas: inicjalizacji tych operacji i transferu danych potrzebnych do wykonania tych operacji. Ten typ granulacji odpowiada strukturze pętli zagnieżdżonej, w której zrównoleglenie jest dokonane w najbardziej zewnętrznej pętli gniazda.
Granulacja drobnoziarnista [59] występuje wtedy, gdy czas wykonania występujących w programie operacji przetwarzania danych jest krótszy, niż łączny czas: inicjalizacji tych operacji i transferu danych potrzebnych do wykonania tych operacji. Ten typ granulacji odpowiada strukturze pętli zagnieżdżonej, w której zrównoleglenie jest dokonane w którejś z wewnętrznych pętli gniazda [12] [13].
Oba typy granulacji przedstawiono poglądowo na Rysunku 4. Struktury pętli zagnieżdżonych, reprezentujących granulację gruboziarnistą oraz granulację drobnoziarnistą, przedstawiono na Rysunku 5.

[2] Pętla programowa to ciąg operacji powtarzanych dopóki nie zostaną spełnione warunki określające koniec takiego postępowania. Pojedyncze powtórzenie zdefiniowanego w pętli ciągu operacji określa się mianem iteracji. Do kontroli przebiegu realizacji pętli, tzn. do określenia która z kolei iteracja pętli jest wykonywana, służy zmienna sterująca pętli.



Rysunek 4 *Granulacja/równoległość grubo- i drobnoziarnista (na podstawie [14])*

Granulacja/równoległość gruboziarnista	**Granulacja/równoległość drobnoziarnista**
parfor	seqfor
seqfor	parfor
…	seqfor
seqfor	…

parfor – pętla wykonywana w sposób równoległy
seqfor – pętla wykonywana w sposób sekwencyjny

Rysunek 5 *Struktury pętli zagnieżdżonych, reprezentujących granulację gruboziarnistą oraz granulację drobnoziarnistą*

Granulacja gruboziarnista jest standardowo stosowana przy zrównolegleniu programów wykonywanych na maszynach wieloprocesorowych z pamięcią dzieloną [47].

Często dokonuje się transformacji kodu źródłowego do postaci semantycznie równoważnej, stosując blokowanie (ang. *tiling*). Blokowanie jest stosowane dla pętli zagnieżdżonych, a jego

ideą jest podział przestrzeni iteracji pętli na mniejsze fragmenty (bloki) [1] [89]. Ma to na celu zwiększenie ponownego użycia danych już znajdujących się w pamięci podręcznej procesora i zmniejszenie ilości transferów danych z pamięci głównej (RAM) do pamięci podręcznej procesora, a w konsekwencji, skrócenie czasu wykonywania programu – co jest obserwowane w przetwarzaniu sekwencyjnym [87] [88].

Jednym z najistotniejszych problemów związanych z blokowaniem jest dobór wymiarów bloków, na które jest dzielona przestrzeń iteracji pętli [35] [68]. Wymiary bloków powinny być dobrane tak, by ponowne użycie danych znajdujących się w pamięci podręcznej procesora było jak największe.

Ideę blokowania przedstawiono na Rysunkach 6 i 7. Na Rysunku 6 przedstawiono kod źródłowy pętli, realizującej mnożenie macierzy – w wersji bez blokowania i z blokowaniem przy zastosowaniu bloku kwadratowego o boku *B*. Na Rysunku 7 przedstawiono sposób dostępu do danych dla pętli z Rysunku 6.

a) mnożenie macierzy – bez blokowania

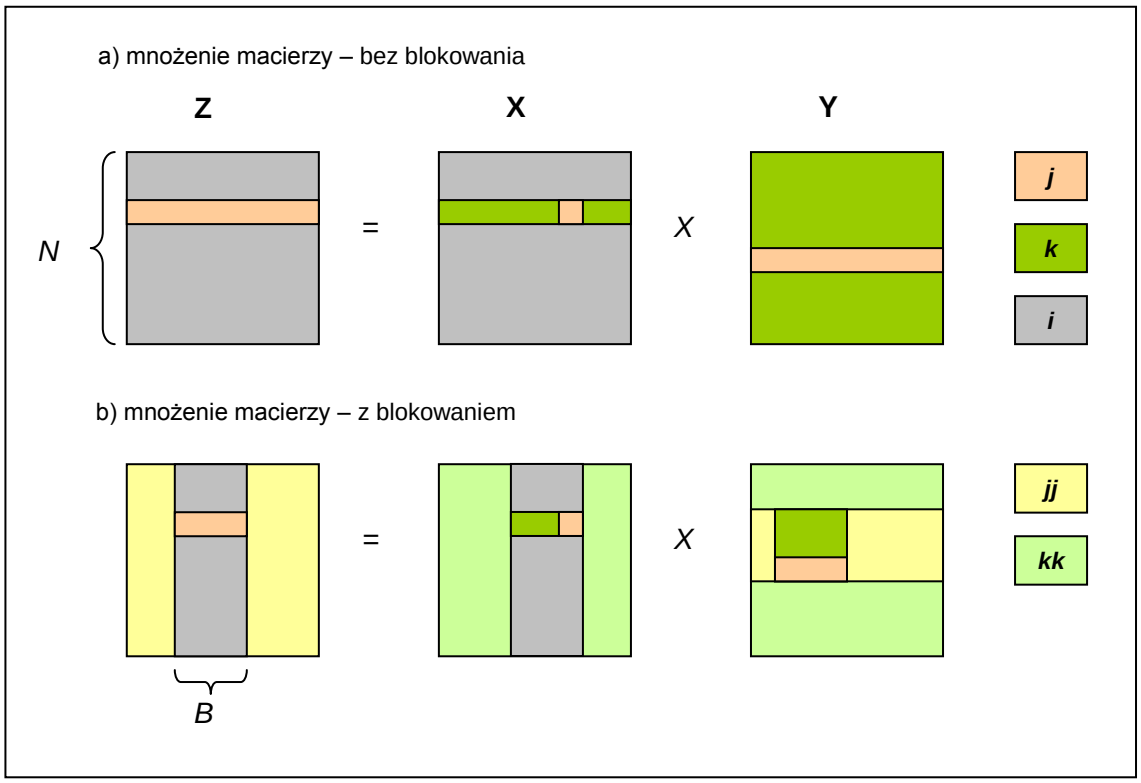
```
for i := 1 to N do
    for k := 1 to N do
        r = X[i,k];
        for j := 1 to N do
            Z[i,j] += r*Y[k,j];
```

b) mnożenie macierzy – z blokowaniem

```
for kk := 1 to N by B do
    for jj := 1 to N by B do
        for i := 1 to N do
            for k := kk to min(kk+B-1, N) do
                r = X[i,k];
                for j := jj to min(jj+B-1, N) do
                    Z[i,j] += r*Y[k,j]
```

min(x,y) – oznacza mniejszą z liczb x, y

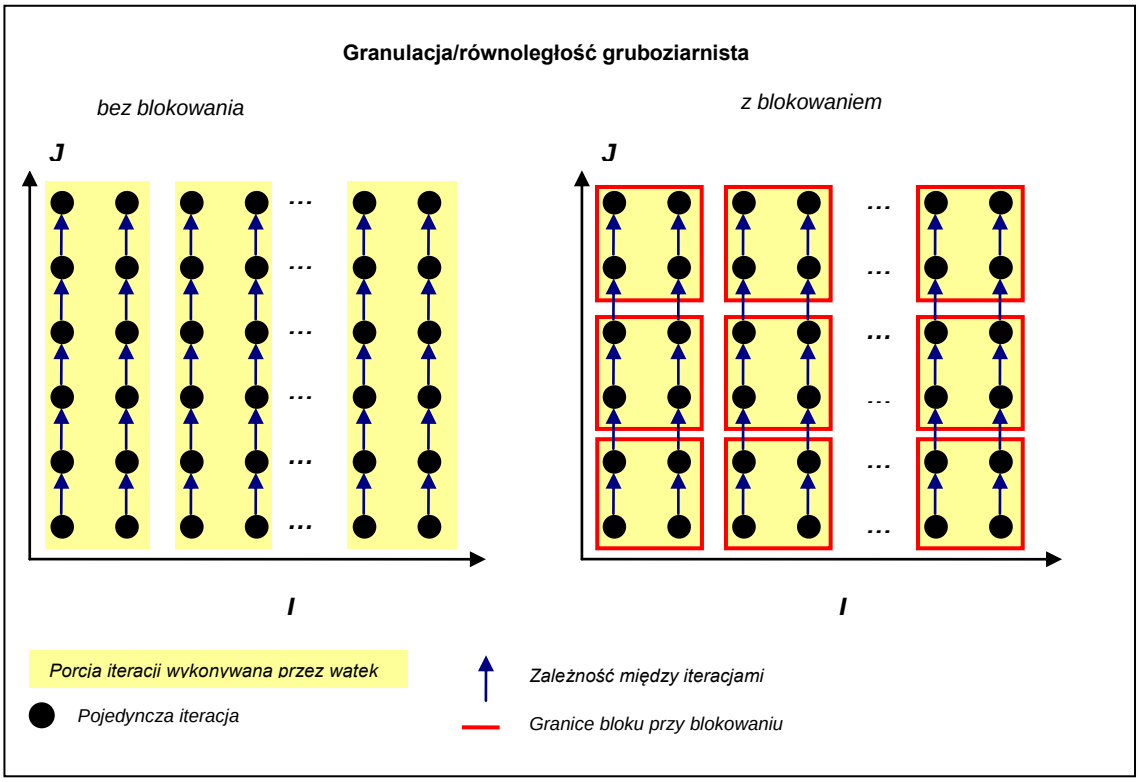
Rysunek 6 Kod źródłowy mnożenia macierzy – w wersji bez blokowania i z blokowaniem (na podstawie [56])



Rysunek 7 *Sposób dostępu do danych dla mnożenia macierzy – w wersji bez blokowania i z blokowaniem (na podstawie [56])*

Duża skuteczność blokowania jako techniki optymalizacji kompilacji w przetwarzaniu sekwencyjnym pozwala przypuszczać, że ta transformacja może być skuteczna także w przetwarzaniu równoległym, a jej zastosowanie w kompilacji iteracyjnej może przyczynić się do otrzymania kodu o wyższej jakości (tj. o krótszym czasie wykonania). Z tego względu, w modelu ogólnym, opracowywanym w ramach niniejszej pracy, uwzględniono blokowanie.

Model ogólny, którego opracowanie jest celem niniejszej pracy, ma być przeznaczony dla komputerów wieloprocesorowych z pamięcią dzieloną. Ze względu na to założenie oraz przedstawione powyżej informacje na temat ziarnistości i blokowania pętli, przewidywany zakres stosowalności modelu to: gruboziarniste, równoległe pętle programowe (bez blokowania i z blokowaniem – patrz Rysunek 8), zrównoleglone w standardzie OpenMP.
OpenMP (Open Multi-Processing) to standard, określający specyfikację dyrektyw kompilatora, bibliotek i zmiennych środowiskowych na potrzeby przetwarzania równoległego w językach: Fortran, C i C++, opartego o mechanizm pamięci współdzielonej i wątki [30] [55] [83].
Wybór standardu OpenMP wynika z powszechnego stosowania tego standardu [15] [23] [29] [46].



Rysunek 8 *Granulacja/równoległość gruboziarnista bez i z blokowaniem*

2.4. Podsumowanie

W rozdziale 2 dokonano przeglądu aktualnego stanu wiedzy na temat przetwarzania równoległego, koncentrując się na miernikach oceny wydajności aplikacji równoległych oraz podejściach do tworzenia aplikacji równoległych.

Na tym tle, przeanalizowano stosowane obecnie podejścia do oszacowania wydajności aplikacji równoległych i wskazano potencjalny obszar usprawnień (skrócenie czasu trwania kompilacji iteracyjnej programów zrównoleglonych, poprzez zastosowanie oszacowań w celu wstępnej selekcji postaci kodu źródłowego o najlepszych rokowaniach odnośnie czasu wykonania).

3. OPRACOWANIE OGÓLNEGO MODELU STATYSTYCZNEGO ORAZ MODELI SZCZEGÓLNYCH

Model to uproszczony opis wybranego aspektu rzeczywistości. Oznacza to, że model jest skoncentrowany na tych właściwościach rozpatrywanego obiektu/zjawiska, które są istotne z punktu widzenia prowadzonej analizy, pomijając częściowo lub całkowicie pozostałe właściwości rozpatrywanego obiektu/zjawiska.

W modelach matematycznych (do których zalicza się również modele statystyczne) przedmiotowy opis stanowią funkcje, które wiążą ze sobą zmienne, reprezentujące merytorycznie istotne właściwości rozpatrywanego obiektu/zjawiska.

W kontekście niniejszej pracy, oznacza to, że aby opracować model ogólny, będący celem pracy, oraz przykładowe modele szczególne uszczegółowiające go dla konkretnych sytuacji, należy określić:

- założenia odnośnie granic modelowanego aspektu rzeczywistości, czyli przewidywany zakres stosowalności modelu ogólnego oraz modeli szczególnych,
- jakie właściwości modelowanego aspektu rzeczywistości są merytorycznie istotne oraz jaka jest ich reprezentacja matematyczna w postaci zmiennych zależnych i niezależnych modelu ogólnego,
- postać funkcji wiążącej ze sobą zmienne modelu ogólnego.

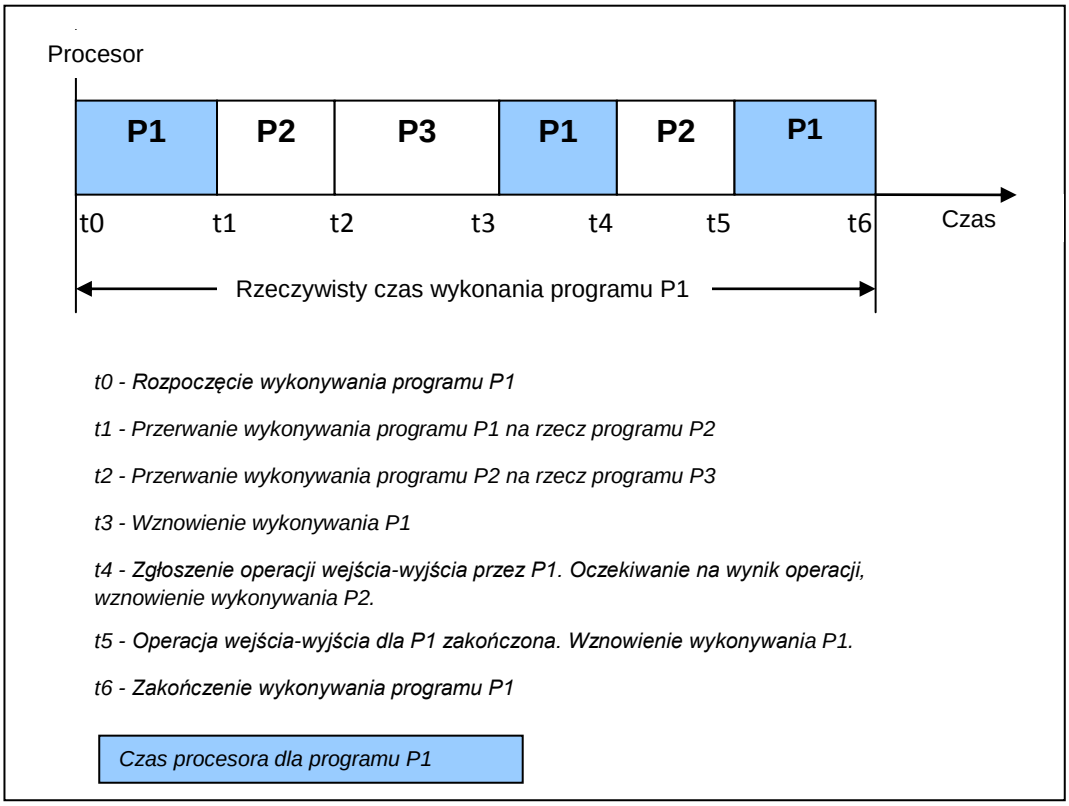
Przewidywany zakres stosowalności modelu ogólnego, opracowywanego w ramach niniejszej pracy, został przedstawiony w rozdziale 2.3.
Przedmiotem rozważań przedstawionych w niniejszym rozdziale jest wybór zmiennych, rodzaju i postaci modelu ogólnego oraz uzasadnienie dokonanego wyboru, jak również: wyznaczenie wartości parametrów modeli szczególnych, zakres stosowalności modeli szczególnych, ocena jakości modelu ogólnego i modeli szczególnych oraz stosowanie modelu ogólnego i modeli szczególnych w praktyce.
Wybrane problemy związane z opracowaniem modelu ogólnego i przykładowych modeli szczególnych zostały przedstawione w pracach: [95], [96], [97], [98], [99], [100], [101], w których przygotowaniu autorka niniejszej dysertacji uczestniczyła jako autor lub współautor.

3.1. Zmienne i postać modelu ogólnego

Przewidywane zastosowanie modelu ogólnego to szacowanie czasu wykonania gruboziarnistych, równoległych pętli programowych, zrównoleglonych w standardzie OpenMP C/C++. Dlatego też, zmienna zależna modelu ogólnego powinna odzwierciedlać ten czas.
Problemem, który się tu pojawia, jest: którą miarę (rzeczywisty czas wykonania programu czy czas procesora, spędzony na wykonywaniu programu) użyć, aby obiektywnie skwantyfikować w modelu czas wykonania programu.

Rzeczywisty czas wykonania programu (ang. *elapsed real time, real time, wall clock time*) to czas mierzony od momentu uruchomienia programu do momentu jego zakończenia,

wyrażony w jednostkach czasu rzeczywistego (np. sekundy lub mikrosekundy). Rzeczywisty czas wykonania programu obejmuje: czas przetwarzania instrukcji wykonywanych w programie oraz czas oczekiwania związany z wszelkimi zdarzeniami, które mogą mieć miejsce podczas wykonywania programu, jak np. oczekiwanie na zakończenie operacji wejścia-wyjścia (I/O), obsługę przerwania zgłoszonego przez inny program itd. [27].
Czas procesora (ang. *CPU time*) to czas, który obejmuje wyłącznie przetwarzanie instrukcji wykonywanych w programie. W środowisku wieloprocesorowym, czas ten jest mierzony liczbą taktów zegara procesora, spędzonych na wykonywaniu programu przez wszystkie wątki programu (ang. *total CPU time*) [27].
Na Rysunku 9 przedstawiono różnicę pomiędzy rzeczywistym czasem wykonania programu a czasem procesora, spędzonym na wykonywaniu programu.



Rysunek 9 Rzeczywisty czas wykonania programu a czas procesora (na podstawie [27])

Rzeczywisty czas wykonania programu jest całkowicie zależny od stanu chwilowego środowiska, w którym został uruchomiony program. Z uwagi na to, że nie można z góry przewidzieć stanu, w jakim znajdzie się system operacyjny (a w konsekwencji, środowisko uruchomienia programu) w danym momencie, rzeczywisty czas wykonania programu nie może być wykorzystany w modelu ogólnym jako miara czasu wykonania programu.
Czas procesora, spędzony na wykonywaniu programu, jest zależny jedynie od budowy procesora i stałych w czasie parametrów środowiska sprzętowego (takich jak: rozmiar pamięci podręcznej procesora, rozmiar pamięci RAM, itd.); w związku z tym, może być wykorzystany w modelu ogólnym jako miara czasu wykonania programu.

Dlatego też, w modelu ogólnym za zmienną zależną przyjęto czas procesora.

Przewidywany zakres stosowalności modelu ogólnego to gruboziarniste, równoległe pętle programowe, zrównoleglone w standardzie OpenMP C/C++. Aby modele szczególne, utworzone w oparciu o model ogólny, były zadawalająco dokładne, w modelu ogólnym należy uwzględnić, poprzez zmienne niezależne, właściwości specyficzne dla powyższego zakresu stosowalności modelu ogólnego i oddające naturę problemu opisywanego przez model, a zarazem możliwe do wyrażenia w sposób ilościowy i istotnie wpływające na wielkość zmiennej zależnej modelu – tutaj, czas wykonania zrównoleglonego programu (pętli programowej). Potencjalnymi właściwościami, które należy uwzględnić w modelu ogólnym, są:

a) struktura programu równoległego i rodzaj równoległości, reprezentowanej w programie,
b) specyfika problemu rozwiązywanego w sposób równoległy,
c) parametry środowiska sprzętowego, w którym będzie wykonywany zrównoleglony program.

Poszczególne właściwości zostały skwantyfikowane w następujący sposób:

a) Struktura programu równoległego i rodzaj równoległości, reprezentowanej w programie

W standardzie OpenMP C/C++, przyjętym sposobem zrównoleglenia programu jest wielowątkowość [83]. Wielowątkowość to wykonywanie programu przez wiele wątków, czyli instancji wykonawczych współdziałających ze sobą w ramach jednego procesu[3]. Każdy z powołanych wątków otrzymuje do wykonania pewną część zadania, które jest realizowane w programie. Czas wykonania zrównoleglonego programu jest zależny od liczby powołanych wątków OpenMP – dlatego też, potencjalną zmienną niezależną modelu ogólnego jest zmienna niezależna (*X4*) określająca liczbę wątków OpenMP, wykonujących program.

W przypadku, gdy zadanie do wykonania jest realizowane w pętli programowej, każdy z powołanych wątków OpenMP otrzymuje do wykonania pewną liczbę iteracji tej pętli. Poszczególne wątki mogą otrzymać jednakową lub niejednakową liczbę iteracji do wykonania – jest to zależne od przyjętego sposobu przydziału iteracji pętli do wątków (równomierne bądź nierównomierne obciążenie poszczególnych wątków iteracjami; przydział iteracji do każdego z wątków w jednej lub wielu transzach).
W związku z tym, czas wykonania całego zadania (pętli programowej) jest zdeterminowany przez czas wykonania tego wątku, który otrzymał do wykonania najwięcej iteracji, a w szczególności – przez wielkość największej porcji iteracji, jaka została przydzielona do wykonania do wspomnianego wątku. Dlatego też, potencjalną zmienną niezależną modelu ogólnego jest zmienna niezależna (*X3*) określająca maksymalną, dla danego sposobu przydziału iteracji do wątków OpenMP, wielkość pojedynczej porcji iteracji przydzielonej do wykonania przez wątek.

[3] Przez "proces" rozumie się tu program komputerowy, będący w trakcie wykonywania.

b) Specyfika problemu rozwiązywanego w sposób równoległy

W aspekcie najbardziej niskopoziomowym, specyfika i różnorodność problemów rozwiązywanych w programach uwidacznia się w liczbie i typie operacji arytmetycznych, jakie ma wykonać procesor. Prostym, ale skutecznym sposobem wyrażenia powyższej obserwacji w sposób ilościowy jest przypisanie różnych wag różnym operacjom arytmetycznym; wagi należy dobrać na podstawie analizy czasów wykonania instrukcji danego procesora. Dzięki takiemu podejściu, uzyskuje się porównywalność operacji arytmetycznych różnych typów (np. dodawanie i mnożenie). Dlatego też, potencjalną zmienną niezależną modelu ogólnego jest zmienna niezależna (*X2*) wskazująca, jaka jest łączna ważona liczba operacji przypadająca na pojedynczy wątek programu.

c) Parametry środowiska sprzętowego, w którym będzie wykonywany zrównoleglony program

Z uwagi na występowanie dysproporcji pomiędzy szybkością procesorów a czasem dostępu do podsystemu pamięci, to właśnie pamięć – a zwłaszcza szybko dostępna pamięć podręczna procesora – jest jednym z tych elementów środowiska sprzętowego, które determinują długość czasu wykonania programu [77].

W idealnej sytuacji, wszystkie dane potrzebne procesorowi podczas wykonywania programu powinny być dostępne w pamięci podręcznej procesora w momencie, gdy zostało na nie zgłoszone zapotrzebowanie, a nie być dopiero wówczas pobierane z pamięci głównej do pamięci podręcznej procesora. Im więcej danych przetwarzanych w programie jest dostępnych w pamięci podręcznej procesora w momencie zgłoszenia na nie zapotrzebowania, tym krótszy jest czas wykonania programu.
Z drugiej strony, pojemność pamięci podręcznej procesora i sposób jej adresowania determinują jaki odsetek danych przetwarzanych w programie jest dostępny w pamięci podręcznej procesora w momencie zgłoszenia zapotrzebowania na przedmiotowe dane.

Oznacza to, że czas wykonania programu jest zależny od:
1/ faktycznej pojemności i sposobu organizacji (asocjacyjności) pamięci podręcznej procesora w danym systemie komputerowym,
2/ minimalnej pojemności pamięci podręcznej procesora (pamięć z odwzorowaniem bezpośrednim), jaka jest niezbędna do pomieszczenia wszystkich danych przetwarzanych w programie, przy założeniu pełnego czasowego i przestrzennego ponownego użycia danych przechowywanych w pamięci podręcznej procesora[4]. Przedmiotową pojemność można oszacować przy pomocy odcisku danych [89]. W celu wyznaczenia odcisku danych dla danego programu wystarczy znać jego kod źródłowy, nie ma potrzeby wykonywania tego programu [88].

[4] Czasowe ponowne użycie danych (ang. *temporal data reuse*) występuje wówczas, gdy dane pobrane z określonej lokalizacji w pamięci są wielokrotnie wykorzystywane w programie. Przestrzenne ponowne użycie danych (ang. *spatial data reuse*) występuje wówczas, gdy w programie są wykorzystywane dane sąsiadujące, w ramach jednej linii pamięci podręcznej, z danymi pobranymi z określonej lokalizacji w pamięci [1] [88].

3/ relacji pomiędzy wielkościami wskazanymi w 1/ i 2/.

W związku z powyższym, potencjalną zmienną niezależną modelu ogólnego jest zmienna niezależna (*X1*), przedstawiająca relację pomiędzy wielkościami wskazanymi w 1/ i 2/.

Lista wszystkich potencjalnych zmiennych modelu ogólnego przedstawia się następująco:

Zmienna zależna:

Yt – szacowany czas procesora dla wykonania pętli programowej przez wszystkie wątki programu, wyrażony liczbą taktów zegara procesora *(wielkość bezwymiarowa)*.

Potencjalne zmienne niezależne:

X1 – wielkość bezwymiarowa, określająca stosunek łącznego rozmiaru pamięci podręcznej L1 i L2, przypadającej na pojedynczy wątek OpenMP, do odcisku danych dla pojedynczego wątku OpenMP; wielkość ta jest wyrażona równaniem (3):

$$X1 = \frac{\left(sL1 \times aL1 + sL2 \times aL2\right)}{D_f} \tag{3}$$

gdzie:

- *sL1* – rozmiar pamięci podręcznej L1, dostępnej dla pojedynczego wątku OpenMP *(wielkość wyrażona w bajtach),*
- *sL2* – rozmiar pamięci podręcznej L2, dostępnej dla pojedynczego wątku OpenMP *(wielkość wyrażona w bajtach),*
- *aL1* – „asocjacyjność" pamięci podręcznej L1 *(wielkość bezwymiarowa),*
- *aL2* – „asocjacyjność" pamięci podręcznej L2 *(wielkość bezwymiarowa),*
- *Df* – odcisk danych dla pojedynczego wątku OpenMP, wyznaczony wg [88] i [56] *(wielkość wyrażona w bajtach).*

X2 – łączna ważona liczba operacji przypadająca na pojedynczy wątek OpenMP *(wielkość bezwymiarowa),*

X3 – maksymalna, dla danego sposobu przydziału iteracji do wątków OpenMP, wielkość pojedynczej porcji iteracji przydzielonej do wykonania przez wątek *(wielkość bezwymiarowa),*

X4 – liczba wątków OpenMP wykonujących program *(wielkość bezwymiarowa).*

Zmienne *X1, X2, X3* oraz *X4* są wzajemnie ze sobą powiązane. Ponadto, wartości, przyjmowane przez zmienne *X1, X2* oraz *X3* są zależne od rozmiaru problemu, który jest rozwiązywany w programie (*N*).

Pomiędzy *X1, X2, X3, X4* oraz *N* występują następujące zależności:

- Wielkość odcisku danych dla pojedynczego wątku OpenMP (w konsekwencji, wartość zmiennej *X1*) jest zależna od sposobu przydziału iteracji do wątków OpenMP (czyli od wartości zmiennych *X3* oraz *X4*) oraz od rozmiaru problemu rozwiązywanego w programie (czyli od wartości *N*).

- Łączna ważona liczba operacji przypadająca na pojedynczy wątek OpenMP (wielkość wyrażona zmienną *X2*) jest zależna od sposobu przydziału iteracji do wątków OpenMP (czyli od wartości zmiennych *X3* oraz *X4*) oraz od rozmiaru problemu rozwiązywanego w programie (czyli od wartości *N*).
- Maksymalna wielkość pojedynczej porcji iteracji przydzielonej do wykonania przez wątek (wielkość wyrażona zmienną *X3*) jest zależna od liczby wątków OpenMP wykonujących program (czyli od wartości zmiennej *X4*) oraz od rozmiaru problemu rozwiązywanego w programie (czyli od wartości *N*).

Biorąc pod uwagę dużą różnorodność czynników, które powinny znaleźć odbicie w modelu ogólnym, jak również złożone, istotne i bardzo trudne do sformalizowania relacje pomiędzy nimi i środowiskiem sprzętowo-programowym, najprostszym sposobem na ujęcie wszystkich tych czynników jednym równaniem dającym możliwy do zaakceptowania błąd oszacowania jest zastosowanie modelu statystycznego, otrzymanego drogą analizy regresji (model regresyjny). W niniejszej pracy, przedmiotową analizę regresji przeprowadzono wykorzystując regresję liniową opartą o klasyczną metodę najmniejszych kwadratów, przyjmując za zmienną objaśnianą (zależną) *Yt* natomiast za potencjalne zmienne objaśniające (niezależne) zmienne: *X1, X2, X3, X4*. Wybór ostatecznych zmiennych niezależnych modelu został omówiony w dalszej części niniejszego rozdziału. Przyjęcie niewielkiej listy potencjalnych zmiennych niezależnych modelu (tylko 4 zmienne) było podyktowane dążeniem do uniknięcia nadmiernego dopasowania modelu. Nadmierne dopasowanie modelu występuje wówczas, gdy liczba zmiennych niezależnych modelu z uwzględnieniem wyrazu wolnego (*k*) jest równa lub nieznacznie mniejsza od liczby obserwacji (*n*), na podstawie których budowany jest model. W takim wypadku, uzyskany model będzie się charakteryzował bardzo wysokim dopasowaniem do danych – niemniej, jego wartość prognostyczna będzie znikoma, z uwagi na to, że jego liczba stopni swobody *df* (gdzie: $df = n - k$) będzie równa zeru lub bliska zera, co z kolei oznacza, że w takim modelu wcale lub jedynie w bardzo niewielkim stopniu uwzględniono czynnik losowy.

Zgodnie z założeniami regresji linowej, zależność występująca pomiędzy obserwowanymi wartościami zmiennej zależnej *Y* oraz odpowiadającymi im wartościami zmiennych niezależnych *X1*, *X2*, …, *Xp* jest zależnością wyrażoną równaniem (4):

$$Y_i = a_0 + a_1 X1_i + a_2 X2_i + \cdots + a_p Xp_i + \varepsilon_i = Yt_i + \varepsilon_i \tag{4}$$

gdzie:

i – numery kolejnych obserwacji ($i = 1, \ldots, n$),

$a_0, \ldots, a_p$ – parametry o nieznanych wartościach dokładnych; wartości tych parametrów są szacowane w oparciu o klasyczną metodę najmniejszych kwadratów,

$X1_i, \ldots, Xp_i$ – znane wartości zmiennych niezależnych *X1, X2, …, Xp*, odpowiadające wartości *Y* obserwowanej w i-tej obserwacji,

Y_i – wartość zmiennej zależnej *Y* obserwowana w i-tej obserwacji,

Yt_i – teoretyczna (przewidywana) wartość zmiennej zależnej *Y* dla i-tej obserwacji (wartość Yt_i leży na oszacowanej powierzchni regresji),

ε_i – błąd (reszta, składnik resztowy) dla i-tej obserwacji.

Równanie (4) jest słuszne w przypadku, gdy zależność, występująca pomiędzy empirycznymi wartościami zmiennej zależnej i zmiennych niezależnych jest zależnością liniową lub zależnością, którą można sprowadzić do zależności liniowej – tj. zależnością potęgową, wykładniczą, logarytmiczną lub hiperboliczną.

Dlatego też, dla omówionych w niniejszym rozdziale potencjalnych zmiennych niezależnych: *X1*, *X2*, *X3*, *X4* oraz zmiennej zależnej *Yt* opracowywany w ramach pracy model ogólny, będący modelem regresyjnym opartym o regresję liniową wykorzystującą klasyczną metodę najmniejszych kwadratów, może przyjąć jedną z następujących postaci:

- postać liniową, wyrażoną równaniem (5):

$$Yt = a1 \times X1 + a2 \times X2 + a3 \times X3 + a4 \times X4 \tag{5}$$

- postać potęgową, wyrażoną równaniem (6):

$$Yt = X1^{a1} \times X2^{a2} \times X3^{a3} \times X4^{a4} \tag{6}$$

  którą można sprowadzić do równoważnej postaci liniowej, wyrażonej równaniem (7):

$$Yt^* = a1 \times X1^* + a2 \times X2^* + a3 \times X3^* + a4 \times X4^* \tag{7}$$

  gdzie:

  $Yt^* = \log Yt$; $X1^* = \log X1$; $X2^* = \log X2$; $X3^* = \log X3$; $X4^* = \log X4$

- postać wykładniczą, wyrażoną równaniem (8):

$$Yt = a1^{X1} \times a2^{X2} \times a3^{X3} \times a4^{X4} \tag{8}$$

  którą można sprowadzić do równoważnej postaci linowej, wyrażonej równaniem (9):

$$Yt^* = a1^* \times X1 + a2^* \times X2 + a3^* \times X3 + a4^* \times X4 \tag{9}$$

  gdzie:

  $Yt^* = \log Yt$; $a1^* = \log a1$; $a2^* = \log a2$; $a3^* = \log a3$; $a4^* = \log a4$

- postać logarytmiczną, wyrażoną równaniem (10):

$$Yt = a1 \times \log X1 + a2 \times \log X2 + a3 \times \log X3 + a4 \times \log X4 \tag{10}$$

  którą można sprowadzić do równoważnej postaci linowej, wyrażonej równaniem (11):

$$Yt = a1 \times X1^* + a2 \times X2^* + a3 \times X3^* + a4 \times X4^* \tag{11}$$

  gdzie:

$X1^* = \log X1$; $X2^* = \log X2$; $X3^* = \log X3$; $X4^* = \log X4$

- postać hiperboliczną, wyrażoną równaniem (12):

$$Yt = a1 \times \frac{1}{X1} + a2 \times \frac{1}{X2} + a3 \times \frac{1}{X3} + a4 \times \frac{1}{X4} \tag{12}$$

którą można sprowadzić do równoważnej postaci linowej, wyrażonej równaniem (13):

$$Yt = a1 \times X1^* + a2 \times X2^* + a3 \times X3^* + a4 \times X4^* \tag{13}$$

gdzie:

$$X1^* = \frac{1}{X1} \; ; \quad X2^* = \frac{1}{X2} \; ; \quad X3^* = \frac{1}{X3} \; ; \quad X4^* = \frac{1}{X4}$$

Uwaga : W równaniach (5) ÷ (13) nie uwzględniono a0 z uwagi na to, że dla modelowanego zjawiska parametr a0 nie ma sensu praktycznego.

Na podstawie przedstawionego w niniejszym rozdziale opisu zmiennych *X1*, *X2*, *X3*, *X4*, *Yt* i ich wzajemnych powiązań nasuwało się przypuszczenie, że zależność występująca pomiędzy zmiennymi: *X1*, *X2*, *X3*, *X4* oraz *Yt* ma charakter potęgowy.

Przypuszczenie to zweryfikowano w oparciu o dane empiryczne, zebrane dla dwóch pętli: nonInterf (dane empiryczne zebrane dla tej pętli podano w Tabeli 14 – patrz strona 87) oraz matmul (dane empiryczne zebrane dla tej pętli podano w oraz Tabeli 15 – patrz strona 88) – przeprowadzając analizę regresji liniowej dla każdej z wymienionych powyżej postaci modelu (patrz równania (5) ÷ (13)) i dodatkowo zakładając, że w modelu:

1/ występują zmienne: *Yt, X1, X2, X3, X4,*
2/ występują wyłącznie zmienne *Yt, X1,*
3/ występują wyłącznie zmienne *Yt, X2,*
4/ występują wyłącznie zmienne *Yt, X3,*
5/ występują wyłącznie zmienne *Yt, X4.*

Celem analizy regresji liniowej dla przypadków 2/ ÷ 5/ było zbadanie charakteru zależności występujących pomiędzy zmienną zależną *Yt* a każdą z potencjalnych zmiennych niezależnych.
Wartości współczynników determinacji uzyskane dla przypadków 1/ ÷ 5/ przedstawiono w Tabeli 1 oraz w Tabeli 2.

Tabela 1 Wartości współczynników determinacji dla różnych możliwych postaci modelu ogólnego – dla pętli nonInterf

Przypadek	**1/**	**2/**	**3/**	**4/**	**5/**
Postać modelu	$R^2_{Yt.X1,X2,X3,X4}$	$R^2_{Yt.X1}$	$R^2_{Yt.X2}$	$R^2_{Yt.X3}$	$R^2_{Yt.X4}$
liniowy	0,9738484	0,0602237	0,9239709	0,6125842	0,6390095
potęgowy	0,9999580	0,8968516	0,9957804	0,9653380	0,9203893
wykładniczy	0,9845407	0,3399016	0,7284349	0,8848293	0,9194891
logarytmiczny	0,9557310	0,4977595	0,7366662	0,6611192	0,6387514
hiperboliczny	0,9458243	0,9239709	0,0602237	0,5872795	0,5997874

Tabela 2 Wartości współczynników determinacji dla różnych możliwych postaci modelu ogólnego – dla pętli matmul

Przypadek	**1/**	**2/**	**3/**	**4/**	**5/**
Postać modelu	$R^2_{Yt.X1,X2,X3,X4}$	$R^2_{Yt.X1}$	$R^2_{Yt.X2}$	$R^2_{Yt.X3}$	$R^2_{Yt.X4}$
liniowy	0,9506216	0,0002301	0,9286567	0,3616036	0,4771490
potęgowy	0,9999514	0,6540767	0,9982205	0,9119271	0,9183616
wykładniczy	0,9645971	0,1066810	0,5703056	0,4310208	0,9170599
logarytmiczny	0,8230448	0,8095558	0,5858303	0,5074892	0,4774223
hiperboliczny	0,8098927	0,7669016	0,0014395	0,3219836	0,4602693

Dla obu pętli, najwyższą wartość R^2 dla przypadku 1/ uzyskano stosując model potęgowy wyrażony równaniem (6). Ponadto, dla obu pętli, stosując ww. model potęgowy uzyskano bardzo dobre wyniki dopasowania także dla wszystkich pozostałych przypadków, co świadczy o potęgowym charakterze zależności pomiędzy zmienną zależną a każdą z potencjalnych zmiennych niezależnych.

Dla wspomnianego modelu potęgowego i wszystkich możliwych kombinacji potencjalnych zmiennych niezależnych (tj. zmiennych: *X1, X2, X3, X4*) obliczono także wartość skorygowanego współczynnika determinacji, wykorzystując w tym celu równanie (14):

$$\bar{R}^2 = R^2 - \left(1 - R^2\right)\frac{p}{n - p - 1} \tag{14}$$

gdzie:

$\bar{R}^2$ – skorygowany współczynnik determinacji,
R^2 – współczynnik determinacji,
p – liczba zmiennych niezależnych,
n – wielkość (liczba elementów) próby.

Na podstawie wartości skorygowanego współczynnika determinacji można stwierdzić czy wprowadzenie do modelu kolejnej zmiennej niezależnej (objaśniającej) faktycznie zwiększa

stopień w jakim model wyjaśnia kształtowanie się badanego zjawiska, czy może tylko prowadzi do zwiększenia wartości współczynnika determinacji nie poprawiając w żaden sposób wartości predykcyjnych modelu.

Jeżeli wprowadzenie do modelu konkretnej, kolejnej zmiennej niezależnej faktycznie zwiększa stopień, w jakim model wyjaśnia kształtowanie się badanego zjawiska, wówczas wartość skorygowanego współczynnika determinacji jest wyższa niż w przypadku, gdy przedmiotowej zmiennej niezależnej nie uwzględniono w modelu. Dlatego też, w modelu regresyjnym należy uwzględnić jako zmienne niezależne ten podzbiór zbioru potencjalnych zmiennych niezależnych, dla którego uzyskano najwyższą wartość skorygowanego współczynnika determinacji.

Wartości skorygowanych współczynników determinacji dla modelu potęgowego wyrażonego równaniem (6) (patrz strona 24) i wszystkich możliwych kombinacji potencjalnych zmiennych niezależnych (tj. zmiennych: *X1*, *X2*, *X3*, *X4*) przedstawiono w Tabeli 3 (dla pętli nonInterf) oraz w Tabeli 4 (dla pętli matmul).

Tabela 3 Wartości skorygowanych współczynników determinacji dla różnych możliwych kombinacji potencjalnych zmiennych niezależnych – dla pętli nonInterf i modelu potęgowego wyrażonego równaniem (6)

Zmienne modelu				R^2	Skorygowane R^2
x1				0,8968516	0,8919398
	x2			0,9957804	0,9955795
		x3		0,9653380	0,9636874
			x4	0,9203893	0,9165983
x1	x2			0,9982909	0,9981200
x1		x3		0,9663456	0,9629802
x1			x4	0,9300614	0,9230676
	x2	x3		0,9966362	0,9962998
	x2		x4	0,9959525	0,9955477
		x3	x4	0,9702187	0,9672406
x1	x2	x3		0,9983220	0,9980570
x1	x2		x4	0,9999401	0,9999306
x1		x3	x4	0,9747436	0,9707558
	x2	x3	x4	0,9970075	0,9965350
x1	x2	x3	x4	0,9999580	0,9999486

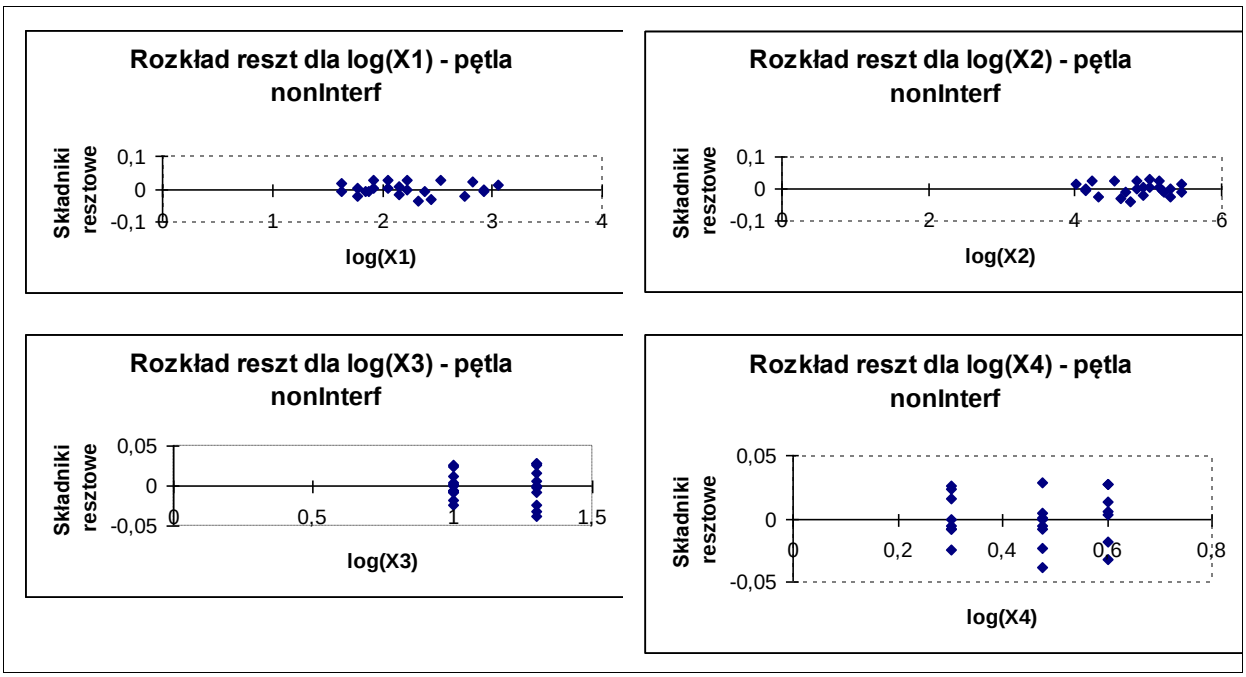
Tabela 4 Wartości skorygowanych współczynników determinacji dla różnych możliwych kombinacji potencjalnych zmiennych niezależnych – dla pętli matmul i modelu potęgowego wyrażonego równaniem (6)

Zmienne modelu				R^2	Skorygowane R^2
x1				0,6540767	0,6458404
	x2			0,9982205	0,9981782
		x3		0,9119271	0,9098301
			x4	0,9183616	0,9164178
x1	x2			0,9994628	0,9994366

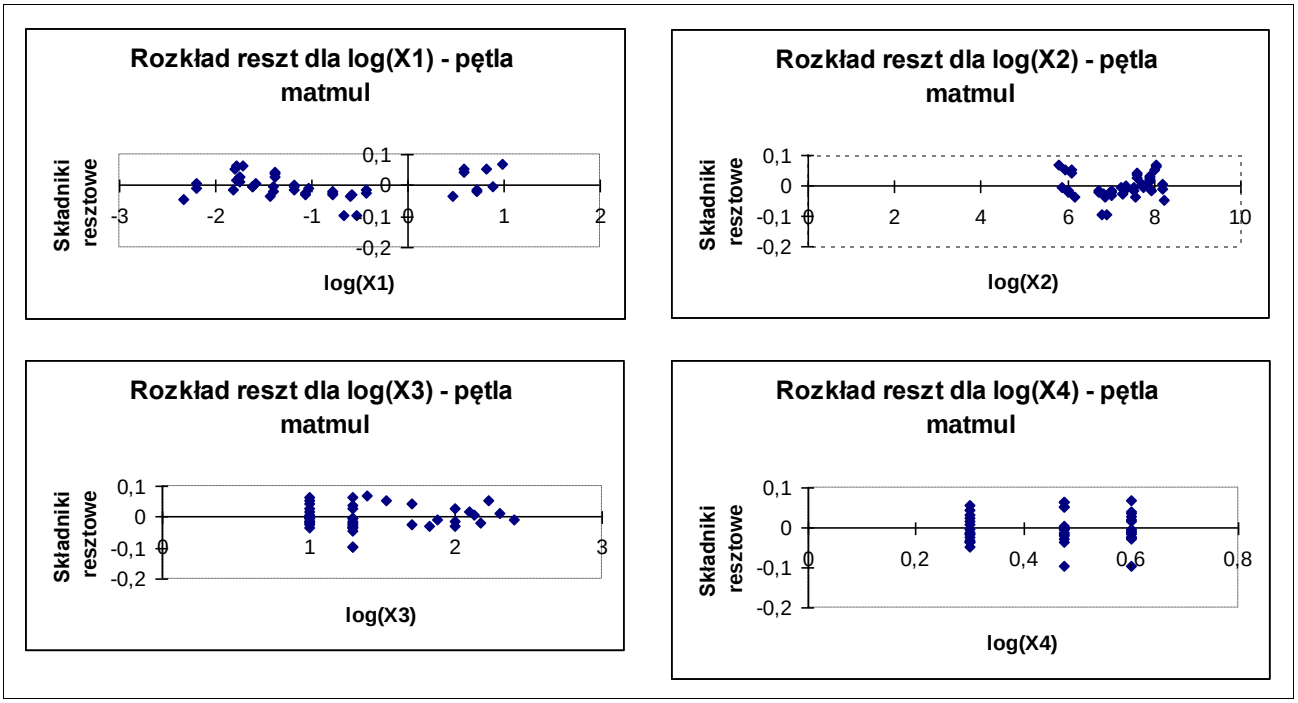
Zmienne modelu				R^2	Skorygowane R^2
x1		x3		0,9383487	0,9353413
x1			x4	0,9655228	0,9638410
	x2	x3		0,9982451	0,9981595
	x2		x4	0,9982402	0,9981543
		x3	x4	0,9549970	0,9528018
X1	x2	x3		0,9994630	0,9994227
x1	x2		x4	0,9999501	0,9999463
x1		x3	x4	0,9796433	0,9781166
	x2	x3	x4	0,9982646	0,9981345
x1	x2	x3	x4	0,9999514	0,9999464

Zarówno dla pętli nonInterf, jak i dla pętli mamtul, najwyższą wartość skorygowanego współczynnika determinacji uzyskano uwzględniając w modelu potęgowym wyrażonym równaniem (6) wszystkie potencjalne zmienne niezależne, tj. zmienne: *X1, X2, X3, X4*. Należy także zauważyć, że w obu przypadkach, tj. dla pętli nonInterf i matmul, wprowadzanie do modelu kolejnych zmiennych ze zbioru *X1, X2, X3, X4* skutkowało zwiększeniem wartości skorygowanego współczynnika determinacji – co świadczy o tym, że każda ze zmiennych *X1, X2, X3, X4* ma istotne znaczenie w wyjaśnieniu kształtowania się wartości zmiennej zależnej *Yt*.

Zarówno dla pętli nonInterf, jak i dla pętli mamtul, analiza wykresów reszt, przedstawiających przyporządkowanie reszt (uzyskanych dla modelu potęgowego wyrażonego równaniem (6), przedstawionego na stronie 24) wartościom zmiennych niezależnych *X1, X2, X3, X4* potwierdza słuszność decyzji o uwzględnieniu wszystkich tych zmiennych w opracowywanym modelu. Na wykresach reszt (patrz Rysunek 10 oraz Rysunek 11) reszty są rozłożone losowo, nie wykazują żadnej tendencji tj. wzrostu lub spadku ze wzrostem wartości którejś ze zmiennych niezależnych (co świadczyłoby o nieujęciu w modelu czynnika o charakterze systematycznym); wariancja reszt jest stała. Taka sytuacja (homoskedastyczność) jest zgodna z ogólnymi założeniami analizy regresji.



Rysunek 10 *Wykresy reszt dla zmiennych niezależnych uliniowionego modelu wyrażonego równaniem (6) – dla pętli nonInterf*



Rysunek 11 *Wykresy reszt dla zmiennych niezależnych uliniowionego modelu wyrażonego równaniem (6) – dla pętli matmul*

Dla obu przypadków, tj. dla pętli nonInterf i matmul, sprawdzono także istotność równania regresji otrzymanego dla modelu potęgowego wyrażonego równaniem (6) (patrz strona 24) i uwzględniającego zmienne: *X1, X2, X3, X4*. W tym celu, wykorzystano zestaw następujących hipotez:

H_0: $\beta_1 = 0 \wedge \beta_2 = 0 \wedge \ldots \wedge \beta_k = 0$

H_1: $\beta_1 \neq 0 \vee \beta_2 \neq 0 \vee \ldots \vee \beta_k \neq 0$

Aby dokonać weryfikacji hipotez, obliczono statystykę testową *F*, która ma rozkład F-Snedecora z *k* oraz (*N*-*k*) stopniami swobody:

$$F(k, N-k) = \frac{R^2 / k}{(1-R^2)/(N-k)} \tag{15}$$

gdzie:

R^2 – współczynnik determinacji,

k – liczba zmiennych niezależnych,

N – wielkość (liczba elementów) próby

oraz obliczono graniczny poziom istotności dla statystyki *F* wyznaczonej wg równania (15). Graniczny poziom istotności (*p*) dla danej wartości statystyki testowej jest miarą prawdopodobieństwa popełnienia błędu pierwszego rodzaju, tzn. odrzucenia hipotezy zerowej, która jest w rzeczywistości prawdziwa. W przypadku przedmiotowej statystyki *F* oznacza to, że:

- przyjmując poziom istotności $\alpha \geq p$ należy odrzucić hipotezę zerową mówiącą, że rozpatrywane równanie regresji jako całość jest statystycznie nieistotne, należy natomiast przyjąć hipotezę alternatywną mówiącą, że równanie regresji jako całość jest statystycznie istotne,
- przyjmując poziom istotności $\alpha < p$ brak jest podstaw do odrzucenia hipotezy zerowej mówiącej, że rozpatrywane równanie regresji jako całość jest statystycznie nieistotne.

Wyniki weryfikacji hipotezy o istotności równania regresji (model potęgowy wyrażony równaniem (6), ze zmiennymi niezależnymi: *X1, X2, X3, X4*) dla pętli nonInterf i matmul przedstawiono w Tabeli 5.

Tabela 5 Wyniki weryfikacji hipotezy o istotności równania regresji (model potęgowy wyrażony równaniem (6), ze zmiennymi niezależnymi: X1, X2, X3, X4) – dla pętli nonInterf i matmul

Pętla	Wartość statystyki *F*	Graniczny poziom istotności *p*
nonInterf	112 988,28	$2{,}52 \times 10^{-39}$
matmul	205 702,56	$9{,}74 \times 10^{-84}$

Dla obu przypadków, tj. dla pętli nonInterf i matmul, uzyskano bardzo małe wartości granicznego poziomu istotności (*p*). Oznacza to, że można przyjąć, że w obu przypadkach rozpatrywane równania regresji jako całość są statystycznie istotne.

Dla obu przypadków, tj. dla pętli nonInterf i matmul, sprawdzono także czy reszty uzyskane dla modelu potęgowego wyrażonego równaniem (6) (patrz strona 24) mają rozkład zbliżony do rozkładu normalnego N(μ, σ). Jeżeli rozkład reszt jest normalny lub zbliżony do normalnego oznacza to, że prognozy uzyskane przy pomocy modelu są „stabilne" – tzn. spełniają regułę 3 sigm (czyli, niemal wszystkie prognozy odchylają się od wartości faktycznie obserwowanych o nie więcej niż 3σ) i tylko znikomy odsetek prognoz uzyskanych wg modelu (poniżej 1 % wszystkich prognoz) odchyla się od wartości faktycznie obserwowanych o więcej niż 3σ.

W celu sprawdzenia czy reszty uzyskane dla modelu potęgowego wyrażonego równaniem (6) mają rozkład zbliżony do normalnego, wykorzystano zestaw następujących hipotez:

H_0: rozkład reszt jest rozkładem normalnym
H_1: rozkład reszt nie jest rozkładem normalnym

Aby dokonać weryfikacji hipotez, wykorzystano test Kołmogorowa-Smirnowa.

Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu potęgowego wyrażonego równaniem (6), ze zmiennymi niezależnymi: *X1, X2, X3, X4*) dla pętli nonInterf i matmul przedstawiono w Tabeli 6.

Tabela 6 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu potęgowego wyrażonego równaniem (6), ze zmiennymi niezależnymi: X1, X2, X3, X4) – dla pętli nonInterf i matmul

Pętla	**Wartość statystyki testowej Kołmogorowa-Smirnowa**	**Graniczny poziom istotności *p***
nonInterf	0,1054	0,9372
matmul	0,1074	0,6507

Dla obu przypadków, tj. dla pętli nonInterf i matmul, uzyskano bardzo wysokie wartości granicznego poziomu istotności (*p*). Oznacza to, że przy uzyskanych wartościach statystyk testowych przyjęcie hipotezy alternatywnej (H_1) zamiast hipotezy zerowej (H_0) implikuje bardzo wysokie prawdopodobieństwo popełnienia błędu pierwszego rodzaju, tzn. odrzucenia hipotezy zerowej, która jest w rzeczywistości prawdziwa. Z tego względu, można przyjąć, że w obu przypadkach brak jest podstaw do odrzucenia hipotezy zerowej – a co za tym idzie, w obu przypadkach analizowany rozkład reszt jest rozkładem normalnym.

Na podstawie przedstawionych powyżej wyników analizy regresji, na potrzeby problemu rozpatrywanego w niniejszej pracy przyjęto, jako model ogólny, regresyjny model

statystyczny ze zmiennymi: *Yt, X1, X2, X3, X4* oraz parametrami *a1, a2, a3, a4*, wyrażony w postaci funkcji potęgowej. Model ten jest postaci:

$$Yt = X1^{a1} \times X2^{a2} \times X3^{a3} \times X4^{a4} \tag{16}$$

gdzie:

a1, a2, a3, a4 – parametry, których wartości wyznaczono na podstawie analizy regresji, przeprowadzonej dla danych empirycznych zebranych w docelowym środowisku sprzętowo-programowym dla specjalnie przygotowanej próbki.

Wyznaczanie wartości zmiennej *X1* zostało omówione w rozdziale 3.1.2.1. Wyznaczanie odcisku danych, którego znajomość jest niezbędna do wyznaczenia wartości zmiennej *X1*, zostało omówione w rozdziale 3.1.1.
Wyznaczanie wartości zmiennej *X2* zostało omówione w rozdziale 3.1.2.2.
Wyznaczanie wartości zmiennej *X3* zostało omówione w rozdziale 3.1.2.3.
Wyznaczanie wartości zmiennej *X4* zostało omówione w rozdziale 3.1.2.4.

W dalszej części pracy, model wyrażony równaniem (16) będzie określany jako model (16) lub model ogólny (16).

3.1.1 Wyznaczanie odcisku danych

Wyznaczenie odcisku danych (*Df* wg równania (3) – patrz strona 22) dla pojedynczego wątku OpenMP wymaga dobrej znajomości specyfiki działania pamięci podręcznej procesora oraz jej miejsca w hierarchii pamięci systemu komputerowego. Oba te zagadnienia mają kluczowe znaczenie dla problematyki niniejszej pracy i jako takie, wymagają szerszego omówienia.

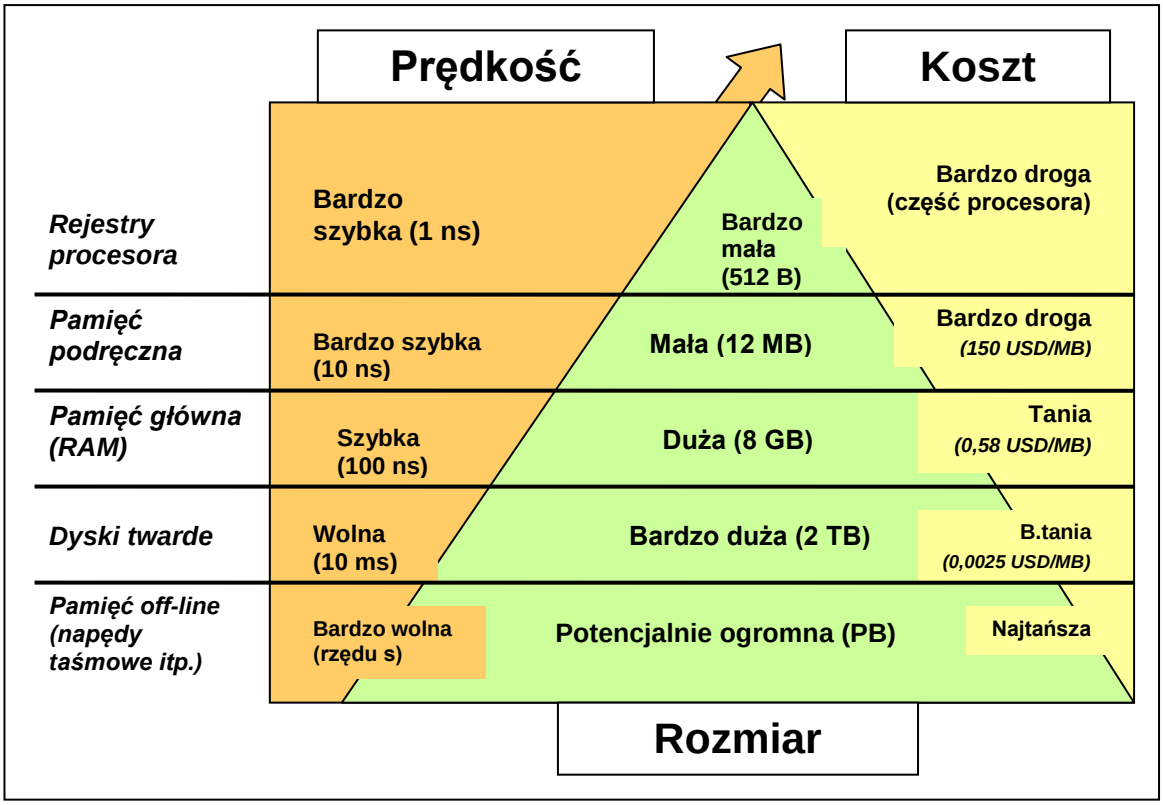
Wprowadzenie hierarchii pamięci w systemach komputerowych ma na celu zniwelowanie dysproporcji pomiędzy szybkością, z jaką procesory są w stanie przetwarzać rozkazy operacji arytmetycznych i logicznych, a czasem dostępu do podsystemu pamięci – pobranie danych ze wskazanych lokalizacji w pamięci RAM trwa kilka lub nawet kilkanaście razy dłużej, niż wykonanie elementarnych operacji arytmetycznych lub logicznych na tych danych. W rezultacie, pamięć stanowi wąskie gardło w, realizowanym na poziomie całego systemu komputerowego, przetwarzaniu danych [4] [10] [60] [61] [86] [91] [92].

Ideą hierarchicznej organizacji pamięci jest podział całej pamięci, dostępnej w systemie komputerowym, na poziomy różniące się położeniem (tj. odległością fizyczną) w stosunku do procesora, a w konsekwencji, czasem dostępu [41] [60] [77] [85].
Poszczególne poziomy pamięci, uszeregowane rosnąco ze względu na odległość od procesora i czas dostępu, to:

- rejestry procesora,
- pamięć podręczna procesora (ang. *CPU cache*) pierwszego poziomu (L1),
- pamięć podręczna procesora (ang. CPU *cache*) wyższych poziomów (L2, L3; poziom L3 jest używany głównie w zastosowaniach specjalistycznych),
- pamięć główna (RAM),
- pamięć zewnętrzna (np. dyski magnetyczne),
- pamięć offline (np. napędy taśmowe).

W dalszej części pracy, pamięć podręczna procesora będzie nazywana pamięcią podręczną.

Na Rysunku 12 przedstawiono poglądowo hierarchię pamięci w aspektach: czasu dostępu do pamięci, rozmiaru pamięci oraz kosztu wytworzenia pamięci w przeliczeniu na bit jej pojemności.

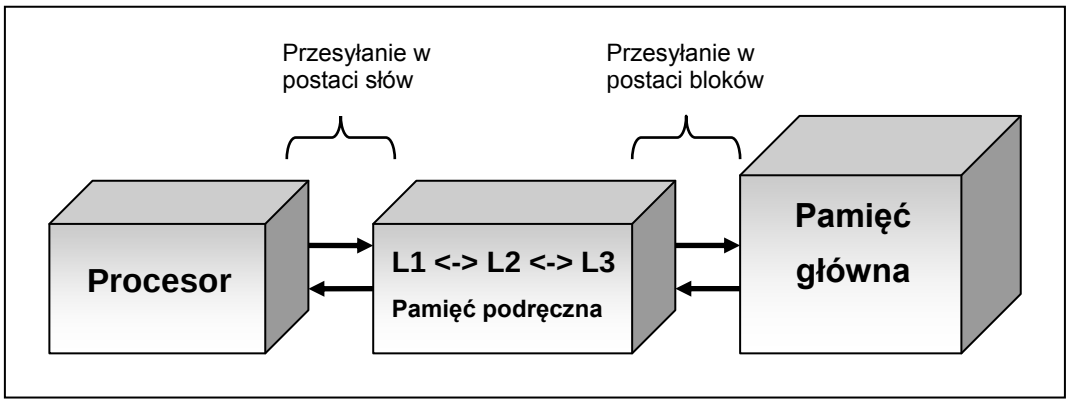


Rysunek 12 Hierarchia pamięci [78]

Wykonanie programu, zapisanego w postaci kodu maszynowego, polega na umieszczeniu kodu maszynowego programu w pamięci operacyjnej komputera (czyli pamięci, do której procesor ma dostęp bezpośrednio, a nie za pośrednictwem kanałów wejścia-wyjścia) i wskazaniu procesorowi adresu pierwszej instrukcji programu. Po wykonaniu pierwszej instrukcji programu, procesor będzie wykonywał kolejne instrukcje programu, aż do poprawnego (tj. zgodnego z intencjami twórcy/użytkownika programu) lub błędnego (spowodowanego np. awarią sprzętu) zakończenia programu.

W zależności od przeznaczenia, program może realizować różne operacje na danych.
Podczas wykonywania programu w systemie komputerowym, optymalna sytuacja to taka, gdy dane potrzebne procesorowi w związku z wykonywaniem programu znajdują się w rejestrach procesora (czyli na najwyższym poziomie hierarchii pamięci, położonym najbliżej procesora) – wówczas dane te są od razu dostępne dla procesora. Jednakże, rejestry procesora mają bardzo niewielką pojemność. Wynika to stąd, że im bliżej procesora jest położona pamięć, tym wyższy jest koszt ekonomiczny jej wytworzenia w przeliczeniu na bit pojemności – w rezultacie, im bliżej procesora jest położona pamięć, tym mniejsza jej pojemność. W konsekwencji, rejestry procesora mogą pomieścić bardzo niewielką ilość danych.
Dlatego też, kolejny poziom hierarchii pamięci systemu komputerowego – pamięć podręczna – jest newralgiczny dla komunikacji pomiędzy procesorem a pamięcią główną i z tego względu, ma zasadniczy wpływ na czas wykonania programu [22] [70] [71] [85] [92].

Komunikację pomiędzy procesorem a pamięcią główną w systemach komputerowych, w których występuje pamięć podręczna, przedstawiono poglądowo na Rysunku 13.



Rysunek 13 Komunikacja pomiędzy procesorem a pamięcią główną (na podstawie [77])

Pamięć podręczna to pamięć o niewielkim rozmiarze i krótkim czasie dostępu, w której przechowywane są kopie danych znajdujących się w pamięci głównej. Ponieważ pamięć podręczna ma mniejszą pojemność, niż pamięć główna, może zawierać kopie tylko części danych z pamięci głównej.

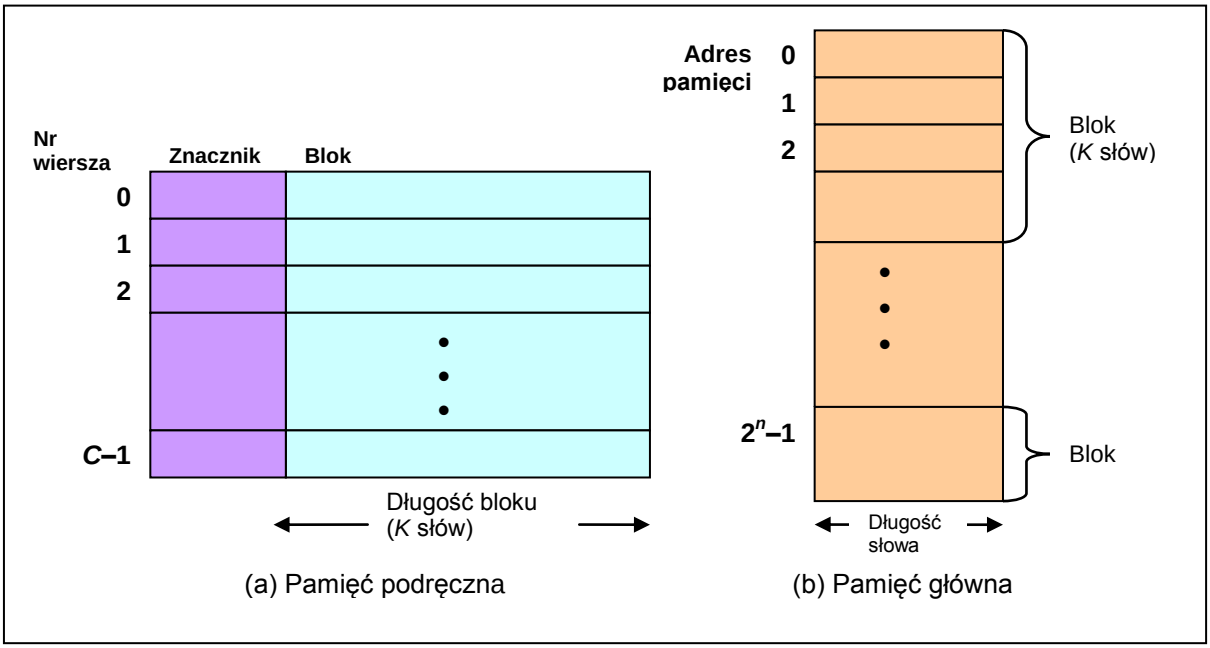
Wyróżnia się trzy poziomy pamięci podręcznej: L1, L2 oraz L3. Pamięć L1 jest pamięcią o najmniejszym rozmiarze i najkrótszym czasie dostępu, natomiast pamięć L3 jest pamięcią o największym rozmiarze i najdłuższym czasie dostępu.

Kiedy podczas wykonywania programu procesor zgłasza zapotrzebowanie na dane przechowywane w konkretnych lokalizacjach pamięci głównej, w pierwszej kolejności następuje sprawdzenie czy kopie tych danych znajdują się w pamięci podręcznej L1. Jeżeli dane nie zostaną odnalezione w pamięci podręcznej L1, wówczas sprawdzane są kolejno pamięci podręczne wyższych poziomów (L2 oraz, jeśli występuje, L3). Jeżeli dane nie zostaną odnalezione w pamięci podręcznej wyższych poziomów, wówczas z pamięci głównej pobierany jest blok danych zawierający żądane dane; blok ten jest ładowany do pamięci podręcznej i dopiero z pamięci podręcznej żądane dane są przesyłane do procesora.

Odnalezienie poszukiwanych danych w pamięci podręcznej dowolnego poziomu nazywa się trafieniem (ang. *cache hit*), natomiast nieodnalezienie tych danych w pamięci podręcznej dowolnego poziomu nazywa się chybieniem (ang. *cache miss*) [38].

Blok danych, pobrany z pamięci głównej do pamięci podręcznej, jest zapisywany w wierszu pamięci podręcznej, wskazanym przez algorytm odwzorowania bloków pamięci głównej na wiersze pamięci podręcznej. Taki algorytm jest niezbędny, ponieważ pamięć podręczna zawiera mniej wierszy, niż liczba bloków w pamięci głównej – a więc trzeba dysponować jakimś sposobem określenia czy i w którym wierszu pamięci podręcznej aktualnie znajduje się (lub, jeśli nie ma go w pamięci podręcznej – do którego jej wiersza powinien zostać wczytany) dany blok pamięci głównej. Wczytanie bloku pamięci głównej do wiersza pamięci

podręcznej jest równoznaczne z zastąpieniem dotychczasowej zawartości wiersza pamięci podręcznej wczytanym blokiem.
Na Rysunku 14 przedstawiono poglądowo relację pomiędzy strukturą pamięci głównej a strukturą pamięci podręcznej.



Rysunek 14 Struktura pamięci podręcznej i pamięci głównej [77]

Organizacja pamięci podręcznej jest uzależniona od sposobu (algorytmu) odwzorowania bloków pamięci głównej na wiersze pamięci podręcznej. Stosuje się trzy różne sposoby organizacji pamięci podręcznej:

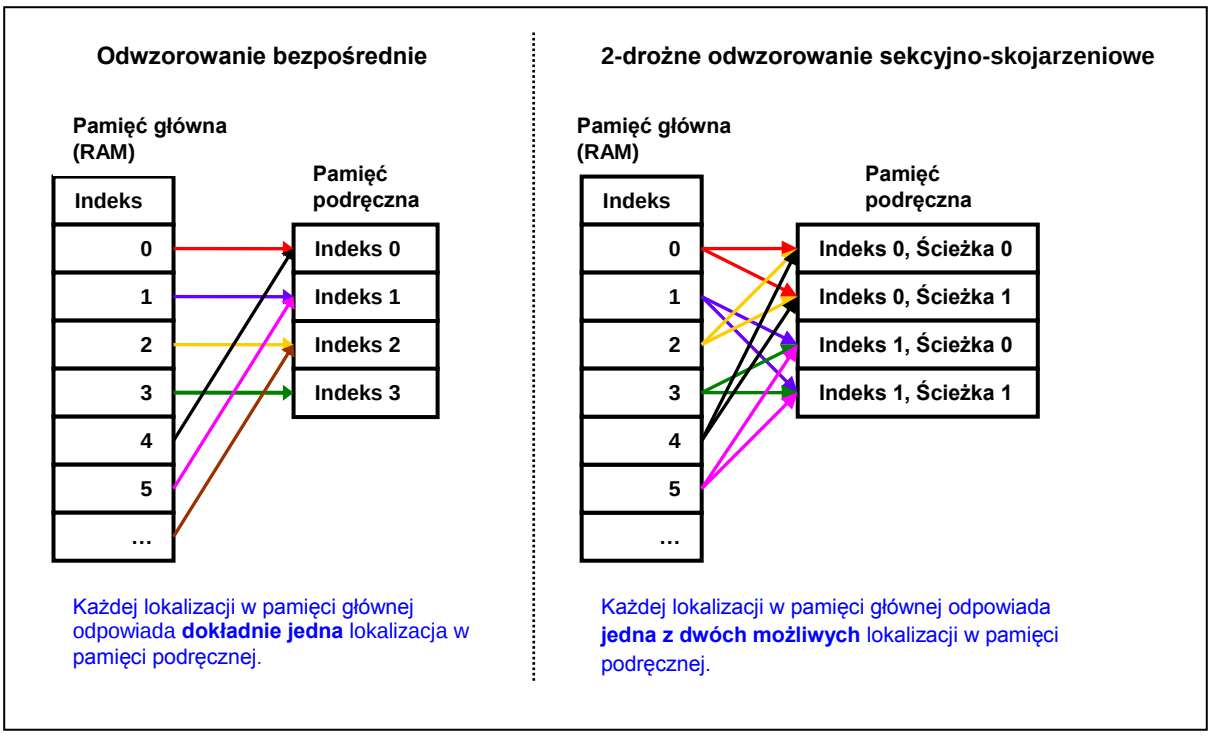
- pamięć podręczna z odwzorowaniem bezpośrednim,
- pamięć podręczna z odwzorowaniem skojarzeniowym (asocjacyjnym),
- pamięć podręczna z odwzorowaniem sekcyjno-skojarzeniowym.

Przy odwzorowaniu bezpośrednim, każdy blok pamięci głównej jest odwzorowywany na jeden i tylko jeden możliwy wiersz pamięci podręcznej. Podstawową wadą tego odwzorowania jest to, że wiele różnych bloków pamięci głównej może być odwzorowywanych na jeden i ten sam wiersz pamięci podręcznej. Jeżeli wykonywany program często odwołuje się do danych, które są przechowywane w różnych blokach pamięci głównej odwzorowywanych na ten sam wiersz pamięci podręcznej, skutkuje to: wzmożoną komunikacją pomiędzy pamięcią główną a pamięcią podręczną, potencjalnie mniejszym ponownym użyciem danych i w rezultacie, wydłużeniem czasu wykonywania programu.
Przy odwzorowaniu skojarzeniowym (asocjacyjnym) każdy blok pamięci głównej może zostać odwzorowany na dowolny wiersz pamięci podręcznej; wybór docelowego wiersza następuje podczas wczytywania nowego bloku do pamięci podręcznej. Realizacja fizyczna takiego odwzorowania jest skomplikowana i to stanowi jego podstawową wadę.

Przy odwzorowaniu sekcyjno-skojarzeniowym, pamięć podręczna jest dzielona na sekcje; każda sekcja liczy k wierszy (tzw. k-drożne odwzorowanie sekcyjno-skojarzeniowe). Każdy blok pamięci głównej jest odwzorowany na dokładnie jedną sekcję; w ramach tej sekcji przedmiotowy blok pamięci głównej może być wczytany do dowolnego z jej k wierszy. Odwzorowanie sekcyjno-skojarzeniowe to wariant pośredni pomiędzy odwzorowaniem bezpośrednim a odwzorowaniem skojarzeniowym.

W przypadku odwzorowań skojarzeniowego oraz sekcyjno-skojarzeniowego, do wyboru docelowego wiersza pamięci podręcznej stosuje się algorytmy zastępowania; najpopularniejsze z nich to: zastępowanie najdawniej użytego bloku (LRU, ang. *least recently used*), zastępowanie bloku najdłużej pozostającego w pamięci podręcznej (FIFO, ang. *first in first out*), zastępowanie bloku najrzadziej używanego (LFU, ang. *least frequently used*), przypadkowy wybór spośród kandydujących wierszy.

Różnice pomiędzy odwzorowaniem bezpośrednim a sekcyjno-skojarzeniowym przedstawiono w sposób poglądowy na Rysunku 15.



Rysunek 15 Porównanie odwzorowania bezpośredniego i sekcyjno-skojarzeniowego [26]

Z uwagi na specyfikę działania pamięci podręcznej, w wielu przypadkach występuje rozbieżność pomiędzy łączną, deklarowaną w kodzie źródłowym ilością danych do przetworzenia w programie a faktyczną, łączną ilością danych, które zostaną pobrane do pamięci podręcznej podczas wykonywania programu w celu zaspokojenia zgłaszanego przez program zapotrzebowania na dane.

To, ile bajtów danych zostanie faktycznie pobranych do pamięci podręcznej w związku z wykonywaniem programu zależy w głównej mierze od następujących czynników:

- w jakich lokalizacjach (tj. pod jakimi adresami) pamięci RAM są przechowywane dane, na które program zgłasza zapotrzebowanie,
- kolejności, w jakiej dane te są pobierane z pamięci RAM,
- rozmiaru pamięci podręcznej i sposobu jej organizacji.

Ww. czynniki decydują o tym, czy w trakcie wykonywania programu dane znajdujące się w pamięci podręcznej, które można by ponownie wykorzystać na potrzeby operacji wykonywanych w programie, zostaną ponownie wykorzystane w pełnym możliwym zakresie – czy, być może, dane te zostaną ponownie wykorzystane jedynie w pewnym zakresie lub wcale, co z kolei będzie skutkować koniecznością ponownego pobierania tych danych z pamięci głównej do pamięci podręcznej, wydłużając czas wykonania programu. Oznacza to, że ww. czynniki determinują faktyczne czasowe i przestrzenne ponowne użycie danych podczas wykonywania programu.
Jeżeli podczas wykonywania programu obserwuje się wysokie czasowe i przestrzenne ponowne użycie danych znajdujących się w pamięci podręcznej oznacza to, że program cechuje się wysoką lokalnością na poziomie pamięci podręcznej, czyli zgłaszane w związku z wykonywaniem programu zapotrzebowanie na dane jest w dużym stopniu zaspokajane lokalnie – danymi znajdującymi się w pamięci podręcznej, bez odwoływania się do pamięci głównej.

Jeżeli dany program można zapisać w różnych, ale semantycznie równoważnych, postaciach, to postać, dla której do pamięci podręcznej podczas wykonywania programu zostanie pobrana najmniejsza łączna ilość danych, cechuje się najlepszym ponownym użyciem danych.

W przeciwieństwie do ponownego użycia danych – łączna ilość danych, pobrana do pamięci podręcznej podczas wykonywania programu, jest wielkością zagregowaną, a co więcej, dobrze oddającą specyfikę działania pamięci podręcznej i jej związek z czasem wykonania programu. Faktyczna łączna ilość danych, pobrana do pamięci podręcznej podczas wykonywania programu, jest znana dopiero po wykonaniu programu. W związku z tym, łączna ilość danych, pobrana do pamięci podręcznej podczas wykonywania programu została uwzględniona w modelu (16) (patrz strona 32) szacunkowo – poprzez odcisk danych (*Df*).

Odcisk danych to wielkość szacunkowa, wyznaczana dla konkretnego programu, na podstawie kodu źródłowego tego programu.

Odcisk danych wskazuje szacunkową minimalną pojemność pamięci podręcznej z odwzorowaniem bezpośrednim, niezbędną do równoczesnego pomieszczenia wszystkich danych przetwarzanych w programie, przy założeniu pełnego czasowego i przestrzennego ponownego użycia danych przechowywanych w pamięci podręcznej. Czyli, odcisk danych

wskazuje szacunkową minimalną ilość danych, jaka zostanie pobrana z pamięci głównej do pamięci podręcznej podczas wykonywania programu.

Ilość danych, jaka zostanie pobrana z pamięci głównej do pamięci podręcznej podczas wykonywania programu (w konsekwencji, także oszacowanie tej ilości poprzez odcisk danych) jest zależna od występowania tzw. interferencji (ang. *interference*, [1] [24] [56] [81]).

Interferencja zachodzi w sytuacji, gdy wiersz pamięci podręcznej zawierający dane, które można ponownie użyć w programie, zostaje nadpisany nowymi danymi pomimo, że w pamięci podręcznej jest wolne miejsce, w którym można byłoby ulokować nowo wprowadzane dane – jednak sposób organizacji pamięci podręcznej wymusza nadpisanie konkretnego, zajętego wcześniej wiersza pamięci podręcznej [1] [24] [32] [81].

Wyróżnia się interferencję własną (ang. *self interference*) oraz interferencję krzyżową (ang. *cross interference*). Interferencja własna zachodzi w sytuacji, gdy dane znajdujące się w pamięci podręcznej oraz dane, które mają je zastąpić, są elementami jednej i tej samej tablicy. Interferencja krzyżowa zachodzi wtedy, gdy dane znajdujące się w pamięci podręcznej oraz dane, które mają je zastąpić, są wyrażone przez różne zmienne lub są elementami różnych tablic [24] [56].

W niniejszej pracy, odcisk danych dla danych przetwarzanych w gruboziarnistych, równoległych pętlach programowych wyznaczono w oparciu o metodę zaproponowaną przez Wolfe'a [88], z uwzględnieniem wpływu interferencji w oparciu o metodę zaproponowaną przez Lam [56]. Dla blokowania, dodatkowo przy wyznaczaniu odcisku danych uwzględniono wielkość bloku, opierając się na wnioskach przedstawionych przez Lam [56].
W dalszej części rozdziału przedstawiono sposób wyznaczenia odcisku danych dla następujących trzech sytuacji, które mogą wystąpić dla struktur pętli, uwzględnionych w modelu:

a) gdy nie występuje interferencja,
b) gdy występuje interferencja,
c) dla blokowania.

a) Wyznaczanie odcisku danych bez uwzględnienia interferencji

Metoda zaproponowana przez Wolfe'a [88] i przedstawiona poniżej pozwala, na podstawie znajomości samego kodu źródłowego programu (czyli, bez konieczności wykonywania programu w środowisku docelowym), w oparciu o współczynniki ponownego użycia danych, oszacować ponowne użycie danych zapisanych w zmiennych skalarnych i zmiennych tablicowych[5] o indeksach afinicznych (liniowych).

[5] W programowaniu komputerów, zmienna to konstrukcja posiadająca: nazwę (zwaną także identyfikatorem), miejsce przechowywania w podsystemie pamięci komputera, wartość oraz, zazwyczaj, typ, określający rodzaj danych zapisanych w zmiennej (a w konsekwencji, sposób reprezentacji wartości zmiennej w miejscu jej przechowywania). W trakcie wykonywania programu, wartość zmiennej może ulegać zmianom na skutek operacji wykonywanych w programie, natomiast nazwa i miejsce przechowywania zmiennej są stałe przez cały czas

Na podstawie współczynników ponownego użycia danych, jest wyznaczany zbiorczy odcisk danych dla całej analizowanej pętli.
Współczynniki ponownego użycia danych są wyznaczane dla wszystkich występujących w pętli odwołań do pamięci głównej, realizowanych przy pomocy zmiennych skalarnych i tablicowych (o indeksach afinicznych). Wartości wspomnianych współczynników są obliczane na podstawie liczby iteracji pętli oraz postaci indeksów zmiennych tablicowych.

Dla każdego, występującego w pętli odwołania do pamięci głównej obliczane są następujące współczynniki: własnego-czasowego ponownego użycia (ang. s*elf-temporal reuse factor*), własnego-przestrzennego ponownego użycia (ang. *self-spatial reuse factor*) własnego-ponownego użycia (ang. *self-reuse factor)* oraz odcisk danych (ang. *data footprint*).
Na potrzeby zdefiniowania ww. współczynników, w dalszej części pracy przez „pętlę *k*" będzie się rozumieć pętlę programową mającą N_k (N_k >1) iteracji, której zmienną sterującą jest zmienna *k*.

Współczynnik własnego-czasowego ponownego użycia obliczany dla występującego w pętli *k* odwołania wyrażonego zmienną *v* wskazuje, w ilu iteracjach pętli *k* może być wykorzystana (czyli, ponownie użyta) wartość pobrana z lokalizacji pamięci głównej wskazanej przez zmienną *v*. Dla wyrażenia tego współczynnika wprowadza się oznaczenie: $R_{k(temp)}(v)$.
Możliwe wartości współczynnika $R_{k(temp)}(v)$ to 1 lub N_k.
Współczynnik $R_{k(temp)}(v)$ będzie miał wartość 1, jeżeli pobrana wartość *v* może zostać użyta w dokładnie jednej iteracji pętli *k*. Taka sytuacja zachodzi wtedy, gdy zmienna *v* jest zmienną tablicową, wyrażoną przy pomocy tablicy jedno- lub wielowymiarowej i w przynajmniej jednym indeksie przedmiotowej tablicy występuje zmienna *k*. Czyli współczynnik $R_{k(temp)}(v)$ będzie miał wartość 1, jeśli wartość zmiennej *v* jest zależna od *k*.
Współczynnik $R_{k(temp)}(v)$ będzie miał wartość N_k, jeżeli pobrana wartość może zostać użyta we wszystkich iteracjach pętli *k.* Taka sytuacja zachodzi wtedy, gdy zmienna *v* jest zmienną skalarną lub zmienną tablicową, wyrażoną przy pomocy tablicy jedno- lub wielowymiarowej i w żadnym indeksie przedmiotowej tablicy nie występuje zmienna *k*. Tak więc, współczynnik $R_{k(temp)}(v)$ będzie miał wartość N_k, jeśli wartość zmiennej *v* jest niezależna od *k*.

Powyższe rozważania zilustrowano przykładem.

Przykład 1: Obliczyć wartość współczynnika $R_{k(temp)}(v)$ gdy:
1/ $v = a$

istnienia zmiennej. W kodzie źródłowym programu, poprzez nazwę zmiennej można odwołać się do jej wartości lub miejsca przechowywania.
Zmienne skalarne to zmienne, które mogą przechowywać jednocześnie tylko jedną wartość liczbową. Zmienne tablicowe to zmienne, które mogą przechowywać jednocześnie wiele wartości tego samego typu – w postaci tablicy, której poszczególne komórki są dostępne poprzez jednoznacznie je identyfikujący klucz. W językach C/C++, przedmiotowy klucz nazywa się indeksem; indeks jest liczbą naturalną. Tablice mogą być jedno i wielowymiarowe; w matematyce odpowiednikiem tablicy jednowymiarowej (o jednym indeksie) jest ciąg, tablicy dwuwymiarowej (o 2 indeksach) – macierz, tablicy więcej niż dwuwymiarowej (mającej więcej niż 2 indeksy) – macierz wielowymiarowa.

2/ $v = b[i][j]$

3/ $v = c[i][2k+j][p]$

Rozwiązanie:

1/ Zmienna *a* jest zmienną skalarną; jej wartość nie zależy od *k* – w związku z tym, $R_{k(temp)}(v)=N_k$.

2/ Zmienna $b[i][j]$ jest zmienną tablicową o 2 indeksach; żaden z indeksów nie jest zależny od *k* – w związku z tym, wartość $b[i][j]$ jest niezależna od *k*, w konsekwencji, $R_{k(temp)}(v)=N_k$.

3/ Zmienna $c[i][2k+j][p]$ jest zmienną tablicową o 3 indeksach; jeden z indeksów tej zmiennej $[2k+j]$ jest zależny od *k* – w związku z tym, wartość $c[i][2k+j][p]$ jest zależna od *k*, w konsekwencji, $R_{k(temp)}(v)=1$.

Współczynnik własnego-przestrzennego ponownego użycia obliczany dla występującego w pętli *k* odwołania wyrażonego zmienną *v* wskazuje ile różnych elementów bloku pamięci głównej (pobranego do wiersza pamięci podręcznej), zawierającego wartość z lokalizacji pamięci wskazanej przez zmienną *v*, może być wykorzystanych (czyli, ponownie użytych) w programie, bez konieczności dodatkowego pobierania tych elementów z pamięci głównej.

Dla wyrażenia tego współczynnika wprowadza się oznaczenie: $R_{k(spat)}(v)$.

Możliwe wartości współczynnika $R_{k(spat)}(v)$ to 1 lub $\max(1, L/a_{sk})$ gdzie:

L – liczba elementów mieszczących się w wierszu pamięci podręcznej, o typie identycznym z typem wartości pobranej z lokalizacji wskazanej przez odwołanie wyrażone zmienną *v*,

a_{sk} – współczynnik stojący przy zmiennej *k* w tym indeksie rozpatrywanego odwołania, który to indeks jest odpowiedzialny za odwzorowanie w pamięci kolejno po sobie następujących elementów tablicy związanej z rozpatrywanym odwołaniem,

$\max(x,y)$ – większa z liczb *x*, *y*.

W językach C i C++, tablice odwzorowywane są w pamięci w porządku wierszowym, czyli wiersz po wierszu. Dla tablicy wielowymiarowej, która jest odwzorowywana w pamięci w porządku wierszowym, jej ostatni indeks jest odpowiedzialny za odwzorowanie w pamięci elementów tablicy następujących kolejno po sobie.

W przypadku programów zapisanych w językach C i C++, oznacza to, że współczynnik $R_{k(spat)}(v)$ może przyjąć wartość większą od 1 tylko wtedy, gdy zmienna *k* pojawia się wyłącznie w ostatnim indeksie odwołania wyrażonego zmienną *v*, natomiast stojący przy zmiennej *k* współczynnik a_{sk} przyjmuje wartość mniejszą niż rozmiar linii pamięci podręcznej (rozmiar linii pamięci podręcznej jest tu wyrażony liczbą elementów o typie identycznym z typem zmiennej *v*, mieszczących się w linii pamięci podręcznej). Tak więc, przy wierszowym odwzorowaniu tablic w pamięci, współczynnik $R_{k(spat)}(v)$ może przyjąć wartość większą od 1 tylko wówczas, gdy wyłącznie ostatni indeks zmiennej *v* jest zależny od *k*.

Powyższe rozważania zilustrowano przykładem.

Przykład 2: Zakłada się, że wiersz pamięci podręcznej może pomieścić L = 16 elementów o typie identycznym z typem zmiennej *v.* Zakładając wierszowe odwzorowanie tablic w pamięci, obliczyć wartość współczynnika $R_{k(spat)}(v)$ gdy:

1/ $v = a$

2/ $v = b[i][j]$

3/ $v = c[i][2k+j][p]$

4/ $v = d[i][j][2k]$

Rozwiązanie:

1/ Zmienna *a* jest zmienną skalarną; jej wartość nie zależy w żaden sposób od k – w związku z tym, $R_{k(spat)}(v) = 1$.

2/ Zmienna $b[i][j]$ jest zmienną tablicową o 2 indeksach; żaden z indeksów nie jest zależny od k – w związku z tym, $R_{k(spat)}(v) = 1$.

3/ Zmienna $c[i][2k+j][p]$ jest zmienną tablicową o 3 indeksach; tylko jeden z indeksów tej zmiennej, $[2k+j]$, jest zależny od k, nie jest to jednak ostatni indeks – w związku z tym, $R_{k(spat)}(v) = 1$.

4/ Zmienna $d[i][j][2k]$ jest zmienną tablicową o 3 indeksach; tylko jeden z indeksów tej zmiennej, $[2k]$, jest zależny od k, jest to zarazem ostatni indeks – w związku z tym: $R_{k(spat)}(v) = \max(1, L/a_{sk}) = \max(1, 16/2) = \max(1, 8) = 8$.

Współczynniki własnego-czasowego i własnego-przestrzennego ponownego użycia dla występującego w pętli k odwołania wyrażonego zmienną v są wykorzystywane do wyznaczenia współczynnika własnego-ponownego użycia. Dla wyrażenia tego współczynnika wprowadza się oznaczenie: $R_k(v)$.

Wartość $R_k(v)$ wyznacza się następująco:

$$R_k(v) = \begin{cases} R_{k(temp)}(v), \text{ gdy } R_{k(temp)}(v) > 1 \\ R_{k(spat)}(v), \text{ gdy } R_{k(temp)}(v) = 1 \end{cases} \tag{17}$$

Znając wartość $R_k(v)$, można wyznaczyć odcisk danych dla występującego w pętli k odwołania wyrażonego zmienną *v.* Dla wyrażenia przedmiotowego odcisku danych wprowadza się oznaczenie: $F_k(v)$. Odcisk danych $F_k(v)$ wskazuje minimalną (przy założeniu maksymalnego możliwego ponownego użycia danych) liczbę bloków pamięci głównej, które są pobierane do wierszy pamięci podręcznej podczas wykonywania pętli k, aby zaspokoić zapotrzebowanie procesora na dane przechowywane w lokalizacji pamięci głównej wskazanej przez zmienną *v.* Tak więc, odcisk danych $F_k(v)$ wskazuje dla ilu wierszy pamięci podręcznej dotychczasowa zawartość zostanie w tej sytuacji zastąpiona „nową” zawartością.

Odcisk danych $F_k(v)$ wyznacza się następująco:

$$F_k(v) = \frac{N_k}{R_k(v)} \tag{18}$$

Odcisk danych $F_k(v)$ wyznaczony wg równania (18) nie uwzględnia relacji pomiędzy pętlą *k* a innymi pętlami, które mogą występować w programie i być powiązane z pętlą *k* w ten sposób, że pętla *k j*est wykonywana w ramach innej pętli lub inna pętla jest wykonywana w ramach pętli *k.* Oznacza to, że jeżeli pętla *k* jest wykonywana w ramach pętli zagnieżdżonej (czyli pętli, która może zawierać w sobie kolejną pętlę lub wiele pętli) – odcisk danych $F_k(v)$ nie bierze pod uwagę pozycji pętli *k* w gnieździe pętli.
Pozycję pętli *k* w gnieździe pętli i wynikające stąd inne, niż w przypadku odcisku danych $F_k(v)$*,* zapotrzebowanie na pamięć podręczną wyraża skumulowany odcisk danych.
Skumulowany odcisk danych dla występującego w pętli *k* odwołania wyrażonego zmienną *v* oznacza się jako $F_r^*(v)$*.* Skumulowany odcisk danych $F_r^*(v)$ wyznacza się następująco:

$$F_r^*(v) = b\prod_{i=1}^{r}\frac{N_i}{R_i(v)} \;=\; b\prod_{i=1}^{r}F_i(v) \tag{19}$$

gdzie:

b – rozmiar linii pamięci podręcznej, wyrażony w bajtach,
r – poziom w m-poziomowym gnieździe pętli (licząc od najbardziej wewnętrznej pętli gniazda), na którym znajduje się pętla *k,*
N_i – liczba iteracji pętli *i,*
$R_i(v)$ – współczynnik własnego-ponownego użycia dla występującego w pętli *i* odwołania wyrażonego zmienną *v,*
$F_i(v)$ – odcisk danych dla występującego w pętli *i* odwołania wyrażonego zmienną *v.*

Skumulowane odciski danych są wykorzystywane do wyznaczenia zbiorczego odcisku danych dla całej analizowanej pętli.
Zbiorczy odcisk danych dla całej analizowanej pętli zagnieżdżonej jest sumą skumulowanych odcisków danych wyznaczonych, na poziomie najbardziej zewnętrznej pętli gniazda, dla wszystkich odwołań do pamięci głównej, występujących we wszystkich pętlach pętli zagnieżdżonej. Zbiorczy odcisk danych jest wyrażony następującym równaniem:

$$F_{total} = \sum_{p=1}^{P} F_t^*(v_p) \tag{20}$$

gdzie:

F_{total} – zbiorczy odcisk danych,
P – łączna liczba wszystkich różnych odwołań do pamięci, występujących we wszystkich pętlach pętli zagnieżdżonej,
$F_t^*(v_p)$ – skumulowany odcisk danych dla występującego w pętli *t* (najbardziej zewnętrznej pętli gniazda) odwołania wyrażonego zmienną v_p*.*

Zbiorczy odcisk danych wyznaczony wg równania (20) wskazuje szacunkową minimalną ilość danych (wyrażoną w bajtach) jaka zostanie pobrana z pamięci głównej do pamięci podręcznej z odwzorowaniem bezpośrednim podczas wykonywania programu (tutaj: zagnieżdżonej pętli programowej), przy założeniu pełnego czasowego i pełnego przestrzennego ponownego użycia danych przechowywanych w pamięci podręcznej.

b) Wyznaczanie odcisku danych z uwzględnieniem interferencji

W przedstawionym w punkcie a) sposobie wyznaczenia zbiorczego odcisku danych (patrz równanie (20)) nie uwzględniono interferencji. Interferencja to zjawisko wynikające ze specyfiki organizacji i działania pamięci podręcznej, a polegające na zastępowaniu zawartości wierszy pamięci podręcznej, zawierających dane, które można ponownie użyć w programie, nowymi danymi, pomimo że w pamięci podręcznej znajduje się wolne miejsce, w którym można byłoby ulokować nowe dane. Sytuację zastąpienia zawartości pojedynczego wiersza pamięci podręcznej, zawierającego dane, które można ponownie użyć w programie, nowymi danymi, pomimo, że w pamięci podręcznej znajduje się wolne miejsce, w którym można byłoby ulokować nowe dane, w dalszej części pracy określa się mianem „chybienia spowodowanego interferencją”.

W sytuacji, gdy podczas wykonywania programu zachodzi interferencja, należy uwzględnić jej wpływ na ilość danych pobieranych do pamięci podręcznej podczas wykonywania programu. W niniejszej pracy, wpływ ten uwzględniono w oparciu o podejście zaproponowane przez Lam [56] i przedstawione poniżej.

Zgodnie z równaniem (18) (patrz strona 43), odcisk danych $F_k(v)$ *d*la pętli *k* wyznacza się następująco:

$$F_k(v) = \frac{N_k}{R_k(v)}$$

Przytoczone powyżej równanie (18) nie uwzględnia interferencji; Lam [56] proponuje uwzględnić ją równaniem następującej postaci:

$$F_{k(interf)}(v) = N_k\left(\frac{1}{R_k(v)} + \frac{R_k(v)-1}{R_k(v)} M_k(v)\right) = F_k(v)\left[1 + (R_k(v)-1)M_k(v)\right] \quad (21)$$

gdzie:

$F_{k(interf)(}(v)$ – odcisk danych dla występującego w pętli *k* odwołania wyrażonego zmienną *v*, z uwzględnieniem interferencji,

$M_k(v)$ – udział chybień w pamięci podręcznej dla występującego w pętli *k* odwołania wyrażonego zmienną *v.*

Jeżeli zmienna *v* jest ponownie używana w pętli *k*, a współczynnik własnego-ponownego użycia tej zmiennej wynosi $R_k(v)$, wówczas wykonywana na poziomie pętli *k* liczba odwołań do pamięci RAM w związku z operacjami wymagającymi użycia wartości zmiennej *v* jest

redukowana $R_k(v)$ razy, tj. z N_k do $N_k / R_k(v)$. Oznacza to, że tylko w $N_k / R_k(v)$ iteracjach pętli k następuje pobranie wartości zmiennej v z pamięci RAM do pamięci podręcznej. Taka sytuacja ma miejsce w przypadku braku interferencji.
Jeśli jednak zachodzi interferencja, wówczas zachodzi konieczność dodatkowego pobrania wartości zmiennej v z pamięci RAM do pamięci podręcznej, przez co ilość danych pobranych do pamięci podręcznej zwiększa się o $[N_k - N_k / R_k(v)] M_k(v) = N_k [(R_k(v) – 1) / R_k(v)] M_k(v)$.
Udział chybień w pamięci podręcznej, $M_k(v)$, wyznacza się następująco:

$$M_k(v) = 1 - (1 - S_k(v)) \prod_{u \in V - \{v\}} \left(1 - \frac{F_k(u)}{C_{rows}} \right) \tag{22}$$

gdzie:

V – zbiór wszystkich zmiennych, występujących w pętli k,

u – oznacza zmienną występująca w pętli k,

C_{rows} – rozmiar pamięci podręcznej, wyrażony liczbą wierszy pamięci podręcznej,

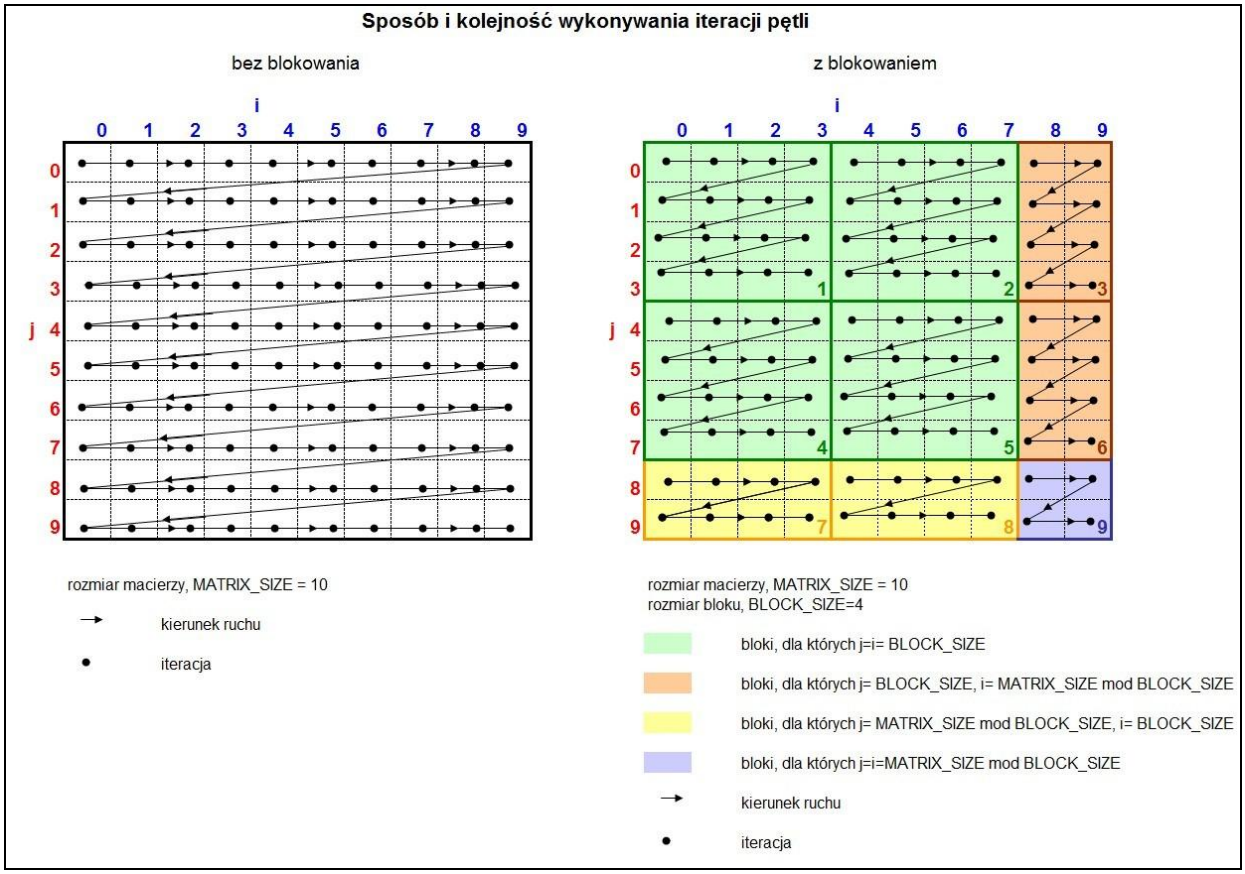
$S_k(v)$ – wskazuje, jaka część pamięci podręcznej została wypełniona danymi w związku z interferencją własną zmiennej v.

$F_k(u) / C_{rows}$ wskazuje, jaka szacunkowo część pamięci podręcznej jest zajęta przez dane, pobrane do pamięci podręcznej w związku z operacjami wykonywanymi w pętli k na zmiennej u. $(1 - F_k(u) / C_{rows})$ wskazuje, jaka szacunkowo część pamięci podręcznej nie jest zajęta przez ww. dane.
Jeżeli $M_k(v) = 1$, to wówczas $F_{k(interf)}(v) = N_k$. Jeżeli $M_k(v) < 1$, to wówczas $F_{k(interf)}(v) < N_k$.
Jeżeli $S_k(v) = 1$, to wówczas $M_k(v) = 1$ co w takiej sytuacji oznacza, że bez względu na siłę oddziaływania interferencji krzyżowej, $F_{k(interf)}(v) = N_k$.

Znając $F_{k(interf)}(v)$ – czyli skorygowaną o interferencję wartość odcisku danych $F_k(v)$, można obliczyć wartości skumulowanych odcisków danych (stosując równanie (19), poprzez podstawienie do niego w miejsce $F_i(v)$ wartości $F_{i(interf)}(v)$), co pozwala obliczyć zbiorczy odcisk danych z uwzględnieniem interferencji.

c) Wyznaczenie odcisku danych dla blokowania

Jeżeli w pętli, dla której ma być obliczony zbiorczy odcisk danych, zastosowano blokowanie blokiem kwadratowym o wielkości boku *BLOCK_SIZE* – obliczając odcisk danych należy uwzględnić specyficzną dla blokowania kolejność wykonywania iteracji pętli, w których zastosowano blokowanie (patrz Rysunek 16). Poszczególne bloki, powstałe na skutek blokowania, są wykonywane w kolejności zaznaczonej na rysunku cyframi, umieszczonymi w prawych dolnych narożnikach bloków.



Rysunek 16 Wpływ blokowania na kolejność wykonywania iteracji pętli

Przy blokowaniu przestrzeni iteracji, wyznaczonej przez 2 pętle: pętlę *j* oraz pętlę *i* (dla uproszczenia założono, że: $N_j = N_i = MATRIX\_SIZE$) blokiem kwadratowym o boku *BLOCK_SIZE*, przestrzeń iteracji jest dzielona na bloki 4 typów:

- bloki, dla których $j = i = BLOCK\_SIZE$ (bloki wyróżnione kolorem zielonym na Rysunku 16)
- bloki, dla których:
  $j = BLOCK\_SIZE$,
  $i = N_i \bmod BLOCK\_SIZE = MATRIX\_SIZE \bmod BLOCK\_SIZE$
  (bloki wyróżnione kolorem pomarańczowym na Rysunku 16)
- bloki, dla których:
  $j = N_j \bmod BLOCK\_SIZE = MATRIX\_SIZE \bmod BLOCK\_SIZE$,
  $i = BLOCK\_SIZE$
  (bloki wyróżnione kolorem żółtym na Rysunku 16)
- bloki, dla których:
  $j = N_j \bmod BLOCK\_SIZE = MATRIX\_SIZE \bmod BLOCK\_SIZE$,
  $i = N_i \bmod BLOCK\_SIZE = MATRIX\_SIZE \bmod BLOCK\_SIZE$
  (bloki wyróżnione kolorem fioletowym na Rysunku 16)

Wyznaczenie odcisku danych dla sytuacji, przedstawionej na Rysunku 16, sprowadza się do wyznaczenia odcisku danych (ewentualnie skorygowanego o wpływ interferencji) osobno dla każdego typu bloków – tj. osobno dla bloków wyróżnionych na zielono, pomarańczowo, żółto i fioletowo.

Jeżeli poddane blokowaniu pętla *j* oraz pętla *i* są zagnieżdżone w innej pętli (załóżmy, że jest to pętla *p,* która nie jest zagnieżdżona w innej pętli), to wówczas przy obliczaniu zbiorczego odcisku danych dla pętli *p* należy – obliczając odcisk danych na poziomie pętli *j* oraz na poziomie pętli *i* – uwzględnić, w sposób opisany powyżej, podział przestrzeni iteracji (*j*, *i*) na bloki 4 różnych typów.

Istotnym problemem związanym z blokowaniem jest dobór wielkości boku bloku. Lam [56] zaproponowała algorytm pozwalający znaleźć graniczny rozmiar boku bloku kwadratowego, nie powodujący interferencji własnej. Dla pamięci podręcznej z odwzorowaniem bezpośrednim, algorytm ten przedstawiono na Rysunku 17.

```
maxWidth := min(N, C);
addr := N/2;
   dopóki prawda powtarzaj
      addr := addr + C;
      di := addr div N;
      dj := abs((addr mod N) – N/2);
      jeżeli di > min(maxWidth, dj)
         zwróć B := min(maxWidth,di);
      maxWidth := min(maxWidth, dj);
   koniec dopóki;
```

gdzie:

N, *C*, *B*, *addr*, *di*, *dj*, *maxWidth* – liczby całkowite

N	–	rozmiar macierzy kwadratowej poddanej blokowaniu
C	–	liczba słów (typu całkowitego), mieszczących się w pamięci podręcznej
B	–	graniczny rozmiar boku bloku kwadratowego, nie powodujący interferencji własnej
abs(*x*)	–	wartość bezwzględna liczby *x*
a mod *b*	–	*a* modulo *b*, reszta z dzielenia *a* przez *b*
a div *b*	–	całkowity wynik dzielenia *a* przez *b*, z pominięciem części ułamkowej. Czyli, *a* div *b* = [*a* – (*a* mod *b*)] / *b*
max(*x*,*y*)	–	oznacza większą z liczb *x*, *y*
min(*x*,*y*)	–	oznacza mniejszą z liczb *x*, *y*

Rysunek 17 Wyznaczanie granicznego rozmiaru boku bloku kwadratowego przy blokowaniu – dla pamięci podręcznej z odwzorowaniem bezpośrednim [56]

Wynikiem algorytmu przedstawionego na Rysunku 17 jest B, czyli graniczny rozmiar boku bloku kwadratowego, nie powodujący interferencji własnej. Jeżeli $B \geq N$, wówczas blokowanie nie ma sensu praktycznego.

Relacja pomiędzy faktycznie zastosowanym rozmiarem boku bloku kwadratowego (*BLOCK_SIZE*), *B* a interferencją własną przedstawia się następująco:

Jeżeli	$0 < BLOCK\_SIZE \leq B$	to	nie występuje interferencja własna.
Jeżeli	$B < BLOCK\_SIZE < N$	to	występuje interferencja własna.
Jeżeli	$BLOCK\_SIZE \geq N$	to	blokowanie nie ma sensu praktycznego.

Zgodnie z przytoczonym poniżej równaniem (22):

$$M_k(v) = 1 - (1 - S_k(v)) \prod_{u \in V - \{v\}} \left(1 - \frac{F_k(u)}{C_{rows}}\right)$$

W niniejszej pracy założono, że przy blokowaniu, w sytuacji, gdy $B < BLOCK\_SIZE < N$, interferencja własna zostanie uwzględniona w równaniu (22) w taki sam sposób, jak interferencja krzyżowa, przez co równanie (22) zostanie sprowadzone do postaci równania (23):

$$M_k(v) = 1 - \prod_{u \in V} \left(1 - \frac{F_k(u)}{C_{rows}}\right) \tag{23}$$

W niniejszej pracy zakłada się, że w przypadku wielopoziomowych pamięci podręcznych, algorytm przedstawiony na Rysunku 17 będzie wykonywany w odniesieniu do pamięci podręcznej L1 [71].

Algorytm przedstawiony na Rysunku 17 można zastosować również dla pamięci podręcznej z odwzorowaniem sekcyjno-skojarzeniowym – uwzględniając asocjacyjność pamięci podręcznej z odwzorowaniem sekcyjno-skojarzeniowym w rozmiarze macierzy kwadratowej *N*. Oznacza to, że dla pamięci podręcznej z odwzorowaniem sekcyjno-skojarzeniowym za *N* należy podstawić (*N* x *asocjacyjność pamięci podręcznej L1*). Asocjacyjności pamięci podręcznej z odwzorowaniem sekcyjno-skojarzeniowym nie należy uwzględniać w wartości *C*.

3.1.2 Wyznaczanie wartości zmiennych niezależnych

W ramach niniejszej pracy, wszystkie badania eksperymentalne zostały przeprowadzone w środowisku przedstawionym w Tabeli 7.

Tabela 7 Specyfikacja środowiska badań eksperymentalnych

Procesor	Intel Core 2 Quad Q6600 (technologia wykonania: 65 nm)
Liczba rdzeni	4
Liczba wątków sprzętowych	4

Pamięć podręczna L1	4 x 32 KB Odwzorowanie: 8-drożne sekcyjno-skojarzeniowe Rozmiar wiersza pamięci podręcznej: 64 B
Pamięć podręczna L2	2 x 4 096 KB Odwzorowanie: 16-drożne sekcyjno-skojarzeniowe Rozmiar wiersza pamięci podręcznej: 64 B
Pamięć podręczna L3	Brak
System operacyjny	Linux Slax 6.1.2
Kompilator	gcc 4.2.4
Wersja OpenMP, użyta na potrzeby zrównoleglenia	OpenMP v2.5
Optymalizacja na poziomie kompilacji	Wyłączona (kompilacja z opcją –O0)

Przy wyznaczaniu wartości zmiennych niezależnych *X1, X2, X3, X4* modelu ogólnego (16) (patrz strona 32) uwzględniono specyfikę środowiska badań eksperymentalnych.

3.1.2.1. Wyznaczenie wartości zmiennej *X1*

Zmienna X1 to wielkość bezwymiarowa, określająca stosunek łącznego rozmiaru pamięci podręcznej L1 i L2, przypadającej na pojedynczy wątek OpenMP, do odcisku danych *Df* dla pojedynczego wątku OpenMP.

Podejście przedstawione w rozdziale 3.1.1 jest adekwatne do wyznaczania odcisku danych *Df* dla pamięci podręcznych z odwzorowaniem bezpośrednim.

Jeżeli w systemie komputerowym zastosowano pamięć podręczną z odwzorowaniem bezpośrednim, to wówczas zmienna *X1* przyjmuje wartość wyrażoną następującym równaniem:

$$X1 = \frac{sL1 + sL2}{D_f} \tag{24}$$

gdzie:

sL1 – rozmiar pamięci podręcznej L1, dostępnej dla pojedynczego wątku OpenMP *(wielkość wyrażona w bajtach),*

sL2 – rozmiar pamięci podręcznej L2, dostępnej dla pojedynczego wątku OpenMP *(wielkość wyrażona w bajtach),*

Df – odcisk danych dla pojedynczego wątku OpenMP *(wielkość wyrażona w bajtach).*

Równanie (24) nie jest adekwatne dla systemu komputerowego, w którym zastosowano pamięć podręczną z odwzorowaniem sekcyjno-skojarzeniowym, z uwagi na niezgodność pomiędzy odwzorowaniem wykorzystanym w pamięci podręcznej a odwzorowaniem, dla którego (wg zastosowanego podejścia) wyznaczono wartość *Df*.

Dlatego też, jeżeli w systemie komputerowym zastosowano pamięci podręczne z odwzorowaniem sekcyjno-skojarzeniowym, należy przeliczyć ich rozmiary na rozmiary

pamięci podręcznych z odwzorowaniem bezpośrednim, które są odpowiednikami wspomnianych pamięci z odwzorowaniem sekcyjno-skojarzeniowym.
Uwzględniając takie przeliczenie, równanie (24) dla obliczenia zmiennej *X1* przyjmuje postać przedstawioną równaniem (3) (patrz strona 22), przytoczonym poniżej:

$$X1 = \frac{\left(sL1 \times aL1 + sL2 \times aL2\right)}{D_f}$$

gdzie:
aL1 – „asocjacyjność" pamięci podręcznej L1 *(wielkość bezwymiarowa),*
aL2 – „asocjacyjność" pamięci podręcznej L2 *(wielkość bezwymiarowa).*

Równanie (3) jest prawdziwe także dla pamięci podręcznych L1 oraz L2 z odwzorowaniem bezpośrednim – dla tych pamięci, w równaniu (3) należy dokonać następujących podstawień: *aL1* = 1, *aL2* = 1.

Dla przedstawionego w Tabeli 7 (patrz strona 48) środowiska sprzętowego z procesorem Intel Core 2 Quad Q6600, wartości *sL1, sL2, aL1, aL2* wynoszą:

sL1 = 32 768 B; *aL1* = 8
sL2 = 4 194 304 B; *aL2* = 16

Uwagi:
- Zaproponowany w niniejszej pracy model ogólny, wyrażony równaniem (16), nie jest adekwatny dla pamięci podręcznych z odwzorowaniem skojarzeniowym.
- Pomimo, że dla procesorów Intel jest stosowana pamięć podręczna typu „inclusive" (tzn. dane znajdujące się w pamięci L1 są generalnie zawarte, jako kopia, w pamięci L2 i pamięci L3, jeśli pamięć L3 występuje [64]), w modelu zaproponowanym w niniejszej pracy dla uproszczenia przyjęto, że wartość *sL2* jest równa rozmiarowi całej pamięci podręcznej L2, dostępnej dla pojedynczego wątku OpenMP.
- Pamięć podręczna L3 jest używana głównie w zastosowaniach specjalistycznych, dlatego też pominięto ją w modelu (16).

3.1.2.2. Wyznaczenie wartości zmiennej *X2*

W ramach badań eksperymentalnych niniejszej pracy, zbadano pętle wzorcowe, omówione w rozdziale 3.2, oraz wybrane pętle programowe z zestawu testów referencyjnych NAS Parallel Benchmarks [40] [63]. Pętle, które poddano badaniom, zostały wykonane w środowisku przedstawionym w Tabeli 7 (patrz strona 48), dla danych typu całkowitego. Oznacza to, że wszystkie wykonywane w ww. pętlach operacje (tj. przypisanie, dodawanie, odejmowanie, mnożenie) były wykonywane na liczbach całkowitych.
Zgodnie z Tabelą 7, do badań wykorzystano procesor Intel Core 2 Quad Q6600, wykonany w technologii 65 nm. Na podstawie przedstawionych w [34] czasów wykonania rozkazów dla

procesorów z rodziny Intel Core 2 wyprodukowanych w technologii 65 nm, przyjęto następujące wagi dla poszczególnych operacji realizowanych w pętlach:

- przypisanie, dodawanie, odejmowanie liczb całkowitych – waga 1,
- mnożenie liczb całkowitych – waga 1,5.

W związku z powyższym, łączną ważoną liczbę operacji dla pętli *k* można wyznaczyć korzystając z następującego równania:

$$o_{weighted\_total}(k) = \sum_{t \in T} o_t(k) \times w_t(k) \tag{25}$$

gdzie:

$o_{weighted\_total}(k)$ – łączna ważona liczba operacji dla pętli *k,*

T – uwzględnione w modelu typy operacji występujących w pętli *k*. Do zbioru *T* należą następujące operacje wykonywane na liczbach całkowitych: przypisanie, dodawanie, odejmowanie, mnożenie.

$o_t(k)$ – łączna liczba operacji typu *t* dla pętli *k* (wyznaczana poprzez zliczenie wystąpień operacji typu *t* w kodzie źródłowym programu),

$w_t(k)$ – waga operacji typu *t*.

Wartość $o_{weighted\_total}(k)$ nie uwzględnia pozycji pętli *k* w gnieździe pętli. Po uwzględnieniu pozycji pętli *k* w gnieździe pętli, równanie (25) przyjmuje następującą postać:

$$o^*_{weighted\_total}(k) = \sum_{t \in T} o^*_t(k) \times w_t(k) \tag{26}$$

gdzie:

$o^*_{weighted\_total}(k)$ – łączna ważona liczba operacji dla pętli *k* i wszystkich pętli zagnieżdżonych wewnątrz pętli *k,*

$o^*_t(k)$ – łączna liczba operacji typu *t* dla pętli *k* i wszystkich pętli zagnieżdżonych wewnątrz pętli *k* (wyznaczana poprzez zliczenie wystąpień operacji typu *t* w kodzie źródłowym programu).

Wartość zmiennej *X2*, czyli łączna ważona liczba operacji przypadająca na pojedynczy wątek OpenMP, jest wyznaczana przy pomocy równania (26), zastosowanego do wykonywanej przez pojedynczy wątek OpenMP najbardziej zewnętrznej pętli w pętli zagnieżdżonej.

3.1.2.3. Wyznaczenie wartości zmiennej *X3*

Zmienna *X3* modelu (16) (patrz strona 32) wskazuje maksymalną, dla danego sposobu przydziału iteracji do wątków OpenMP, wielkość pojedynczej porcji iteracji przydzielonej do wykonania przez wątek. Założono tutaj, że iteracje są przydzielane do wątków OpenMP zgodnie z planowaniem „static" [5], tzn. każdy wątek otrzymuje do wykonania w przybliżeniu jednakową liczbę iteracji, w porcjach o zadanej przez programistę wielkości *chunk_size*. Jeżeli programista nie określił wielkości *chunk_size*, kompilator przydzieli do każdego z

wątków wykonujących program w przybliżeniu jednakową liczbę iteracji, równą (w przybliżeniu) ilorazowi liczby wszystkich iteracji do wykonania i liczby wątków przewidzianych do wykonania przedmiotowych iteracji.
W dalszej części pracy, przydział iteracji do wątków w porcjach o zdefiniowanej przez programistę wielkości będzie określany jako „wymuszony", natomiast przydział iteracji do wątków w porcjach o wielkości określonej przez kompilator będzie określany jako „domyślny".

Oznacza to, że wartość zmiennej X3 można wyznaczyć przy pomocy następującego równania:

$$X3 = \begin{cases} \left\lceil \frac{N_{outermost}}{num\_threads} \right\rceil & \text{dla domyślnego przydziału iteracji do wątków} \\ chunk\_size & \text{dla wymuszonego przydziału iteracji do wątków} \end{cases} \quad (27)$$

gdzie:

$N_{outermost}$ – liczba iteracji do wykonania w najbardziej zewnętrznej pętli w pętli zagnieżdżonej (uwaga: $N_{outermost}$ dotyczy wyłącznie najbardziej zewnętrznej pętli gniazda i nie uwzględnia w żaden sposób liczby iteracji pętli zagnieżdżonych w najbardziej zewnętrznej pętli gniazda),

num_threads – liczba wątków wykonujących program (tutaj: analizowaną pętlę programową).

Wartość zmiennej *X3* determinuje maksymalną liczbę iteracji do wykonania przez pojedynczy wątek OpenMP w najbardziej zewnętrznej pętli w pętli zagnieżdżonej, a przez to ma wpływ na wartość *Df* oraz na wartość *X2*. Dlatego też, przy wyznaczaniu *Df* oraz *X2* należy przyjąć, że liczba iteracji wykonywanych w najbardziej zewnętrznej pętli w pętli zagnieżdżonej jest dana równaniem (28):

$$\left\lceil \frac{N_{outermost}}{num\_threads \times X3} \right\rceil \times X3 \quad (28)$$

3.1.2.4. Wyznaczenie wartości zmiennej *X4*

Zmienna *X4* modelu (16) (patrz strona 32) to liczba wątków OpenMP wykonujących program (tutaj: analizowaną pętlę programową). Wartość przyjmowana przez zmienną *X4* może być zmieniana przez programistę.
Z uwagi na to, że w ramach niniejszej pracy do badań eksperymentalnych wykorzystano procesor Intel Core 2 Quad Q6600 o 4 rdzeniach, założono, że zmienna *X4* może przyjmować następujące wartości: 2 lub 3 lub 4.

3.2. Wyznaczenie wartości parametrów modeli szczególnych

W zaproponowanym w niniejszej pracy modelu ogólnym (model (16)) występują następujące parametry: *a1, a2, a3, a4.* Parametry te wskazują jak – w danym środowisku sprzętowo-programowym[6] (nazywanym w dalszej części pracy „środowiskiem") – czynniki wyrażone poszczególnymi zmiennymi niezależnymi modelu ogólnego wpływają na czas wykonania programu, wyrażony zmienną zależną modelu ogólnego. Oznacza to, że poprzez nadanie przedmiotowym parametrom modelu ogólnego odpowiednich wartości, można dostosować go do specyfiki konkretnego środowiska, tworząc dedykowany dla tego środowiska model szczególny.

Podstawowym problemem, który się tu pojawia, jest jak wyznaczyć charakterystyczne dla danego środowiska wartości parametrów *a1, a2, a3, a4.*

Planowanym przeznaczeniem modelu ogólnego (16) jest zastosowanie go jako narzędzia usprawniającego kompilację iteracyjną i pozwalającego przez to skrócić czas wykonania kompilacji iteracyjnej. Z uwagi na planowane przeznaczenie modelu ogólnego, przyjęty sposób wyznaczenia wartości parametrów modeli szczególnych powinien mieć charakter metody, którą można zastosować dla dowolnego środowiska. Na podstawie tych przesłanek, podjęto decyzję o wyznaczaniu wartości parametrów *a1, a2, a3, a4* dla danego środowiska drogą analizy statystycznej danych empirycznych, zebranych w środowisku w metodyczny sposób.

Na potrzeby zebrania wspomnianych danych empirycznych, opracowano dedykowane do tego celu pętle wzorcowe.

Opracowane pętle wzorcowe muszą spełniać wszystkie, przedstawione we wcześniejszych rozdziałach, założenia modelu ogólnego zaproponowanego w niniejszej pracy. Spośród tych założeń, najistotniejszymi – z punktu widzenia ich wpływu na specyficzne właściwości pętli wzorcowych – są założenia dotyczące wyznaczania wartości odcisku danych. Dlatego też, w oparciu o przedstawiony w rozdziale 3.1.1 sposób wyznaczania odcisku danych, utworzono pętle wzorcowe cechujące się następującymi właściwościami:

- pętla wzorcowa 1: występowanie ponownego użycia danych przy braku interferencji,
- pętla wzorcowa 2: występowanie ponownego użycia danych oraz interferencji.

Z uwagi na to, że w praktyce rzadko spotykane są pętle, w których nie występuje ponowne użycie danych, sytuację taką pominięto w opracowanych pętlach wzorcowych.

Ostateczna postać pętli wzorcowej 1 i ostateczna postać pętli wzorcowej 2 są zależne od rodzaju odwzorowania, występującego w pamięci podręcznej L1 oraz L2 systemu komputerowego.

W ramach niniejszej pracy, wszystkie badania eksperymentalne zostały przeprowadzone w środowisku z pamięcią podręczną z odwzorowaniem sekcyjno-skojarzeniowym (patrz Tabela

[6] Przez "środowisko sprzętowo-programowe" rozumie się tutaj parametry systemu komputerowego, wymienione w Tabeli 7.

7 na stronie 48) – dlatego też, przy ustaleniu ostatecznej postaci pętli wzorcowych, uwzględniono ten właśnie rodzaj odwzorowania.

W przypadku odwzorowania sekcyjno-skojarzeniowego, jeżeli w programie (tutaj: pętli programowej) nie występuje czasowe ponowne użycie danych a łączna, deklarowana w kodzie źródłowym ilość danych do przetworzenia w programie nie przekracza rozmiaru pamięci podręcznej L2, prawdopodobieństwo wystąpienia interferencji jest pomijalnie małe. Wynika to stąd, że gdy jedynym występującym w programie rodzajem ponownego użycia danych jest przestrzenne ponowne użycie danych – dane pobrane do pamięci podręcznej są w pełni wykorzystywane przez procesor w bardzo krótkim czasie od momentu ich pobrania do pamięci podręcznej. Tak więc, z samej natury przestrzennego ponownego użycia danych, prawdopodobieństwo zastąpienia pobranych do pamięci podręcznej danych, które mają być jeszcze użyte w programie, innymi danymi, jest niewielkie; przy pamięci podręcznej z odwzorowaniem sekcyjno-skojarzeniowym prawdopodobieństwo to staje się pomijalnie małe. W konsekwencji, przyjęto założenie, że brak interferencji jest tożsamy z niewystępowaniem w programie czasowego ponownego użycia danych.

Jeżeli w programie występuje czasowe ponowne użycie danych, wówczas prawdopodobieństwo zastąpienia pobranych do pamięci podręcznej danych, które mają być jeszcze użyte w programie, innymi danymi, będzie tym większe, im większe czasowe ponowne użycie danych znajdujących się w pamięci podręcznej. Wynika to stąd, że przy czasowym ponownym użyciu danych, dane pobrane do pamięci podręcznej są wykorzystywane przez procesor (tj. ponownie używane) przed długi czas od momentu pobrania wspomnianych danych do pamięci podręcznej. W konsekwencji, przyjęto założenie, że występowanie interferencji jest tożsame z występowaniem w programie czasowego ponownego użycia danych.

W oparciu o przedstawione powyżej wnioski odnośnie wpływu czasowego ponownego użycie danych na występowanie interferencji, pętlę wzorcową 1 oraz pętlę wzorcową 2 skonkretyzowano w następujący sposób:

Tabela 8 Konkretyzacja pętli wzorcowych

Założenia	**Konkretyzacja**

Pętla wzorcowa 1
Występowanie ponownego użycia danych przy braku interferencji
	Pętla nonInterf
	`int ma[N][N],mb[N][N],mc[N][N],md[N][N],me[N][N];`
	`int i, j, N;`
	
	`for (i = 0; i <= N-1; i++) {`
	`  for (j= 0; j <= N-1; j++) {`
	`  ma[i][j]=1;`
	`  mb[i][j]=mc[i][j]+md[i][j]*me[i][j];`
	`  } //endfor j`
	`} //endfor i`

Założenia	Konkretyzacja

Pętla wzorcowa 2
Występowanie ponownego użycia danych oraz interferencji
	Pętla matmul
	int ma[N][N],mb[N][N],mc[N][N];
	int i, j, k, r, N;
	
	for (i = 0; i <= N-1; i++) {
	for (k=0; k <= N-1; k++) {
	r = ma[i][k];
	for (j=0; j <= N-1; j++){
	mc[i][j]= mc[i][j] + r*mb[k][j];
	} //endfor j
	} //endfor k
	} //endfor i

Należy tu podkreślić, że przyjęty sposób konkretyzacji pętli wzorcowych (patrz Tabela 8) nie jest jedynym słusznym sposobem konkretyzacji pętli wzorcowej 1 oraz pętli wzorcowej 2; jest to jeden z wielu możliwych sposób konkretyzacji. Poprzez przyjęcie innej konkretyzacji pętli wzorcowej 1 oraz pętli wzorcowej 2 można uzyskać modele szczególne o innych przestrzeniach dziedziny, niż modele szczególne zbudowane w oparciu o pętle wzorcowe wskazane w Tabeli 8 (to zagadnienie jest omawiane w rozdziale 3.3), co dowodzi dużej uniwersalności opracowanego podejścia. W związku z powyższym, przyjęty sposób konkretyzacji pętli wzorcowych – poprzez pętle: nonInterf oraz matmul – ma charakter przykładowej konkretyzacji, przyjętej na potrzeby utworzenia przykładowych modeli szczególnych dla modelu ogólnego (16).

Dla przyjętych pętli wzorcowych, nonInterf oraz matmul, zebrano, w środowisku scharakteryzowanym w Tabeli 7 (patrz strona 48), dane empiryczne na podstawie których wyznaczono drogą analizy regresji wartości parametrów *a1, a2, a3, a4*. W celu uzyskania reprezentatywnych dla analizowanego środowiska danych empirycznych, założono, że dla każdej z pętli wzorcowych:

1/ Łączny rozmiar danych, przetwarzanych w pętli, nie przekracza rozmiaru pamięci podręcznej L2 dostępnej dla procesora.
2/ Różnica względna pomiędzy średnią liczbą porcji iteracji, przypadającą na pojedynczy wątek OpenMP, a maksymalną liczbą porcji iteracji, przypadającą na pojedynczy wątek OpenMP przy danym sposobie przydziału iteracji do wątków nie przekracza 50% (wartość przyjęta a priori).

Założenie 1/ oraz założenie 2/ mają charakter ogólny, tzn. obowiązują dla dowolnej konkretyzacji pętli wzorcowej 1 oraz pętli wzorcowej 2 i dla dowolnego środowiska, w którym zastosowano pamięć podręczną z odwzorowaniem sekcyjno-skojarzeniowym (a nie tylko dla arbitralnie przyjętych pętli wzorcowych nonInterf oraz matmul oraz środowiska, przedstawionego w Tabeli 7).

Uwagi odnośnie założenia 1/:

W przypadku większości pętli programowych, niemal całe realizowane w pętli przetwarzanie danych to przetwarzanie danych zapisanych w zmiennych tablicowych. Dlatego też, założenie dotyczące relacji łącznego rozmiaru danych przetwarzanych w pętli do rozmiaru pamięci podręcznej L2 sprowadza się do dobrania odpowiedniego rozmiaru macierzy przetwarzanych w pętli. Rozmiar ten jest tożsamy z rozmiarem problemu rozwiązywanego w pętli (w przypadku pętli nonInterf i matmul, wspomniane rozmiary są wyrażone zmienną *N*).

Założenie 1/ wyrażono następującym równaniem:

$$\lambda = \frac{total\_matrix\_size(N)}{L2\_per\_processor} \leq 1 \tag{29}$$

gdzie:

total_matrix_size(*N*) – łączny rozmiar danych zapisanych w zmiennych tablicowych przetwarzanych w pętli *(wielkość wyrażona w bajtach),* dla rozmiaru problemu *N*,

L2_per_processor – rozmiar pamięci podręcznej L2, dostępnej dla pojedynczego procesora *(wielkość wyrażona w bajtach).*

W pętlach nonInterf i matmul są przetwarzane dane typu całkowitego, natomiast dla środowiska scharakteryzowanego w Tabeli 7 jeden element typu całkowitego zajmuje w pamięci głównej 4 bajty. W związku z tym, wartość *total_matrix_size*(*N*) dla pętli nonInterf oraz matmul wyniesie, odpowiednio:

$$total\_matrix\_size(N)_{nonInterf} = 5 \times N^2 \times size\_int = 5 \times N^2 \times 4 = 20 \times N^2 \tag{30}$$

$$total\_matrix\_size(N)_{matmul} = 3 \times N^2 \times size\_int = 3 \times N^2 \times 4 = 12 \times N^2 \tag{31}$$

gdzie:

size_int – rozmiar jednego elementu typu całkowitego, wyrażony w bajtach.

Dla środowiska przedstawionego w Tabeli 7, wartość *L2_per_processor* wynosi 4 194 304 B. Uwzględniając przedstawione powyżej uwagi odnośnie założenia 1/, dla pętli wzorcowych nonInterf i matmul przyjęto następujące wartości *N* (patrz Tabela 9):

Tabela 9 Zależność pomiędzy N, total_matrix_size(N) oraz λ dla pętli wzorcowych matmul oraz nonInterf

Pętla wzorcowa	***N***	***total_matrix_size*(*N*)**	***λ***
matmul	100	120 000	0,0286
	200	480 000	0,1144
	300	1 080 000	0,2575
	400	1 920 000	0,4578
	500	3 000 000	0,7153
nonInterf	100	200 000	0,0477
	200	800 000	0,1907
	300	1 800 000	0,4292

Pętla wzorcowa	*N*	*total_matrix_size*(*N*)	*λ*
	400	3 200 000	0,7629

Uwagi odnośnie założenia 2/:

Przyjęcie założenia 2/ wynika stąd, że przy wymuszonym sposobie przydziału iteracji do wątków, mogą wystąpić istotne różnice pomiędzy łącznymi liczbami iteracji wykonywanych przez poszczególne wątki.
Przedmiotowe założenie wyrażono następującym równaniem:

$$\theta = \frac{no\_chunks_{\max} - no\_chunks_{average}}{no\_chunks_{average}} \le 0{,}5 \tag{32}$$

gdzie:

no_chunks_{max} – maksymalna liczba porcji iteracji, przypadająca na pojedynczy wątek OpenMP, przy danym sposobie przydziału iteracji do wątków,

$no\_chunks_{average}$ – średnia liczba porcji iteracji, przypadająca na pojedynczy wątek OpenMP, przy danym sposobie przydziału iteracji do wątków.

Wartości no_chunks_{max}, $no\_chunks_{average}$ można wyznaczyć przy pomocy następujących równań:

$$no\_chunks_{\max} = \left\lceil \frac{N_{outermost}}{num\_threads \times X3} \right\rceil \tag{33}$$

$$no\_chunks_{average} = \frac{N_{outermost}}{num\_threads \times X3} \tag{34}$$

gdzie:

$N_{outermost}$ – liczba iteracji do wykonania w najbardziej zewnętrznej pętli w pętli zagnieżdżonej (uwaga: $N_{outermost}$ dotyczy wyłącznie najbardziej zewnętrznej pętli gniazda i nie uwzględnia w żaden sposób liczby iteracji pętli zagnieżdżonych w najbardziej zewnętrznej pętli gniazda),

num_threads – liczba wątków wykonujących program (tutaj: analizowaną pętlę programową),

X3 – maksymalna, dla danego sposobu przydziału iteracji do wątków OpenMP, wielkość pojedynczej porcji iteracji przydzielonej do wykonania przez wątek.

Uwzględniając wnioski wypływające z założenia 1/ oraz przedstawione powyżej uwagi odnośnie założenia 2/, na potrzeby zebrania danych empirycznych, będących podstawą do

wyznaczenia drogą analizy regresji wartości parametrów *a1, a2, a3, a4* dla obranych pętli wzorcowych, przyjęto próby przedstawione w Tabelach: 10 i 11.

Tabela 10 Próba do wyznaczenia wartości parametrów a1, a2, a3, a4 – dla pętli wzorcowej nonInterf

Lp.	Pętla wzorcowa	*N*	Liczba wątków wykonujących pętlę	Maksymalna wielkość pojedynczej porcji iteracji przydzielonej do wykonania przez wątek	Sposób przydziału iteracji do wątków
1	nonInterf	100	2	10	wymuszony
2	nonInterf	100	2	20	wymuszony
3	nonInterf	100	3	10	wymuszony
4	nonInterf	100	3	20	wymuszony
5	nonInterf	100	4	10	wymuszony
6	nonInterf	200	2	10	wymuszony
7	nonInterf	200	2	20	wymuszony
8	nonInterf	200	3	10	wymuszony
9	nonInterf	200	3	20	wymuszony
10	nonInterf	200	4	10	wymuszony
11	nonInterf	200	4	20	wymuszony
12	nonInterf	300	2	10	wymuszony
13	nonInterf	300	2	20	wymuszony
14	nonInterf	300	3	10	wymuszony
15	nonInterf	300	3	20	wymuszony
16	nonInterf	300	4	10	wymuszony
17	nonInterf	300	4	20	wymuszony
18	nonInterf	400	2	10	wymuszony
19	nonInterf	400	2	20	wymuszony
20	nonInterf	400	3	10	wymuszony
21	nonInterf	400	3	20	wymuszony
22	nonInterf	400	4	10	wymuszony
23	nonInterf	400	4	20	wymuszony

Tabela 11 Próba do wyznaczenia wartości parametrów a1, a2, a3, a4 – dla pętli wzorcowej matmul

Lp.	Pętla wzorcowa	*N*	Liczba wątków wykonujących pętlę	Maksymalna wielkość pojedynczej porcji iteracji przydzielonej do wykonania przez wątek	Sposób przydziału iteracji do wątków
1	matmul	100	2	10	wymuszony
2	matmul	100	2	50	domyślny
3	matmul	100	2	20	wymuszony
4	matmul	100	3	34	domyślny
5	matmul	100	3	20	wymuszony
6	matmul	100	3	10	wymuszony
7	matmul	100	4	10	wymuszony
8	matmul	100	4	25	domyślny
9	matmul	200	2	100	domyślny
10	matmul	200	2	10	wymuszony
11	matmul	200	2	20	wymuszony

Lp.	Pętla wzorcowa	*N*	Liczba wątków wykonujących pętlę	Maksymalna wielkość pojedynczej porcji iteracji przydzielonej do wykonania przez wątek	Sposób przydziału iteracji do wątków
12	matmul	200	3	67	domyślny
13	matmul	200	3	10	wymuszony
14	matmul	200	3	20	wymuszony
15	matmul	200	4	50	domyślny
16	matmul	200	4	10	wymuszony
17	matmul	200	4	20	wymuszony
18	matmul	300	2	150	domyślny
19	matmul	300	2	10	wymuszony
20	matmul	300	2	20	wymuszony
21	matmul	300	3	100	domyślny
22	matmul	300	3	10	wymuszony
23	matmul	300	3	20	wymuszony
24	matmul	300	4	75	domyślny
25	matmul	300	4	10	wymuszony
26	matmul	300	4	20	wymuszony
27	matmul	400	2	200	domyślny
28	matmul	400	2	10	wymuszony
29	matmul	400	2	20	wymuszony
30	matmul	400	3	134	domyślny
31	matmul	400	3	10	wymuszony
32	matmul	400	3	20	wymuszony
33	matmul	400	4	100	domyślny
34	matmul	400	4	10	wymuszony
35	matmul	400	4	20	wymuszony
36	matmul	500	2	250	domyślny
37	matmul	500	2	10	wymuszony
38	matmul	500	2	20	wymuszony
39	matmul	500	3	167	domyślny
40	matmul	500	3	10	wymuszony
41	matmul	500	3	20	wymuszony
42	matmul	500	4	125	domyślny
43	matmul	500	4	10	wymuszony
44	matmul	500	4	20	wymuszony

Pętle wzorcowe nonInterf i matmul zostały wykonane w środowisku scharakteryzowanym w Tabeli 7 (patrz strona 48), dla konfiguracji wskazanych odpowiednio w Tabeli 10 oraz Tabeli 11.

Dla każdej z ww. konfiguracji, zmierzono – przy pomocy funkcji clock() zdefiniowanej w bibliotece standardowej języka C – czas procesora (wyrażony liczbą taktów zegara procesora) spędzony na wykonywaniu pętli wzorcowej przez wszystkie wątki programu.

Każda konfiguracja została wykonana w środowisku, przedstawionym w Tabeli 7, pięciokrotnie; wyniki pięciokrotnego wykonania uśredniono na potrzeby dalszej analizy statystycznej.

W ramach pojedynczego wykonania danej konfiguracji, pętla wzorcowa była wykonywana *p*-krotnie, przy czym *p* było dobierane eksperymentalnie tak, by rzeczywisty czas *p*-krotnego wykonania pętli wzorcowej wynosił co najmniej minutę, natomiast na potrzeby dalszej analizy statystycznej wyniki *p*-krotnego wykonania uśredniono. Zastosowanie takiego podejścia było podyktowane tym, że w przypadku jednokrotnych wykonań pętli wzorcowej, w wielu przypadkach różnice pomiędzy zmierzonymi przy pomocy funkcji clock() czasami procesora były rzędu 100%, co w oczywisty sposób wykluczało zasadność przeprowadzania analizy statystycznej takich wyników. Przy *p*-krotnym wykonaniu pętli wzorcowej, różnice pomiędzy zmierzonymi przy pomocy funkcji clock() czasami procesora były rzędu kilku – kilkunastu procent.
Opisany powyżej sposób wykonania pomiarów ma charakter ogólny, tzn. obowiązuje dla dowolnej konkretyzacji pętli wzorcowej 1 oraz pętli wzorcowej 2 i dla dowolnego środowiska, w którym zastosowano pamięć podręczną z odwzorowaniem sekcyjno-skojarzeniowym (a nie tylko dla arbitralnie przyjętych pętli wzorcowych nonInterf oraz matmul oraz środowiska, przedstawionego w Tabeli 7).

Na podstawie analizy regresji zmierzonych w powyższy sposób czasów procesora oraz wartości zmiennych *X1, X2, X3, X4* obliczonych wg zaproponowanego w pracy modelu (16) (patrz strona 32), dla każdej z przyjętych pętli wzorcowych wyznaczono wartości parametrów *a1, a2, a3, a4* modelu ogólnego (16). Dane empiryczne oraz wartości zmiennych *X1, X2, X3, X4,* dla których przeprowadzono analizę regresji, przedstawiono w rozdziale 4.
Wyniki analizy regresji przedstawiają się następująco:

- dla pętli nonInterf:

  a1 = -0,325431
  a2 = 0,675172
  a3 = -0,082602
  a4 = 0,981967
  R^2 = 0,999958

- dla pętli matmul:

  a1 = -0,298695
  a2 = 0,623738
  a3 = 0,014426
  a4 = 0,962976
  R^2 = 0,999951

W związku z tym, dla pętli wzorcowej nonInterf model ogólny (16) przyjmuje postać modelu szczególnego:

$$Yt = X1^{-0,325431} \times X2^{0,675172} \times X3^{-0,082602} \times X4^{0,981967} \quad (35)$$

Natomiast dla pętli wzorcowej matmul model ogólny (16) przyjmuje postać modelu szczególnego:

$$Yt = X1^{-0,298695} \times X2^{0,623738} \times X3^{0,014426} \times X4^{0,962976} \quad (36)$$

Model wyrażony równaniem (35), opracowany dla środowiska przedstawionego w Tabeli 7, jest adekwatny dla pętli nonInterf oraz dla innych pętli, spełniających założenia przedstawione w rozdziale 3.3, w których występuje ponowne użycie danych przy braku interferencji. Model ten w dalszej części pracy będzie określany jako model (35) lub model szczególny (35).

Model wyrażony równaniem (36), opracowany dla środowiska przedstawionego w Tabeli 7, jest adekwatny dla pętli matmul oraz dla innych pętli, spełniających założenia przedstawione w rozdziale 3.3, w których występuje ponowne użycie danych oraz interferencja. Model ten w dalszej części pracy będzie określany jako model (36) lub model szczególny (36).
Model (36) jest także adekwatny dla pętli, w których zastosowano blokowanie. Wynika to stąd, że blokowanie ma sens praktyczny jedynie wówczas, gdy w poddanej blokowaniu pętli zachodzi czasowe ponowne użycie danych [24], co – jak wyjaśniono w rozdziale 3.2 – dla środowiska, w którym zastosowano pamięć podręczną z odwzorowaniem sekcyjno-skojarzeniowym, jest tożsame z występowaniem interferencji.

W modelu (36) uwzględniono zarówno interferencję własną jak i interferencję krzyżową.
Uwzględnienie interferencji własnej w modelu (36) wynika z przedstawionego przez Lam [56] sposobu wyznaczania granicznego rozmiaru bloku kwadratowego nie powodującego interferencji własnej. Przedstawiony przez Lam algorytm (patrz rozdział 3.1.1) został wykorzystany dla pętli wzorcowej matmul, w celu oceny czy dla *N* wskazanych w Tabeli 11 zachodzi interferencja własna. Dla wszystkich *N*, wymienionych w Tabeli 11, stwierdzono występowanie interferencji własnej.

Należy tu podkreślić, że zarówno model szczególny (35) jak i model szczególny (36) są przykładowymi modelami szczególnymi, utworzonymi na podstawie modelu ogólnego (16) dla przyjętego środowiska i przyjętych postaci pętli wzorcowych. Dzięki wykorzystaniu koncepcji pętli wzorcowych o ściśle zdefiniowanych cechach charakterystycznych, ale wielu różnych możliwych konkretyzacjach, oraz z uwagi na wyznaczanie wartości parametrów *a1, a2, a3, a4* modeli szczególnych w oparciu o zbierane w metodyczny sposób dane empiryczne, charakterystyczne dla przyjętego środowiska – model ogólny (16) można wykorzystać dla bardzo szerokiego spektrum przypadków, tworząc dla nich dedykowane modele szczególne.

3.3. Zakres stosowalności modeli szczególnych

Przedstawiony poniżej problem zakresu stosowalności modeli szczególnych dotyczy wszystkich modeli szczególnych utworzonych dla modelu ogólnego (16). Dla lepszego zobrazowania, problem ten został omówiony na przykładzie przyjętych w pracy, przykładowych modeli szczególnych – modelu (35) oraz modelu (36).

Wyprowadzenie modelu (35) oraz modelu (36) jest oparte o dane uzyskane drogą empiryczną. Na podstawie tych danych, stosując analizę regresji, uzyskano funkcje, które z bardzo dobrą dokładnością oddają przebieg analizowanego zjawiska odpowiednio dla pętli nonInterf oraz matmul – w przestrzeniach, których granice wynikają z parametrów prób przyjętych do wyprowadzenia modelu (35) oraz modelu (36) (patrz Tabela 10 i Tabela 11).

Bardzo dobre dopasowanie modelu (35) oraz modelu (36) do wskazanych powyżej danych empirycznych nie daje żadnych podstaw do stwierdzenia, drogą teoretyczną, jaka będzie dokładność ww. modeli poza granicami, wynikającymi z parametrów prób przyjętych do wyprowadzenia tych modeli. W związku z tym, oszacowania przeprowadzane przy pomocy modelu (35) oraz modelu (36) powinny być przeprowadzane wyłącznie w przestrzeniach, których granice wynikają z parametrów prób przyjętych do wyprowadzenia modelu (35) oraz modelu (36).

Powyższe założenie o charakterze ogólnym ma zastosowanie zarówno do pętli wzorcowych (nonInterf, matmul) dla których wyprowadzono model (35) oraz model (36), jak i do wszystkich innych pętli (nazywanych dalej pętlami niewzorcowymi), wskazanych w rozdziale 3.2, dla których te modele można zastosować. W praktyce, przedmiotowe założenie ogólne przekłada się na następujące założenia szczegółowe:

1/ Wartość λ, wyznaczona wg równania (29) (patrz strona 56) dla pętli niewzorcowej, nie może przekraczać wartości λ wyznaczonych dla odpowiadającej jej pętli wzorcowej (patrz Tabela 9, strona 56). Założenie to jest wyrażone następującym równaniem:

$$\lambda_{min}(referenceLoop) \leq \lambda \leq \lambda_{max}(referenceLoop) \tag{37}$$

gdzie:

λ – wartość λ wyznaczona dla pętli niewzorcowej,
$\lambda_{min}(referenceLoop)$ – najmniejsza wartość λ dla pętli wzorcowej,
$\lambda_{max}(referenceLoop)$ – największa wartość λ dla pętli wzorcowej.

2/ Wartość θ, wyznaczona wg równania (32) (patrz strona 57) dla pętli niewzorcowej, nie może przekraczać 0,5.

3/ Faktyczny czas wykonania analizowanej pętli w środowisku przedstawionym w Tabeli 7 musi być tego samego rzędu wielkości, co czas wykonania odpowiadającej jej pętli wzorcowej. Założenie to jest wyrażone następującym równaniem:

$$\gamma_{min}(referenceLoop) \leq \gamma \leq \gamma_{max}(referenceLoop) \tag{38}$$

gdzie:

γ – faktyczny czas procesora spędzony na wykonywaniu analizowanej pętli przez wszystkie wątki programu, wyrażony liczbą taktów zegara procesora,

$\gamma_{min}(referenceLoop)$ – najkrótszy (spośród czasów zmierzonych dla całej próby przedstawionej w rozdziale 3.2) faktyczny czas procesora spędzony na wykonywaniu pętli wzorcowej przez wszystkie wątki programu, wyrażony liczbą taktów zegara procesora,

$\gamma_{max}(referenceLoop)$ – najdłuższy (spośród czasów zmierzonych dla całej próby przedstawionej w rozdziale 3.2) faktyczny czas procesora spędzony na wykonywaniu pętli wzorcowej przez wszystkie wątki programu, wyrażony liczbą taktów zegara procesora.

Wprowadzenie założenia wyrażonego równaniem (38) wynika stąd, że mogą istnieć pętle, dla których założenie 1/ oraz 2/ są spełnione, niemniej pętle te, pomimo podobieństwa do pętli wzorcowych pod względem występowania ponownego użycia danych oraz pod względem występowania interferencji, pod innymi względami są na tyle różne od pętli wzorcowych, że ich czasy wykonania są zupełnie innego rzędu wielkości, niż czasy wykonania pętli wzorcowych. W takiej sytuacji, oszacowania wykonane przy pomocy modelu (35) lub modelu (36) mogą być obarczone bardzo wysokim, niemożliwym do zaakceptowania błędem.

Dane empiryczne na podstawie których wyznaczono wartości $\gamma_{min}(referenceLoop)$, $\gamma_{max}(referenceLoop)$ dla poszczególnych pętli wzorcowych przedstawiono w rozdziale 4.2.1.

Wartości $\gamma_{min}(referenceLoop)$, $\gamma_{max}(referenceLoop)$ dla poszczególnych pętli wzorcowych są następujące:

$\gamma_{min}(nonInterf)$ = 151,47; $\gamma_{max}(nonInterf)$ = 2 287,73
$\gamma_{min}(matmul)$ = 9 752,33; $\gamma_{max}(matmul)$ = 1 206 440,00

Po uwzględnieniu omówionych powyżej założeń, zakres stosowalności modelu (35) oraz modelu (36) przedstawia się następująco:

Tabela 12 Zakres stosowalności modelu (35)

Model (35)	$Yt = X1^{-0,325431} \times X2^{0,675172} \times X3^{-0,082602} \times X4^{0,981967}$
Pętla wzorcowa	nonInterf
Pętle niewzorcowe, dla których można zastosować model	Pętle, w których występuje ponowne użycie danych przy braku interferencji *(Brak interferencji jest tożsamy z niewystępowaniem w pętli czasowego ponownego użycia danych.)*
Zakres wartości λ	[0,047684; 0,762939]
Zakres wartości θ	[0; 0,5]

Zakres wartości γ	[151,47; 2 287,73]

Tabela 13 Zakres stosowalności modelu (36)

Model (36)	$Yt = X1^{-0,298695} \times X2^{0,623738} \times X3^{0,014426} \times X4^{0,962976}$
Pętla wzorcowa	matmul
Pętle niewzorcowe, dla których można zastosować model	Pętle, w których występuje ponowne użycie danych oraz interferencja, w tym pętle z blokowaniem *(Występowanie interferencji jest tożsame z występowaniem w pętli czasowego ponownego użycia danych.)*
Zakres wartości λ	[0,028610; 0,715256]
Zakres wartości θ	[0; 0,5]
Zakres wartości γ	[9 752,33; 1 206 440,00]

3.4. Ocena jakości modelu ogólnego i modeli szczególnych

Ocena jakości modelu ogólnego, którego opracowanie jest celem niniejszej pracy, sprowadza się do oceny jakości modeli szczególnych utworzonych w oparciu o model ogólny. Z uwagi na fakt, że dla zaproponowanego w pracy modelu ogólnego (16) (patrz strona 32) utworzono dwa przykładowe modele szczególne, tj. model (35) (patrz strona 60) oraz model (36) (patrz strona 61), ocena jakości zostanie przedstawiona na przykładzie modelu (35) oraz modelu (36). Dla obu ww. modeli szczególnych, ocena jakości jest przeprowadzana w dwóch aspektach: jakościowym i ilościowym.

Celem oceny jakości omawianych modeli szczególnych w aspekcie jakościowym jest stwierdzenie czy model (35) oraz model (36) pozwalają z zadawalającą dokładnością oszacować czas wykonania pętli niewzorcowych, spełniających założenia odnośnie zakresu stosowalności ww. modeli. Przez analogię do definicji dokładności pomiaru, wg której dokładność pomiaru jest pojęciem jakościowym, określającym zgodność wyniku pomiaru z wartością rzeczywistą [45] [79] – w niniejszej pracy, przez dokładność oszacowania rozumie się zgodność wyniku oszacowania z wartością zmierzoną empirycznie. Tak rozumiana dokładność oszacowania jest uznawana za zadawalającą, jeżeli dla danego rozmiaru problemu rozwiązywanego w pętli, kierunek zmian zmierzonych czasów wykonania poszczególnych wersji danej pętli programowej oraz kierunek zmian odpowiadających im czasów oszacowanych wg opracowanych modeli szczególnych jest identyczny.

Celem oceny jakości omawianych modeli szczególnych w aspekcie ilościowym jest określenie wielkości błędów oszacowań dokonanych wg tych modeli.

Oba aspekty oceny jakości omawianych modeli szczególnych znajdują swoje odzwierciedlenie w wynikach weryfikacji hipotezy o normalności rozkładu reszt uzyskanych dla tych modeli. Jeżeli, w rezultacie przeprowadzonego testu Kołmogorowa-Smirnowa, rozkład reszt uzyskanych dla ww. modeli będzie można uznać za normalny – to ocena

jakości ww. modeli w aspekcie jakościowym i ilościowym będzie pozytywna. (Zagadnienie normalności rozkładu reszt i jego znaczenie dla problematyki niniejszej pracy zostało omówione na stronie 31.)

a) Ocena jakości modelu (35) oraz modelu (36) w aspekcie jakościowym

Przewidywanym obszarem zastosowania modelu (35) oraz modelu (36) jest kompilacja iteracyjna. Celem zastosowania ww. modeli szczególnych jest skrócenie czasu wykonania kompilacji iteracyjnej, poprzez wytypowanie – drogą oszacowań przeprowadzonych przy pomocy przedmiotowych modeli – spośród różnych, analizowanych postaci kodu źródłowego danego programu tych postaci, których czas wykonania w docelowym środowisku wykonania programu będzie krótszy, niż czas wykonania pozostałych, analizowanych postaci kodu źródłowego programu.
W związku z powyższym, ocena jakości opracowanych modeli szczególnych w aspekcie jakościowym sprowadza się do wykazania, że omawiane modele realizują ten cel, co oznacza, że w aspekcie jakościowym opracowane modele szczególne pozwalają na osiągnięcie zadawalającej dokładności oszacowania. Aby wykazać powyższe, należy wykonać następujące czynności:

1/ Wygenerować kilka pętli niewzorcowych, spełniających założenia opracowanego modelu szczególnego podlegającego ocenie.
2/ Dla każdej z przyjętych pętli niewzorcowych, wybrać kilka różnych rozmiarów (*N*) problemu, rozwiązywanego w ramach pętli.
3/ Dla poszczególnych, przyjętych pętli niewzorcowych i dla każdej wybranej dla nich wartości *N*, wygenerować kilka różnych postaci kodu źródłowego programu (postaci te powinny różnić się liczbą wątków, wykonujących program oraz sposobem przydziału iteracji pętli do poszczególnych wątków).
4/ Każdą postać kodu źródłowego, o której mowa w punkcie 3/, wykonać w docelowym środowisku wykonania programu (tj. tu w środowisku, przedstawionym w Tabeli 7 – patrz strona 48).
5/ Dla poszczególnych, przyjętych pętli niewzorcowych i dla każdej wybranej dla nich wartości *N*, uszeregować postaci kodu źródłowego, o których mowa w punkcie 3/, od postaci o najdłuższym (w przeliczeniu na jeden wątek) czasie wykonania w docelowym środowisku wykonania do postaci o najkrótszym (w przeliczeniu na jeden wątek) czasie wykonania w docelowym środowisku wykonania.
6/ Dla każdego ciągu, o którym mowa w punkcie 5/, wyznaczyć (wystarczy graficznie) funkcję trendu liniowego.
7/ Dla każdego ciągu, o którym mowa w punkcie 5/, wyznaczyć:
   7a/ wartości *Yt* (w przeliczeniu na jeden wątek) wg opracowanego modelu, odpowiadające kolejnym wyrazom ciągu,
   7b/ funkcję trendu liniowego dla wartości *Yt*, o których mowa w punkcie 7a/ (wystarczy to zrobić graficznie).

8/ Dla każdego ciągu, o którym mowa w punkcie 5/, porównać funkcję trendu, o której mowa w punkcie 6/ z funkcją trendu, o której mowa w punkcie 7b/. Jeżeli w każdym przypadku obie funkcje trendów są funkcjami malejącymi, oceniany model w aspekcie jakościowym spełnia stawiane mu wymagania, co kończy procedurę wykazania właściwej jakości w aspekcie jakościowym.

b) Ocena jakości modelu (35) oraz modelu (36) w aspekcie ilościowym

Ocena jakości modelu (35) oraz modelu (36) w aspekcie ilościowym sprowadza się do oceny wielkości błędów oszacowań, dokonanych wg ww. modeli szczególnych.
Należy tu podkreślić, że ocena jakości wspomnianych modeli szczególnych w aspekcie ilościowym jest drugorzędna w stosunku do oceny w aspekcie jakościowym, co wynika z przewidywanego zastosowania modeli. Z punktu widzenia zastosowania obu modeli szczególnych, jest bowiem kluczowe, by modele te pozwalały uszeregować różne postaci kodów źródłowych danej pętli malejąco wg czasu wykonania, bez konieczności wykonywania przedmiotowych kodów źródłowych w docelowym środowisku. Jeżeli modele pozwalają dokonać takiego uszeregowania, to uzyskane błędy oszacowań mają mniejsze znaczenie.

Do oceny jakości opracowanych modeli szczególnych w aspekcie ilościowym wykorzystano wielkości, wyznaczone na podstawie znajomości błędu względnego oszacowania.

Błąd względny oszacowania, o którym mowa powyżej, jest wyznaczany w następujący sposób:

$$\delta_x = \left| \frac{x - x_0}{x_0} \right| \times 100\% \tag{39}$$

gdzie:

δ_x – błąd względny oszacowania,
x – wartość oszacowana,
x_0 – wartość wyznaczona eksperymentalnie.

Na podstawie wyników badań eksperymentalnych, przedstawionych w rozdziale 4.2.2, zaproponowano a posteriori następujące wartości graniczne dla kryteriów oceny jakości modeli szczególnych w aspekcie ilościowym:

- Średnia z błędów względnych oszacowania dla wszystkich postaci kodu źródłowego, przyjętych dla danej pętli niewzorcowej i dla danego N, nie przekracza 55 punktów procentowych.
- Maksymalny błąd względny oszacowania dla wszystkich postaci kodu źródłowego, przyjętych dla danej pętli niewzorcowej i dla danego N, nie przekracza 65 punktów procentowych.

Przedstawione powyżej wartości graniczne zostały zaproponowane w oparciu o analizę wyników empirycznych uzyskanych dla 307 różnych kodów źródłowych. Z uwagi na bardzo

dużą liczbę i różnorodność przeanalizowanych kodów źródłowych, zaproponowane wartości graniczne można przyjąć za adekwatne do oceny jakości (w aspekcie ilościowym) oszacowań uzyskanych przy pomocy opracowanych modeli szczególnych także dla innych pętli niewzorcowych, niż pętle niewzorcowe które wykorzystano do uzyskania wartości granicznych.

c) Źródła błędów przy ocenie jakości modelu (35) oraz modelu (36)

Z racji tego, że zarówno model (35) jak i model (36) są uproszczonym obrazem pewnego fragmentu rzeczywistości (co z kolei jest immanentną cechą wszystkich modeli), oszacowania otrzymane przy pomocy ww. modeli szczególnych są jedynie pewnym przybliżeniem faktycznych wartości szacowanych wielkości. Dlatego też czynniki, determinujące jakość oszacowań otrzymanych przy pomocy modelu (35) oraz modelu (36) dla pętli niewzorcowych, są głównym źródłem błędów popełnianych przy ocenie jakości wspomnianych modeli szczególnych zarówno w aspekcie jakościowym jak i w aspekcie ilościowym. Lista czynników, o których mowa powyżej, przedstawia się następująco:

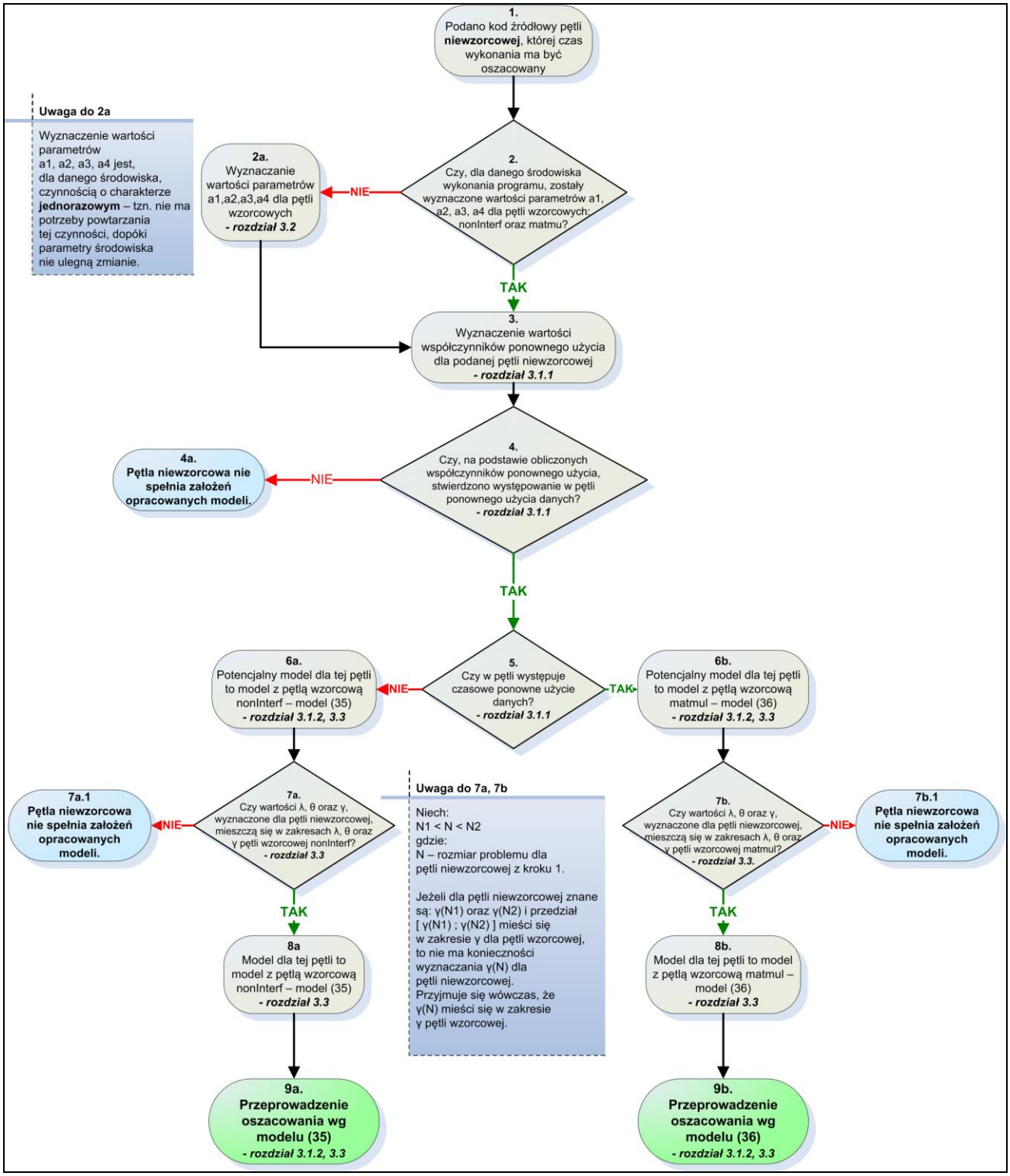
- Błąd, popełniany przy oszacowaniu wartości odcisku danych (*Df*).
  Z uwagi na to, że odcisk danych jest wyznaczany w sposób szacunkowy, a samo oszacowanie jest oparte na licznych uproszczeniach, bez których przeprowadzenie oszacowania wartości odcisku danych nie byłoby możliwe – oszacowana wartość odcisku danych może znacząco różnić się od wartości rzeczywistej. Wyznaczenie odcisku danych w sposób szacunkowy oraz złożona procedura wyznaczenia jego wartości powodują, że błąd, popełniany przy oszacowaniu wartości odcisku danych, bardzo silnie wpływa na jakość oszacowań otrzymanych przy pomocy modelu (35) oraz modelu (36).
- Błąd, popełniany przy przypisaniu wag poszczególnym operacjom wykonywanym przez pojedynczy wątek.
  W tym przypadku, błąd wynika z tego, że przyjęte wagi są pewnym uśrednieniem danych empirycznych. Jednakże z uwagi na to, że dla wszystkich pętli (wzorcowych i niewzorcowych) przyjęto identyczne wagi, przedmiotowy błąd w umiarkowanym stopniu wpływa na jakość oszacowań otrzymanych przy pomocy modelu (35) oraz modelu (36).
- Błąd, popełniany w związku z przyjęciem, dla pętli niewzorcowych, wartości parametrów *a1, a2, a3, a4* wyznaczonych dla pętli wzorcowych.
  W tym przypadku, błąd wynika z tego, że dla pętli niewzorcowych przyjmuje się a priori wartości *a1, a2, a3, a4*, które zostały wyznaczone dla pętli wzorcowych na podstawie danych empirycznych, stosując analizę regresji. Oznacza to, że już dla samych pętli wzorcowych, wyznaczone wartości *a1, a2, a3, a4* są obarczone pewnym błędem; błąd ten może być znacząco większy gdy tak wyznaczone wartości *a1, a2, a3, a4* stosuje się dla pętli niewzorcowych i z tego względu, silnie wpływa na jakość oszacowań otrzymanych przy pomocy modelu (35) oraz modelu (36).

3.5. Stosowanie modelu ogólnego i modeli szczególnych w praktyce

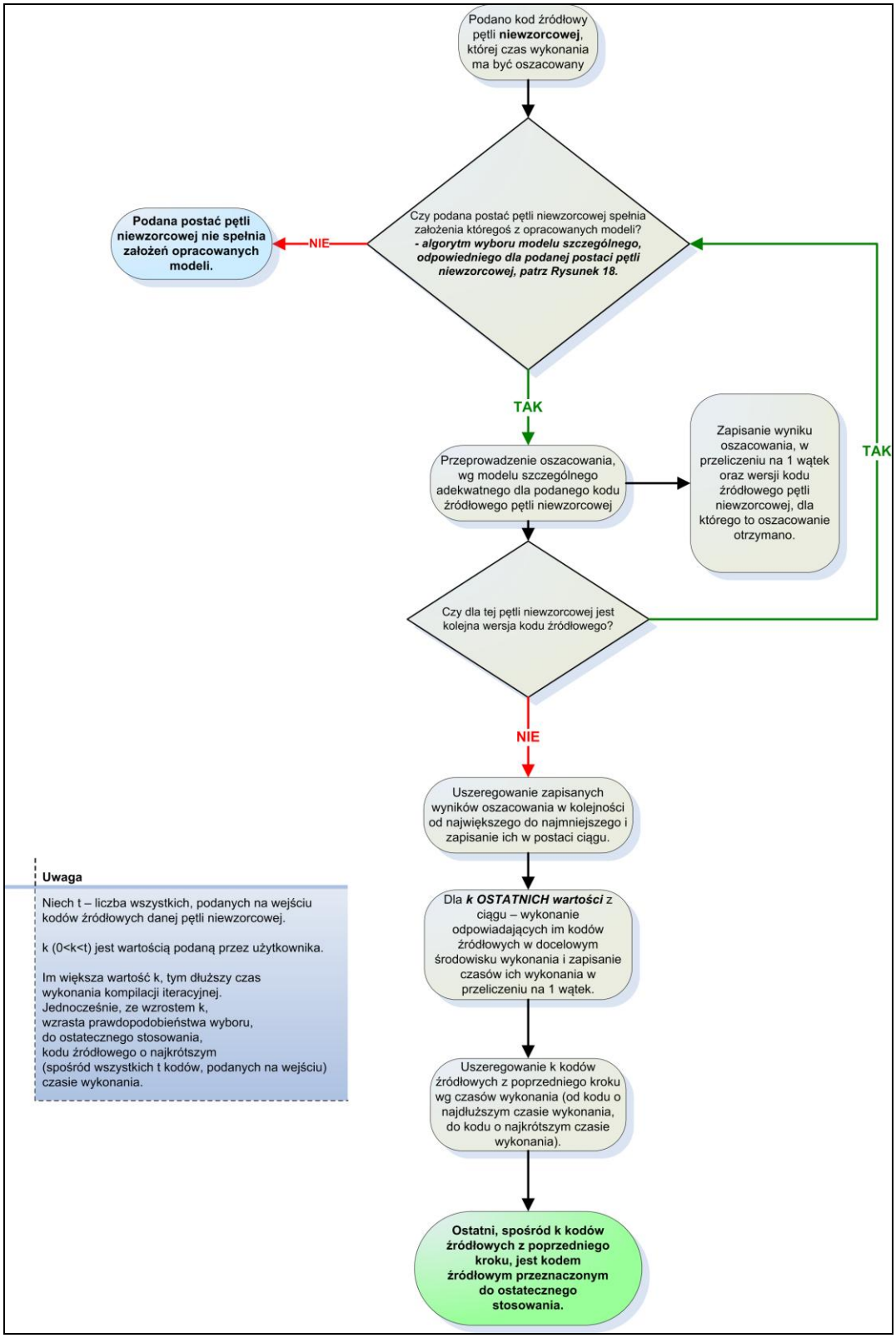
Stosowanie modelu ogólnego, którego opracowanie jest celem niniejszej pracy, sprowadza się do wykorzystania go do budowy modeli szczególnych dla konkretnych środowisk wykonania programu, z uwzględnieniem (poprzez przyjętą konkretyzację pętli wzorcowych) cech specyficznych dla pętli niewzorcowych, dla których ww. modele szczególne będą stosowane. Tak utworzone modele szczególne można wykorzystać w kompilacji iteracyjnej, celem skrócenia czasu jej wykonania.

Na Rysunku 18 przedstawiono – na przykładzie modeli szczególnych (tj. modelu (35) oraz modelu (36)) opracowanych dla pętli nonInterf, matmul i środowiska scharakteryzowanego w Tabeli 7 (patrz strona 48) – w jaki sposób dokonać wyboru modelu szczególnego, adekwatnego dla pętli niewzorcowej, której czas wykonania ma być oszacowany.

Na Rysunku 19 przedstawiono przykład zastosowania modeli szczególnych w kompilacji iteracyjnej. Procedura przedstawiona na Rysunku 19 nie została zaimplementowana w kompilatorze iteracyjnym, z uwagi na bardzo dużą złożoność niezbędnej implementacji oraz fakt, że przedmiotowa implementacja nie stanowiła celu niniejszej pracy. W oparciu o wyniki badań empirycznych przedstawione w Tabelach 17 ÷ 64 (patrz strony 93 ÷ 140), dokonano jednak oszacowania potencjalnego zysku czasowego, jaki można by osiągnąć stosując w praktyce procedurę przedstawioną na Rysunku 19. Wyniki oszacowania przedstawiono w Tabeli 69 (patrz strona 150) i omówiono na stronie 151.



Rysunek 18 Algorytm wyboru modelu, adekwatnego dla danej postaci pętli niewzorcowej – na przykładzie modeli szczególnych (35) i (36)



Rysunek 19 Przykład zastosowania modeli szczególnych w kompilacji iteracyjnej

3.6. Przykład zastosowania opracowanych modeli szczególnych do oszacowania czasu wykonania pętli niewzorcowej

Przykład przedstawiony poniżej ilustruje zastosowanie opracowanych modeli szczególnych – tj. modelu szczególnego (35), wyznaczonego dla pętli wzorcowej nonInterf oraz dla modelu szczególnego (36), wyznaczonego dla pętli wzorcowej matmul – do oszacowania czasu wykonania pętli niewzorcowej.

Przykład:
Należy oszacować czas wykonania pętli CG_cg_3 w środowisku przedstawionym w Tabeli 7 (patrz strona 48), dla wielkości problemu N = 75 000, przy założeniu, że pętla CG_cg_3 będzie wykonywana w sposób równoległy – przez 2 wątki OpenMP. Iteracje są przydzielane do wątków w sposób wymuszony, w porcjach liczących 15 000 iteracji.
Kod źródłowy pętli CG_cg_3 (wersja sekwencyjna, dla dowolnego N) przedstawia się następująco:

Kod źródłowy pętli CG_cg_3

```
int q[N], z[N], r[N], x[N], p[N];

for(j = 0; j < N; j++){
    q[j] = 0.0;
    z[j] = 0.0;
    r[j] = x[j];
    p[j] = r[j];
} //endfor j
```

Rozwiązanie:
Do rozwiązania powyższego problemu wykorzystano algorytm, przedstawiony na Rysunku 18 (patrz strona 69).
Kod źródłowy pętli CG_cg_3 po zrównolegleniu przedstawia się następująco:

Kod źródłowy pętli CG_cg_3 po zrównolegleniu

```
int N = 75 000;
int NUM_THREADS = 2;
int CHUNK_SIZE=15 000;

int q[N], z[N], r[N], x[N], p[N];

omp_set_num_threads(NUM_THREADS); //ustawienie liczby wątków


#pragma omp parallel for private(j) schedule(static,CHUNK_SIZE)
for(j = 0; j < N; j++){
    q[j] = 0.0;
    z[j] = 0.0;
    r[j] = x[j];
    p[j] = r[j];
} //endfor j
```

Z uwagi na to, że znany jest kod źródłowy pętli niewzorcowej (krok 1 ww. algorytmu) oraz wartości parametrów *a1, a2, a3, a4* dla pętli wzorcowych nonInterf i matmul dla środowiska przedstawionego w Tabeli 7 (krok 2 ww. algorytmu), można przystąpić do wykonania kroku 3 algorytmu, czyli wyznaczenia wartości współczynników ponownego użycia.

Pętla CG_cg_3 nie jest pętlą zagnieżdżoną. Zmienną sterującą tej pętli jest zmienna *j*, łączna liczba iteracji pętli *j* (do wykonania łącznie przez wszystkie wątki) wynosi 75 000. W pętli występują następujące, różne odwołania do pamięci głównej: *q*[*j*], *z*[*j*], *r*[*j*], *x*[*j*], *p*[*j*]. Wszystkie odwołania są wyrażone zmiennymi tablicowymi o jednym indeksie. Z racji tego, że jedyny indeks każdej ze zmiennych występujących w kodzie źródłowym jest zależny od *j* – w pętli CG_cg_3 zachodzi przestrzenne ponowne użycie danych, natomiast nie zachodzi czasowe ponowne użycie danych.
Rozmiar wiersza pamięci podręcznej L1 dla środowiska przedstawionego w Tabeli 7 to 64 bajty. Jeden element typu całkowitego zajmuje w pamięci głównej 4 bajty. W związku z tym, w jednym wierszu pamięci podręcznej mieści się $L = 64/4 = 16$ elementów typu całkowitego. Dla wszystkich rozpatrywanych odwołań a_{sj}, czyli współczynnik stojący przy zmiennej *j* w ostatnim indeksie odwołania, przyjmuje wartość 1.
W związku z powyższym, współczynniki ponownego użycia obliczone dla występujących w pętli *j* odwołań wyrażonych poszczególnymi zmiennymi przyjmują następujące wartości:

$R_{j(temp)}(q) = 1$
$R_{j(temp)}(z) = 1$
$R_{j(temp)}(r) = 1$
$R_{j(temp)}(x) = 1$

$R_{j(temp)}(p) = 1$

$R_{j(spat)}(q) = \max(1, L/a_{sj}) = \max(1, 16/1) = \max(1, 16) = 16$
$R_{j(spat)}(z) = \max(1, L/a_{sj}) = \max(1, 16/1) = \max(1, 16) = 16$
$R_{j(spat)}(r) = \max(1, L/a_{sj}) = \max(1, 16/1) = \max(1, 16) = 16$
$R_{j(spat)}(x) = \max(1, L/a_{sj}) = \max(1, 16/1) = \max(1, 16) = 16$
$R_{j(spat)}(p) = \max(1, L/a_{sj}) = \max(1, 16/1) = \max(1, 16) = 16$

Na podstawie równania (17) (patrz strona 42):

$R_j(q) = R_{j(spat)}(q) = 16$
$R_j(z) = R_{j(spat)}(z) = 16$
$R_j(r) = R_{j(spat)}(r) = 16$
$R_j(x) = R_{j(spat)}(x) = 16$
$R_j(p) = R_{j(spat)}(p) = 16$

Wśród wartości współczynników własnego-ponownego użycia wyznaczonych dla pętli CG_cg_3, występują wartości większe od 1, co świadczy o występowaniu ponownego użycia danych w pętli CG_cg_3 (krok 4 algorytmu przedstawionego na Rysunku 18 – patrz strona 69). Ponieważ wszystkie współczynniki własnego-czasowego ponownego użycia, wyznaczone dla pętli CG_cg_3 przyjmują wartość 1, w pętli CG_cg_3 nie zachodzi czasowe ponowne użycie danych (krok 5 algorytmu przedstawionego na Rysunku 18). W związku z tym, potencjalnym modelem szczególnym dla analizowanej sytuacji jest model z pętlą wzorcową nonInterf (krok 6a algorytmu przedstawionego na Rysunku 18).

Zgodnie z krokiem 7a algorytmu przedstawionego na Rysunku 18, należy wyznaczyć wartości λ, θ, γ dla pętli CG_cg_3 i sprawdzić czy mieszczą się w zakresach wartości λ, θ, γ wyznaczonych dla pętli wzorcowej nonInterf.

Wartość λ jest wyznaczana na podstawie równania (29) (patrz strona 56), przyjmując następujące podstawienia:

$total\_matrix\_size(N) = 5 \times N \times size\_int = 5 \times N \times 4 = 20 \times N$
$L2\_per\_processor = 4\ 194\ 304$ B

gdzie:

size_int – rozmiar jednego elementu typu całkowitego, wyrażony w bajtach.

Ponieważ:

$total\_matrix\_size(N{=}75\ 000) = 1\ 500\ 000$ B

więc:

$\lambda = 1\ 500\ 000 / 4\ 194\ 304 = 0{,}3576$

Wyznaczona wartość λ mieści się w zakresie:

$[\lambda_{min}(nonInterf) ; \lambda_{max}(nonInterf)] = [0{,}0477; 0{,}7629]$

W celu wyznaczenia wartości *θ* (patrz równanie (32) na stronie 57), należy w pierwszej kolejności wyznaczyć wartości no_chunks_{max}, $no\_chunks_{average}$ oraz *X3*.
Zgodnie z równaniem (27) (patrz strona 52), z racji tego, że zastosowano wymuszony przydział iteracji do wątków:

X3 = 15 000.

Na podstawie równania (33) (patrz strona 57):

$$no\_chunks_{\max} = \left\lceil \frac{N_{outermost}}{num\_threads \times X3} \right\rceil = \left\lceil \frac{75000}{2 \times 15000} \right\rceil = \lceil 2{,}5 \rceil = 3$$

Na podstawie równania (34) (patrz strona 57):

$$no\_chunks_{average} = \frac{N_{outermost}}{num\_threads \times X3} = \frac{75000}{2 \times 15000} = 2{,}5$$

Na podstawie równania (32) (patrz strona 57):

$$\theta = \frac{no\_chunks_{\max} - no\_chunks_{average}}{no\_chunks_{average}} = \frac{3 - 2{,}5}{2{,}5} = 0{,}2 < 0{,}5$$

Wyznaczona wartość *θ* mieści się w zakresie [0; 0,5].

W przypadku wartości *γ* zakładamy, że mieści się ona w zakresie:

[γ_{min}(*nonInterf*); γ_{max}(*nonInterf*) = [151,47; 2 287,73].

(Wyniki pomiarów, przedstawione w Tabeli 17 na stronie 93 potwierdzają słuszność tego założenia.)

Ponieważ wartości *λ, θ, γ* dla pętli CG_cg_3 mieszczą się w zakresach wartości *λ, θ, γ* wyznaczonych dla pętli wzorcowej nonInterf – model szczególny z pętlą wzorcową nonInterf można wykorzystać do oszacowania czasu wykonania pętli CG_cg_3 (krok 8a algorytmu przedstawionego na Rysunku 18).
W celu przeprowadzenia oszacowania (krok 9a algorytmu przedstawionego na Rysunku 18), należy wyznaczyć wartości zmiennych *X1, X2, X3, X4.*

a) Wyznaczenie wartości zmiennej *X1*

Aby wyznaczyć wartość zmiennej *X1*, w pierwszej kolejności należy wyznaczyć *Df*, czyli odcisk danych dla pojedynczego wątku OpenMP.

Zgodnie z równaniem (18) (patrz strona 43), odcisk danych dla występującego w pętli *k* odwołania wyrażonego zmienną *v* wyznacza się następująco:

$$F_k(v) = \frac{N_k}{R_k(v)}$$

Analizowana pętla CG_cg_3 jest wykonywana przez 2 wątki OpenMP; iteracje są przydzielane do wątków w sposób wymuszony, w porcjach liczących 15 000 iteracji.
Zgodnie z wcześniejszymi wyliczeniami:

$$no\_chunks_{\max} = \left\lceil \frac{N_{outermost}}{num\_threads \times X3} \right\rceil = \left\lceil \frac{75\,000}{2 \times 15\,000} \right\rceil = \lceil 2{,}5 \rceil = 3$$

W związku z tym wątek, który otrzyma do wykonania no_chunks_{max}, porcji iteracji, wykona no_chunks_{max} x 15 000 = 45 000 iteracji.
Dlatego też, wyznaczając odcisk danych dla pojedynczego wątku OpenMP, przy wyznaczaniu wartości $F_j(v)$ dla poszczególnych odwołań, występujących w analizowanej pętli CG_cg_3, należy przyjąć, że N_j = 45 000.
Wówczas:

$F_j(q) = N_j / R_j(q) = 45\,000 / 16 = 2\,812{,}5$
$F_j(z) = N_j / R_j(z) = 45\,000 / 16 = 2\,812{,}5$
$F_j(r) = Nj / R_j(r) = 45\,000 / 16 = 2\,812{,}5$
$F_j(x) = N_j / R_j(x) = 45\,000 / 16 = 2\,812{,}5$
$F_j(p) = N_j / R_j(p) = 45\,000 / 16 = 2\,812{,}5$

Zgodnie z równaniem (19) (patrz strona 43), skumulowany odcisk danych wyznacza się następująco:

$$F_r^*(v) = b\prod_{i=1}^{r}\frac{N_i}{R_i(v)} = b\prod_{i=1}^{r}F_i(v)$$

Analizowana pętla CG_cg_3 nie jest pętlą zagnieżdżoną, w związku z tym, r=1 a wartości skumulowanych odcisków danych dla poszczególnych odwołań występujących w tej pętli przyjmują następujące wartości:

$F_1^*(q) = b \times F_j(q) = 64 \times 2812{,}5 = 180\,000$
$F_1^*(z) = b \times F_j(z) = 64 \times 2812{,}5 = 180\,000$
$F_1^*(r) = b \times F_j(r) = 64 \times 2812{,}5 = 180\,000$
$F_1^*(x) = b \times F_j(x) = 64 \times 2812{,}5 = 180\,000$
$F_1^*(p) = b \times F_j(p) = 64 \times 2812{,}5 = 180\,000$

Zgodnie z równaniem (20) (patrz strona 43), zbiorczy odcisk danych wyznacza się następująco:

$$F_{total} = \sum_{p=1}^{P} F_t^*(v_p)$$

Dla analizowanej pętli CG_cg_3, zbiorczy odcisk danych wyniesie:

$F_{total} = F_1^*(q) + F_1^*(z) + F_1^*(r) + F_1^*(x) + F_1^*(p) = 180\,000 \times 5 = 900\,000$

Czyli, poszukiwany odcisk danych dla pojedynczego wątku OpenMP wyniesie:

$Df = F_{total} = 900\,000$

Znając wartość *Df*, można wyznaczyć wartość zmiennej *X1*, korzystając z równania (3) (patrz strona 22). Dla przedstawionego w Tabeli 7 (patrz strona 48) środowiska sprzętowego, wartości *sL1, sL2, aL1, aL2* wynoszą:

sL1 = 32 768 B; *aL1* = 8
sL2 = 4 194 304 B; *aL2* = 16

W związku z tym, zgodnie z równaniem (3), wartość zmiennej *X1* wynosi:

$$X1 = \frac{(sL1 \times aL1 + sL2 \times aL2)}{D_f} = \frac{(32\,768 \times 8 + 4\,194\,304 \times 16)}{900000} = 74{,}8567$$

b) Wyznaczenie wartości zmiennej *X2*

Wartość zmiennej *X2* jest wyznaczana na podstawie przytoczonego poniżej równania (26) (patrz także strona 51), zastosowanego do wykonywanej przez pojedynczy wątek OpenMP najbardziej zewnętrznej pętli w pętli zagnieżdżonej:

$$o^*_{weighted\_total}(k) = \sum_{t \in T} o^*_t(k) \times w_t(k)$$

W analizowanej pętli CG_cg_3 są wykonywane 4 operacje przypisania (operacja przypisania ma wagę 1 – patrz rozdział 3.1.2.2). Maksymalna liczba iteracji, wykonywanych przez pojedynczy wątek OpenMP wynosi no_chunks_{max} x 15 000 = 45 000 iteracji.
W związku z powyższym, wartość zmiennej *X2* wynosi:

$X2 = o^*_{weighted\_total}(j)$ = 45 000 x 4 x 1 = 180 000

c) Wyznaczenie wartości zmiennej *X3* i *X4*

Zgodnie z wcześniejszymi wyliczeniami przeprowadzonymi dla niniejszego przykładu, wartość zmiennej *X3* wynosi 15 000. Natomiast wartość zmiennej *X4* wynika bezpośrednio z przyjętych założeń i wynosi 2.

Podstawiając powyższe wartości, tj.:

X1 = 74,8567
X2 = 180 000
X3 = 15 000
X4 = 2

do równania modelu szczególnego (35) (patrz strona 60), otrzymuje się:

$$Yt = X1^{-0{,}325431} \times X2^{0{,}675172} \times X3^{-0{,}082602} \times X4^{0{,}981967} =$$
$$= 74{,}8567^{-0{,}325431} \times 180\,000^{0{,}675172} \times 15\,000^{-0{,}082602} \times 2^{0{,}981967} = 774{,}34$$

Wartość *Yt*, po przeliczeniu na 1 wątek na podstawie równania (41) (patrz strona 90), wynosi:

$$Yt(per\_thread) = \frac{Yt}{X4^{a4}} = 392{,}04$$

3.7. Prace pokrewne

Model ogólny, którego opracowanie jest celem niniejszej pracy, powinien umożliwiać oszacowanie czasu wykonania programów o określonych właściwościach, zapewniając zadawalającą dokładność oszacowania[7]. Model ogólny, spełniający powyższe wymagania, umożliwi dokonanie wstępnej selekcji tych postaci (transformacji) kodu źródłowego programu, które w danym środowisku sprzętowym będą miały najlepsze rokowania odnośnie czasu wykonania – co oznacza, że przedmiotowy model ogólny będzie adekwatny do zastosowania w kompilacji iteracyjnej w celach optymalizacyjnych oraz że będzie można go również wykorzystać w kompilacji optymalizującej.

Ideą kompilacji optymalizującej jest zastosowanie podczas kompilacji programu różnego rodzaju transformacji, które przekształcają program do postaci semantycznie równoważnych o korzystniejszych – niż w przypadku niezastosowania transformacji – wybranych właściwościach programu. Kompilacja optymalizująca jest zorientowana na poprawę konkretnych właściwości programu, zazwyczaj czasu wykonania programu. Inne właściwości, często optymalizowane w ramach kompilacji optymalizującej, to: rozmiar postaci wykonywalnej programu, przewidywane zużycie energii elektrycznej podczas wykonywania programu [25].

Jednym z podstawowych problemów kompilacji optymalizującej jest wybór, spośród transformacji możliwych do zastosowania dla danego programu, sekwencji transformacji gwarantujących, że dla postaci programu uzyskanej wskutek zastosowania przedmiotowej sekwencji transformacji optymalizowana właściwość programu znajduje się w punkcie optimum globalnego [6]. Przyjmując, że optymalizowaną właściwością programu jest czas wykonania programu, praktycznym aspektem problemu wyboru optymalnej sekwencji transformacji jest utworzenie modelu/metody pozwalających na oszacowanie czasu wykonania różnych, semantycznie równoważnych postaci danego programu [73] [84].
Najprostszy przypadek to program wykonywany w sposób sekwencyjny na maszynie jednoprocesorowej o architekturze von Neumanna – w tym przypadku, czas wykonania programu jest zależny liniowo od liczby wykonanych instrukcji, a więc można go oszacować wykorzystując funkcję liniową jednej zmiennej. Jednakże, wraz z rozwojem sprzętu komputerowego i pojawieniem się: hierarchii pamięci, różnych poziomów równoległości (równoległość na poziomie rozkazu, wątku, procesu), przewidywania rozgałęzień i spekulacji [41] [57] – ten prosty model liniowy okazał się niewystarczający do oszacowania czasu wykonania programu. Problemem, który wciąż nie znalazł rozwiązania, jest: jaka jest dokładna postać (tj. zmienne oraz typ zależności funkcyjnej) funkcji, pozwalającej wyznaczyć

[7] Przyjęta w niniejszej pracy definicja "zadawalającej dokładności oszacowania" jest przedstawiona w rozdziale 1.2.

czas wykonania programu oraz jakie są wartości parametrów występujących w tej funkcji [9] [84].

W ramach prób choćby częściowego rozwiązania tego problemu, naukowcy z różnych ośrodków przedstawili wiele różnych propozycji metod oszacowania czasu wykonania programu, wykorzystując do ich budowy m.in. następujące podejścia [9] [84] i ich różne kombinacje:

- analizę regresji,
- specjalnie zaprojektowane algorytmy (np. zliczające liczbę instrukcji określonego rodzaju lub dokonujące analizy zależności danych w celu wyznaczenia najdłuższej ścieżki wykonywania programu), ukierunkowane na oszacowanie czasu wykonania programu,
- analizę porównawczą zorientowaną na wskazanie – bez szacowania czasu wykonania – która z dwóch podanych wersji programu zostanie wykonana szybciej w danym środowisku sprzętowym.

Poniżej przedstawiono, na tle modelu autorskiego, przykłady zaproponowanych w literaturze przedmiotu metod i modeli utworzonych w oparciu o ww. podejścia.

W [11] przedstawiono metodę opracowania modeli do oszacowania czasu wykonania programu, dedykowanych dla konkretnych aplikacji równoległych i rozproszonych. Zaproponowana metoda jest oparta na analizie regresji liniowej i zakłada, że dla każdego programu i środowiska wykonania, na podstawie zebranych w tym środowisku danych empirycznych, jest tworzony dedykowany model do oszacowania czasu wykonania programu. Zmienne niezależne modeli dedykowanych są wyłaniane na podstawie wyników testów istotności t-Studenta, badających istotność wpływu potencjalnych zmiennych niezależnych na zmienną zależną w oparciu o dane empiryczne zebrane w środowisku wykonania. Za potencjalne zmienne niezależne przyjmuje się specyficzne właściwości rozpatrywanego programu (np. rząd macierzy dla programu realizującego mnożenie macierzy) oraz specyficzne właściwości środowiska wykonania programu (np. liczba wątków wykonujących program w sposób równoległy). Wybór postaci analitycznej modelu jest dokonywany na podstawie wykresu rozrzutu, wykorzystywanego do wstępnego określenia rodzaju zależności funkcyjnej, oraz wartości współczynnika determinacji R^2. Opracowane w ten sposób modele charakteryzują się bardzo wysokim dopasowaniem do danych empirycznych i, jako takie, są dobrym narzędziem do szacowania czasu wykonania programu w analizowanych przedziałach zmienności zmiennych niezależnych – niemniej, z uwagi na duże nakłady czasowe związane z ich opracowaniem (dla każdego programu, jest opracowywany osobny model), podejście przedstawione w [11] jest nieadekwatne do realizacji celu postawionego w niniejszej pracy. W przeciwieństwie do podejścia przedstawionego w [11], modele szczególne utworzone na bazie autorskiego modelu ogólnego mogą być stosowane – bez konieczności dokonywania jakichkolwiek adaptacji – dla wielu różnych programów, spełniających założenia modeli szczególnych.

W [58] przedstawiono metodę umożliwiającą oszacowanie i minimalizację najgorszego możliwego czasu wykonania programu (ang. *WCET, worst-case execution time* [49] [69]), przeznaczoną do wykorzystania w kompilatorach optymalizujących w celu oceny czy po zastosowaniu w trakcie kompilacji konkretnego programu określonej transformacji (optymalizacji) czas wykonania uzyskanej postaci wykonywalnej programu będzie krótszy, niż w przypadku niezastosowania przedmiotowej transformacji. Przyjęto założenie, że dla każdej możliwej transformacji jest tworzony osobny, dedykowany model. Podstawowa lista zmiennych niezależnych dla tworzonego modelu obejmuje 73 cechy charakterystyczne programu; w zależności od potrzeb, w modelu mogą być uwzględnione wszystkie lub tylko wybrane cechy z listy. Zmienną zależną modelu jest WCET. Dobór postaci analitycznej oraz parametrów modelu jest dokonywany w oparciu o jedną z 6 popularnie stosowanych metod uczenia maszynowego (drzewa decyzyjne – ang. *Decision Trees* [8] [54] [72], lasy losowe – ang. *Random Forests* [17] [42] [54], liniowa maszyna wektorów nośnych – ang. *Linear Support Vector Machines (SVM)* [28] [48] [54], maszyna wektorów nośnych z radialnymi funkcjami jądrowymi – ang. *SVMs with RBF kernel* [44] [74] [75], metoda K-najbliższych sąsiadów – ang. *k-Nearest Neighbor (kNN)* [2] [33] [67], naiwny klasyfikator Bayesa – ang. *naive Bayes* [18] [54] [93]). Ostateczna postać poszukiwanego modelu jest wyznaczana w oparciu o tę metodę, dla której, po przeprowadzeniu optymalizacji wartości parametrów specyficznych dla danej metody przy pomocy dedykowanego algorytmu genetycznego, uzyskano największą redukcję WCET dla zbioru uczącego. Z uwagi na to, że opisana metoda jest ukierunkowana na oszacowanie czasu wykonania programu dla konkretnej transformacji oraz czasochłonność procesu uczenia dla pojedynczej transformacji, podejście przedstawione w [58] jest nieadekwatne do realizacji celu postawionego w niniejszej pracy. W przeciwieństwie do podejścia przedstawionego w [58], modele szczególne utworzone na bazie autorskiego modelu ogólnego mogą być stosowane dla dowolnej transformacji, dla której są spełnione założenia modeli szczególnych.

W [65] przedstawiono predyktor turniejowy. Jest to model, który dla danych wejściowych – charakterystyk wydajnościowych programu oraz dwóch różnych sekwencji transformacji optymalizujących – wskazuje tę sekwencję transformacji, po zastosowaniu której czas wykonania programu będzie krótszy, niż w przypadku zastosowania drugiej sekwencji transformacji z analizowanej pary. Charakterystyki wydajnościowe programu, o których mowa powyżej, mają charakter dynamiczny, tzn. opisują zachowanie programu podczas jego wykonywania i są zbierane przy pomocy specjalistycznego oprogramowania do profilowania (w tym celu, wykorzystano HPCToolkit [43] [80]). Postać analityczna modelu jest wyznaczana przy pomocy regresji liniowej. Do konstrukcji zbioru uczącego i testowego na potrzeby wyznaczenia i weryfikacji modelu wykorzystuje się walidację krzyżową [31] [37] [53] typu leave-one-out.

W modelu zaproponowanym w [65] zmienne niezależne to charakterystyki programu wyznaczane dynamicznie (tj. w trakcie wykonywania programu) [19] [94] – w praktyce, przekłada się to na konieczność przeprowadzenia profilowania przy każdorazowym

wykorzystaniu modelu dla nowego programu. W modelu autorskim w zmiennych niezależnych uwzględniono charakterystyki programu wyznaczane statycznie, tj. na podstawie samej postaci kodu źródłowego programu, bez konieczności wykonywania go. Zastosowany w [65] sposób konstrukcji zbioru uczącego powoduje, że opracowanie modelu jest bardziej czasochłonne, niż w przypadku modelu autorskiego. Z tych względów, podejście przedstawione w [65] nie jest adekwatne do realizacji celu postawionego w niniejszej pracy.

W [66] przedstawiono propozycję modelu który, w oparciu o charakterystyki programu wyznaczane statycznie, pozwala ocenić prawdopodobieństwo tego, że po zastosowaniu w trakcie kompilacji konkretnej metody programu określonej transformacji (optymalizacji) czas wykonania uzyskanej postaci wykonywalnej metody programu będzie krótszy, niż w przypadku niezastosowania przedmiotowej transformacji. Postać analityczna modelu jest wyznaczana przy pomocy regresji logistycznej, natomiast do konstrukcji zbioru uczącego i testowego na potrzeby wyznaczenia i weryfikacji modelu wykorzystuje się walidację krzyżową typu leave-one-out. Podobne podejście przedstawiono w [20].
Z uwagi na to, że podejście przedstawione w [66] i [20] jest ukierunkowane na ocenę skuteczności konkretnej transformacji, jak również czasochłonność procesu uczenia dla pojedynczej transformacji, podejście to jest nieadekwatne do realizacji celu postawionego w niniejszej pracy.

Reasumując, należy stwierdzić, że wszystkie rozwiązania, zaprezentowane w przedstawionych tu pracach pokrewnych, dostarczają dobrych narzędzi do szacowania czasu wykonania programu w analizowanych przedziałach zmienności zmiennych niezależnych, jednakże kosztem czasochłonnego procesu budowy modeli oraz wąskiego zakresu stosowalności uzyskanych modeli (modele są dedykowane dla konkretnych programów lub konkretnych transformacji optymalizujących). Z uwagi na te ich właściwości, omówione rozwiązania pokrewne nie są adekwatne do realizacji celu postawionego w niniejszej pracy. Wykorzystując model ogólny, zaproponowany w rozwiązaniu autorskim, można utworzyć modele szczególne o szerokim zakresie stosowalności (modele są dedykowane dla programów o konkretnych właściwościach w zakresie ponownego użycia danych), pozwalające uzyskać zadawalającą dokładność oszacowania (co wykazano w rozdziale 4 i 5) przy wykorzystaniu prostszego i mniej czasochłonnego – niż w przypadku pozostałych rozwiązań przedstawionych w niniejszym rozdziale – procesu uczącego. Rozwiązanie autorskie, z uwagi na jego właściwości, jest podejściem najbardziej właściwym do realizacji celu postawionego w niniejszej pracy.

3.8. Podsumowanie

W rozdziale 3 przedstawiono propozycję autorskiego modelu, pozwalającego na oszacowanie czasu wykonania programów zrównoleglonych oraz sposób, w jaki model ten został opracowany. Aby przybliżyć kolejne etapy opracowywania finalnego modelu, w

rozdziale zostały przedstawione wyniki autorskich analiz, obejmujących następujące zagadnienia:

- dobór zmiennych modelu,
- wybór postaci modelu,
- wyznaczenie wartości parametrów modelu,
- ocena jakości modelu,
- zastosowanie modelu w praktyce.

Zaproponowany model porównano z rozwiązaniami, przedstawionymi w pracach pokrewnych – wszystkie te rozwiązania okazały się nieadekwatne do realizacji celu niniejszej pracy.

4. BADANIA EKSPERYMENTALNE

Celem badań eksperymentalnych, przeprowadzonych w ramach niniejszej pracy, było:

- wyznaczenie wartości parametrów *a1, a2, a3, a4* dla przykładowych modeli szczególnych (tj. dla modelu (35) oraz dla modelu (36)),
- przeprowadzenie oceny jakości przykładowych modeli szczególnych (tj. modelu (35) oraz modelu (36)) w aspekcie jakościowym i ilościowym.

Sposób przeprowadzenia badań eksperymentalnych został przedstawiony w rozdziale 4.1. Uzyskane wyniki badań eksperymentalnych zostały przedstawione w rozdziale 4.2, natomiast ich omówienie oraz wnioski zostały przedstawione w rozdziale 5.

4.1. Sposób przeprowadzenia badań eksperymentalnych

Badania eksperymentalne zostały przeprowadzone w środowisku przedstawionym w Tabeli 7 (patrz strona 48).

Wyznaczenie wartości parametrów *a1, a2, a3, a4* dla przykładowych modeli szczególnych (tj. dla modelu (35) – patrz strona 60, oraz dla modelu (36) – patrz strona 61) zostało przeprowadzone dla prób, wskazanych w Tabelach: 10 i 11 (patrz strona 58).

Na potrzeby przeprowadzenia oceny jakości przykładowych modeli szczególnych (tj. modelu (35) oraz modelu (36)) w aspekcie jakościowym i ilościowym oszacowano, przy pomocy modelu (35) oraz modelu (36), czasy wykonania pętli niewzorcowych, wybranych z zestawu testów referencyjnych NAS Parallel Benchmarks [40] [63] (wersja 3.3, dla OpenMP) i porównano uzyskane tą drogą oszacowania z czasami wykonania przyjętych pętli niewzorcowych w środowisku przedstawionym w Tabeli 7. Pomiar czasów wykonania przyjętych pętli niewzorcowych przeprowadzono wg podejścia, przedstawionego w rozdziale 3.2 (patrz strona 59). Kody źródłowe wszystkich pętli, które były obiektem badań, zapisano w języku C.

Wspomniane powyżej NAS Parallel Benchmarks (w dalszej części pracy określane skrótowo jako NPB) to zbiór testów referencyjnych opracowanych przez NASA Advanced Supercomputing Division, mających postać programów komputerowych i przeznaczonych do oceny wydajności komputerów równoległych.
Na potrzeby przeprowadzenia badań eksperymentalnych niniejszej pracy, jako pętle niewzorcowe przyjęto wybrane pętle wyselekcjonowane z zestawu NPB w oparciu o następujące kryteria (rozpatrywane łącznie):

- Jeżeli pętla jest zagnieżdżona (czyli może zawierać w sobie kolejną pętlę lub wiele pętli), to musi istnieć możliwość zrównoleglenia tej struktury na poziomie najbardziej zewnętrznej pętli całego gniazda. Jeżeli pętla nie jest zagnieżdżona, to musi istnieć możliwość jej zrównoleglenia.
- Pętla nie może być pętlą sparametryzowaną – tzn. liczba iteracji pętli musi być wyrażona liczbą (a nie parametrem, którego wartość staje się znana dopiero

w momencie uruchomienia programu). Jeżeli pętla jest zagnieżdżona – to żadna z pętli, tworzących gniazdo, nie może być pętlą sparametryzowaną.
- W pętli występuje ponowne użycie danych.
- Wszystkie występujące w pętli odwołania do pamięci głównej są realizowane poprzez zmienne skalarne lub zmienne tablicowe o indeksach afinicznych (liniowych).
- W pętli nie mogą występować instrukcje skoku (np. goto) oraz instrukcje warunkowe (np. if).

W oparciu o kryteria, przedstawione powyżej, jako pętle niewzorcowe dla oceny jakości przykładowych modeli szczególnych (tj. modelu (35) oraz modelu (36)) przyjęto następujące pętle wybrane z zestawu NPB:

- dla modelu (35):
  - ✓ CG_cg_3,
  - ✓ CG_cg_4,
  - ✓ FT_auxfnct_2,
  - ✓ LU_HP_pintgr_11,
  - ✓ MG_mg_3,
  - ✓ UA_diffuse_2,
- dla modelu (36):
  - ✓ UA_diffuse_3,
  - ✓ UA_diffuse_4,
  - ✓ UA_transfer_11,
  - ✓ UA_transfer_16,
- dla modelu (36), z blokowaniem pętli niewzorcowych :
  - ✓ UA_diffuse_3,
  - ✓ UA_diffuse_4,
  - ✓ UA_transfer_11,
  - ✓ UA_transfer_16.

Kody źródłowe ww. pętli przedstawiono w rozdziale 4.2.

4.2. Wyniki badań eksperymentalnych

W niniejszym rozdziale przedstawiono wyniki badań eksperymentalnych, przeprowadzonych w środowisku przedstawionym w Tabeli 7 (patrz strona 48).

4.2.1 Wyznaczenie wartości parametrów *a1, a2, a3, a4* dla pętli wzorcowych nonInterf oraz matmul

Wartości parametrów *a1, a2, a3, a4* dla pętli wzorcowych nonInterf oraz matmul wyznaczono w oparciu o dane, zestawione odpowiednio w Tabeli 14 (patrz strona 87) oraz Tabeli 15 (patrz strona 88).

W celu wyznaczenia wartości powyższych parametrów, dla przedmiotowych danych przeprowadzono regresję liniową. Przeprowadzenie regresji liniowej wymagało sprowadzenia przyjętego modelu ogólnego (16) (patrz strona 32), przytoczonego poniżej:

$$Yt = X1^{a1} \times X2^{a2} \times X3^{a3} \times X4^{a4}$$

do następującej, równoważnej postaci liniowej:

$$\log(Yt) = a1 \times \log(X1) + a2 \times \log(X2) + a3 \times \log(X3) + a4 \times \log(X4) \quad (40)$$

Aby wyznaczyć, dla pętli wzorcowych nonInterf oraz matmul, nieznane wartości parametrów *a1, a2, a3, a4* przeprowadzono analizę regresji dla zależności wyrażonej równaniem (40). Za rzeczywiste wartości zmiennej *Yt* przyjęto wartości *y* zmierzone w środowisku przedstawionym w Tabeli 7 dla prób przedstawionych odpowiednio w Tabeli 14 oraz Tabeli 15. Za *X1, X2, X3, X4* przyjęto wartości, wyznaczone dla prób przedstawionych odpowiednio w Tabeli 14 oraz Tabeli 15, w oparciu o podejście przedstawione w rozdziałach 3.1.2.1, 3.1.2.2, 3.1.2.3 i 3.1.2.4.

Wyniki przeprowadzonej w opisany powyżej sposób analizy regresji przedstawiają się następująco:

- dla pętli nonInterf:

  a1 = -0,325431
  a2 = 0,675172
  a3 = -0,082602
  a4 = 0,981967
  R^2 = 0,999958

  Uzyskana wartość współczynnika determinacji R^2 jest bardzo bliska jedności, co świadczy o tym, że dla pętli wzorcowej nonInterf, przyjęty model ogólny (16) bardzo dobrze odzwierciedla kształtowanie się wartości zmiennej *Yt* w zależności od wartości zmiennych niezależnych *X1, X2, X3, X4.*

  Po uwzględnieniu w równaniu modelu ogólnego (16) przedstawionych powyżej wartości parametrów *a1, a2, a3, a4,* model ogólny (16) przyjmuje postać modelu szczególnego (35) (patrz strona 60), przytoczonego poniżej:

  $$Yt = X1^{-0,325431} \times X2^{0,675172} \times X3^{-0,082602} \times X4^{0,981967}$$

- dla pętli matmul:

  $a1 = -0{,}298695$
  $a2 = 0{,}623738$
  $a3 = 0{,}014426$
  $a4 = 0{,}962976$
  $R^2 = 0{,}999951$

  Uzyskana wartość współczynnika determinacji R^2 jest bardzo bliska jedności, co świadczy o tym, że dla pętli wzorcowej matmul, przyjęty model ogólny (16) bardzo dobrze odzwierciedla kształtowanie się wartości zmiennej *Yt* w zależności od wartości zmiennych niezależnych *X1, X2, X3, X4.*
  Po uwzględnieniu w równaniu modelu ogólnego (16) przedstawionych powyżej wartości parametrów *a1, a2, a3, a4,* model ogólny (16) przyjmuje postać modelu szczególnego (36) (patrz strona 61), przytoczonego poniżej:

$$Yt = X1^{-0,298695} \times X2^{0,623738} \times X3^{0,014426} \times X4^{0,962976}$$

W oparciu o dane, zestawione w Tabeli 14 oraz Tabeli 15, przyjęto następujące wartości graniczne λ i γ dla analizowanych pętli wzorcowych:

- dla pętli nonInterf:

  $\lambda_{min}(nonInterf) = 0{,}0477$; $\lambda_{max}(nonInterf) = 0{,}7629$
  $\gamma_{min}(nonInterf) = 151{,}47$; $\gamma_{max}(nonInterf) = 2\,287{,}73$

- dla pętli matmul:

  $\lambda_{min}(matmul) = 0{,}0286$; $\lambda_{max}(matmul) = 0{,}7153$
  $\gamma_{min}(matmul) = 9\,752{,}33$; $\gamma_{max}(matmul) = 1\,206\,440{,}00$

Oznaczenia, przyjęte w Tabeli 14 oraz Tabeli 15:

N – rozmiar problemu rozwiązywanego w pętli,
Df – odcisk danych dla pojedynczego wątku OpenMP,
X1 – wielkość bezwymiarowa, określająca stosunek łącznego rozmiaru pamięci podręcznej L1 i L2, przypadającej na pojedynczy wątek OpenMP, do odcisku danych dla pojedynczego wątku OpenMP; wielkość ta jest wyrażona równaniem (3) (patrz strona 22),
X2 – łączna ważona liczba operacji przypadająca na pojedynczy wątek OpenMP,
X3 – maksymalna, dla danego sposobu przydziału iteracji do wątków OpenMP, wielkość pojedynczej porcji iteracji przydzielonej do wykonania przez wątek,
X4 – liczba wątków OpenMP wykonujących program,
γ – faktyczny (tj. zmierzony empirycznie) czas procesora spędzony na wykonywaniu pętli programowej przez wszystkie wątki programu, wyrażony liczbą taktów zegara procesora,

λ – stosunek łącznego rozmiaru danych zapisanych w zmiennych tablicowych przetwarzanych w pętli (wielkość wyrażona w bajtach), dla rozmiaru problemu N do rozmiaru pamięci podręcznej L2, dostępnej dla pojedynczego procesora (wielkość wyrażona w bajtach), wyznaczany wg równania (29) (patrz strona 56),

θ – różnica względna pomiędzy średnią liczbą porcji iteracji, przypadającą na pojedynczy wątek OpenMP, a maksymalną liczbą porcji iteracji, przypadającą na pojedynczy wątek OpenMP przy danym sposobie przydziału iteracji do wątków, wyznaczana wg równania (32) (patrz strona 57).

Tabela 14 Wyniki uzyskane dla pętli wzorcowej nonInterf

Lp.	Pętla wzorcowa	N	Sposób przydziału iteracji do wątków	Df	X1	X2	X3	X4	γ	λ	θ
1	nonInterf	100	wymuszony	100 000,00	673,71	17 500,00	10	2	151,47	0,0477	0,0000
2	nonInterf	100	wymuszony	120 000,00	561,43	21 000,00	20	2	153,92	0,0477	0,2000
3	nonInterf	100	wymuszony	80 000,00	842,14	14 000,00	10	3	169,01	0,0477	0,2000
4	nonInterf	100	wymuszony	80 000,00	842,14	14 000,00	20	3	161,20	0,0477	0,2000
5	nonInterf	100	wymuszony	60 000,00	1 122,85	10 500,00	10	4	175,27	0,0477	0,2000
6	nonInterf	200	wymuszony	400 000,00	168,43	70 000,00	10	2	574,44	0,1907	0,0000
7	nonInterf	200	wymuszony	400 000,00	168,43	70 000,00	20	2	577,34	0,1907	0,0000
8	nonInterf	200	wymuszony	280 000,00	240,61	49 000,00	10	3	588,44	0,1907	0,0500
9	nonInterf	200	wymuszony	320 000,00	210,53	56 000,00	20	3	591,46	0,1907	0,2000
10	nonInterf	200	wymuszony	200 000,00	336,86	35 000,00	10	4	603,72	0,1907	0,0000
11	nonInterf	200	wymuszony	240 000,00	280,71	42 000,00	20	4	597,76	0,1907	0,2000
12	nonInterf	300	wymuszony	900 000,00	74,86	157 500,00	10	2	1 276,64	0,4292	0,0000
13	nonInterf	300	wymuszony	960 000,00	70,18	168 000,00	20	2	1 277,80	0,4292	0,0667
14	nonInterf	300	wymuszony	600 000,00	112,29	105 000,00	10	3	1 296,96	0,4292	0,0000
15	nonInterf	300	wymuszony	600 000,00	112,29	105 000,00	20	3	1 295,28	0,4292	0,0000
16	nonInterf	300	wymuszony	480 000,00	140,36	84 000,00	10	4	1 307,20	0,4292	0,0667
17	nonInterf	300	wymuszony	480 000,00	140,36	84 000,00	20	4	1 304,96	0,4292	0,0667
18	nonInterf	400	wymuszony	1 600 000,00	42,11	280 000,00	10	2	2 258,47	0,7629	0,0000
19	nonInterf	400	wymuszony	1 600 000,00	42,11	280 000,00	20	2	2 257,47	0,7629	0,0000
20	nonInterf	400	wymuszony	1 120 000,00	60,15	196 000,00	10	3	2 275,20	0,7629	0,0500
21	nonInterf	400	wymuszony	1 120 000,00	60,15	196 000,00	20	3	2 270,27	0,7629	0,0500
22	nonInterf	400	wymuszony	800 000,00	84,21	140 000,00	10	4	2 287,73	0,7629	0,0000
23	nonInterf	400	wymuszony	800 000,00	84,21	140 000,00	20	4	2 284,07	0,7629	0,0000

W Tabeli 14 wyróżniono następujące wartości graniczne:

$\lambda_{min}(nonInterf) = 0,0477$; $\lambda_{max}(nonInterf) = 0,7629$

$\gamma_{min}(nonInterf)$ = 151,47; $\gamma_{max}(nonInterf)$ = 2 287,73

Tabela 15 Wyniki uzyskane dla pętli wzorcowej matmul

Lp.	Pętla wzorcowa	N	Sposób przydziału iteracji do wątków	Df	X1	X2	X3	X4	γ	λ	θ
1	matmul	100	wymuszony	17 788 894,66	3,79	1 255 000,00	10	2	9 774,33	0,0286	0,0000
2	matmul	100	domyślny	17 788 894,66	3,79	1 255 000,00	50	2	9 773,00	0,0286	0,0000
3	matmul	100	wymuszony	23 158 800,40	2,91	1 506 000,00	20	2	9 752,33	0,0286	0,2000
4	matmul	100	domyślny	10 422 453,29	6,46	853 400,00	34	3	9 788,33	0,0286	0,0200
5	matmul	100	wymuszony	13 005 097,66	5,18	1 004 000,00	20	3	9 760,00	0,0286	0,2000
6	matmul	100	wymuszony	13 005 097,66	5,18	1 004 000,00	10	3	9 758,33	0,0286	0,2000
7	matmul	100	wymuszony	8 822 557,56	7,64	753 000,00	10	4	9 805,33	0,0286	0,2000
8	matmul	100	domyślny	6 961 492,66	9,68	627 500,00	25	4	9 792,67	0,0286	0,0000
9	matmul	200	domyślny	415 810 462,89	0,16	10 020 000,00	100	2	77 655,00	0,1144	0,0000
10	matmul	200	wymuszony	415 810 462,89	0,16	10 020 000,00	10	2	77 557,50	0,1144	0,0000
11	matmul	200	wymuszony	415 810 462,89	0,16	10 020 000,00	20	2	77 495,00	0,1144	0,0000
12	matmul	200	domyślny	258 553 565,02	0,26	6 713 400,00	67	3	77 690,00	0,1144	0,0050
13	matmul	200	wymuszony	272 150 126,42	0,25	7 014 000,00	10	3	77 615,00	0,1144	0,0500
14	matmul	200	wymuszony	318 530 716,01	0,21	8 016 000,00	20	3	77 592,50	0,1144	0,2000
15	matmul	200	domyślny	184 464 824,39	0,37	5 010 000,00	50	4	77 702,50	0,1144	0,0000
16	matmul	200	wymuszony	184 464 824,39	0,37	5 010 000,00	10	4	77 700,00	0,1144	0,0000
17	matmul	200	wymuszony	227 424 341,91	0,30	6 012 000,00	20	4	77 652,50	0,1144	0,2000
18	matmul	300	domyślny	1 664 862 035,00	0,04	33 795 000,00	150	2	260 376,00	0,2575	0,0000
19	matmul	300	wymuszony	1 664 862 035,00	0,04	33 795 000,00	10	2	259 192,00	0,2575	0,0000
20	matmul	300	wymuszony	1 794 143 776,90	0,04	36 048 000,00	20	2	259 224,00	0,2575	0,0667
21	matmul	300	domyślny	1 025 734 857,04	0,07	22 530 000,00	100	3	260 696,00	0,2575	0,0000
22	matmul	300	wymuszony	1 025 734 857,04	0,07	22 530 000,00	10	3	259 344,00	0,2575	0,0000
23	matmul	300	wymuszony	1 025 734 857,04	0,07	22 530 000,00	20	3	259 368,00	0,2575	0,0000
24	matmul	300	domyślny	725 082 330,47	0,09	16 897 500,00	75	4	261 040,00	0,2575	0,0000

Lp.	Pętla wzorcowa	N	Sposób przydziału iteracji do wątków	Df	X1	X2	X3	X4	γ	λ	θ
25	matmul	300	wymuszony	783 573 527,44	0,09	18 024 000,00	10	4	259 528,00	0,2575	0,0667
26	matmul	300	wymuszony	783 573 527,44	0,09	18 024 000,00	20	4	259 552,00	0,2575	0,0667
27	matmul	400	domyślny	3 808 804 397,43	0,02	80 080 000,00	200	2	617 054,55	0,4578	0,0000
28	matmul	400	wymuszony	3 808 804 397,43	0,02	80 080 000,00	10	2	613 109,09	0,4578	0,0000
29	matmul	400	wymuszony	3 808 804 397,43	0,02	80 080 000,00	20	2	613 290,91	0,4578	0,0000
30	matmul	400	domyślny	2 613 361 697,28	0,03	53 653 600,00	134	3	617 436,36	0,4578	0,0050
31	matmul	400	wymuszony	2 779 948 626,62	0,02	56 056 000,00	10	3	614 036,36	0,4578	0,0500
32	matmul	400	wymuszony	2 779 948 626,62	0,02	56 056 000,00	20	3	613 618,18	0,4578	0,0500
33	matmul	400	domyślny	1 600 145 067,14	0,04	40 040 000,00	100	4	617 127,27	0,4578	0,0000
34	matmul	400	wymuszony	1 600 145 067,14	0,04	40 040 000,00	10	4	614 581,82	0,4578	0,0000
35	matmul	400	wymuszony	1 600 145 067,14	0,04	40 040 000,00	20	4	614 527,27	0,4578	0,0000
36	matmul	500	domyślny	10 560 035 465,71	0,01	156 375 000,00	250	2	1 203 520,00	0,7153	0,0000
37	matmul	500	wymuszony	10 560 035 465,71	0,01	156 375 000,00	10	2	1 197 840,00	0,7153	0,0000
38	matmul	500	wymuszony	14 337 842 300,94	0,00	162 630 000,00	20	2	1 197 680,00	0,7153	0,0400
39	matmul	500	domyślny	4 214 444 629,53	0,02	104 458 500,00	167	3	1 204 680,00	0,7153	0,0020
40	matmul	500	wymuszony	4 090 047 327,87	0,02	106 335 000,00	10	3	1 198 680,00	0,7153	0,0200
41	matmul	500	wymuszony	3 501 721 725,42	0,02	112 590 000,00	20	3	1 198 320,00	0,7153	0,0800
42	matmul	500	domyślny	3 976 095 476,93	0,02	78 187 500,00	125	4	1 206 440,00	0,7153	0,0000
43	matmul	500	wymuszony	4 159 477 061,45	0,02	81 315 000,00	10	4	1 200 840,00	0,7153	0,0400
44	matmul	500	wymuszony	4 421 791 075,08	0,02	87 570 000,00	20	4	1 200 920,00	0,7153	0,1200

W Tabeli 15 wyróżniono następujące wartości graniczne:

$\lambda_{min}(matmul) = 0{,}0286$; $\lambda_{max}(matmul) = 0{,}7153$

$\gamma_{min}(matmul) = 9\,752{,}33$; $\gamma_{max}(matmul) = 1\,206\,440{,}00$

4.2.2 Ocena jakości przykładowych modeli szczególnych

Ocena jakości przykładowych modeli szczególnych została przeprowadzona dla modelu szczególnego (35) (patrz strona 60), wyznaczonego dla pętli wzorcowej nonInterf oraz dla modelu szczególnego (36) (patrz strona 61), wyznaczonego dla pętli wzorcowej matmul.
W celu dokonania przedmiotowej oceny, wartości *Yt* wyznaczone wg ww. modeli szczególnych oraz wartości *γ* zmierzone w środowisku przedstawionym w Tabeli 7 (patrz strona 48) przeliczono na 1 wątek, korzystając z następujących równań:

$$Yt(per\_thread) = \frac{Yt}{X4^{a4}} \tag{41}$$

$$\gamma(per\_thread) = \frac{\gamma}{X4^{a4}} \tag{42}$$

gdzie:

Yt – szacowany czas procesora dla wykonania pętli programowej przez wszystkie wątki programu, wyrażony liczbą taktów zegara procesora, wyznaczony na podstawie modelu szczególnego,
Yt(*per_thread*) – *Yt* w przeliczeniu na jeden wątek,
γ – faktyczny (tj. zmierzony empirycznie) czas procesora spędzony na wykonywaniu pętli programowej przez wszystkie wątki programu, wyrażony liczbą taktów zegara procesora,
γ(*per_thread*) – *γ* w przeliczeniu na jeden wątek,
X4 – liczba wątków OpenMP wykonujących program,
a4 – parametr *a4* odpowiedniego modelu szczególnego.

W oparciu o wartości *Yt*(*per_thread*) oraz *γ*(*per_thread*), wyznaczone dla wszystkich pętli niewzorcowych wskazanych w rozdziale 4.1, oceniono jakość przykładowych modeli szczególnych, tj. modelu (35) oraz modelu (36), w aspekcie jakościowym i ilościowym, wykorzystując w tym celu podejście, przedstawione w rozdziale 3.4.
W dalszych podrozdziałach niniejszego rozdziału przedstawiono wyniki przedmiotowej oceny. Dla każdej z ww. pętli niewzorcowych, zamieszczono:

- kod źródłowy pętli (w wersji sekwencyjnej),
- tabele z wynikami uzyskanymi dla różnych wielkości problemu, rozwiązywanego w pętli,
- wykresy, sporządzone na podstawie danych z tabel jw., wykorzystane do oceny jakości modelu szczególnego (właściwego dla pętli) w aspekcie jakościowym.

Oznaczenia, przyjęte w tabelach:

N – rozmiar problemu rozwiązywanego w pętli,

BLOCK_SIZE – rozmiar boku bloku kwadratowego przy blokowaniu,

λ – stosunek łącznego rozmiaru danych zapisanych w zmiennych tablicowych przetwarzanych w pętli (wielkość wyrażona w bajtach), dla rozmiaru problemu *N* do rozmiaru pamięci podręcznej L2, dostępnej dla pojedynczego procesora (wielkość wyrażona w bajtach), wyznaczany wg równania (29) (patrz strona 56),

θ – różnica względna pomiędzy średnią liczbą porcji iteracji, przypadającą na pojedynczy wątek OpenMP, a maksymalną liczbą porcji iteracji, przypadającą na pojedynczy wątek OpenMP przy danym sposobie przydziału iteracji do wątków, wyznaczana wg równania (32) (patrz strona 57),

Df – odcisk danych dla pojedynczego wątku OpenMP,

X1 – wielkość bezwymiarowa, określająca stosunek łącznego rozmiaru pamięci podręcznej L1 i L2, przypadającej na pojedynczy wątek OpenMP, do odcisku danych dla pojedynczego wątku OpenMP; wielkość ta jest wyrażona równaniem (3) (patrz strona 22),

X2 – łączna ważona liczba operacji przypadająca na pojedynczy wątek OpenMP,

X3 – maksymalna, dla danego sposobu przydziału iteracji do wątków OpenMP, wielkość pojedynczej porcji iteracji przydzielonej do wykonania przez wątek,

X4 – liczba wątków OpenMP wykonujących program,

Yt – szacowany czas procesora dla wykonania pętli programowej przez wszystkie wątki programu, wyrażony liczbą taktów zegara procesora, wyznaczony na podstawie modelu szczególnego,

γ – faktyczny (tj. zmierzony empirycznie) czas procesora spędzony na wykonywaniu pętli programowej przez wszystkie wątki programu, wyrażony liczbą taktów zegara procesora,

Yt(*per_thread*) – *Yt* w przeliczeniu na jeden wątek,

γ(*per_thread*) – *γ* w przeliczeniu na jeden wątek,

$\delta_{Y(per\_thread)}$ – błąd względny oszacowania wartości *Yt*(*per_thread*), wyznaczony na podstawie równania (43), otrzymanego przez podstawienie odpowiednich wielkości do równania (39) (patrz strona 66):

$$\delta_{Y(per\_thread)} = \left| \frac{Yt(per\_thread) - \gamma(per\_thread)}{\gamma(per\_thread)} \right| \times 100\% \tag{43}$$

Omówienie uzyskanych wyników oraz wnioski przedstawiono w rozdziale 5.

Przy ocenie jakości modeli szczególnych, kluczową kwestią jest oszacowanie czasów wykonania pętli niewzorcowych uwzględniając wszystkie założenia modeli, będących przedmiotem oceny. W przypadku opracowanych, przykładowych modeli szczególnych oznacza to, że w celu oszacowania czasów wykonania pętli niewzorcowych należy postępować zgodnie z procedurą, przedstawioną na Rysunku 18 (patrz strona 69). W rozdziale 3.6 przedstawiono przykład zastosowania ww. procedury w praktyce. Dla wszystkich pętli niewzorcowych, będących przedmiotem badań eksperymentalnych niniejszej pracy, oszacowanie czasów wykonania przeprowadzono analogicznie, jak w przykładzie z rozdziału 3.6.

4.2.2.1. Model szczególny wyznaczony dla pętli wzorcowej nonInterf

4.2.2.1.1. Zastosowanie modelu dla pętli niewzorcowej CG_cg_3

Kod źródłowy pętli CG_cg_3 (wersja sekwencyjna) ma następującą postać:

Kod źródłowy pętli CG_cg_3 – wersja sekwencyjna

```
int q[N], z[N], r[N], x[N], p[N];

for(j = 0; j < N; j++){
    q[j] = 0.0;
    z[j] = 0.0;
    r[j] = x[j];
    p[j] = r[j];
} //endfor j
```

Wyniki uzyskane dla pętli CG_cg_3 zestawiono w Tabelach16 ÷ 19 oraz zaprezentowano na wykresach (Rysunki 20 ÷ 22).
W tabelach przyjęto oznaczenia przedstawione w rozdziale 4.2.2 (patrz strona 91).

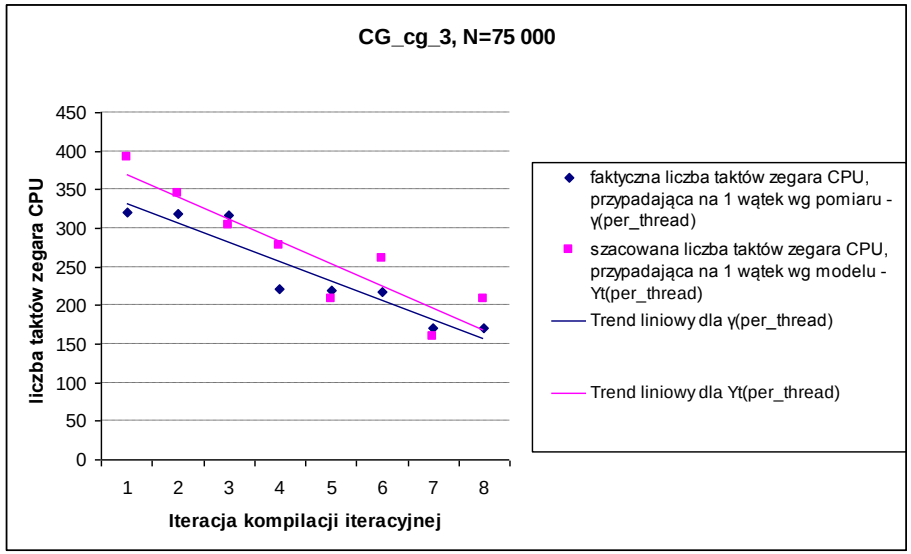
Tabela 16 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (35)) – dla pętli CG_cg_3

N	**Wartość statystyki testowej Kołmogorowa-Smirnowa**	**Graniczny poziom istotności *p***	**Wniosek odnośnie typu rozkładu reszt**
75 000	0,2572	0,5793	Rozkład normalny
118 000	0,2433	0,6454	Rozkład normalny
160 000	0,2506	0,6106	Rozkład normalny

Tabela 17 Wyniki uzyskane dla pętli niewzorcowej CG_cg_3, dla N = 75 000

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	75 000	wymuszony	0,3576	0,2000	900 000,00	74,86	180 000,00	15 000	2	774,34	631,67	392,04	319,81	22,59
2	75 000	wymuszony	0,3576	0,0000	750 000,00	89,83	150 000,00	7 500	2	683,23	629,60	345,91	318,76	8,52
3	75 000	domyślny	0,3576	0,0000	750 000,00	89,83	150 000,00	37 500	2	598,18	627,25	302,85	317,57	4,63
4	75 000	wymuszony	0,3576	0,2000	600 000,00	112,29	120 000,00	7 500	3	813,79	651,21	276,69	221,41	24,97
5	75 000	domyślny	0,3576	0,0000	500 000,00	134,74	100 000,00	25 000	3	613,89	647,85	208,73	220,27	5,24
6	75 000	wymuszony	0,3576	0,2000	600 000,00	112,29	120 000,00	15 000	3	768,51	642,35	261,29	218,40	19,64
7	75 000	domyślny	0,3576	0,0000	375 000,00	179,66	75 000,00	18 750	4	625,30	668,35	160,28	171,32	6,44
8	75 000	wymuszony	0,3576	0,2000	450 000,00	149,71	90 000,00	7 500	4	809,44	662,60	207,48	169,84	22,16
												Średnia dla $\delta_{Yt(per\_thread)}$		**14,27**

Rysunek 20 Wykresy sporządzone na podstawie danych z Tabeli 17



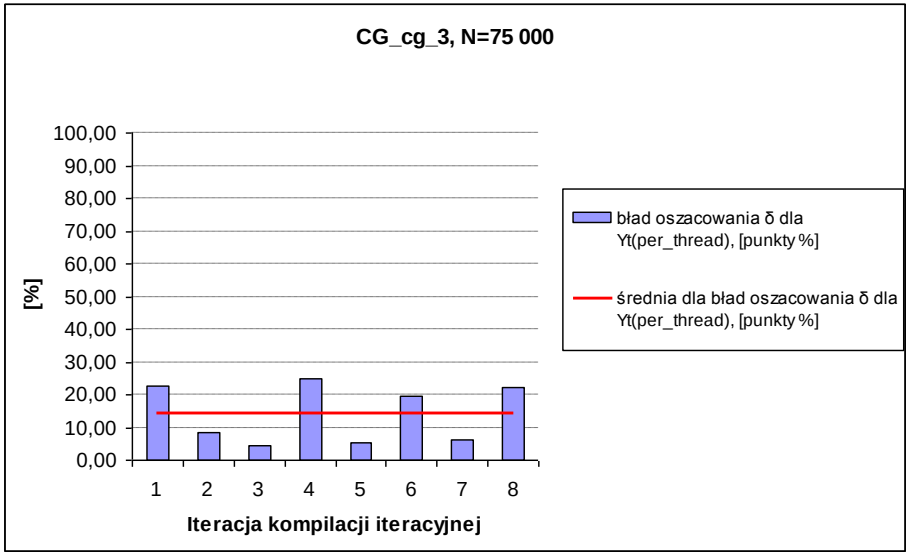
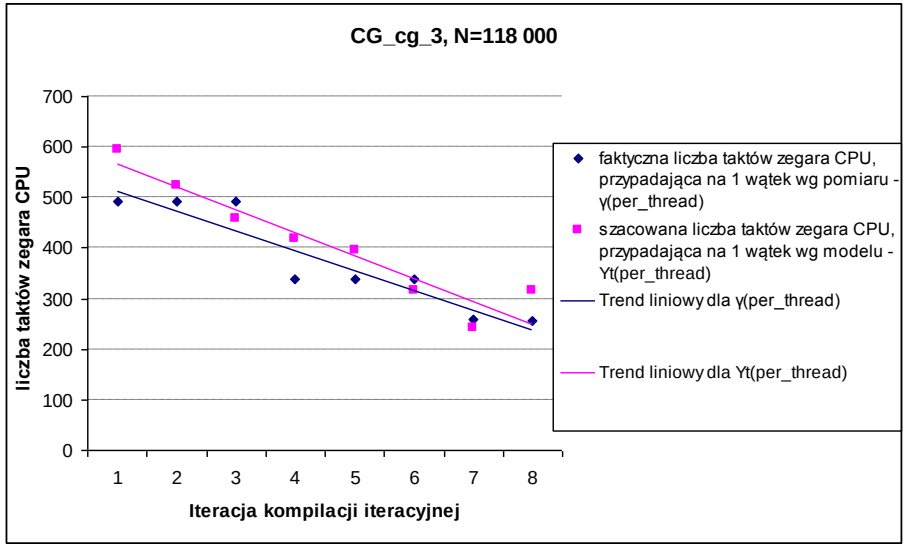


Tabela 18 Wyniki uzyskane dla pętli niewzorcowej CG_cg_3, dla N = 118 000

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	118 000	wymuszony	0,5627	0,2000	1 416 000,00	47,58	283 200,00	23 600	2	1 173,85	974,50	594,31	493,38	20,46
2	118 000	wymuszony	0,5627	0,0000	1 180 000,00	57,09	236 000,00	11 800	2	1 035,73	972,25	524,38	492,24	6,53
3	118 000	domyślny	0,5627	0,0000	1 180 000,00	57,09	236 000,00	59 000	2	906,80	970,00	459,10	491,10	6,52
4	118 000	wymuszony	0,5627	0,2000	944 000,00	71,37	188 800,00	11 800	3	1 233,66	994,79	419,45	338,23	24,01
5	118 000	wymuszony	0,5627	0,2000	944 000,00	71,37	188 800,00	23 600	3	1 165,01	994,00	396,11	337,96	17,20
6	118 000	domyślny	0,5627	0,0000	786 680,00	85,64	157 336,00	39 334	3	930,64	993,50	316,42	337,79	6,33
7	118 000	domyślny	0,5627	0,0000	590 000,00	114,19	118 000,00	29 500	4	947,91	1 013,56	242,98	259,80	6,48
8	118 000	wymuszony	0,5627	0,2000	708 000,00	95,16	141 600,00	11 800	4	1 227,06	1 002,92	314,53	257,08	22,35
												Średnia dla $\delta_{Yt(per\_thread)}$		**13,73**

Rysunek 21 Wykresy sporządzone na podstawie danych z Tabeli 18



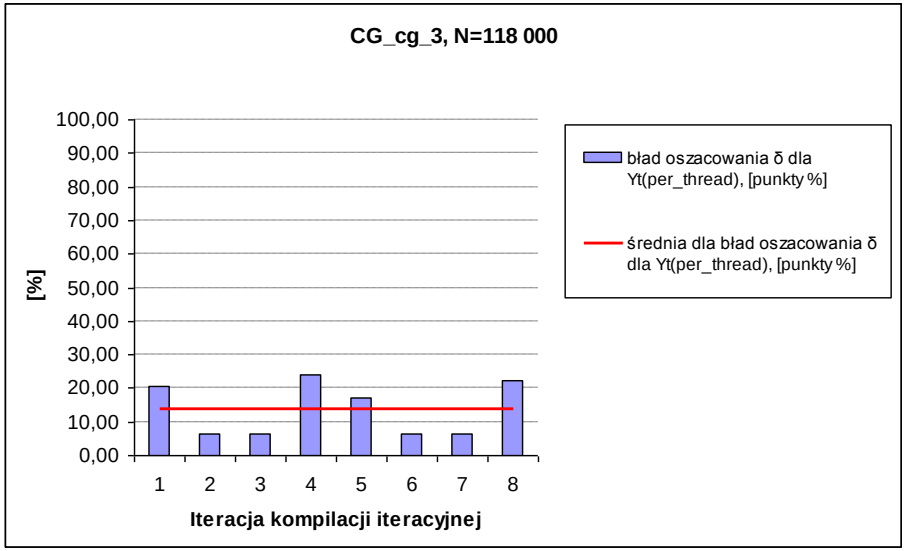
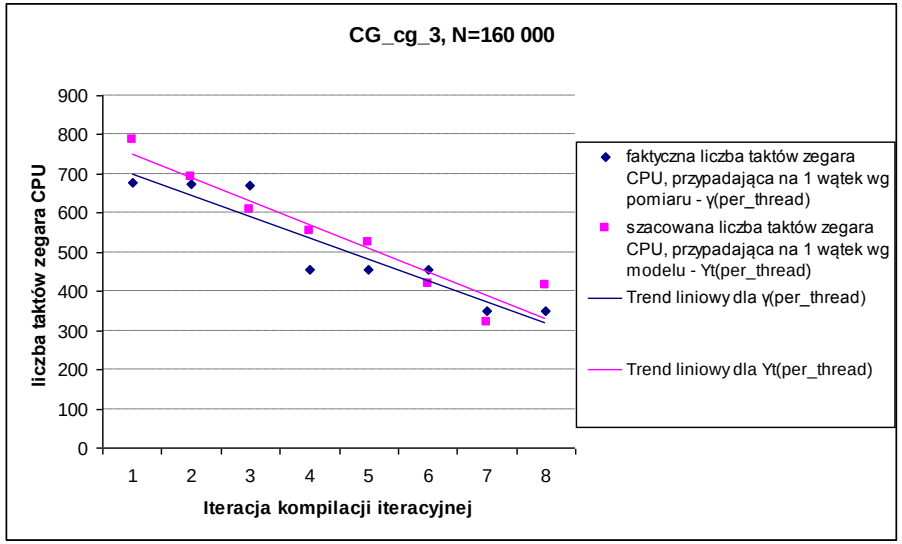
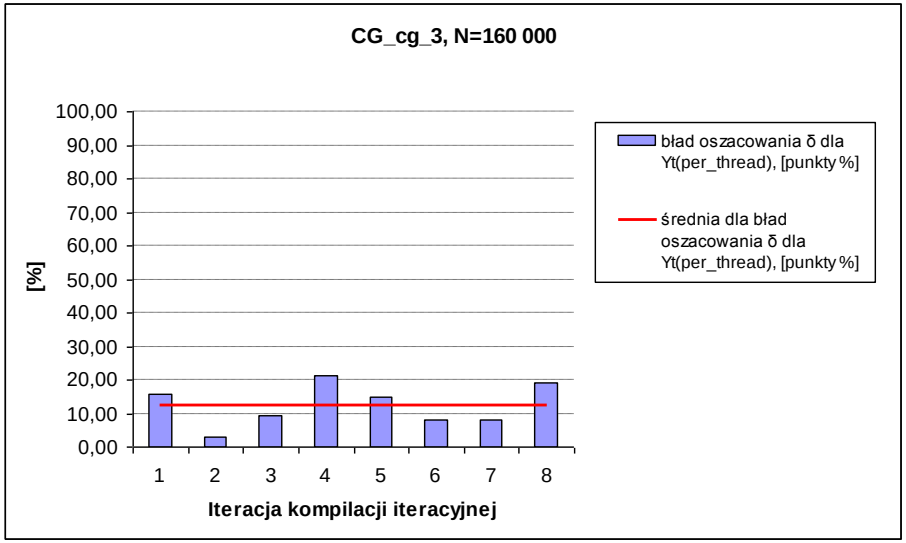


Tabela 19 *Wyniki uzyskane dla pętli niewzorcowej CG_cg_3, dla N = 160 000*

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	160 000	wymuszony	0,7629	0,2000	1 920 000,00	35,09	384 000,00	32 000	2	1 552,41	1 338,86	785,97	677,85	15,95
2	160 000	wymuszony	0,7629	0,0000	1 600 000,00	42,11	320 000,00	16 000	2	1 369,75	1 328,58	693,49	672,65	3,10
3	160 000	domyślny	0,7629	0,0000	1 600 000,00	42,11	320 000,00	80 000	2	1 199,24	1 324,00	607,16	670,33	9,42
4	160 000	wymuszony	0,7629	0,2000	1 280 000,00	52,63	256 000,00	16 000	3	1 631,51	1 343,25	554,72	456,71	21,46
5	160 000	wymuszony	0,7629	0,2000	1 280 000,00	52,63	256 000,00	32 000	3	1 540,72	1 341,93	523,85	456,26	14,81
6	160 000	domyślny	0,7629	0,0000	1 066 680,00	63,16	213 336,00	53 334	3	1 230,77	1 336,13	418,46	454,29	7,89
7	160 000	domyślny	0,7629	0,0000	800 000,00	84,21	160 000,00	40 000	4	1 253,61	1 363,53	321,34	349,51	8,06
8	160 000	wymuszony	0,7629	0,2000	960 000,00	70,18	192 000,00	16 000	4	1 622,79	1 361,92	415,97	349,10	19,15
												Średnia dla $\delta_{Yt(per\_thread)}$		**12,48**

Rysunek 22 *Wykresy sporządzone na podstawie danych z Tabeli 19*





4.2.2.1.2. Zastosowanie modelu dla pętli niewzorcowej CG_cg_4

Kod źródłowy pętli CG_cg_4 (wersja sekwencyjna) ma następującą postać:

Kod źródłowy pętli CG_cg_4 – wersja sekwencyjna

```
int r[N], x[N], sum, d;

for(j = 0; j < N; j++) {
    d = x[j] - r[j];
    sum = sum + d*d;
} //endfor j
```

Wyniki uzyskane dla pętli CG_cg_4 zestawiono w Tabelach 20 ÷ 23 oraz zaprezentowano na wykresach (Rysunki 23 ÷ 25).
W tabelach przyjęto oznaczenia przedstawione w rozdziale 4.2.2 (patrz strona 91).

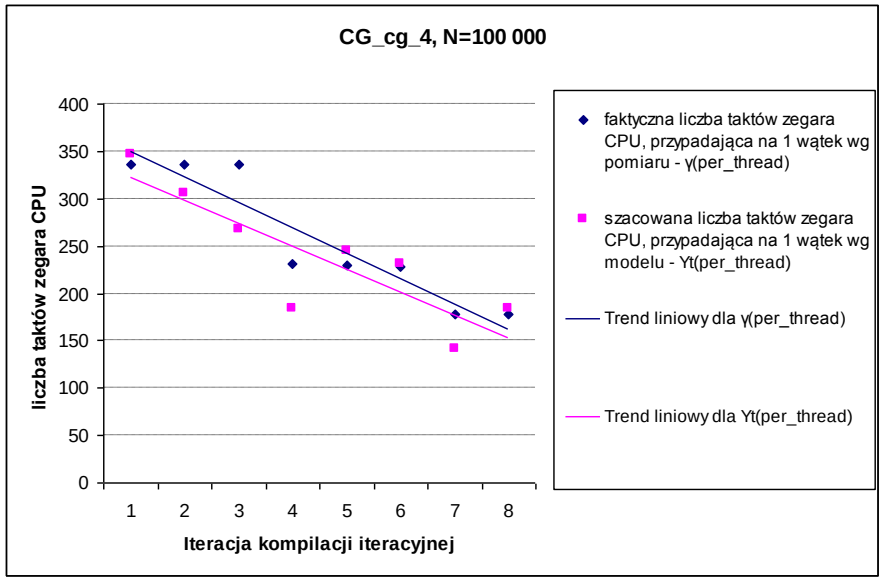
Tabela 20 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (35)) – dla pętli CG_cg_4

N	**Wartość statystyki testowej Kołmogorowa-Smirnowa**	**Graniczny poziom istotności *p***	**Wniosek odnośnie typu rozkładu reszt**
100 000	0,2626	0,5541	Rozkład normalny
215 000	0,2479	0,6233	Rozkład normalny
330 000	0,2444	0,6401	Rozkład normalny

Tabela 21 *Wyniki uzyskane dla pętli niewzorcowej CG_cg_4, dla N = 100 000*

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	100 000	wymuszony	0,1907	0,2000	480 128,00	140,32	210 000,00	20 000	2	683,92	665,38	346,26	336,87	2,79
2	100 000	wymuszony	0,1907	0,0000	400 128,00	168,37	175 000,00	10 000	2	603,46	664,83	305,53	336,60	9,23
3	100 000	domyślny	0,1907	0,0000	400 128,00	168,37	175 000,00	50 000	2	528,34	663,16	267,49	335,75	20,33
4	100 000	domyślny	0,1907	0,0000	266 800,00	252,52	116 669,00	33 334	3	542,26	681,46	184,37	231,70	20,43
5	100 000	wymuszony	0,1907	0,2000	320 128,00	210,45	140 000,00	10 000	3	718,80	676,44	244,39	229,99	6,26
6	100 000	wymuszony	0,1907	0,2000	320 128,00	210,45	140 000,00	20 000	3	678,80	670,49	230,79	227,97	1,24
7	100 000	domyślny	0,1907	0,0000	200 128,00	336,64	87 500,00	25 000	4	552,35	691,50	141,58	177,25	20,12
8	100 000	wymuszony	0,1907	0,2000	240 128,00	280,56	105 000,00	10 000	4	714,99	691,28	183,27	177,19	3,43
													Średnia dla $\delta_{Yt(per\_thread)}$	**10,48**

Rysunek 23 *Wykresy sporządzone na podstawie danych z Tabeli 21*



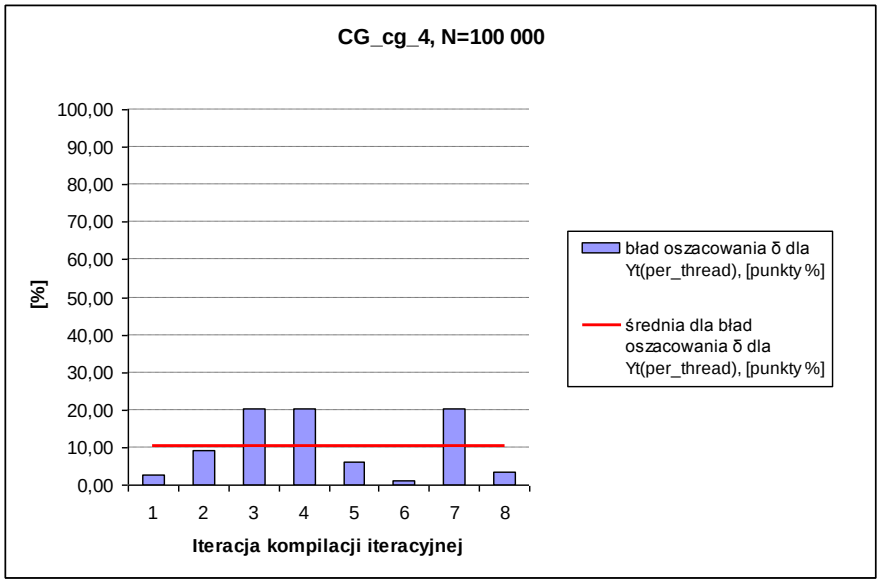
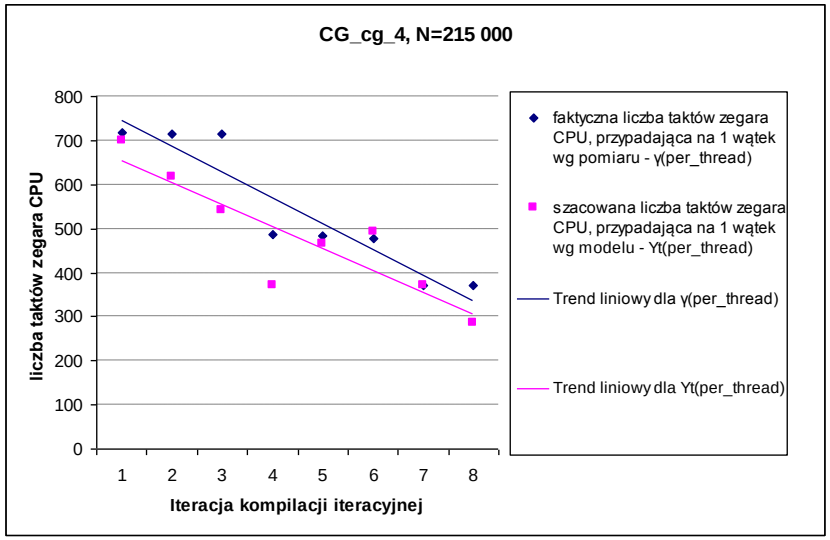


Tabela 22 Wyniki uzyskane dla pętli niewzorcowej CG_cg_4, dla N = 215 000

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	215 000	wymuszony	0,4101	0,2000	1 032 130,00	65,27	451 500,00	43 000	2	1 380,90	1 416,38	699,14	717,09	2,50
2	215 000	wymuszony	0,4101	0,0000	860 128,00	78,33	376 250,00	21 500	2	1 218,44	1 413,17	616,88	715,47	13,78
3	215 000	domyślny	0,4101	0,0000	860 128,00	78,33	376 250,00	107 500	2	1 066,76	1 411,60	540,09	714,68	24,43
4	215 000	domyślny	0,4101	0,0000	573 464,00	117,48	250 834,50	71 667	3	1 094,82	1 427,68	372,24	485,42	23,31
5	215 000	wymuszony	0,4101	0,2000	688 128,00	97,90	301 000,00	43 000	3	1 370,54	1 420,30	465,99	482,91	3,50
6	215 000	wymuszony	0,4101	0,2000	688 128,00	97,90	301 000,00	21 500	3	1 451,30	1 408,67	493,44	478,95	3,03
7	215 000	wymuszony	0,4101	0,2000	516 128,00	130,53	225 750,00	21 500	4	1 443,57	1 444,11	370,03	370,17	0,04
8	215 000	domyślny	0,4101	0,0000	430 128,00	156,63	188 125,00	53 750	4	1 115,18	1 442,10	285,85	369,65	22,67
												Średnia dla $\delta_{Yt(per\_thread)}$		**11,66**

Rysunek 24 Wykresy sporządzone na podstawie danych z Tabeli 22



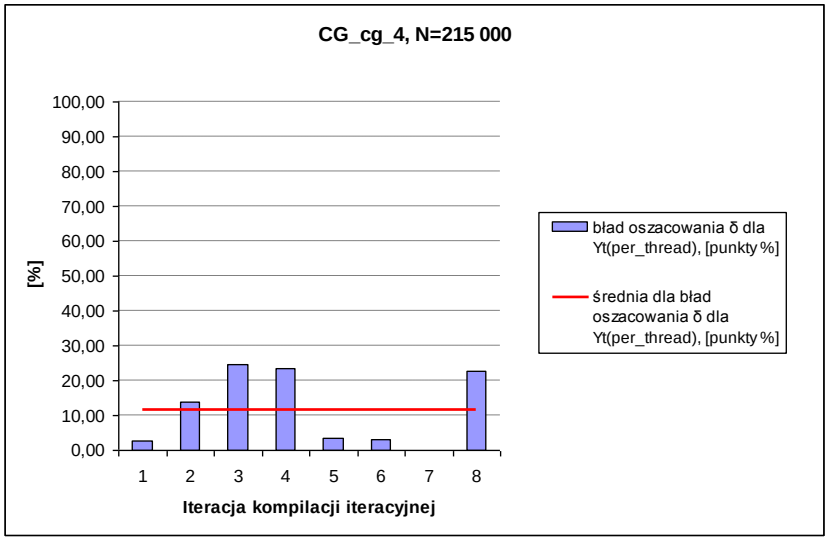
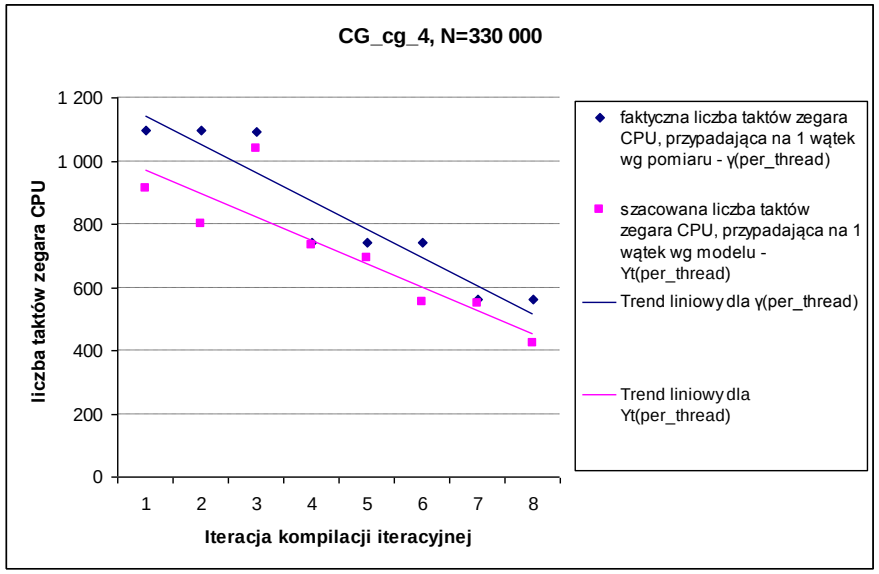
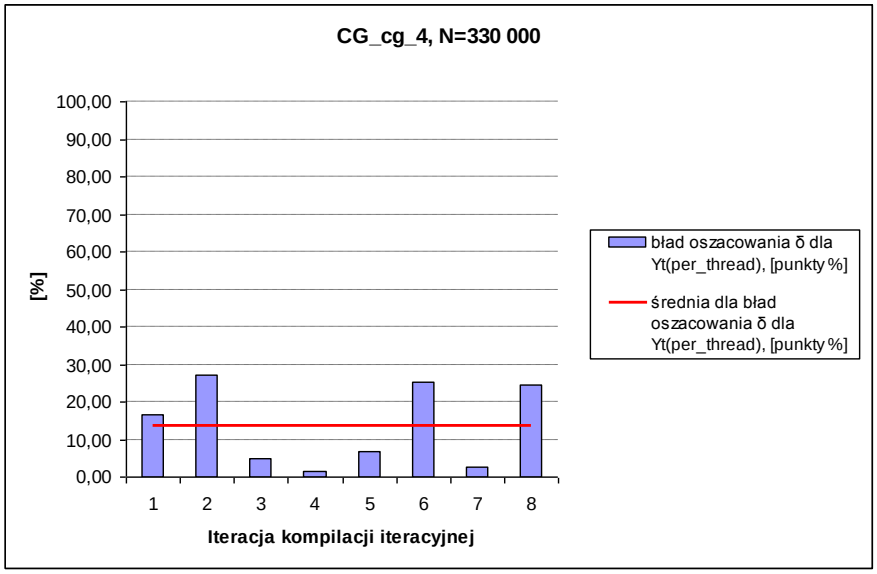


Tabela 23 Wyniki uzyskane dla pętli niewzorcowej CG_cg_4, dla N = 330 000

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	330 000	wymuszony	0,6294	0,0000	1 320 130,00	51,03	577 500,00	33 000	2	1 805,57	2 168,88	914,14	1 098,08	16,75
2	330 000	domyślny	0,6294	0,0000	1 320 130,00	51,03	577 500,00	165 000	2	1 580,80	2 167,40	800,34	1 097,33	27,06
3	330 000	wymuszony	0,6294	0,2000	1 584 130,00	42,53	693 000,00	66 000	2	2 046,33	2 155,40	1 036,03	1 091,25	5,06
4	330 000	wymuszony	0,6294	0,2000	1 056 130,00	63,79	462 000,00	33 000	3	2 150,63	2 180,00	731,22	741,21	1,35
5	330 000	wymuszony	0,6294	0,2000	1 056 130,00	63,79	462 000,00	66 000	3	2 030,95	2 177,00	690,53	740,19	6,71
6	330 000	domyślny	0,6294	0,0000	880 128,00	76,55	385 000,00	110 000	3	1 622,36	2 175,00	551,61	739,51	25,41
7	330 000	wymuszony	0,6294	0,2000	792 128,00	85,05	346 500,00	33 000	4	2 139,15	2 193,94	548,33	562,37	2,50
8	330 000	domyślny	0,6294	0,0000	660 128,00	102,06	288 750,00	82 500	4	1 652,52	2 187,40	423,59	560,69	24,45
												Średnia dla $\delta_{Yt(per\_thread)}$		**13,66**

Rysunek 25 Wykresy sporządzone na podstawie danych z Tabeli 23





4.2.2.1.3. Zastosowanie modelu dla pętli niewzorcowej FT_auxfnct_2

Kod źródłowy pętli FT_auxfnct_2 (wersja sekwencyjna) ma następującą postać:

Kod źródłowy pętli FT_auxfnct_2 – wersja sekwencyjna

```
int y[N][N][N], x[N][N][N], twiddle[N][N][N];

for (k = 0; k < N; k++) {
    for (j = 0; j < N; j++) {
        for (i = 0; i < N; i++) {
           y[k][j][i] = y[k][j][i]*twiddle[k][j][i];
           x[k][j][i] = y[k][j][i];
        } //endfor i
    } //endfor j
} //endfor k
```

Wyniki uzyskane dla pętli FT_auxfnct_2 zestawiono w Tabelach 24 ÷ 27 oraz zaprezentowano na wykresach (Rysunki 26 ÷ 28).
W tabelach przyjęto oznaczenia przedstawione w rozdziale 4.2.2 (patrz strona 91).

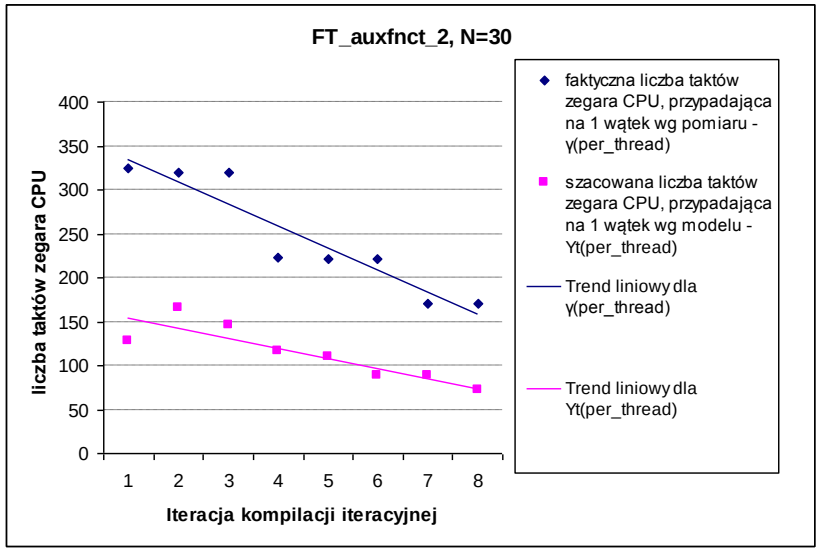
Tabela 24 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (35)) – dla pętli FT_auxfnct_2

N	**Wartość statystyki testowej Kołmogorowa-Smirnowa**	**Graniczny poziom istotności *p***	**Wniosek odnośnie typu rozkładu reszt**
30	0,2294	0,7135	Rozkład normalny
38	0,1853	0,8642	Rozkład normalny
45	0,1912	0,8823	Rozkład normalny

Tabela 25 Wyniki uzyskane dla pętli niewzorcowej FT_auxfnct_2, dla N = 30

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	30	domyślny	0,0772	0,0000	162 000,00	415,87	33 750,00	15	2	253,24	639,96	128,21	324,01	60,43
2	30	wymuszony	0,0772	0,2000	194 400,00	346,56	40 500,00	6	2	327,81	631,27	165,97	319,61	48,07
3	30	wymuszony	0,0772	0,0000	162 000,00	415,87	33 750,00	3	2	289,24	630,28	146,44	319,10	54,11
4	30	wymuszony	0,0772	0,2000	129 600,00	519,84	27 000,00	3	3	344,52	654,60	117,14	222,57	47,37
5	30	wymuszony	0,0772	0,2000	129 600,00	519,84	27 000,00	6	3	325,35	651,93	110,62	221,66	50,09
6	30	domyślny	0,0772	0,0000	108 000,00	623,81	22 500,00	10	3	259,89	650,88	88,36	221,30	60,07
7	30	wymuszony	0,0772	0,2000	97 200,00	693,12	20 250,00	3	4	342,68	668,19	87,84	171,28	48,72
8	30	domyślny	0,0772	0,0667	86 400,00	779,76	18 000,00	8	4	280,88	666,41	72,00	170,82	57,85
												Średnia dla $\delta_{Yt(per\_thread)}$		**53,34**

Rysunek 26 Wykresy sporządzone na podstawie danych z Tabeli 25



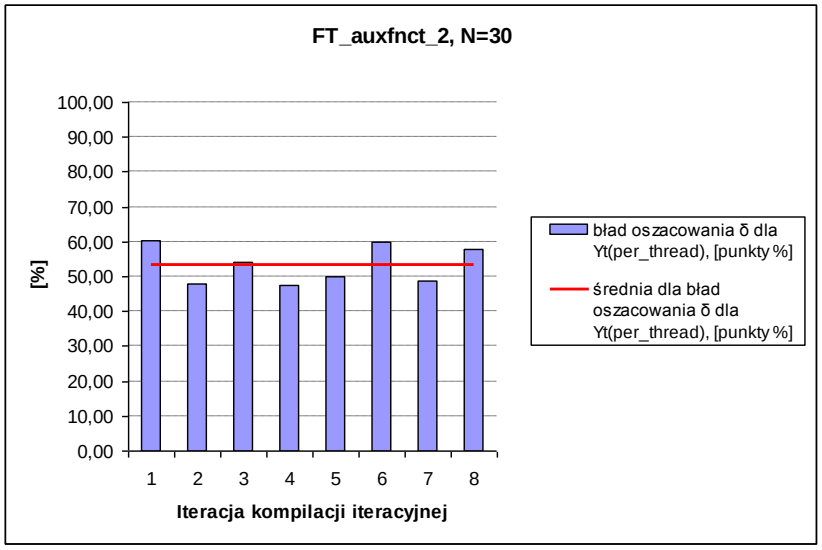
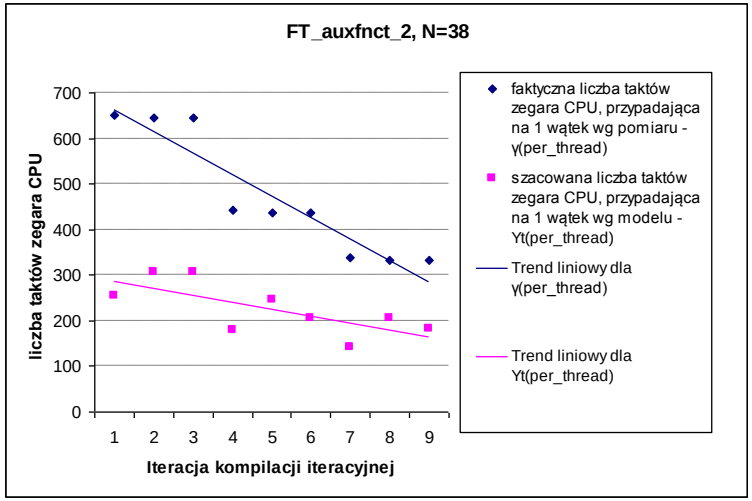


Tabela 26 *Wyniki uzyskane dla pętli niewzorcowej FT_auxfnct_2, dla N = 38*

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	38	domyślny	0,1570	0,0000	329 232,00	204,63	68 590,00	19	2	504,92	1 286,48	255,63	651,33	60,75
2	38	wymuszony	0,1570	0,1053	363 888,00	185,14	75 810,00	7	2	606,08	1 273,16	306,85	644,58	52,40
3	38	wymuszony	0,1570	0,0526	346 560,00	194,40	72 200,00	4	2	604,51	1 271,90	306,06	643,95	52,47
4	38	domyślny	0,1570	0,0263	225 264,00	299,08	46 930,00	13	3	530,69	1 301,72	180,44	442,59	59,23
5	38	wymuszony	0,1570	0,2632	277 248,00	243,00	57 760,00	4	3	720,04	1 286,94	244,81	437,56	44,05
6	38	wymuszony	0,1570	0,1053	242 592,00	277,71	50 540,00	7	3	601,52	1 283,00	204,52	436,22	53,12
7	38	domyślny	0,1570	0,0526	173 280,00	388,80	36 100,00	10	4	553,26	1 323,94	141,82	339,36	58,21
8	38	wymuszony	0,1570	0,4737	242 592,00	277,71	50 540,00	7	4	797,88	1 301,38	204,52	333,58	38,69
9	38	wymuszony	0,1570	0,2632	207 936,00	324,00	43 320,00	4	4	716,19	1 300,86	183,58	333,45	44,95
												Średnia dla $\delta_{Yt(per\_thread)}$		**51,54**

Rysunek 27 *Wykresy sporządzone na podstawie danych z Tabeli 26*



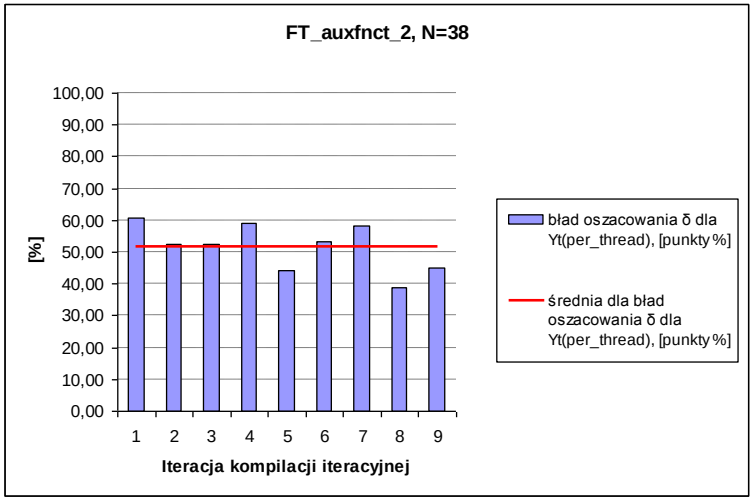
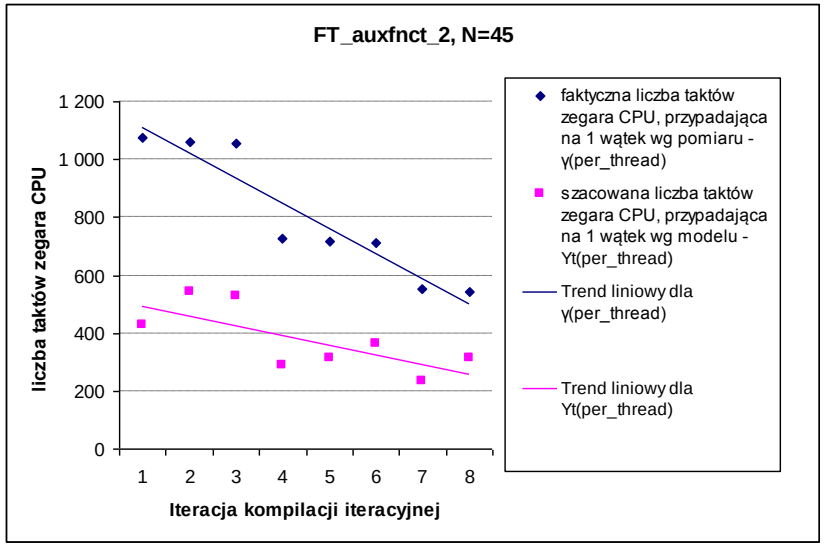
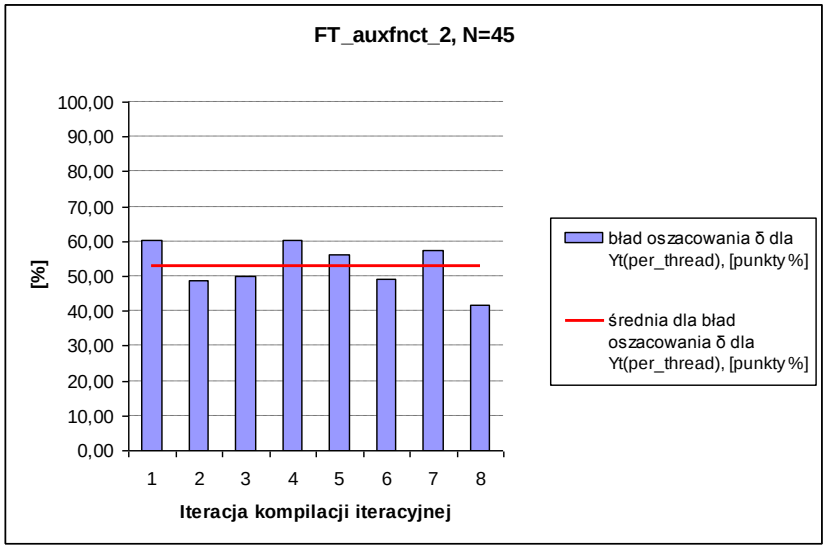


Tabela 27 Wyniki uzyskane dla pętli niewzorcowej FT_auxfnct_2, dla N = 45

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	45	domyślny	0,2607	0,0222	558 900,00	120,54	116 437,50	23	2	843,99	2 120,57	427,30	1 073,62	60,20
2	45	wymuszony	0,2607	0,2000	656 100,00	102,68	136 687,50	9	2	1 070,72	2 092,20	542,09	1 059,26	48,82
3	45	wymuszony	0,2607	0,1111	607 500,00	110,90	126 562,50	5	2	1 040,68	2 086,83	526,88	1 056,54	50,13
4	45	domyślny	0,2607	0,0000	364 500,00	184,83	75 937,50	15	3	848,86	2 145,43	288,62	729,45	60,43
5	45	wymuszony	0,2607	0,0000	364 500,00	184,83	75 937,50	5	3	929,50	2 115,70	316,03	719,34	56,07
6	45	wymuszony	0,2607	0,2000	437 400,00	154,03	91 125,00	9	3	1 062,66	2 097,72	361,31	713,23	49,34
7	45	domyślny	0,2607	0,0667	291 600,00	231,04	60 750,00	12	4	917,40	2 160,43	235,16	553,78	57,54
8	45	wymuszony	0,2607	0,3333	364 500,00	184,83	75 937,50	5	4	1 232,92	2 122,07	316,03	543,95	41,90
												Średnia dla $\delta_{Yt(per\_thread)}$		**53,05**

Rysunek 28 Wykresy sporządzone na podstawie danych z Tabeli 27





4.2.2.1.4. Zastosowanie modelu dla pętli niewzorcowej LU_HP_pintgr_11

Kod źródłowy pętli LU_HP_pintgr_11 (wersja sekwencyjna) ma następującą postać:

Kod źródłowy pętli LU_HP_pintgr_11 – wersja sekwencyjna

```
int phi1[N][N], phi2[N][N];

ki1 = 0;
ki2 = N;
jbeg = 0;
jfin1 = N;

for (k = ki1; k < ki2-1; k++) {
   for (j = jbeg; j < jfin1; j++) {
        frc3 =        frc3 + (  phi1[k][j]
                              + phi1[k][j+1]
                              + phi1[k+1][j]
                              + phi1[k+1][j+1]
                              + phi2[k][j]
                              + phi2[k][j+1]
                              + phi2[k+1][j]
                              + phi2[k+1][j+1] );
   } //endfor j
} //endfor k
```

Wyniki uzyskane dla pętli LU_HP_pintgr_11 zestawiono w Tabelach 28 ÷ 31 oraz zaprezentowano na wykresach (Rysunki 29 ÷ 31).
W tabelach przyjęto oznaczenia przedstawione w rozdziale 4.2.2 (patrz strona 91).

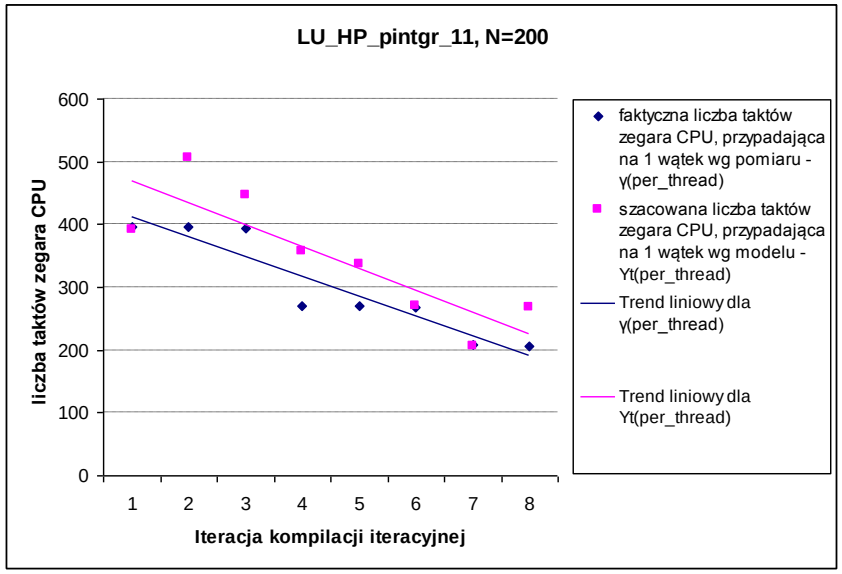
Tabela 28 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (35)) – dla pętli LU_HP_pintgr_11

N	**Wartość statystyki testowej Kołmogorowa-Smirnowa**	**Graniczny poziom istotności *p***	**Wniosek odnośnie typu rozkładu reszt**
200	0,2439	0,6428	Rozkład normalny
265	0,2324	0,6989	Rozkład normalny
330	0,2471	0,627	Rozkład normalny

Tabela 29 Wyniki uzyskane dla pętli niewzorcowej LU_HP_pintgr_11, dla N = 200

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	200	domyślny	0,0763	0,0050	320 064,00	210,49	160 000,00	100	2	772,72	783,74	391,22	396,80	1,41
2	200	wymuszony	0,0763	0,2060	384 064,00	175,42	192 000,00	40	2	1 000,27	780,31	506,42	395,06	28,19
3	200	wymuszony	0,0763	0,0050	320 064,00	210,49	160 000,00	20	2	882,59	778,94	446,84	394,37	13,31
4	200	wymuszony	0,0763	0,2060	256 064,00	263,10	128 000,00	20	3	1 051,26	796,19	357,43	270,71	32,04
5	200	wymuszony	0,0763	0,2060	256 064,00	263,10	128 000,00	40	3	992,76	793,46	337,54	269,78	25,12
6	200	domyślny	0,0763	0,0101	214 464,00	314,14	107 200,00	67	3	796,69	789,76	270,88	268,52	0,88
7	200	domyślny	0,0763	0,0050	160 064,00	420,90	80 000,00	50	4	807,80	814,81	207,06	208,86	0,86
8	200	wymuszony	0,0763	0,2060	192 064,00	350,77	96 000,00	20	4	1 045,67	807,25	268,04	206,92	29,53
												Średnia dla $\delta_{Yt(per\_thread)}$		**16,42**

Rysunek 29 Wykresy sporządzone na podstawie danych z Tabeli 29



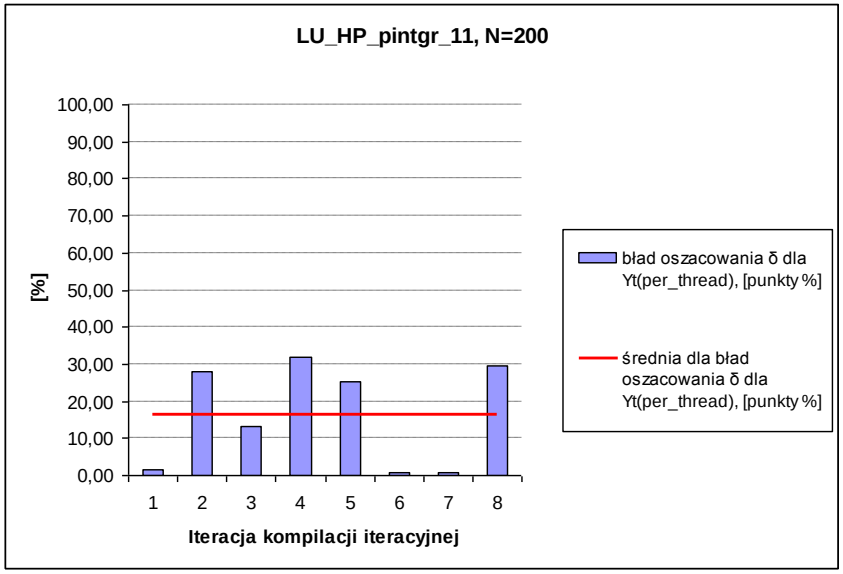
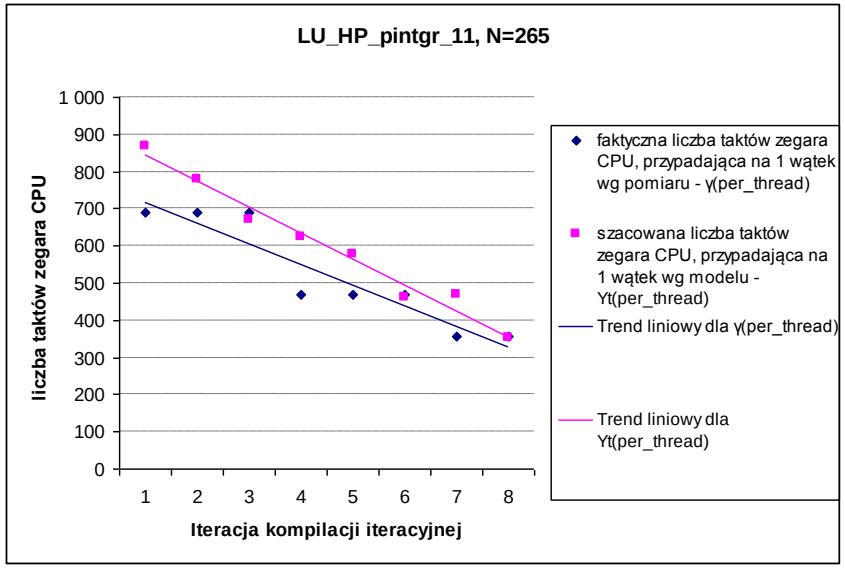


Tabela 30 Wyniki uzyskane dla pętli niewzorcowej LU_HP_pintgr_11, dla N = 265

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	265	wymuszony	0,1339	0,2045	674 224,00	99,92	337 080,00	53	2	1 716,29	1 362,80	868,94	689,97	25,94
2	265	wymuszony	0,1339	0,0227	572 464,00	117,69	286 200,00	27	2	1 540,57	1 361,32	779,97	689,22	13,17
3	265	domyślny	0,1339	0,0000	559 744,00	120,36	279 840,00	132	2	1 321,25	1 361,22	668,94	689,17	2,94
4	265	wymuszony	0,1339	0,2273	457 984,00	147,10	228 960,00	27	3	1 834,98	1 381,22	623,90	469,62	32,85
5	265	wymuszony	0,1339	0,2045	449 504,00	149,88	224 720,00	53	3	1 703,39	1 377,53	579,16	468,36	23,66
6	265	domyślny	0,1339	0,0000	373 184,00	180,53	186 560,00	88	3	1 355,99	1 375,62	461,04	467,71	1,43
7	265	wymuszony	0,1339	0,2273	343 504,00	196,13	171 720,00	27	4	1 825,20	1 396,08	467,85	357,85	30,74
8	265	domyślny	0,1339	0,0000	279 904,00	240,69	139 920,00	66	4	1 381,20	1 395,32	354,04	357,66	1,01
												Średnia dla $\delta_{Yt(per\_thread)}$		**16,47**

Rysunek 30 Wykresy sporządzone na podstawie danych z Tabeli 30



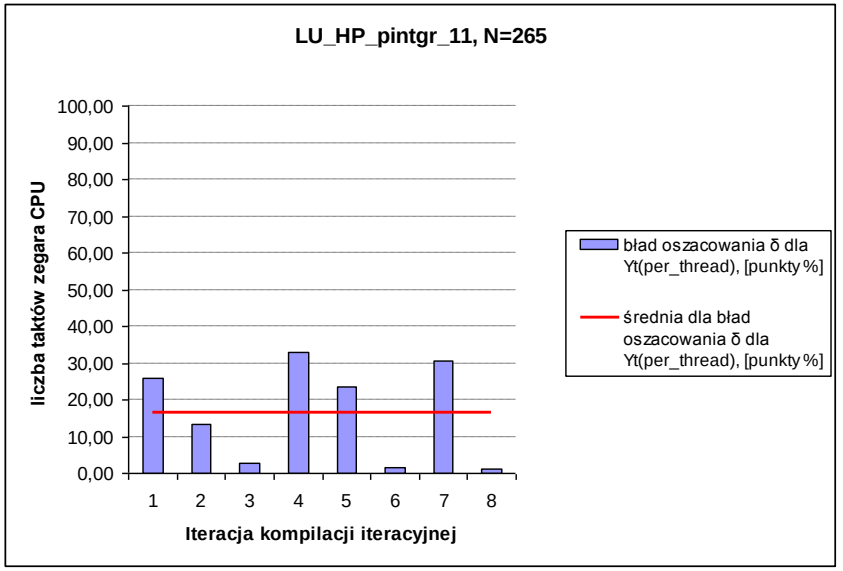
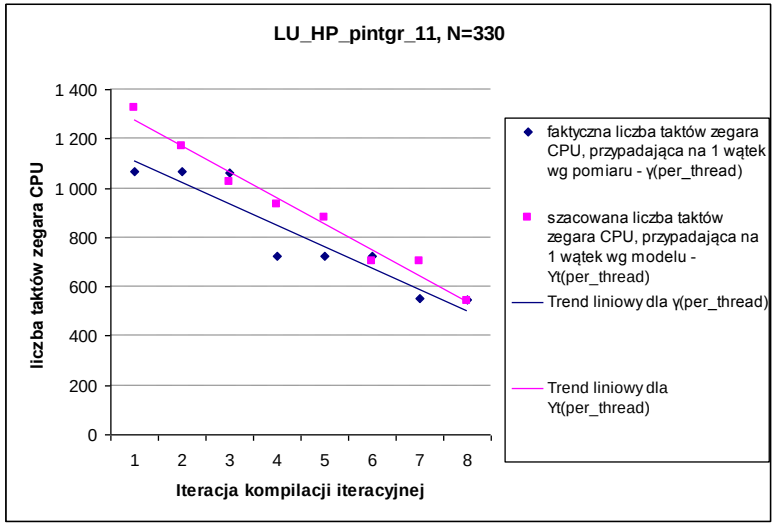
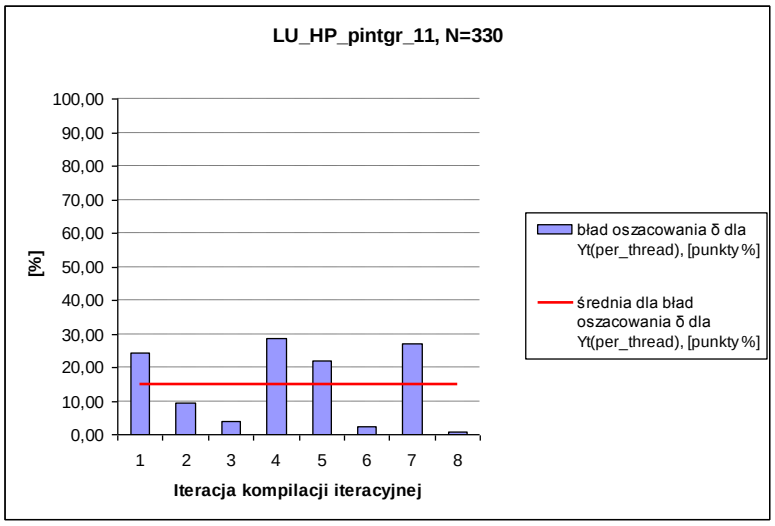


Tabela 31 Wyniki uzyskane dla pętli niewzorcowej LU_HP_pintgr_11, dla N = 330

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	330	wymuszony	0,2077	0,2036	1 045 500,00	64,44	522 720,00	66	2	2 614,36	2 106,00	1 323,62	1 066,24	24,14
2	330	wymuszony	0,2077	0,0030	871 264,00	77,33	435 600,00	33	2	2 306,77	2 104,67	1 167,89	1 065,57	9,60
3	330	domyślny	0,2077	0,0030	871 264,00	77,33	435 600,00	165	2	2 019,62	2 101,67	1 022,51	1 064,05	3,90
4	330	wymuszony	0,2077	0,2036	697 024,00	96,66	348 480,00	33	3	2 747,61	2 133,83	934,19	725,51	28,76
5	330	wymuszony	0,2077	0,2036	697 024,00	96,66	348 480,00	66	3	2 594,71	2 124,20	882,21	722,23	22,15
6	330	domyślny	0,2077	0,0030	580 864,00	115,98	290 400,00	110	3	2 072,71	2 122,00	704,73	721,49	2,32
7	330	wymuszony	0,2077	0,2036	522 784,00	128,87	261 360,00	33	4	2 732,95	2 148,97	700,53	550,84	27,17
8	330	domyślny	0,2077	0,0091	438 304,00	153,71	219 120,00	83	4	2 122,97	2 137,70	544,18	547,95	0,69
												Średnia dla $\delta_{Yt(per\_thread)}$		**14,84**

Rysunek 31 Wykresy sporządzone na podstawie danych z Tabeli 31





4.2.2.1.5. Zastosowanie modelu dla pętli niewzorcowej MG_mg_3

Kod źródłowy pętli MG_mg_3 (wersja sekwencyjna) ma następującą postać:

Kod źródłowy pętli MG_mg_3 – wersja sekwencyjna

```
int m1[N], m2[N], m3[N], mi[N][3], ng[N][3];

for (k = 0; k < N; k++) {
    for (ax = 0; ax < 3; ax++) {
          mi[k][ax] = 2 + ng[k][ax];
    } //endfor ax

    m1[k] = mi[k][0];
    m2[k] = mi[k][1];
    m3[k] = mi[k][2];
} //endfor k
```

Wyniki uzyskane dla pętli MG_mg_3 zestawiono w Tabelach 32 ÷ 35 oraz zaprezentowano na wykresach (Rysunki 32 ÷ 34).
W tabelach przyjęto oznaczenia przedstawione w rozdziale 4.2.2 (patrz strona 91).

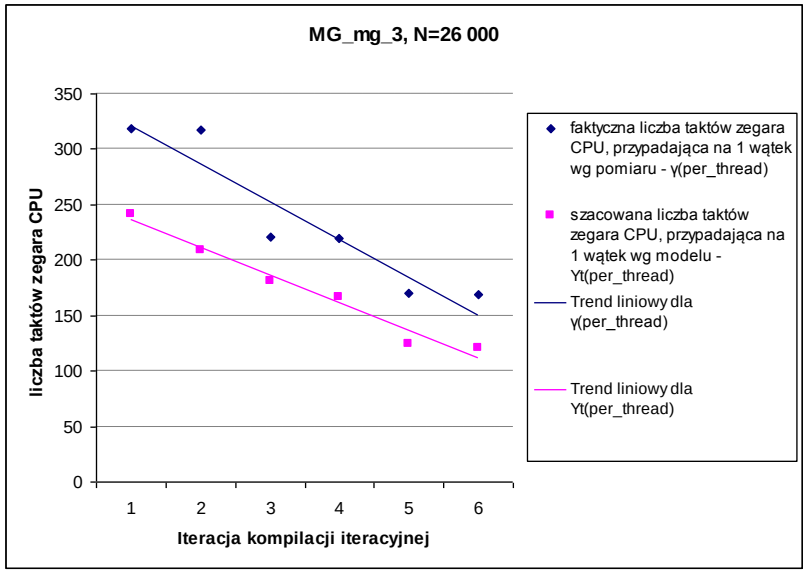
Tabela 32 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (35)) – dla pętli MG_mg_3

N	**Wartość statystyki testowej Kołmogorowa-Smirnowa**	**Graniczny poziom istotności *p***	**Wniosek odnośnie typu rozkładu reszt**
26 000	0,1885	0,9554	Rozkład normalny
57 444	0,2232	0,8681	Rozkład normalny
88 888	0,2363	0,8226	Rozkład normalny

Tabela 33 Wyniki uzyskane dla pętli niewzorcowej MG_mg_3, dla N = 26 000

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	26 000	wymuszony	0,2232	0,2000	561 600,00	119,96	93 600,00	3 900	2	477,37	628,76	241,69	318,34	24,08
2	26 000	wymuszony	0,2232	0,0000	468 000,00	143,96	78 000,00	2 600	2	411,31	627,09	208,24	317,49	34,41
3	26 000	wymuszony	0,2232	0,3500	421 200,00	159,95	70 200,00	3 900	3	533,03	650,88	181,23	221,30	18,10
4	26 000	wymuszony	0,2232	0,2000	374 400,00	179,94	62 400,00	2 600	3	489,91	647,00	166,57	219,98	24,28
5	26 000	wymuszony	0,2232	0,2000	280 800,00	239,93	46 800,00	2 600	4	487,29	663,85	124,91	170,16	26,60
6	26 000	wymuszony	0,2232	0,2000	280 800,00	239,93	46 800,00	3 900	4	471,24	657,08	120,79	168,43	28,28
												Średnia dla $\delta_{Yt(per\_thread)}$		**25,96**

Rysunek 32 Wykresy sporządzone na podstawie danych z Tabeli 33



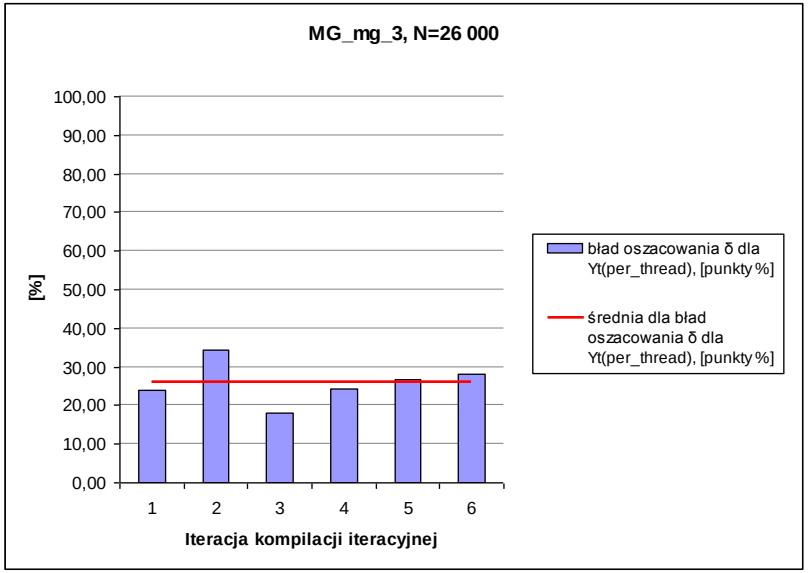
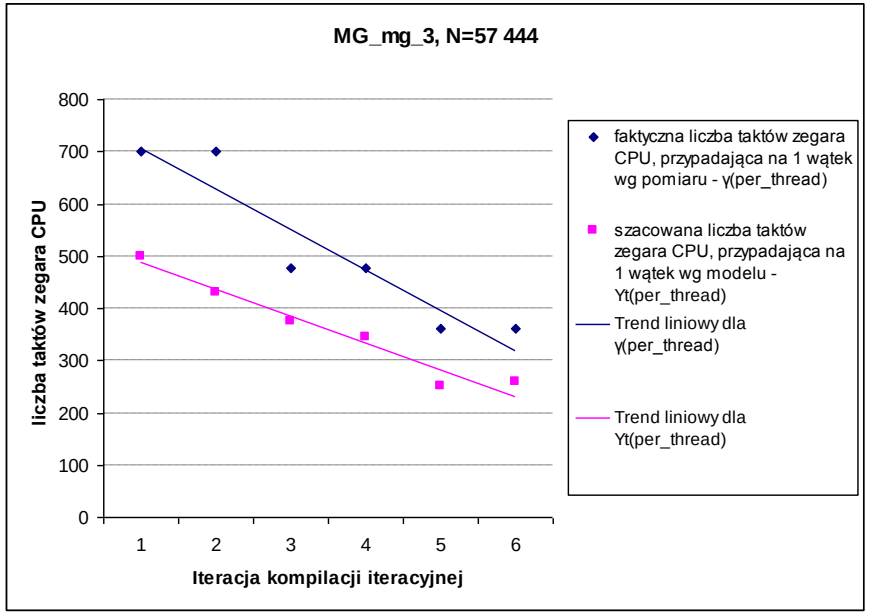


Tabela 34 Wyniki uzyskane dla pętli niewzorcowej MG_mg_3, dla N = 57 444

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	57 444	wymuszony	0,4930	0,2001	1 240 850,00	54,29	206 808,00	8 617	2	988,36	1 384,13	500,39	700,77	28,59
2	57 444	wymuszony	0,4930	0,0001	1 034 100,00	65,15	172 350,00	5 745	2	851,63	1 381,44	431,17	699,41	38,35
3	57 444	wymuszony	0,4930	0,3501	930 636,00	72,39	155 106,00	8 617	3	1 103,61	1 403,63	375,23	477,24	21,37
4	57 444	wymuszony	0,4930	0,2001	827 280,00	81,44	137 880,00	5 745	3	1 014,38	1 400,11	344,89	476,04	27,55
5	57 444	wymuszony	0,4930	0,2001	620 424,00	108,59	103 404,00	8 617	4	975,67	1 413,50	250,09	362,32	30,97
6	57 444	wymuszony	0,4930	0,2001	620 460,00	108,58	103 410,00	5 745	4	1 008,96	1 409,69	258,62	361,34	28,43
												Średnia dla $\delta_{Yt(per\_thread)}$		**29,21**

Rysunek 33 Wykresy sporządzone na podstawie danych z Tabeli 34



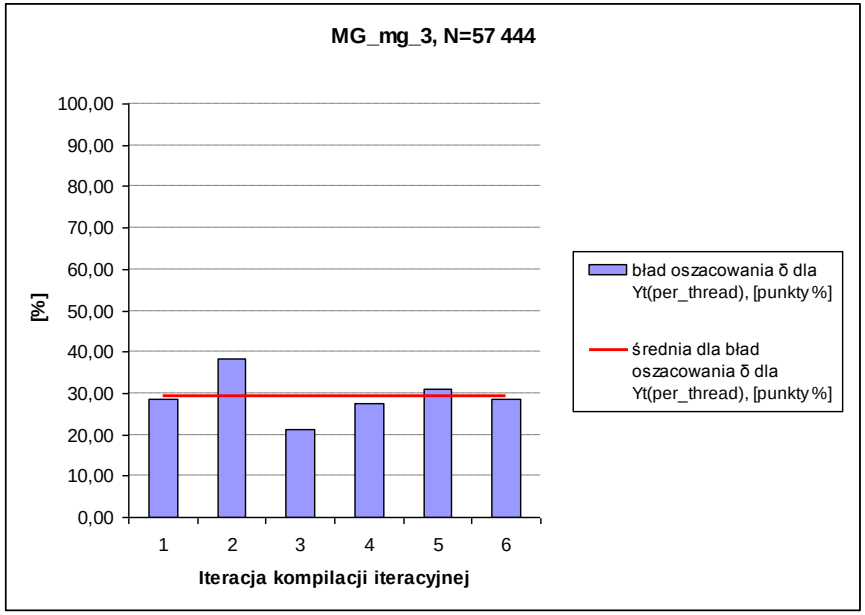
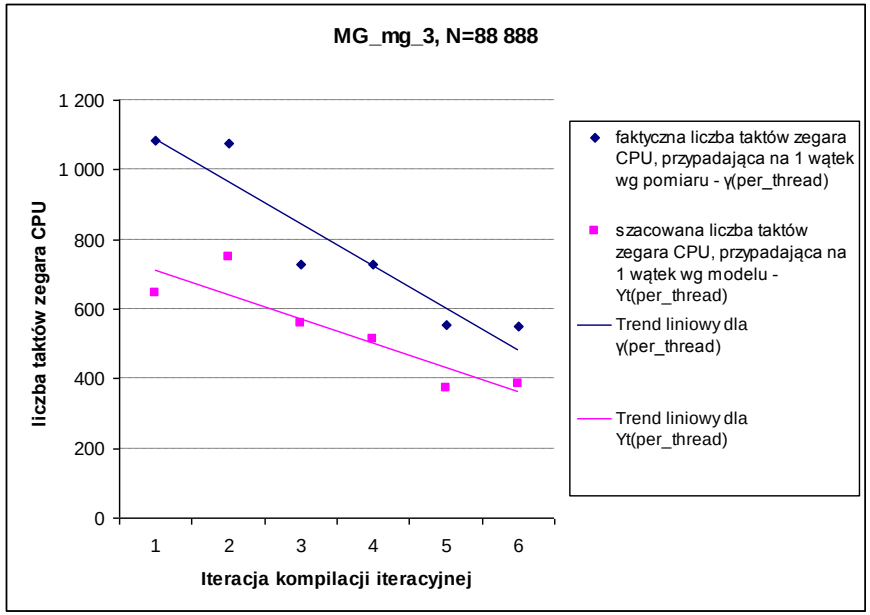
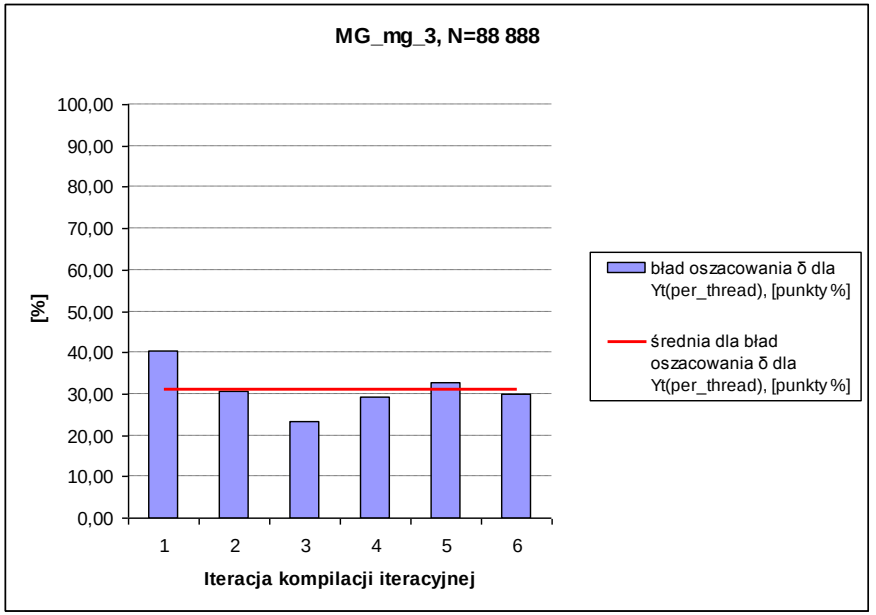


Tabela 35 *Wyniki uzyskane dla pętli niewzorcowej MG_mg_3, dla N = 88 888*

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	88 888	wymuszony	0,7629	0,0001	1 600 200,00	42,10	266 700,00	8 890	2	1 271,50	2 137,39	643,75	1 082,14	40,51
2	88 888	wymuszony	0,7629	0,2000	1 919 950,00	35,09	319 992,00	13 333	2	1 475,51	2 124,86	747,03	1 075,79	30,56
3	88 888	wymuszony	0,7629	0,3500	1 439 960,00	46,79	239 994,00	13 333	3	1 647,56	2 145,43	560,18	729,45	23,21
4	88 888	wymuszony	0,7629	0,2002	1 280 160,00	52,63	213 360,00	8 890	3	1 514,48	2 143,29	514,93	728,72	29,34
5	88 888	wymuszony	0,7629	0,2000	959 976,00	70,18	159 996,00	13 333	4	1 456,57	2 162,00	373,36	554,18	32,63
6	88 888	wymuszony	0,7629	0,2002	960 120,00	70,17	160 020,00	8 890	4	1 506,39	2 151,57	386,13	551,51	29,99
												Średnia dla $\delta_{Yt(per\_thread)}$		**31,04**

Rysunek 34 *Wykresy sporządzone na podstawie danych z Tabeli 35*





4.2.2.1.6. *Zastosowanie modelu dla pętli niewzorcowej UA_diffuse_2*

Kod źródłowy pętli UA_diffuse_2 (wersja sekwencyjna) ma następującą postać:

Kod źródłowy pętli UA_diffuse_2 – wersja sekwencyjna

```
int pmorx[N], dpcmor[N], rmor[N];

for(im = 0; im < N; im++){
    pmorx[im] = dpcmor[im]*rmor[im];
    rho1 = rho1 + rmor[im]*pmorx[im];
} //endfor im
```

Wyniki uzyskane dla pętli UA_diffuse_2 zestawiono w Tabelach 36 ÷ 39 oraz zaprezentowano na wykresach (Rysunki 35 ÷ 37).
W tabelach przyjęto oznaczenia przedstawione w rozdziale 4.2.2 (patrz strona 91).

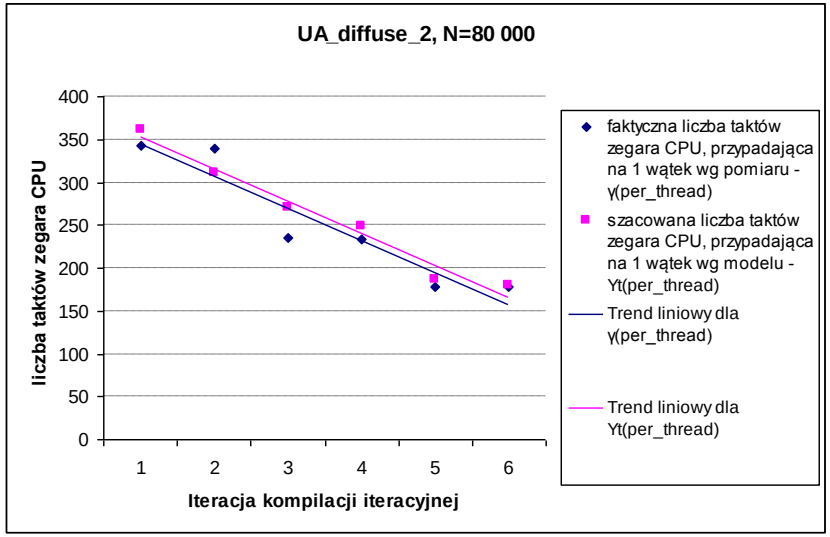
Tabela 36 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (35)) – dla pętli UA_diffuse_2

N	**Wartość statystyki testowej Kołmogorowa-Smirnowa**	**Graniczny poziom istotności *p***	**Wniosek odnośnie typu rozkładu reszt**
80 000	0,2076	0,9138	Rozkład normalny
173 333	0,2188	0,882	Rozkład normalny
266 666	0,2361	0,8234	Rozkład normalny

Tabela 37 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_2, dla N = 80 000

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	80 000	wymuszony	0,2289	0,2000	576 064,00	116,95	192 000,00	12 000	2	712,52	676,04	360,74	342,27	5,40
2	80 000	wymuszony	0,2289	0,0000	480 064,00	140,34	160 000,00	8 000	2	613,93	671,05	310,83	339,75	8,51
3	80 000	wymuszony	0,2289	0,3500	432 064,00	155,93	144 000,00	12 000	3	795,62	690,11	270,51	234,64	15,29
4	80 000	wymuszony	0,2289	0,2000	384 064,00	175,42	128 000,00	8 000	3	731,26	688,20	248,63	233,99	6,26
5	80 000	wymuszony	0,2289	0,2000	288 064,00	233,88	96 000,00	8 000	4	727,36	697,95	186,44	178,90	4,21
6	80 000	wymuszony	0,2289	0,2000	288 064,00	233,88	96 000,00	12 000	4	703,40	695,56	180,30	178,29	1,13
												Średnia dla $\delta_{Yt(per\_thread)}$		**6,80**

Rysunek 35 Wykresy sporządzone na podstawie danych z Tabeli 37



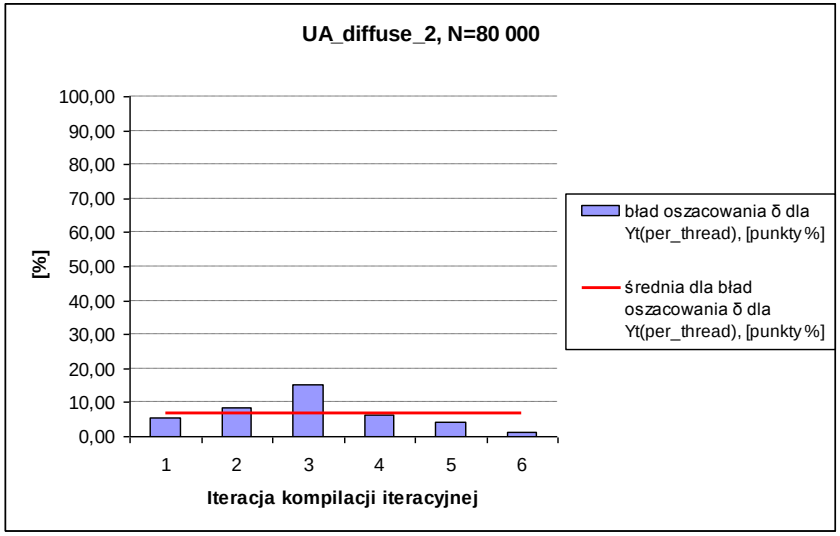
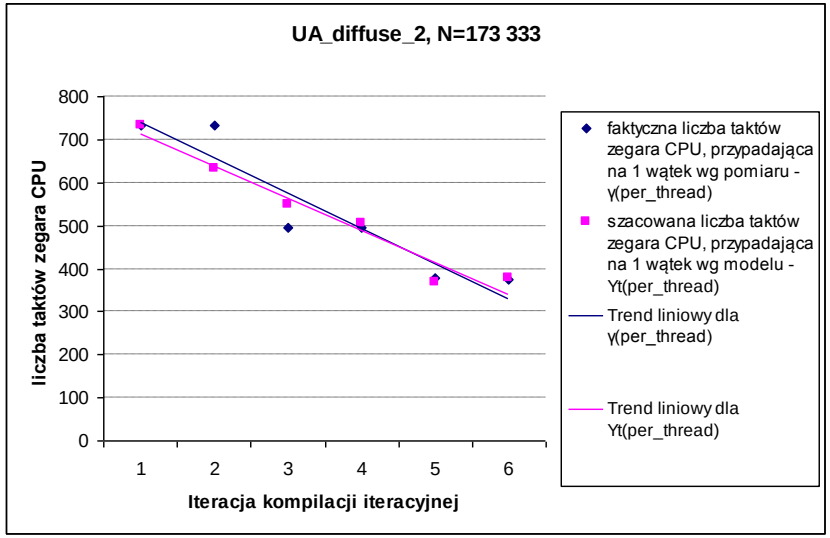


Tabela 38 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_2, dla N = 173 333

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	173 333	wymuszony	0,4959	0,2000	1 248 060,00	53,98	416 000,00	26 000	2	1 448,93	1 447,88	733,58	733,04	0,07
2	173 333	wymuszony	0,4959	0,0000	1 040 100,00	64,77	346 680,00	17 334	2	1 248,48	1 445,22	632,09	731,70	13,61
3	173 333	wymuszony	0,4959	0,3500	936 064,00	71,97	312 000,00	26 000	3	1 617,90	1 458,13	550,09	495,77	10,96
4	173 333	wymuszony	0,4959	0,2000	832 096,00	80,97	277 344,00	17 334	3	1 487,07	1 456,11	505,61	495,08	2,13
5	173 333	wymuszony	0,4959	0,2000	624 064,00	107,96	208 000,00	26 000	4	1 430,36	1 475,25	366,64	378,15	3,04
6	173 333	wymuszony	0,4959	0,2000	624 088,00	107,95	208 008,00	17 334	4	1 479,13	1 466,16	379,14	375,82	0,88
												Średnia dla $\delta_{Yt(per\_thread)}$		**5,12**

Rysunek 36 Wykresy sporządzone na podstawie danych z Tabeli 38



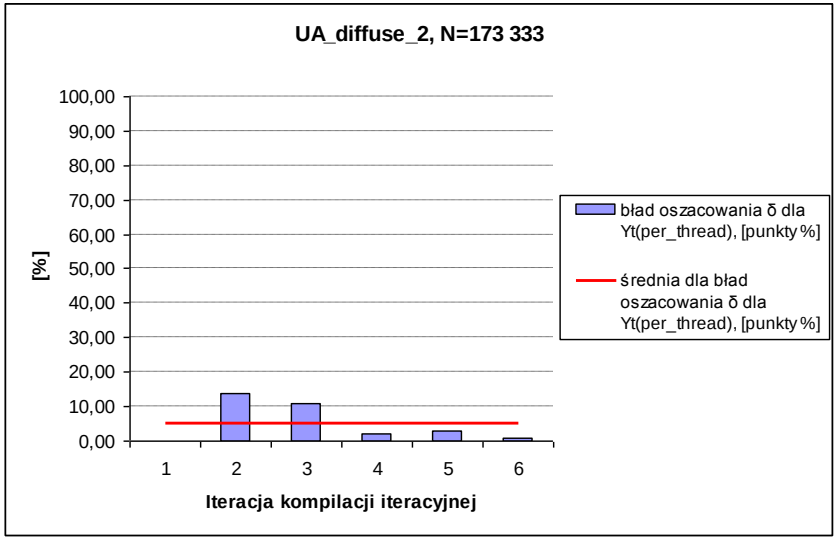
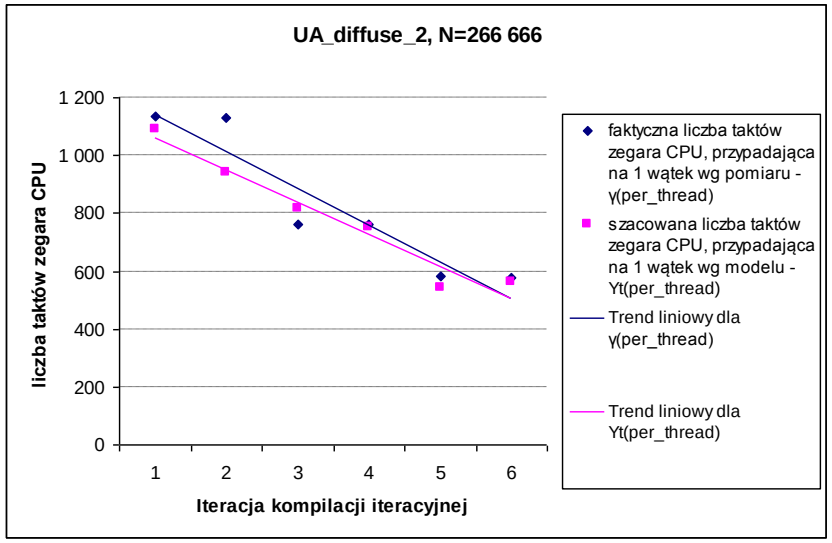
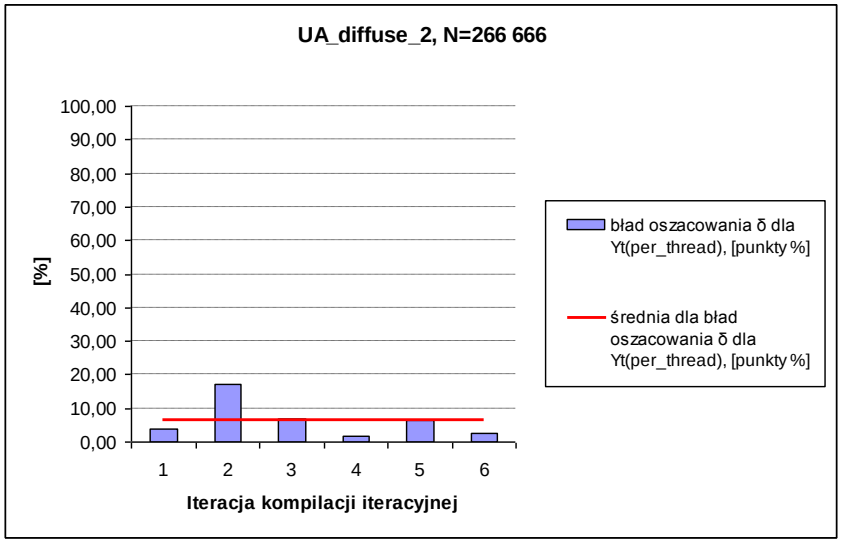


Tabela 39 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_2, dla N = 266 666

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	266 666	wymuszony	0,7629	0,2000	1 920 060,00	35,09	640 000,00	40 000	2	2 151,75	2 242,00	1 089,41	1 135,10	4,03
2	266 666	wymuszony	0,7629	0,0000	1 600 080,00	42,10	533 340,00	26 667	2	1 854,02	2 234,00	938,67	1 131,05	17,01
3	266 666	wymuszony	0,7629	0,3500	1 440 060,00	46,78	480 000,00	40 000	3	2 402,67	2 246,00	816,92	763,65	6,98
4	266 666	wymuszony	0,7629	0,2000	1 280 080,00	52,63	426 672,00	26 667	3	2 208,32	2 243,33	750,84	762,74	1,56
5	266 666	wymuszony	0,7629	0,2000	960 064,00	70,17	320 000,00	40 000	4	2 124,16	2 267,32	544,48	581,18	6,31
6	266 666	wymuszony	0,7629	0,2000	960 076,00	70,17	320 004,00	26 667	4	2 196,53	2 259,17	563,03	579,09	2,77
												Średnia dla $\delta_{Yt(per\_thread)}$		**6,44**

Rysunek 37 Wykresy sporządzone na podstawie danych z Tabeli 39





4.2.2.2. Model szczególny wyznaczony dla pętli wzorcowej matmul

4.2.2.2.1. Zastosowanie modelu dla pętli niewzorcowej UA_diffuse_3

Kod źródłowy pętli UA_diffuse_3 (wersja sekwencyjna) ma następującą postać:

Kod źródłowy pętli UA_diffuse_3 – wersja sekwencyjna

```
int tm1[N][N][N], u[N][N][N], wdtdr[N][N];

for (iz = 0; iz < N; iz++) {
    for (k = 0; k < N; k++) {
        for (j = 0; j < N; j++) {
            for (i = 0; i < N; i++) {
                tm1[iz][j][i] = tm1[iz][j][i]+wdtdr[k][i]*u[iz][j][k];
            } //endfor i
        } //endfor j
    } //endfor k
} //endfor iz
```

Wyniki uzyskane dla pętli UA_diffuse_3 zestawiono w Tabelach 40 ÷ 43 oraz zaprezentowano na wykresach (Rysunki 38 ÷ 40).
Wyniki uzyskane dla pętli UA_diffuse_3 z blokowaniem zestawiono w Tabeli 40, Tabeli 44 i Tabeli 45 oraz zaprezentowano na wykresach (Rysunek 41 i Rysunek 42).

W tabelach przyjęto oznaczenia przedstawione w rozdziale 4.2.2 (patrz strona 91).

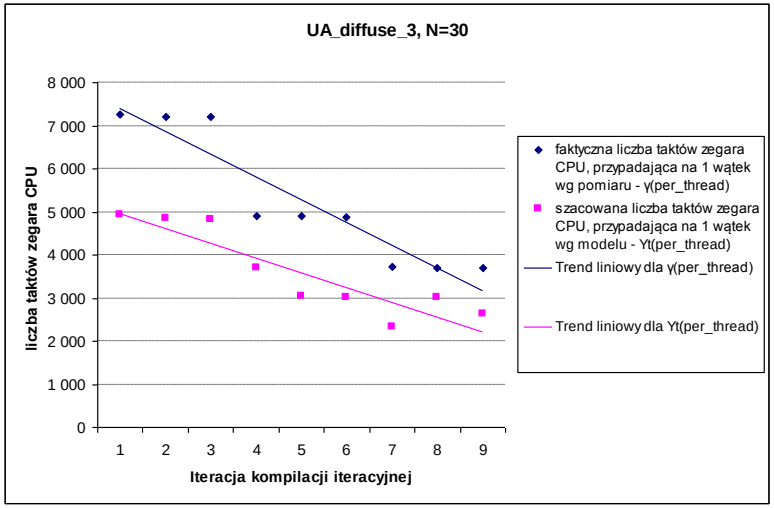
Tabela 40 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (36)) dla pętli UA_diffuse_3

N	**Wartość statystyki testowej Kołmogorowa-Smirnowa**	**Graniczny poziom istotności *p***	**Wniosek odnośnie typu rozkładu reszt**
30	0,1755	0,9017	Rozkład normalny
50	0,205	0,7741	Rozkład normalny
71	0,1721	0,9135	Rozkład normalny
30 (z blokowaniem)	0,1933	0,9465	Rozkład normalny
50 (z blokowaniem)	0,1807	0,8824	Rozkład normalny

Tabela 41 *Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_3, dla N = 30*

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	30	domyślny	0,0524	0,0000	39 284 627,28	1,71	1 012 500,00	15	2	9 608,85	14 133,60	4 929,32	7 250,51	32,01
2	30	wymuszony	0,0524	0,0000	39 284 627,28	1,71	1 012 500,00	5	2	9 457,76	14 047,33	4 851,81	7 206,25	32,67
3	30	wymuszony	0,0524	0,0000	39 284 627,28	1,71	1 012 500,00	3	2	9 388,32	14 025,56	4 816,19	7 195,08	33,06
4	30	wymuszony	0,0524	0,2000	25 943 928,08	2,60	810 000,00	3	3	10 663,32	14 138,89	3 702,00	4 908,62	24,58
5	30	domyślny	0,0524	0,0000	18 445 362,33	3,65	675 000,00	10	3	8 745,72	14 114,00	3 036,26	4 899,98	38,04
6	30	wymuszony	0,0524	0,0000	18 445 362,33	3,65	675 000,00	5	3	8 658,71	14 069,78	3 006,06	4 884,62	38,46
7	30	domyślny	0,0524	0,0667	12 134 905,07	5,55	540 000,00	8	4	8 829,71	14 161,80	2 323,69	3 726,91	37,65
8	30	wymuszony	0,0524	0,3333	18 445 362,33	3,65	675 000,00	5	4	11 422,63	14 102,22	3 006,06	3 711,24	19,00
9	30	wymuszony	0,0524	0,2000	15 137 767,20	4,45	607 500,00	3	4	10 008,97	14 089,11	2 634,03	3 707,78	28,96
												Średnia dla $\delta_{Yt(per\_thread)}$		**31,60**

Rysunek 38 *Wykresy sporządzone na podstawie danych z Tabeli 41*



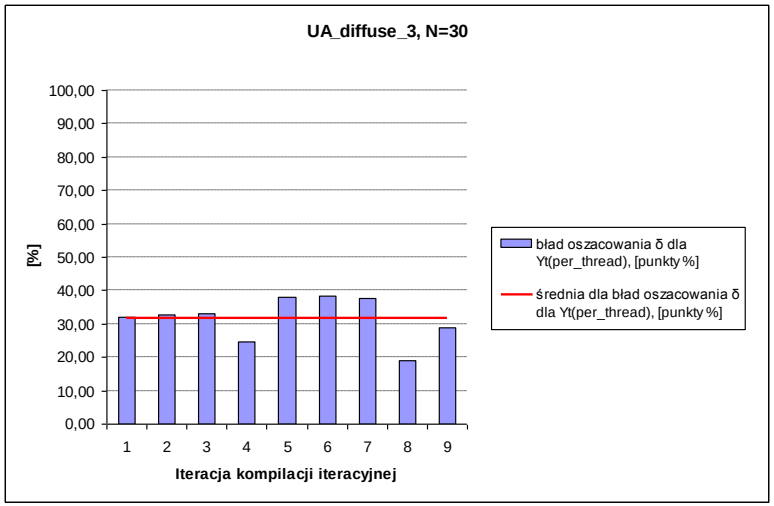
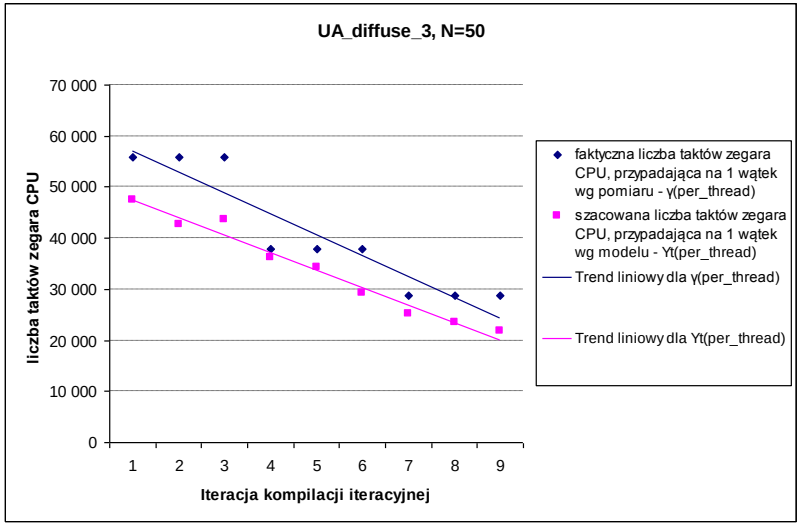


Tabela 42 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_3, dla N = 50

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	50	wymuszony	0,2408	0,1200	892 004 013,45	0,08	8 750 000,00	7	2	92 719,41	109 049,09	47 564,85	55 941,94	14,97
2	50	wymuszony	0,2408	0,0000	798 349 711,04	0,08	7 812 500,00	5	2	83 171,45	108 950,00	42 666,77	55 891,11	23,66
3	50	domyślny	0,2408	0,0000	798 349 711,04	0,08	7 812 500,00	25	2	85 125,08	108 786,67	43 668,98	55 807,32	21,75
4	50	wymuszony	0,2408	0,2600	646 442 264,04	0,10	6 562 500,00	7	3	104 002,70	109 181,82	36 106,76	37 904,80	4,74
5	50	wymuszony	0,2408	0,2000	605 428 187,04	0,11	6 250 000,00	5	3	98 450,30	109 066,67	34 179,13	37 864,82	9,73
6	50	domyślny	0,2408	0,0200	479 393 770,10	0,14	5 312 500,00	17	3	84 446,42	108 865,00	29 317,38	37 794,81	22,43
7	50	wymuszony	0,2408	0,2000	395 563 043,92	0,17	4 687 500,00	5	4	95 584,65	109 008,33	25 154,70	28 687,36	12,31
8	50	wymuszony	0,2408	0,1200	354 511 058,37	0,19	4 375 000,00	7	4	89 041,68	108 945,45	23 432,81	28 670,82	18,27
9	50	domyślny	0,2408	0,0400	314 392 752,07	0,21	4 062 500,00	13	4	82 759,48	108 920,00	21 779,54	28 664,12	24,02
												Średnia dla $\delta_{Yt(per\_thread)}$		**16,88**

Rysunek 39 Wykresy sporządzone na podstawie danych z Tabeli 42



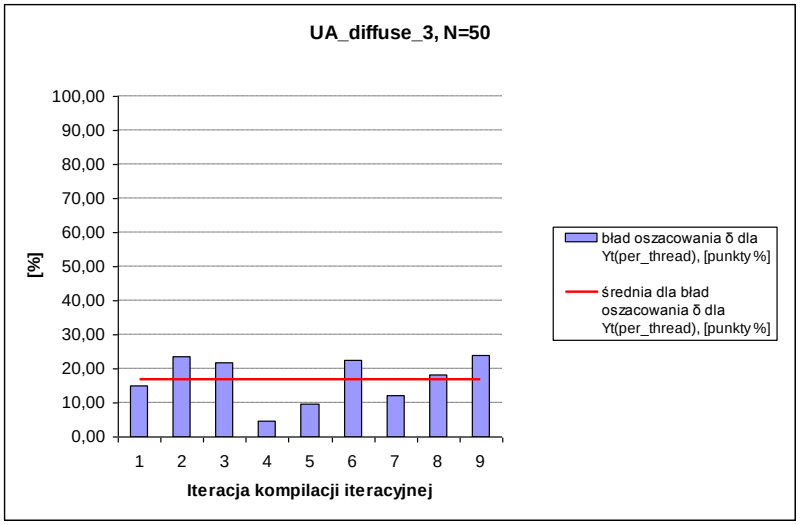
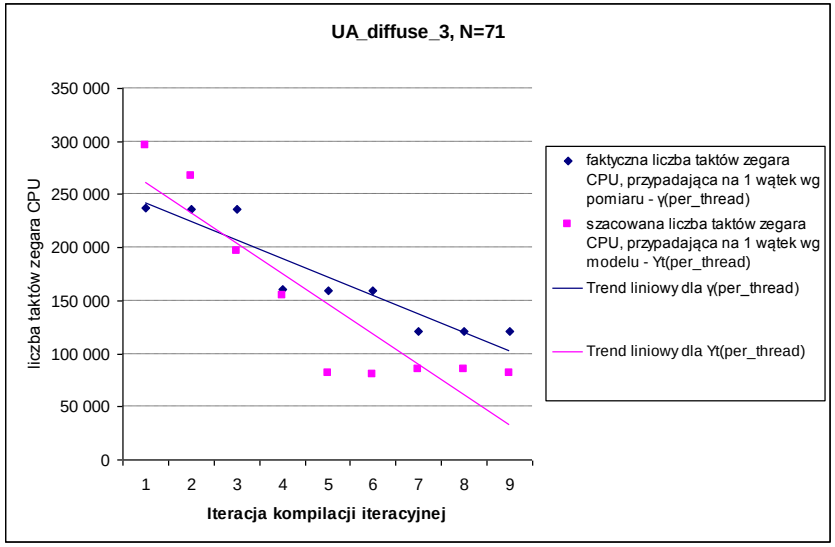


Tabela 43 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_3, dla N = 71

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	71	wymuszony	0,6875	0,2394	17 121 442 037,26	0,00	39 370 210,00	11	2	576 341,83	462 035,71	295 662,07	237 023,29	24,74
2	71	wymuszony	0,6875	0,1831	13 821 992 825,60	0,00	37 580 655,00	7	2	521 767,75	460 813,33	267 665,69	236 396,21	13,23
3	71	domyślny	0,6875	0,0141	6 229 396 844,58	0,01	32 211 990,00	36	2	382 467,05	460 433,33	196 204,74	236 201,27	16,93
4	71	wymuszony	0,6875	0,3944	3 581 245 464,51	0,02	29 527 657,50	11	3	446 023,35	461 428,57	154 846,53	160 194,79	3,34
5	71	wymuszony	0,6875	0,1831	587 744 568,88	0,11	25 053 770,00	7	3	233 125,12	460 086,67	80 934,37	159 728,92	49,33
6	71	domyślny	0,6875	0,0141	746 630 699,83	0,09	21 474 660,00	24	3	231 521,26	459 313,33	80 377,55	159 460,44	49,59
7	71	wymuszony	0,6875	0,2394	1 124 544 729,10	0,06	19 685 105,00	11	4	323 277,21	462 278,57	85 075,80	121 656,33	30,07
8	71	wymuszony	0,6875	0,1831	1 251 986 591,77	0,05	18 790 327,50	7	4	322 157,17	461 600,00	84 781,04	121 477,75	30,21
9	71	domyślny	0,6875	0,0141	1 424 227 886,45	0,05	16 105 995,00	18	4	308 282,30	460 953,33	81 129,64	121 307,57	33,12
												Średnia dla $\delta_{Yt(per\_thread)}$		**27,84**

Rysunek 40 Wykresy sporządzone na podstawie danych z Tabeli 43



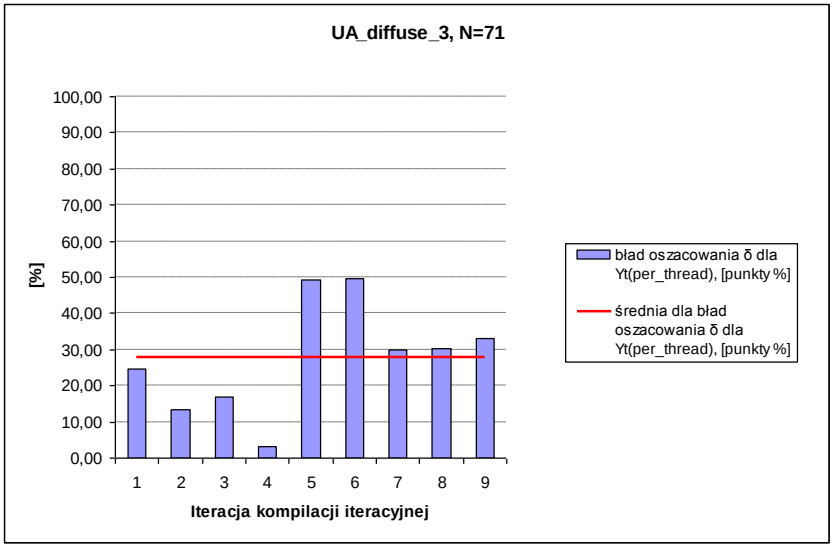
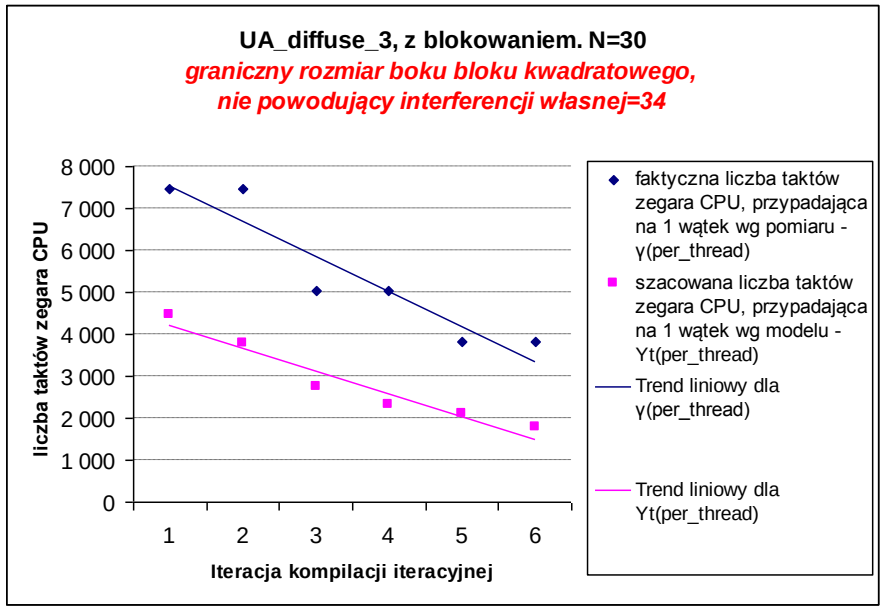


Tabela 44 *Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_3 z blokowaniem, dla N = 30*

Iteracja kompilacji iteracyjnej	N	BLOCK _SIZE	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	30	26	domyślny	0,0524	0,0000	27 888 581,66	2,42	1 012 725,00	15	2	8 675,35	14 582,50	4 450,43	7 480,79	40,51
2	30	16	domyślny	0,0524	0,0000	16 310 778,84	4,13	1 012 725,00	15	2	7 391,04	14 561,25	3 791,58	7 469,89	49,24
3	30	26	domyślny	0,0524	0,0000	12 953 604,04	5,20	675 150,00	10	3	7 870,59	14 546,25	2 732,44	5 050,04	45,89
4	30	16	domyślny	0,0524	0,0000	7 534 031,66	8,94	675 150,00	10	3	6 694,30	14 537,50	2 324,07	5 047,00	53,95
5	30	26	domyślny	0,0524	0,0667	8 512 114,29	7,91	540 120,00	8	4	7 943,43	14 579,00	2 090,45	3 836,71	45,51
6	30	16	domyślny	0,0524	0,0667	4 941 477,17	13,63	540 120,00	8	4	6 752,46	14 564,00	1 777,02	3 832,76	53,64
													Średnia dla $\delta_{Yt(per\_thread)}$		**48,12**

Rysunek 41 *Wykresy sporządzone na podstawie danych z Tabeli 44*



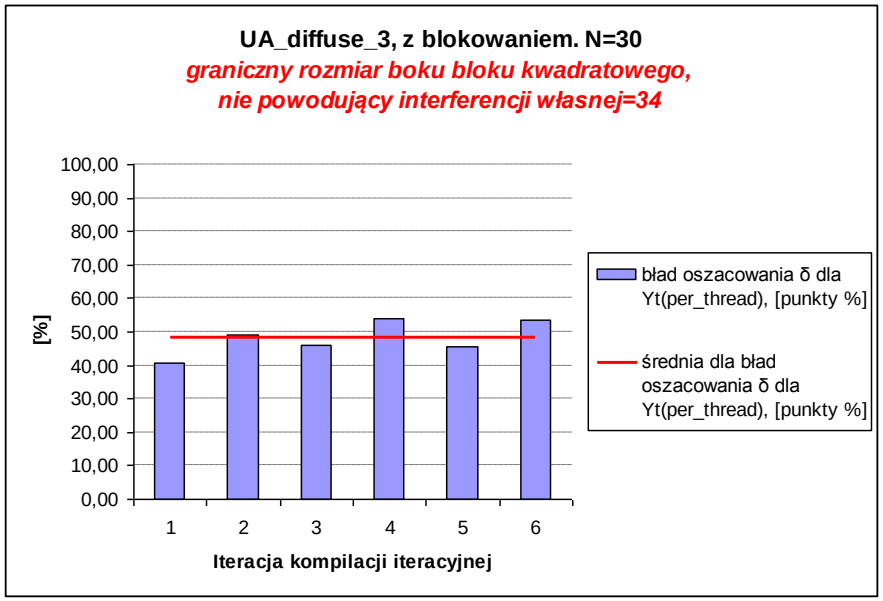
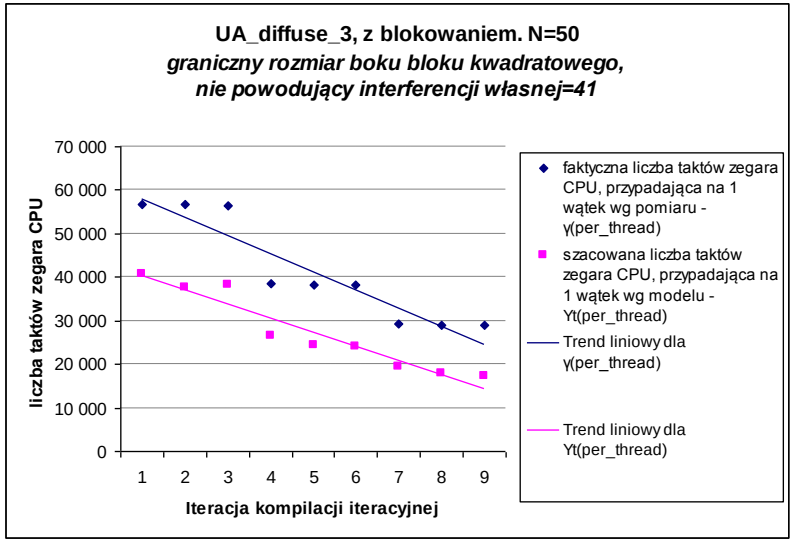
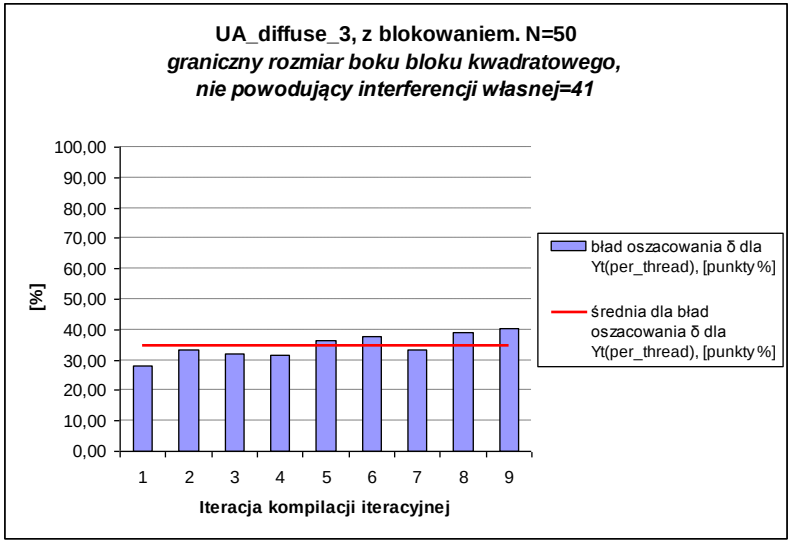


Tabela 45 *Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_3 z blokowaniem, dla N = 50*

Iteracja kompilacji iteracyjnej	N	BLOCK_ SIZE	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	50	41	domyślny	0,2408	0,0000	635 395 178,49	0,11	7 812 875,00	25	2	79 516,11	110 700,00	40 791,59	56 788,85	28,17
2	50	26	domyślny	0,2408	0,0000	488 276 705,21	0,14	7 812 875,00	25	2	73 500,62	110 350,00	37 705,66	56 609,30	33,39
3	50	32	domyślny	0,2408	0,0000	518 123 131,32	0,13	7 812 875,00	25	2	74 814,80	110 118,00	38 379,82	56 490,29	32,06
4	50	41	domyślny	0,2408	0,0200	339 634 813,23	0,20	5 312 755,00	17	3	76 187,82	111 068,00	26 450,23	38 559,63	31,40
5	50	32	domyślny	0,2408	0,0200	260 854 811,76	0,26	5 312 755,00	17	3	70 412,71	110 390,00	24 445,28	38 324,25	36,21
6	50	26	domyślny	0,2408	0,0200	242 552 537,90	0,28	5 312 755,00	17	3	68 899,24	110 350,00	23 919,84	38 310,36	37,56
7	50	41	domyślny	0,2408	0,0400	213 324 165,09	0,32	4 062 695,00	13	4	73 709,05	110 740,00	19 397,77	29 143,08	33,44
8	50	32	domyślny	0,2408	0,0400	159 780 063,29	0,42	4 062 695,00	13	4	67 612,87	110 650,00	17 793,46	29 119,40	38,89
9	50	26	domyślny	0,2408	0,0400	147 692 481,44	0,46	4 062 695,00	13	4	66 042,68	110 430,00	17 380,23	29 061,50	40,19
													Średnia dla $\delta_{Yt(per\_thread)}$		**34,59**

Rysunek 42 *Wykresy sporządzone na podstawie danych z Tabeli 45*





4.2.2.2.2. Zastosowanie modelu dla pętli niewzorcowej UA_diffuse_4

Kod źródłowy pętli UA_diffuse_4 (wersja sekwencyjna) ma następującą postać:

Kod źródłowy pętli UA_diffuse_4 – wersja sekwencyjna

```
int tm2[N][N][N], u[N][N][N], wdtdr[N][N];

for (iz = 0; iz < N; iz++) {
    for (k = 0; k < N; k++) {
        for (j = 0; j < N; j++) {
            for (i = 0; i < N; i++) {
                tm2[iz][j][i] = tm2[iz][j][i]+u[iz][k][i]*wdtdr[j][k];
            } //endfor i
        } //endfor j
    } //endfor k
} // endfor iz
```

Wyniki uzyskane dla pętli UA_diffuse_4 zestawiono w Tabelach 46 ÷ 49 oraz zaprezentowano na wykresach (Rysunki 43 ÷ 45).

Wyniki uzyskane dla pętli UA_diffuse_4 z blokowaniem zestawiono w Tabeli 46 i Tabelach 50 ÷ 52 oraz zaprezentowano na wykresach (Rysunki 46 ÷ 48).

W tabelach przyjęto oznaczenia przedstawione w rozdziale 4.2.2 (patrz strona 91).

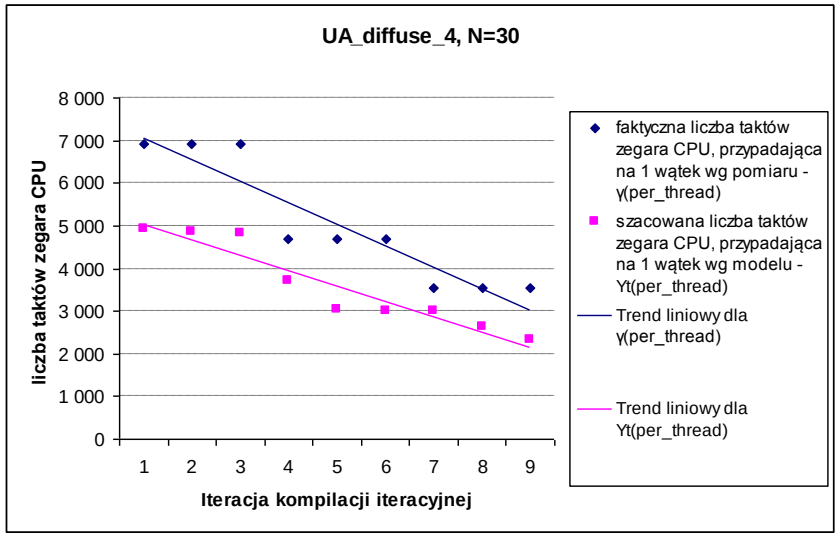
Tabela 46 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (36)) dla pętli UA_diffuse_4

N	**Wartość statystyki testowej Kołmogorowa-Smirnowa**	**Graniczny poziom istotności *p***	**Wniosek odnośnie typu rozkładu reszt**
30	0,1574	0,9546	Rozkład normalny
50	0,2073	0,7626	Rozkład normalny
71	0,1766	0,898	Rozkład normalny
30 (z blokowaniem)	0,2297	0,8463	Rozkład normalny
50 (z blokowaniem)	0,1825	0,8756	Rozkład normalny
66 (z blokowaniem)	0,2672	0,6972	Rozkład normalny

Tabela 47 *Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_4, dla N = 30*

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	30	domyślny	0,0524	0,0000	39 284 627,28	1,71	1 012 500,00	15	2	9 608,85	13 472,00	4 929,32	6 911,11	28,68
2	30	wymuszony	0,0524	0,0000	39 284 627,28	1,71	1 012 500,00	5	2	9 457,76	13 470,00	4 851,81	6 910,08	29,79
3	30	wymuszony	0,0524	0,0000	39 284 627,28	1,71	1 012 500,00	3	2	9 388,32	13 465,33	4 816,19	6 907,69	30,28
4	30	wymuszony	0,0524	0,2000	25 943 928,08	2,60	810 000,00	3	3	10 663,32	13 518,89	3 702,00	4 693,37	21,12
5	30	domyślny	0,0524	0,0000	18 445 362,33	3,65	675 000,00	10	3	8 745,72	13 512,00	3 036,26	4 690,98	35,27
6	30	wymuszony	0,0524	0,0000	18 445 362,33	3,65	675 000,00	5	3	8 658,71	13 487,78	3 006,06	4 682,57	35,80
7	30	wymuszony	0,0524	0,3333	18 445 362,33	3,65	675 000,00	5	4	11 422,63	13 517,78	3 006,06	3 557,43	15,50
8	30	wymuszony	0,0524	0,2000	15 137 767,20	4,45	607 500,00	3	4	10 008,97	13 511,33	2 634,03	3 555,73	25,92
9	30	domyślny	0,0524	0,0667	12 134 905,07	5,55	540 000,00	8	4	8 829,71	13 501,00	2 323,69	3 553,01	34,60
												Średnia dla $\delta_{Yt(per\_thread)}$		**28,55**

Rysunek 43 *Wykresy sporządzone na podstawie danych z Tabeli 47*



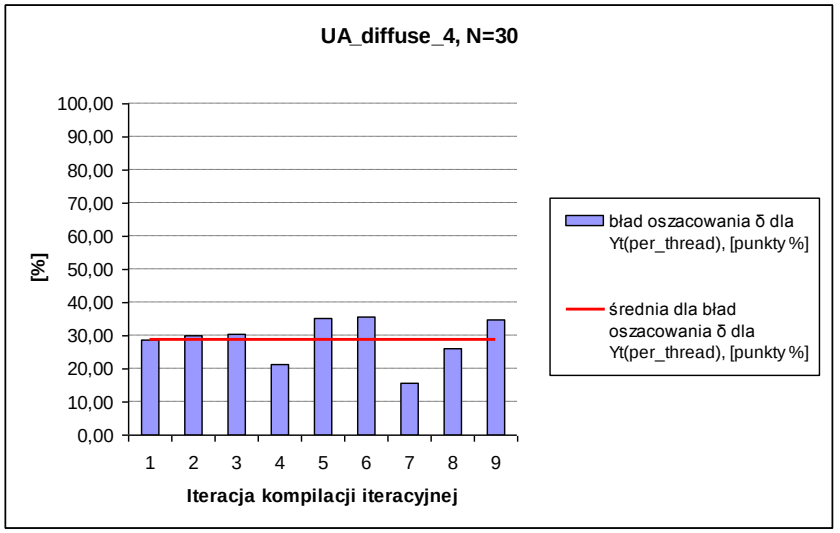
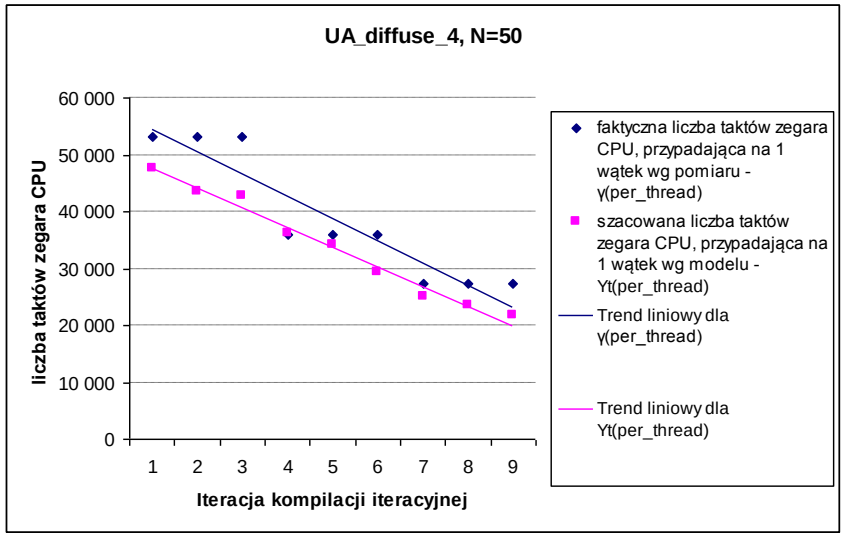


Tabela 48 *Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_4, dla N = 50*

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	50	wymuszony	0,2408	0,1200	892 004 013,45	0,08	8 750 000,00	7	2	92 719,41	103 630,00	47 564,85	53 161,96	10,53
2	50	domyślny	0,2408	0,0000	798 349 711,04	0,08	7 812 500,00	25	2	85 125,08	103 625,00	43 668,98	53 159,39	17,85
3	50	wymuszony	0,2408	0,0000	798 349 711,04	0,08	7 812 500,00	5	2	83 171,45	103 616,67	42 666,77	53 155,12	19,73
4	50	wymuszony	0,2408	0,2600	646 442 264,04	0,10	6 562 500,00	7	3	104 002,70	103 690,00	36 106,76	35 998,20	0,30
5	50	wymuszony	0,2408	0,2000	605 428 187,04	0,11	6 250 000,00	5	3	98 450,30	103 685,00	34 179,13	35 996,46	5,05
6	50	domyślny	0,2408	0,0200	479 393 770,10	0,14	5 312 500,00	17	3	84 446,42	103 653,33	29 317,38	35 985,47	18,53
7	50	wymuszony	0,2408	0,2000	395 563 043,92	0,17	4 687 500,00	5	4	95 584,65	103 805,00	25 154,70	27 318,02	7,92
8	50	wymuszony	0,2408	0,1200	354 511 058,37	0,19	4 375 000,00	7	4	89 041,68	103 802,00	23 432,81	27 317,23	14,22
9	50	domyślny	0,2408	0,0400	314 392 752,07	0,21	4 062 500,00	13	4	82 759,48	103 688,33	21 779,54	27 287,32	20,18
												Średnia dla $\delta_{Yt(per\_thread)}$		**12,70**

Rysunek 44 *Wykresy sporządzone na podstawie danych z Tabeli 48*



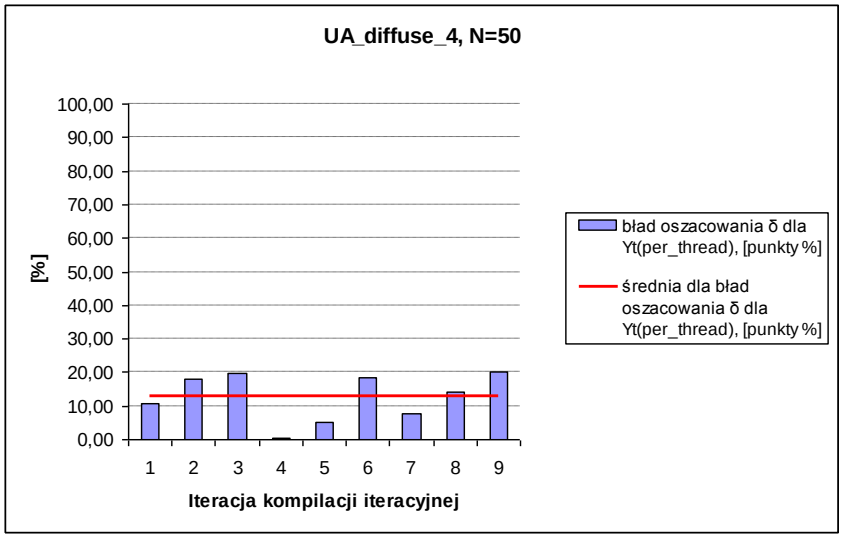
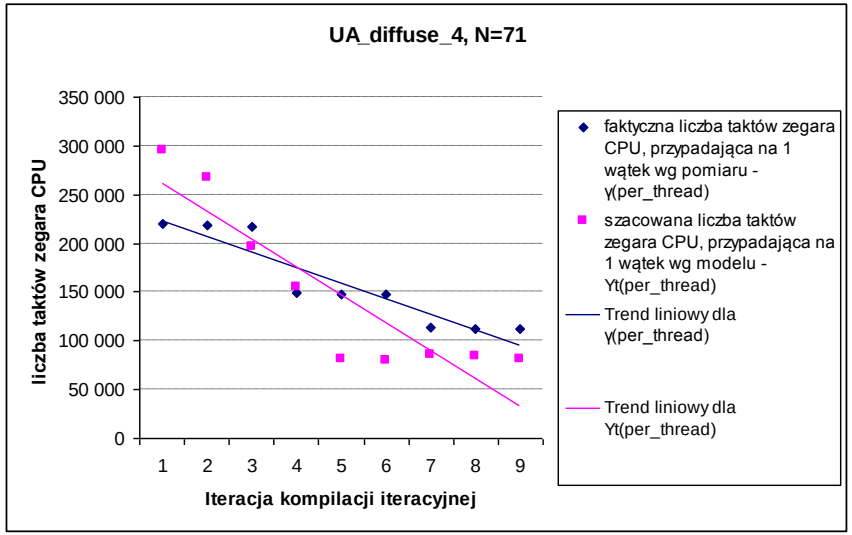


Tabela 49 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_4, dla N = 71

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	71	wymuszony	0,6875	0,2394	17 121 442 037,26	0,00	39 370 210,00	11	2	576 341,83	427 642,86	295 662,07	219 379,83	34,77
2	71	wymuszony	0,6875	0,1831	13 821 992 825,60	0,00	37 580 655,00	7	2	521 767,75	427 246,67	267 665,69	219 176,59	22,12
3	71	domyślny	0,6875	0,0141	6 229 396 844,58	0,01	32 211 990,00	36	2	382 467,05	423 246,67	196 204,74	217 124,59	9,63
4	71	wymuszony	0,6875	0,3944	3 581 245 464,51	0,02	29 527 657,50	11	3	446 023,35	427 807,14	154 846,53	148 522,39	4,26
5	71	wymuszony	0,6875	0,1831	587 744 568,88	0,11	25 053 770,00	7	3	233 125,12	426 920,00	80 934,37	148 214,40	45,39
6	71	domyślny	0,6875	0,0141	746 630 699,83	0,09	21 474 660,00	24	3	231 521,26	423 293,33	80 377,55	146 955,32	45,30
7	71	wymuszony	0,6875	0,2394	1 124 544 729,10	0,06	19 685 105,00	11	4	323 277,21	431 021,43	85 075,80	113 430,49	25,00
8	71	wymuszony	0,6875	0,1831	1 251 986 591,77	0,05	18 790 327,50	7	4	322 157,17	427 780,00	84 781,04	112 577,45	24,69
9	71	domyślny	0,6875	0,0141	1 424 227 886,45	0,05	16 105 995,00	18	4	308 282,30	423 706,67	81 129,64	111 505,49	27,24
												Średnia dla $\delta_{Yt(per\_thread)}$		**26,49**

Rysunek 45 Wykresy sporządzone na podstawie danych z Tabeli 49



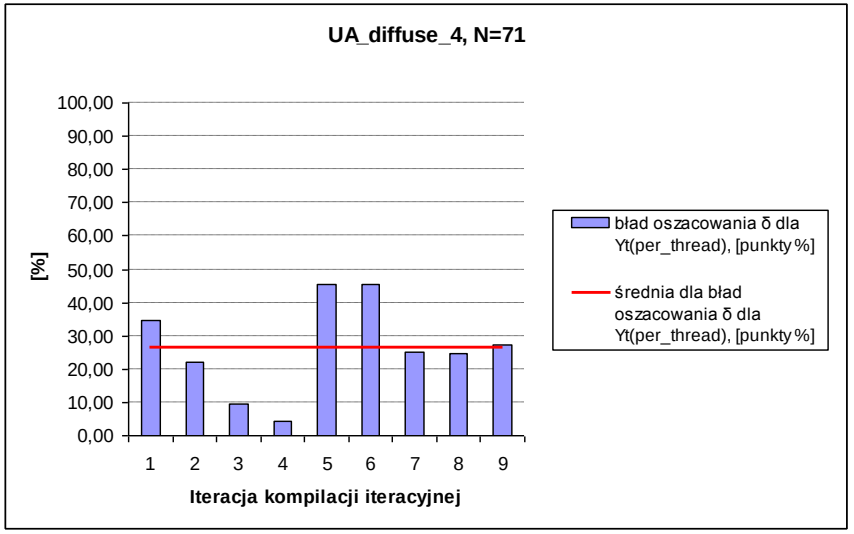
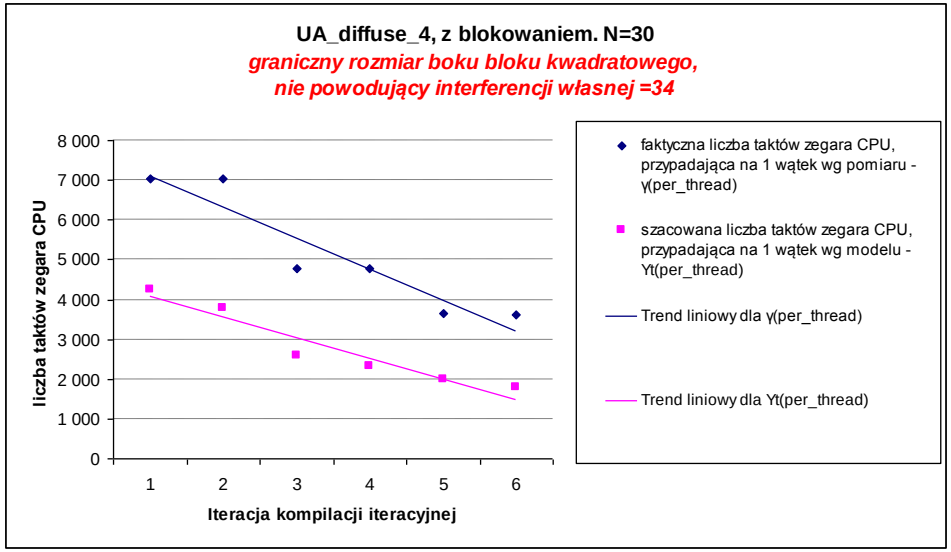


Tabela 50 *Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_4 z blokowaniem, dla N = 30*

Iteracja kompilacji iteracyjnej	N	BLOCK _SIZE	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	30	24	domyślny	0,0524	0,0000	23 728 671,16	2,84	1 012 725,00	15	2	8 266,71	13 746,00	4 240,80	7 051,67	39,86
2	30	16	domyślny	0,0524	0,0000	16 310 778,84	4,13	1 012 725,00	15	2	7 391,04	13 707,00	3 791,58	7 031,66	46,08
3	30	24	domyślny	0,0524	0,0000	11 032 549,03	6,11	675 150,00	10	3	7 502,12	13 785,00	2 604,52	4 785,76	45,58
4	30	16	domyślny	0,0524	0,0000	7 534 031,66	8,94	675 150,00	10	3	6 694,30	13 772,00	2 324,07	4 781,24	51,39
5	30	24	domyślny	0,0524	0,0667	7 244 730,53	9,30	540 120,00	8	4	7 569,98	13 935,20	1 992,17	3 667,28	45,68
6	30	16	domyślny	0,0524	0,0667	4 941 477,17	13,63	540 120,00	8	4	6 752,46	13 805,20	1 777,02	3 633,07	51,09
													Średnia dla $\delta_{Yt(per\_thread)}$		**46,61**

Rysunek 46 *Wykresy sporządzone na podstawie danych z Tabeli 50*



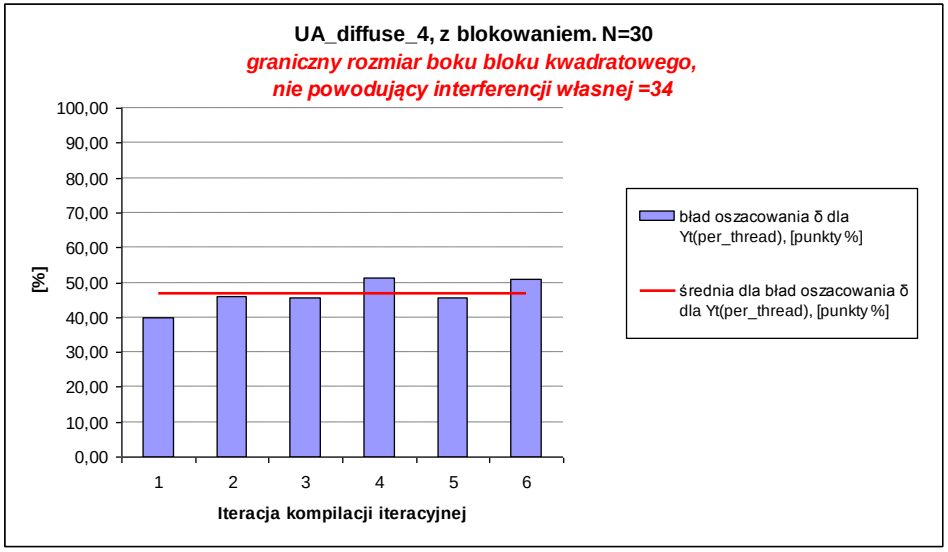
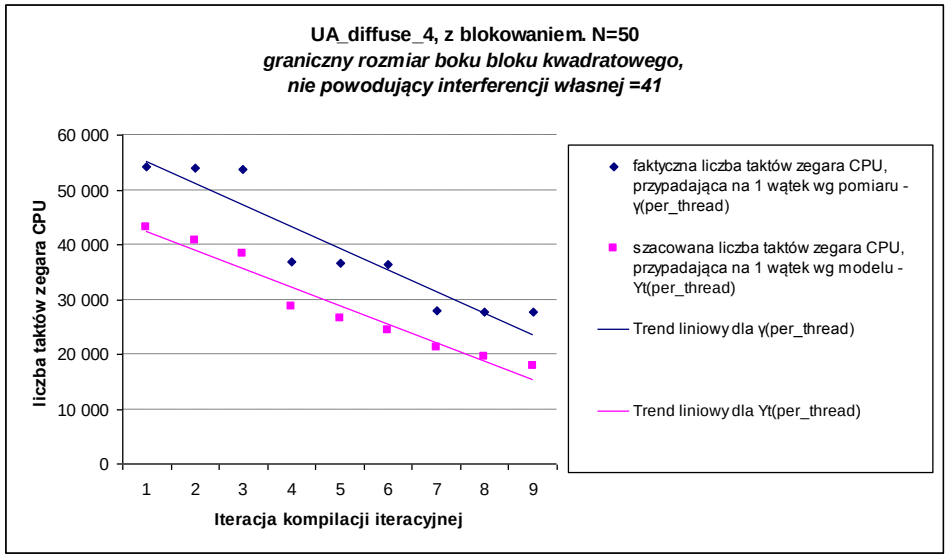


Tabela 51 *Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_4 z blokowaniem, dla N = 50*

Iteracja kompilacji iteracyjnej	N	BLOCK_SIZE	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	50	16	domyślny	0,2408	0,0000	761 658 692,79	0,09	7 813 750,00	25	2	83 945,55	105 758,33	43 063,88	54 253,79	20,63
2	50	41	domyślny	0,2408	0,0000	635 395 178,49	0,11	7 812 875,00	25	2	79 516,11	104 986,67	40 791,59	53 857,93	24,26
3	50	32	domyślny	0,2408	0,0000	518 123 131,32	0,13	7 812 875,00	25	2	74 814,80	104 755,00	38 379,82	53 739,08	28,58
4	50	16	domyślny	0,2408	0,0200	443 166 425,97	0,15	5 313 350,00	17	3	82 495,73	105 945,00	28 640,15	36 781,07	22,13
5	50	41	domyślny	0,2408	0,0200	339 634 813,23	0,20	5 312 755,00	17	3	76 187,82	105 812,50	26 450,23	36 735,07	28,00
6	50	32	domyślny	0,2408	0,0200	260 854 811,76	0,26	5 312 755,00	17	3	70 412,71	104 955,00	24 445,28	36 437,37	32,91
7	50	16	domyślny	0,2408	0,0400	287 458 470,24	0,23	4 063 150,00	13	4	80 582,85	106 398,33	21 206,72	28 000,50	24,26
8	50	41	domyślny	0,2408	0,0400	213 324 165,09	0,32	4 062 695,00	13	4	73 709,05	105 670,83	19 397,77	27 809,05	30,25
9	50	32	domyślny	0,2408	0,0400	159 780 063,29	0,42	4 062 695,00	13	4	67 612,87	105 063,33	17 793,46	27 649,17	35,65
													Średnia dla $\delta_{Yt(per\_thread)}$		**27,41**

Rysunek 47 *Wykresy sporządzone na podstawie danych z Tabeli 51*



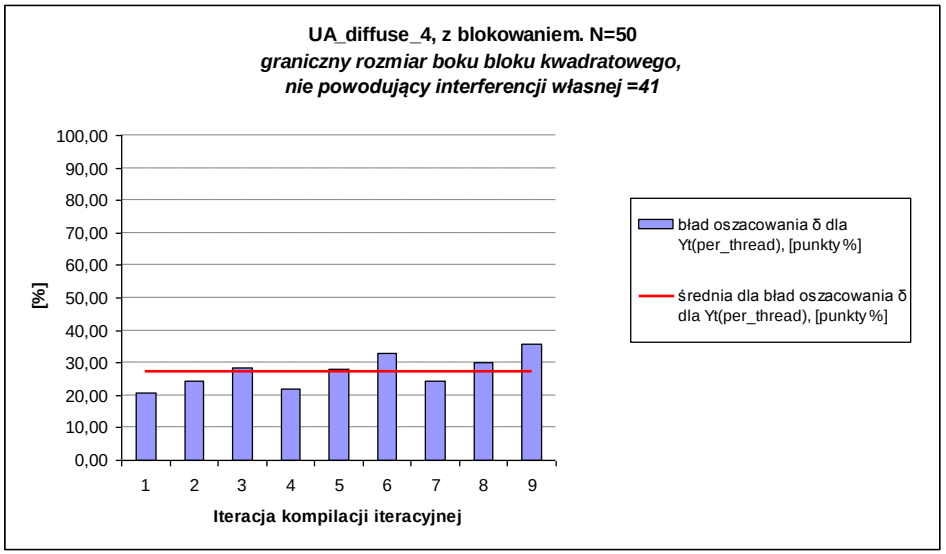
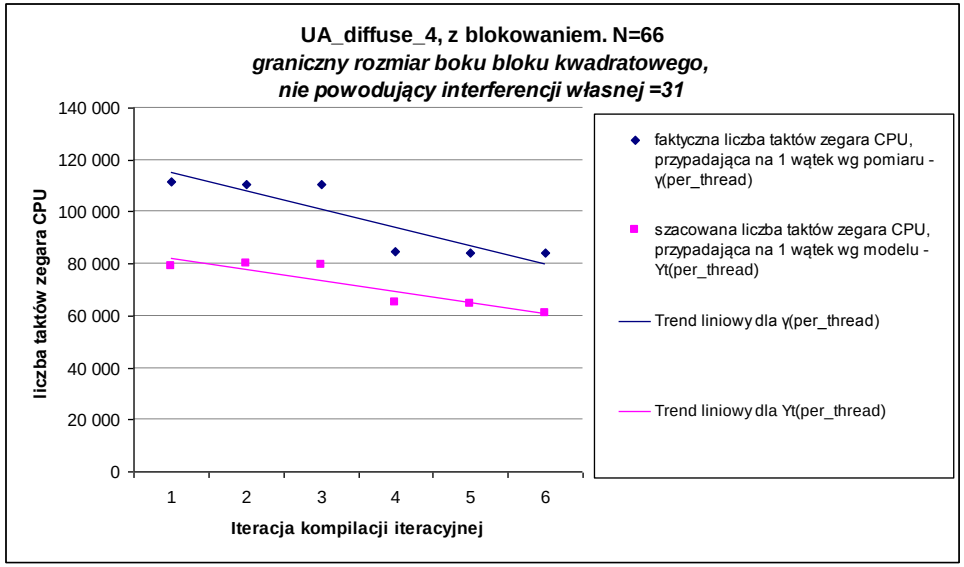
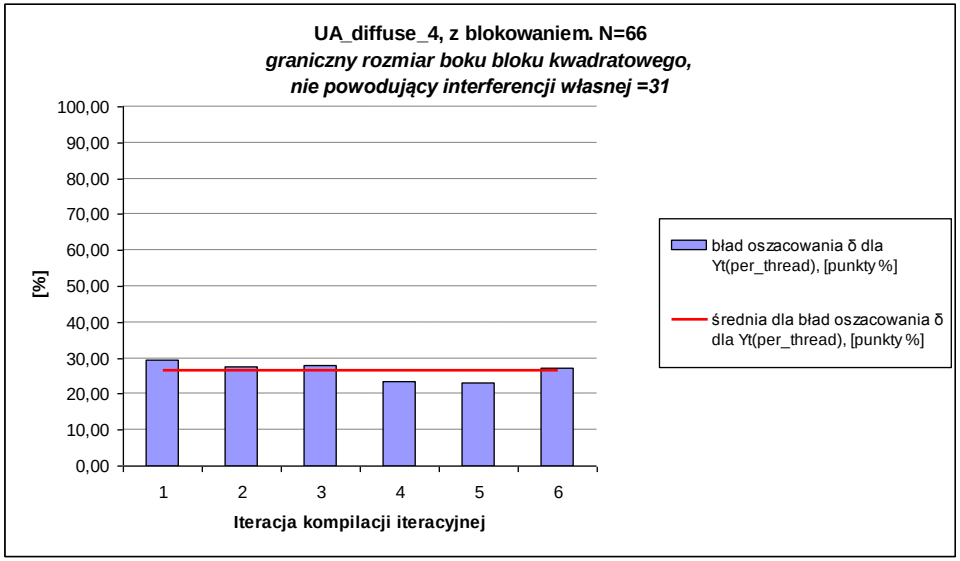


Tabela 52 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_4 z blokowaniem, dla N = 66

Iteracja kompilacji iteracyjnej	N	BLOCK _SIZE	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	66	16	domyślny	0,5525	0,0000	1 335 124 603,24	0,05	15 813 930,00	22	3	227 276,67	321 485,00	78 903,95	111 610,39	29,30
2	66	24	domyślny	0,5525	0,0000	1 399 615 140,73	0,05	15 812 940,00	22	3	230 492,72	318 530,00	80 020,47	110 584,49	27,64
3	66	31	domyślny	0,5525	0,0000	1 365 069 792,07	0,05	15 812 940,00	22	3	228 778,52	318 125,00	79 425,35	110 443,89	28,09
4	66	16	domyślny	0,5525	0,0303	1 208 069 017,73	0,06	12 219 855,00	17	4	246 852,72	322 255,00	64 963,42	84 806,79	23,40
5	66	31	domyślny	0,5525	0,0303	1 184 556 667,91	0,06	12 219 090,00	17	4	245 398,17	319 555,00	64 580,63	84 096,24	23,21
6	66	24	domyślny	0,5525	0,0303	987 872 177,95	0,07	12 219 090,00	17	4	232 443,65	319 035,00	61 171,43	83 959,39	27,14
													Średnia dla $\delta_{Yt(per\_thread)}$		**26,46**

Rysunek 48 Wykresy sporządzone na podstawie danych z Tabeli 52





4.2.2.2.3. Zastosowanie modelu dla pętli niewzorcowej UA_transfer_11

Kod źródłowy pętli UA_transfer_11 (wersja sekwencyjna) ma następującą postać:

Kod źródłowy pętli UA_transfer_11 – wersja sekwencyjna

```
int tmp[N][N], qbnew[2][N][N-2], tmor[N][N];

for (col = 0; col < N; col++) {
    i = 0;
    tmp[col][i] = tmor[col][i];

    for (i = 1; i < N-1; i++) {
        for (j = 0; j < N; j++) {
            tmp[col][i] = tmp[col][i] + qbnew[0][j][i-1]*tmor[col][j];
        } //endfor j
    } //endfor i

} //endfor col
```

Wyniki uzyskane dla pętli UA_transfer_11 zestawiono w Tabelach 53 ÷ 56 oraz zaprezentowano na wykresach (Rysunki 49 ÷ 51).
Wyniki uzyskane dla pętli UA_transfer_11 z blokowaniem zestawiono w Tabeli 53, Tabeli 57 i Tabeli 58 oraz zaprezentowano na wykresach (Rysunek 52 i Rysunek 53).

W tabelach przyjęto oznaczenia przedstawione w rozdziale 4.2.2 (patrz strona 91).

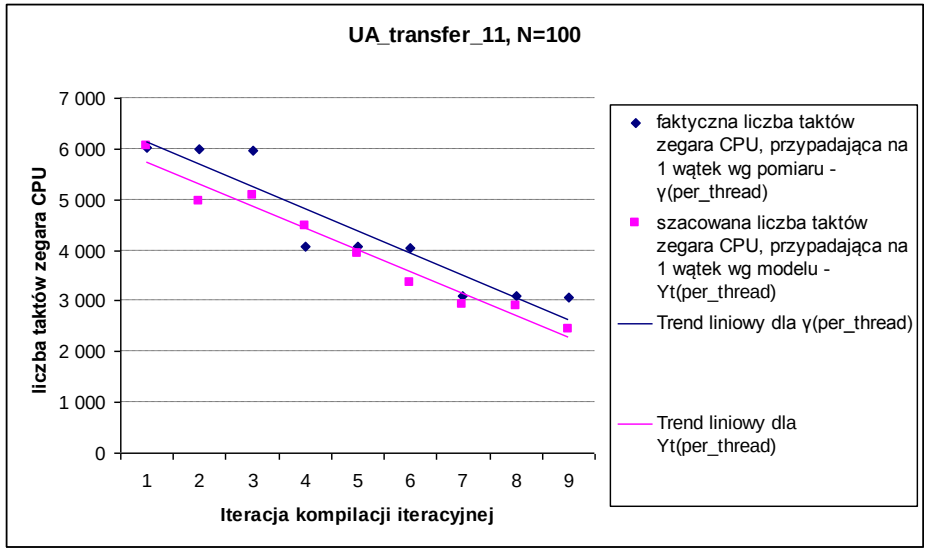
Tabela 53 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (36)) dla pętli UA_transfer_11

N	**Wartość statystyki testowej Kołmogorowa-Smirnowa**	**Graniczny poziom istotności *p***	**Wniosek odnośnie typu rozkładu reszt**
100	0,1954	0,82	Rozkład normalny
267	0,2062	0,768	Rozkład normalny
433	0,2205	0,696	Rozkład normalny
100 (z blokowaniem)	0,2683	0,4583	Rozkład normalny
267 (z blokowaniem)	0,1644	0,985	Rozkład normalny

Tabela 54 *Wyniki uzyskane dla pętli niewzorcowej UA_transfer_11, dla N = 100*

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	100	wymuszony	0,0378	0,2000	35 874 187,78	1,88	1 470 120,00	15	2	11 800,75	11 747,00	6 053,76	6 026,18	0,46
2	100	wymuszony	0,0378	0,0000	27 420 226,29	2,46	1 225 100,00	10	2	9 663,18	11 708,40	4 957,19	6 006,38	17,47
3	100	domyślny	0,0378	0,0000	27 420 226,29	2,46	1 225 100,00	50	2	9 890,16	11 645,60	5 073,63	5 974,17	15,07
4	100	wymuszony	0,0378	0,3500	23 545 785,80	2,86	1 102 590,00	15	3	12 850,80	11 722,00	4 461,43	4 069,54	9,63
5	100	wymuszony	0,0378	0,2000	19 912 078,50	3,38	980 080,00	10	3	11 291,21	11 714,00	3 919,98	4 066,77	3,61
6	100	domyślny	0,0378	0,0200	15 875 478,96	4,24	833 068,00	34	3	9 704,98	11 664,60	3 369,29	4 049,62	16,80
7	100	wymuszony	0,0378	0,2000	13 384 164,51	5,03	735 060,00	15	4	11 120,70	11 730,80	2 926,60	3 087,16	5,20
8	100	wymuszony	0,0378	0,2000	13 384 164,51	5,03	735 060,00	10	4	11 055,84	11 723,20	2 909,53	3 085,16	5,69
9	100	domyślny	0,0378	0,0000	10 498 684,41	6,42	612 550,00	25	4	9 299,18	11 687,60	2 447,24	3 075,79	20,44
												Średnia dla $\delta_{Yt(per\_thread)}$		**10,49**

Rysunek 49 *Wykresy sporządzone na podstawie danych z Tabeli 54*



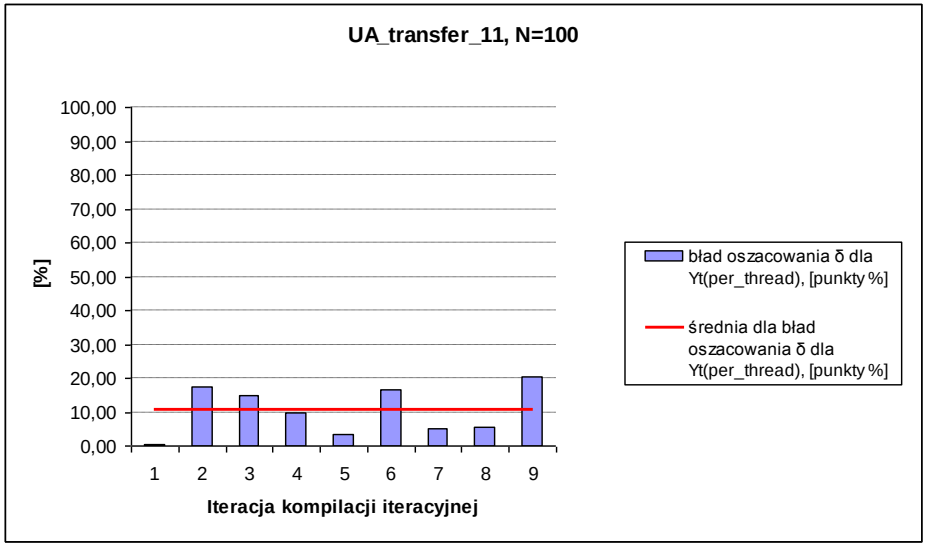
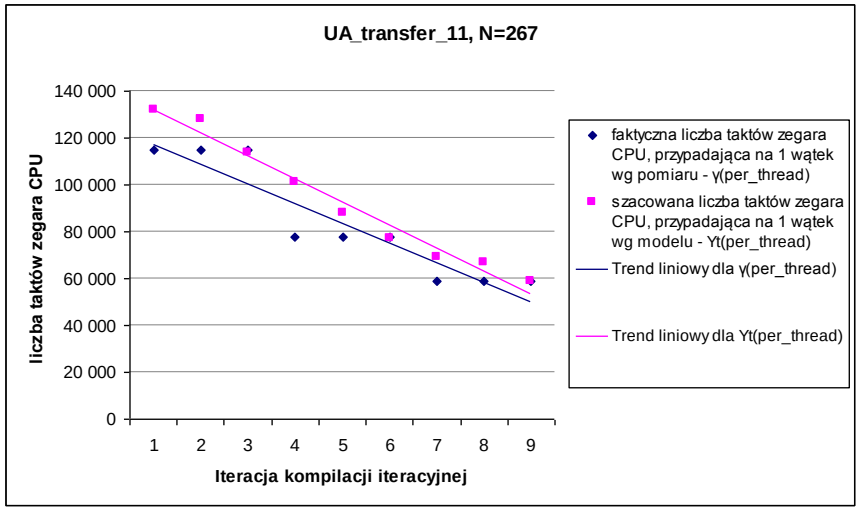


Tabela 55 *Wyniki uzyskane dla pętli niewzorcowej UA_transfer_11, dla N = 267*

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	267	wymuszony	0,2709	0,1985	2 153 974 931,62	0,03	28 302 320,00	40	2	257 301,97	223 850,00	131 995,34	114 834,55	14,94
2	267	wymuszony	0,2709	0,1685	2 098 268 329,00	0,03	27 594 762,00	26	2	249 739,34	223 833,33	128 115,72	114 826,00	11,57
3	267	domyślny	0,2709	0,0037	1 792 192 726,39	0,04	23 703 193,00	134	2	221 884,77	223 666,67	113 826,39	114 740,50	0,80
4	267	wymuszony	0,2709	0,3483	1 598 036 461,77	0,04	21 226 740,00	40	3	290 642,23	223 900,00	100 902,66	77 731,67	29,81
5	267	wymuszony	0,2709	0,1685	1 377 172 683,62	0,05	18 396 508,00	26	3	252 697,34	223 860,00	87 729,28	77 717,78	12,88
6	267	domyślny	0,2709	0,0000	1 171 490 281,33	0,06	15 743 165,50	89	3	222 400,19	223 833,33	77 210,98	77 708,52	0,64
7	267	wymuszony	0,2709	0,1985	1 048 881 791,97	0,06	14 151 160,00	40	4	262 553,33	224 083,33	69 095,29	58 971,27	17,17
8	267	wymuszony	0,2709	0,1685	1 021 727 916,43	0,07	13 797 381,00	26	4	254 834,30	224 073,33	67 063,90	58 968,64	13,73
9	267	domyślny	0,2709	0,0037	873 043 410,47	0,08	11 851 596,50	67	4	224 186,09	223 813,33	58 998,31	58 900,22	0,17
												Średnia dla $\delta_{Yt(per\_thread)}$		**11,30**

Rysunek 50 *Wykresy sporządzone na podstawie danych z Tabeli 55*



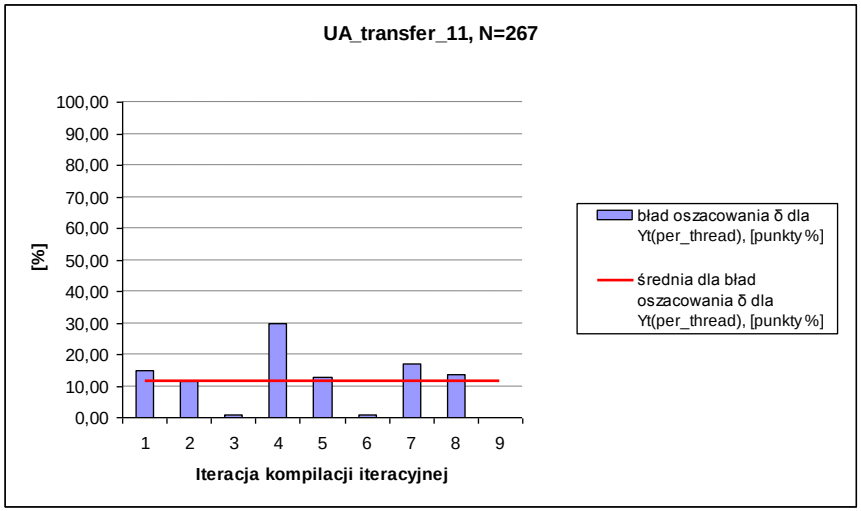
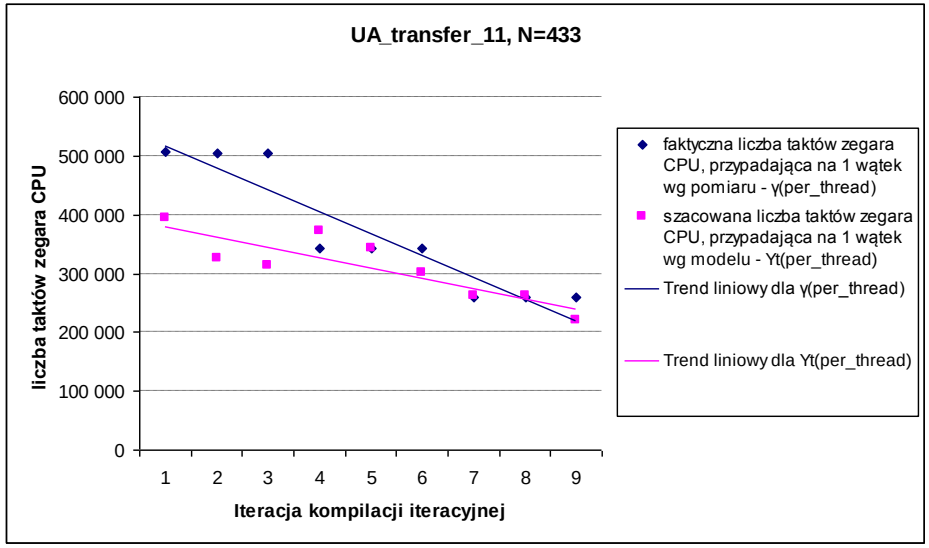


Tabela 56 *Wyniki uzyskane dla pętli niewzorcowej UA_transfer_11, dla N = 433*

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	433	domyślny	0,7136	0,0023	5 389 124 530,15	0,01	101 243 411,50	217	2	767 834,47	989 300,00	393 897,37	507 508,69	22,39
2	433	wymuszony	0,7136	0,1824	2 133 804 395,82	0,03	119 439 232,00	64	2	634 170,26	985 550,00	325 327,93	505 584,95	35,65
3	433	wymuszony	0,7136	0,1917	1 883 454 398,61	0,04	120 372 351,00	43	2	610 426,40	982 750,00	313 147,38	504 148,55	37,89
4	433	wymuszony	0,7136	0,3303	6 065 217 853,10	0,01	89 579 424,00	64	3	1 069 951,18	986 350,00	371 456,41	342 432,48	8,48
5	433	wymuszony	0,7136	0,1917	5 976 108 903,31	0,01	80 248 234,00	43	3	988 905,50	985 483,33	343 319,67	342 131,59	0,35
6	433	domyślny	0,7136	0,0046	5 178 209 361,25	0,01	67 651 127,50	145	3	866 808,37	985 100,00	300 931,04	341 998,51	12,01
7	433	wymuszony	0,7136	0,1917	4 442 949 687,61	0,02	60 186 175,50	43	4	997 889,93	987 050,00	262 611,41	259 758,70	1,10
8	433	wymuszony	0,7136	0,1824	4 392 385 891,54	0,02	59 719 616,00	64	4	995 362,55	986 950,00	261 946,29	259 732,38	0,85
9	433	domyślny	0,7136	0,0069	3 364 058 968,24	0,02	50 854 985,50	109	4	837 897,23	986 150,00	220 506,66	259 521,85	15,03
												Średnia dla $\delta_{Yt(per\_thread)}$		**14,86**

Rysunek 51 *Wykresy sporządzone na podstawie danych z Tabeli 56*



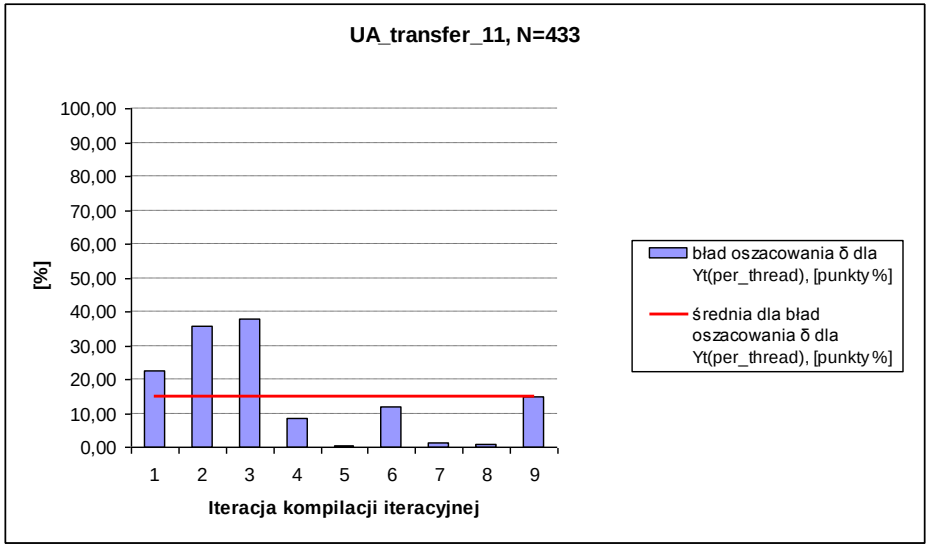
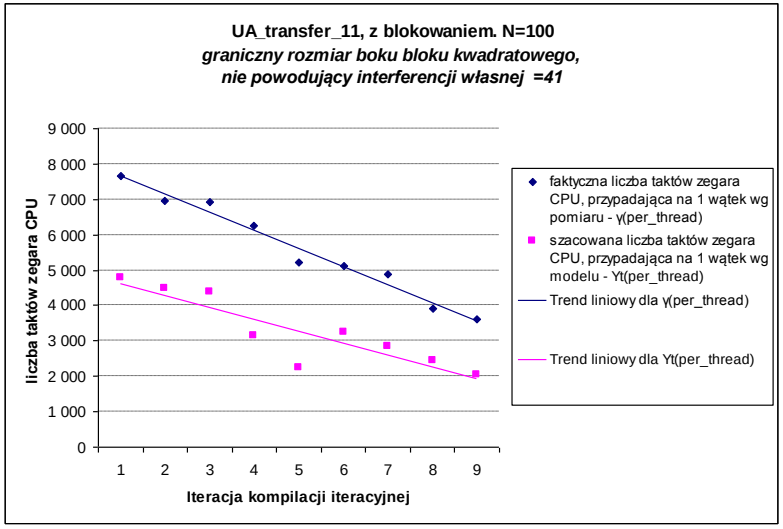


Tabela 57 *Wyniki uzyskane dla pętli niewzorcowej UA_transfer_11 z blokowaniem, dla N = 100*

Iteracja kompilacji iteracyjnej	N	BLOCK _SIZE	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	100	16	domyślny	0,0378	0,0000	22 274 193,03	3,02	1 232 450,00	50	2	9 329,55	14 954,75	4 786,04	7 671,75	37,61
2	100	32	domyślny	0,0378	0,0000	18 017 983,38	3,74	1 227 800,00	50	2	8 736,31	13 595,50	4 481,71	6 974,46	35,74
3	100	41	domyślny	0,0378	0,0000	16 863 338,90	4,00	1 226 750,00	50	2	8 560,62	13 473,50	4 391,58	6 911,88	36,46
4	100	16	domyślny	0,0378	0,0200	12 424 208,39	5,42	838 066,00	34	3	9 053,50	17 987,75	3 143,12	6 244,83	49,67
5	100	16	domyślny	0,0378	0,0000	7 791 996,30	8,65	616 225,00	25	4	8 538,63	19 836,00	2 247,08	5 220,17	56,95
6	100	32	domyślny	0,0378	0,0200	14 114 936,65	4,77	834 904,00	34	3	9 383,04	14 761,75	3 257,52	5 124,86	36,44
7	100	41	domyślny	0,0378	0,0200	9 012 803,76	7,48	834 190,00	34	3	8 202,01	14 031,25	2 847,50	4 871,25	41,54
8	100	32	domyślny	0,0378	0,0000	10 398 235,57	6,48	613 900,00	25	4	9 285,26	14 840,00	2 443,57	3 905,39	37,43
9	100	41	domyślny	0,0378	0,0000	5 593 068,63	12,05	613 375,00	25	4	7 711,18	13 756,50	2 029,33	3 620,25	43,95
													Średnia dla $\delta_{Yt(per\_thread)}$		**41,76**

Rysunek 52 *Wykresy sporządzone na podstawie danych z Tabeli 57*



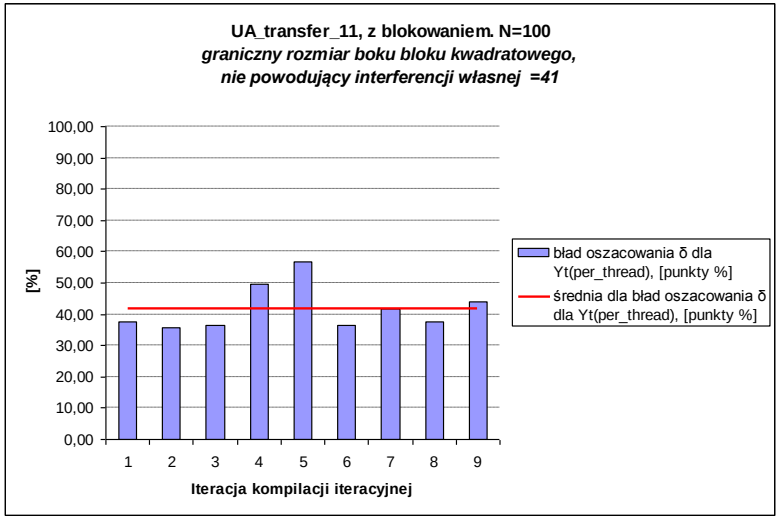
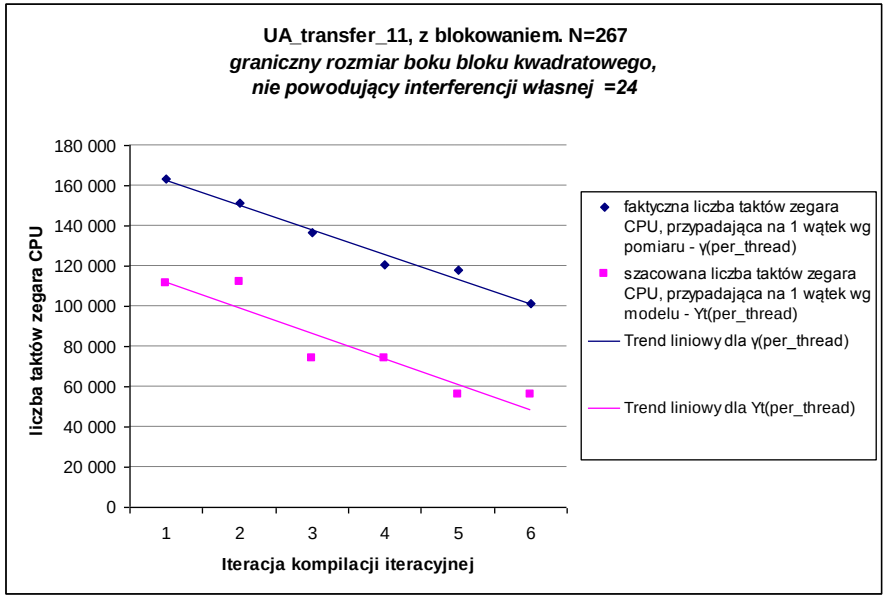
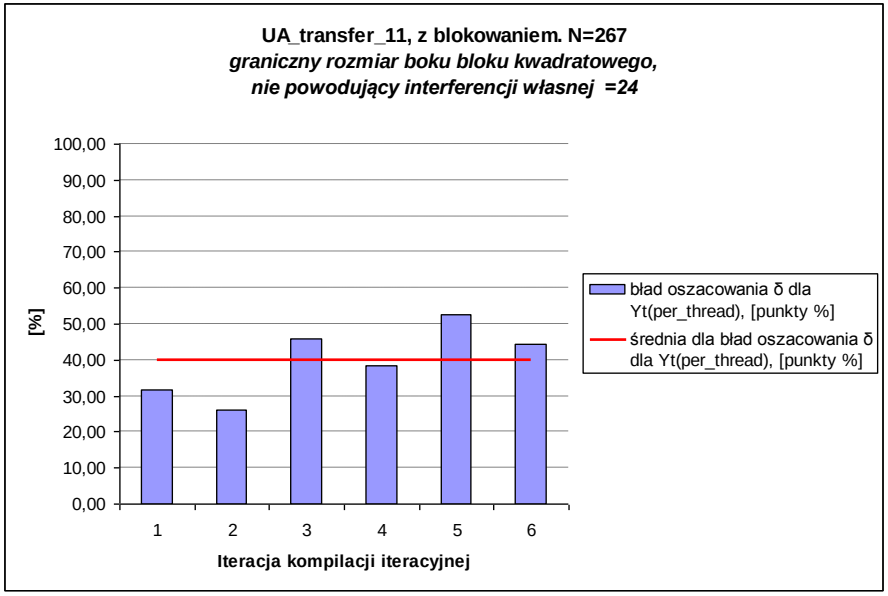


Tabela 58 *Wyniki uzyskane dla pętli niewzorcowej UA_transfer_11 z blokowaniem, dla N = 267*

Iteracja kompilacji iteracyjnej	N	BLOCK_SIZE	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	267	16	domyślny	0,2709	0,0037	1 648 388 629,80	0,04	23 807 981,00	134	2	217 006,27	318 508,57	111 323,73	163 394,19	31,87
2	267	24	domyślny	0,2709	0,0037	1 679 719 950,18	0,04	23 757 061,00	134	2	217 938,92	295 382,86	111 802,18	151 530,75	26,22
3	267	16	domyślny	0,2709	0,0000	1 007 610 359,75	0,07	15 812 763,50	89	3	213 197,27	394 582,86	74 015,99	136 987,87	45,97
4	267	24	domyślny	0,2709	0,0000	1 025 634 945,45	0,07	15 778 943,50	89	3	214 043,32	346 777,14	74 309,71	120 391,09	38,28
5	267	16	domyślny	0,2709	0,0037	718 889 832,84	0,09	11 903 990,50	67	4	212 129,77	448 948,57	55 825,49	118 148,32	52,75
6	267	24	domyślny	0,2709	0,0037	729 751 082,53	0,09	11 875 530,50	67	4	212 764,14	383 965,71	55 992,44	101 046,99	44,59
													Średnia dla $\delta_{Yt(per\_thread)}$		**39,94**

Rysunek 53 *Wykresy sporządzone na podstawie danych z Tabeli 58*





4.2.2.2.4. Zastosowanie modelu dla pętli niewzorcowej UA_transfer_16

Kod źródłowy pętli UA_transfer_16 (wersja sekwencyjna) ma następującą postać:

Kod źródłowy pętli UA_transfer_16 – wersja sekwencyjna

```
int tmor[N][N], temp[N][N], qbnew[2][N][N-2];

for (col = 1; col < N; col++) {
    tmor[col][0] = tmor[col][0]+temp[col][0];

    for (j = 0; j < N; j++) {
        for (i = 1; i < N-1; i++) {
           tmor[col][j] = tmor[col][j] + qbnew[0][j][i-1] *temp[col][i];
        } //endfor i
    } //endfor j
} //endfor col
```

Wyniki uzyskane dla pętli UA_transfer_16 zestawiono w Tabelach 59 ÷ 62 oraz zaprezentowano na wykresach (Rysunki 54 ÷ 56).
Wyniki uzyskane dla pętli UA_transfer_16 z blokowaniem zestawiono w Tabeli 59, Tabeli 63 i Tabeli 64 oraz zaprezentowano na wykresach (Rysunek 57 i Rysunek 58).

W tabelach przyjęto oznaczenia przedstawione w rozdziale 4.2.2 (patrz strona 91).

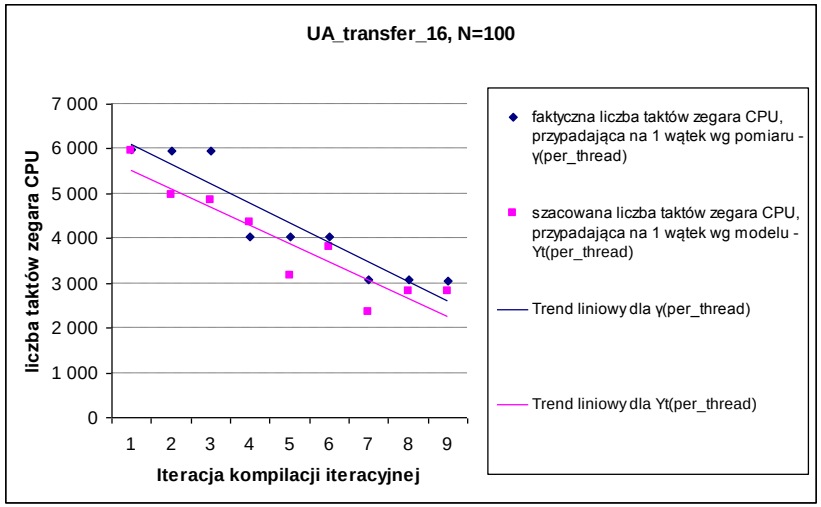
Tabela 59 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (36)) dla pętli UA_transfer_16

N	**Wartość statystyki testowej Kołmogorowa-Smirnowa**	**Graniczny poziom istotności *p***	**Wniosek odnośnie typu rozkładu reszt**
100	0,1612	0,9453	Rozkład normalny
267	0,2191	0,7031	Rozkład normalny
433	0,2269	0,6639	Rozkład normalny
100 (z blokowaniem)	0,185	0,8654	Rozkład normalny
267 (z blokowaniem)	0,1401	0,9895	Rozkład normalny

Tabela 60 *Wyniki uzyskane dla pętli niewzorcowej UA_transfer_16, dla N = 100*

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	100	wymuszony	0,0378	0,2121	34 987 200,98	1,93	1 445 559,00	15	2	11 590,39	11 668,00	5 945,85	5 985,66	0,67
2	100	domyślny	0,0378	0,0101	26 626 285,77	2,53	1 200 549,00	50	2	9 680,73	11 630,00	4 966,19	5 966,16	16,76
3	100	wymuszony	0,0378	0,0101	26 626 285,77	2,53	1 200 549,00	10	2	9 458,56	11 585,20	4 852,22	5 943,18	18,36
4	100	wymuszony	0,0378	0,3636	22 799 647,95	2,95	1 078 044,00	15	3	12 550,31	11 646,00	4 357,11	4 043,16	7,76
5	100	domyślny	0,0378	0,0000	14 609 660,12	4,61	784 032,00	33	3	9 111,62	11 642,00	3 163,29	4 041,77	21,73
6	100	wymuszony	0,0378	0,2121	19 214 604,44	3,51	955 539,00	10	3	10 996,28	11 638,00	3 817,59	4 040,38	5,51
7	100	domyślny	0,0378	0,0101	9 952 429,14	6,77	588 024,00	25	4	8 921,63	11 678,00	2 347,88	3 073,26	23,60
8	100	wymuszony	0,0378	0,2121	12 786 625,65	5,27	710 529,00	15	4	10 740,21	11 645,00	2 826,47	3 064,58	7,77
9	100	wymuszony	0,0378	0,2121	12 786 625,65	5,27	710 529,00	10	4	10 677,57	11 643,00	2 809,98	3 064,05	8,29
												Średnia dla $\delta_{Yt(per\_thread)}$		**12,27**

Rysunek 54 *Wykresy sporządzone na podstawie danych z Tabeli 60*



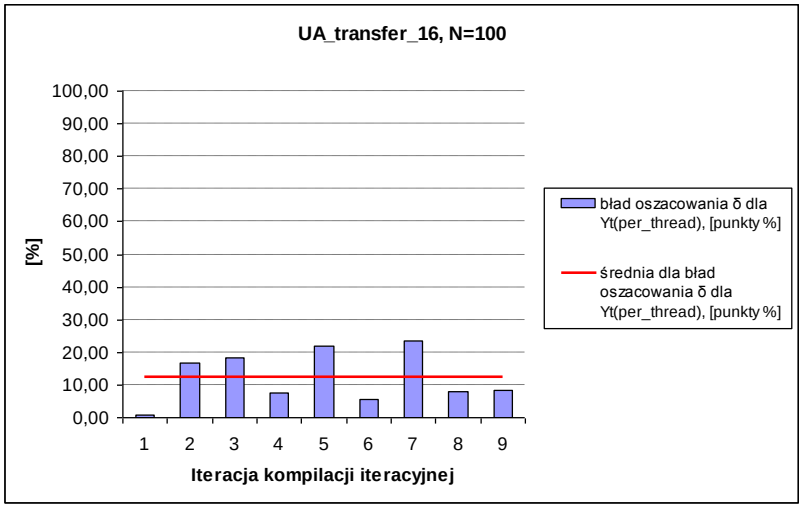
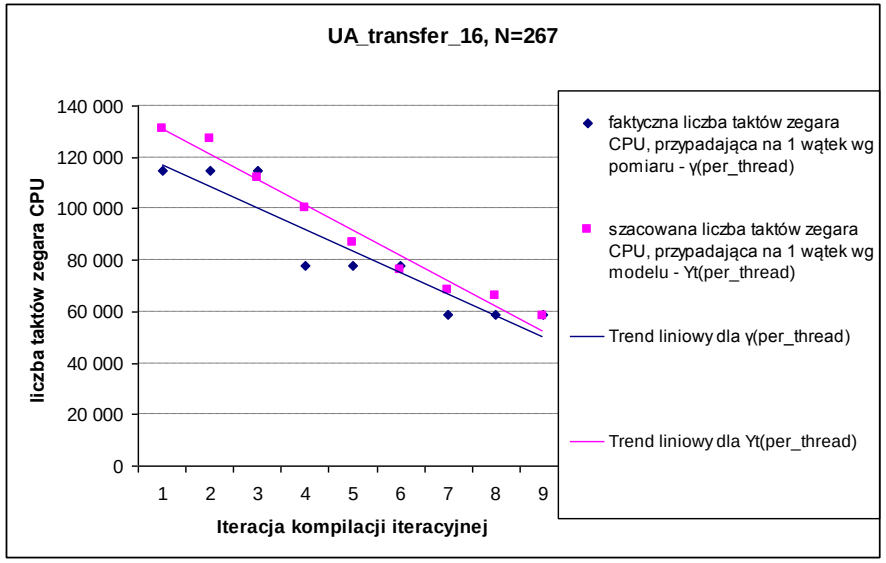


Tabela 61 *Wyniki uzyskane dla pętli niewzorcowej UA_transfer_16, dla N = 267*

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	267	wymuszony	0,2709	0,2030	2 140 047 910,93	0,03	28 125 271,50	40	2	255 800,72	223 740,00	131 225,20	114 778,12	14,33
2	267	wymuszony	0,2709	0,1729	2 084 342 626,32	0,03	27 417 717,50	26	2	248 244,47	223 700,00	127 348,86	114 757,60	10,97
3	267	domyślny	0,2709	0,0000	1 764 416 632,25	0,04	23 349 282,00	133	2	218 765,72	223 646,67	112 226,33	114 730,24	2,18
4	267	wymuszony	0,2709	0,3534	1 584 196 283,42	0,04	21 049 731,50	40	3	288 377,89	223 456,00	100 116,55	77 577,52	29,05
5	267	wymuszony	0,2709	0,1729	1 363 414 662,46	0,05	18 219 515,50	26	3	250 426,00	223 420,00	86 940,74	77 565,03	12,09
6	267	domyślny	0,2709	0,0038	1 157 835 027,97	0,06	15 566 188,00	89	3	220 065,41	223 416,67	76 400,41	77 563,87	1,50
7	267	wymuszony	0,2709	0,2030	1 035 300 459,32	0,07	13 974 191,50	40	4	259 488,39	223 636,00	68 288,70	58 853,55	16,03
8	267	wymuszony	0,2709	0,1729	1 008 164 279,82	0,07	13 620 414,50	26	4	251 783,57	223 420,00	66 261,05	58 796,71	12,70
9	267	domyślny	0,2709	0,0075	859 585 457,14	0,08	11 674 641,00	67	4	221 064,16	223 416,67	58 176,73	58 795,83	1,05
												Średnia dla $\delta_{Yt(per\_thread)}$		**11,10**

Rysunek 55 *Wykresy sporządzone na podstawie danych z Tabeli 61*



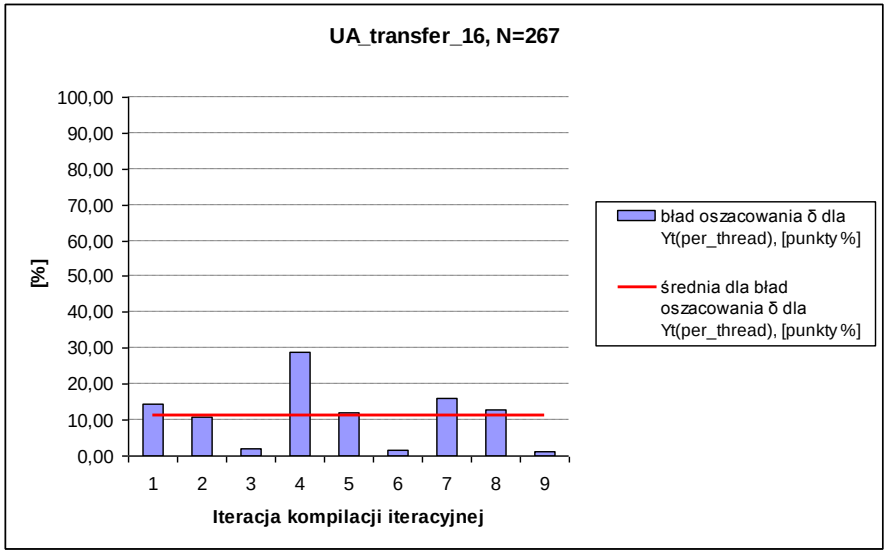
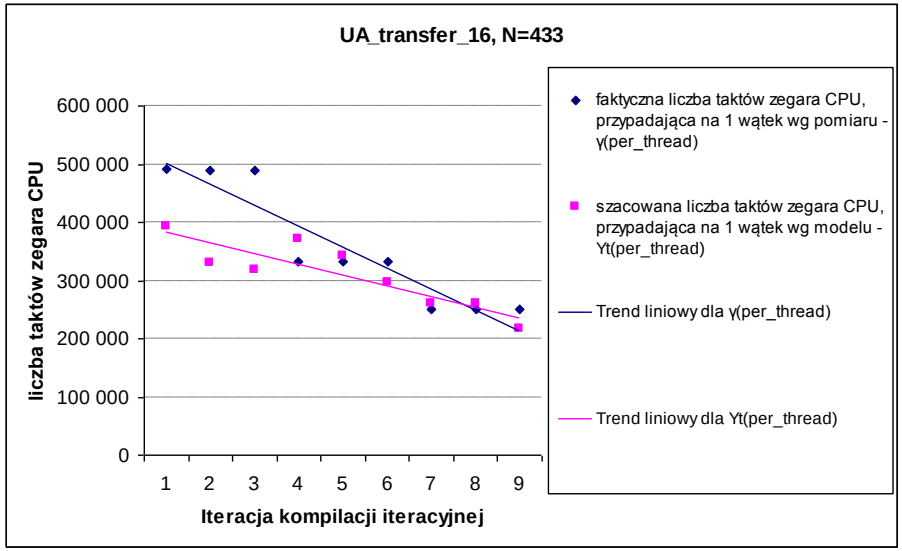


Tabela 62 Wyniki uzyskane dla pętli niewzorcowej UA_transfer_16, dla N = 433

Iteracja kompilacji iteracyjnej	N	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	433	domyślny	0,7136	0,0000	5 479 792 756,14	0,01	100 310 077,50	216	2	767 174,55	956 600,00	393 558,83	490 733,66	19,80
2	433	wymuszony	0,7136	0,1852	2 255 702 684,95	0,03	118 972 417,50	64	2	643 208,48	955 616,67	329 964,51	490 229,22	32,69
3	433	wymuszony	0,7136	0,1944	2 009 725 483,53	0,03	119 905 534,50	43	2	620 866,83	954 750,00	318 503,29	489 784,62	34,97
4	433	wymuszony	0,7136	0,3333	6 072 952 569,82	0,01	89 112 673,50	64	3	1 066 876,51	955 083,33	370 388,97	331 577,58	11,71
5	433	wymuszony	0,7136	0,1944	5 959 148 535,67	0,01	79 781 503,50	43	3	984 478,01	955 050,00	341 782,57	331 566,01	3,08
6	433	domyślny	0,7136	0,0000	5 094 963 736,02	0,01	66 717 865,50	144	3	855 095,12	954 466,67	296 864,54	331 363,50	10,41
7	433	wymuszony	0,7136	0,1944	4 392 385 891,54	0,02	59 719 488,00	43	4	989 667,26	955 833,33	260 447,48	251 543,51	3,54
8	433	wymuszony	0,7136	0,1852	4 341 372 107,80	0,02	59 252 929,50	64	4	987 053,49	955 700,00	259 759,62	251 508,42	3,28
9	433	domyślny	0,7136	0,0000	3 250 781 691,74	0,02	49 921 759,50	108	4	819 733,26	955 640,00	215 726,50	251 492,63	14,22
												Średnia dla $\delta_{Yt(per\_thread)}$		**14,86**

Rysunek 56 Wykresy sporządzone na podstawie danych z Tabeli 62



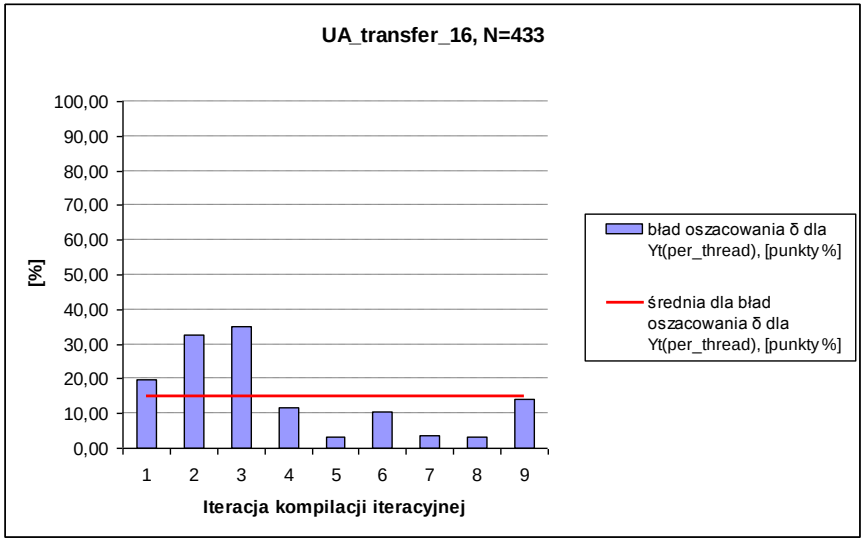
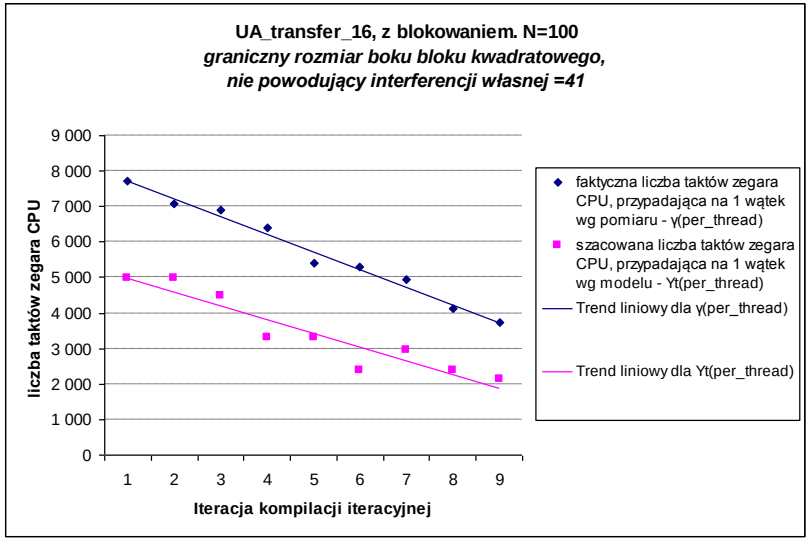


Tabela 63 *Wyniki uzyskane dla pętli niewzorcowej UA_transfer_16 z blokowaniem, dla N = 100*

Iteracja kompilacji iteracyjnej	N	BLOCK _SIZE	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	100	16	domyślny	0,0378	0,0000	25 424 637,06	2,65	1 234 500,00	50	2	9 715,65	15 034,34	4 984,11	7 712,58	35,38
2	100	32	domyślny	0,0378	0,0000	25 424 696,59	2,65	1 228 350,00	50	2	9 685,44	13 819,30	4 968,61	7 089,27	29,91
3	100	41	domyślny	0,0378	0,0000	18 250 705,82	3,69	1 227 000,00	50	2	8 766,30	13 468,01	4 497,09	6 909,06	34,91
4	100	16	domyślny	0,0378	0,0200	14 605 561,71	4,61	839 460,00	34	3	9 511,52	18 408,98	3 302,13	6 391,07	48,33
5	100	32	domyślny	0,0378	0,0200	14 605 561,71	4,61	835 278,00	34	3	9 481,94	15 538,95	3 291,86	5 394,68	38,98
6	100	16	domyślny	0,0378	0,0000	9 603 715,70	7,02	617 250,00	25	4	9 098,24	20 097,64	2 394,35	5 289,03	54,73
7	100	41	domyślny	0,0378	0,0200	10 114 186,28	6,66	834 360,00	34	3	8 490,46	14 243,10	2 947,65	4 944,80	40,39
8	100	32	domyślny	0,0378	0,0000	9 603 715,70	7,02	614 175,00	25	4	9 069,94	15 728,62	2 386,91	4 139,25	42,33
9	100	41	domyślny	0,0378	0,0000	6 470 806,96	10,41	613 500,00	25	4	8 055,38	14 161,84	2 119,91	3 726,92	43,12
													Średnia dla $\delta_{Yt(per\_thread)}$		**40,90**

Rysunek 57 *Wykresy sporządzone na podstawie danych z Tabeli 63*



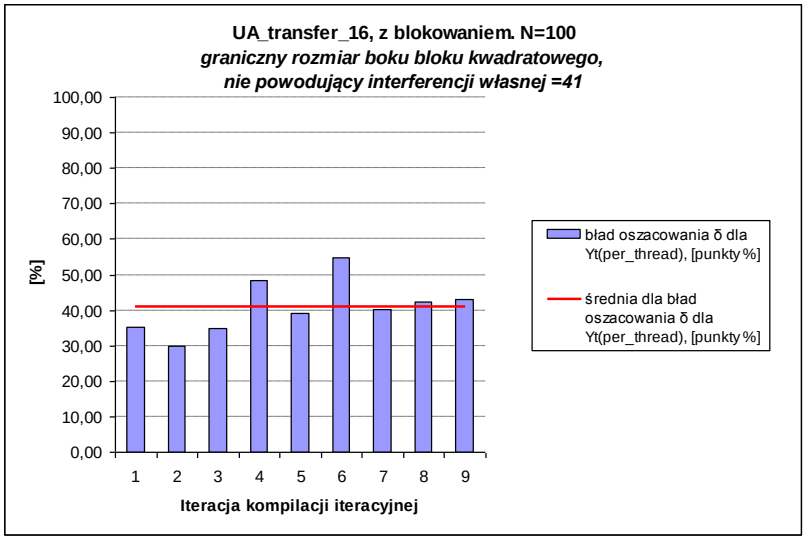
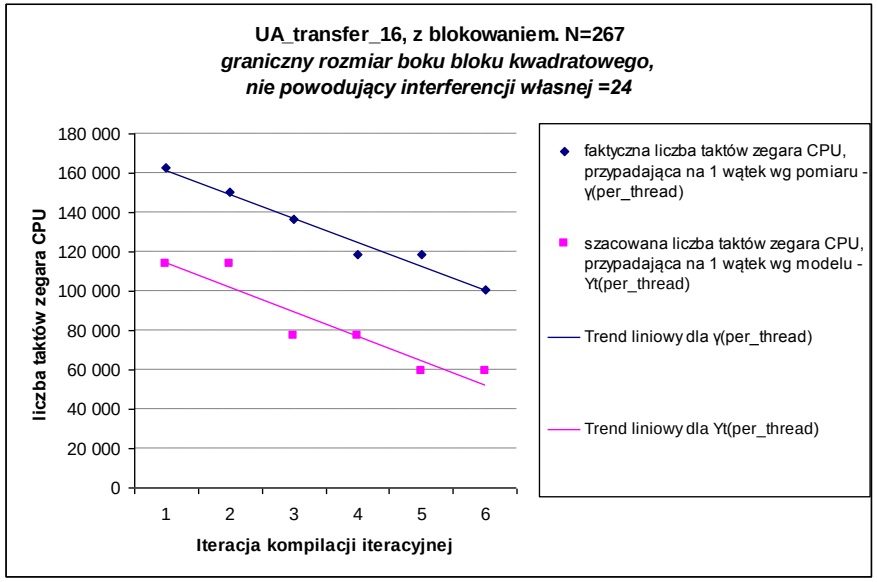
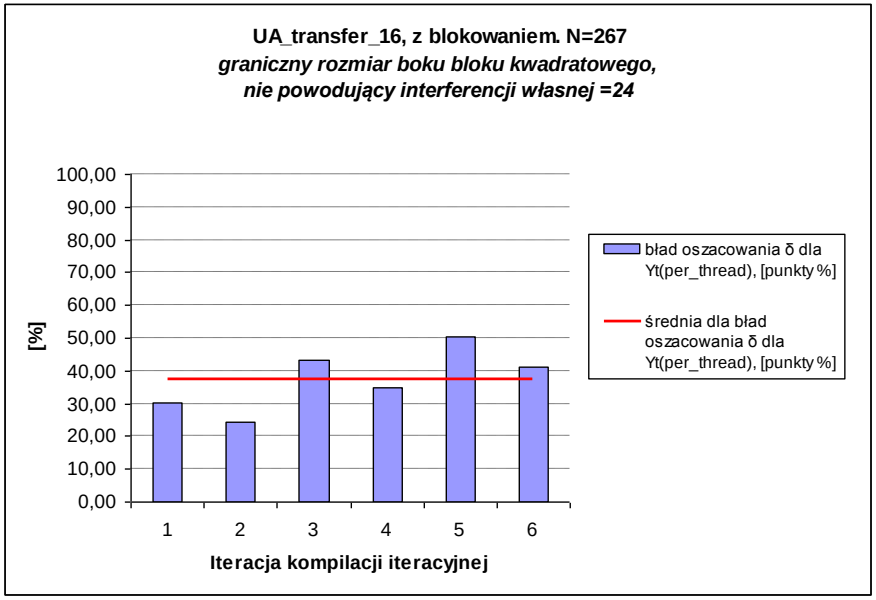


Tabela 64 *Wyniki uzyskane dla pętli niewzorcowej UA_transfer_16 z blokowaniem, dla N = 267*

Iteracja kompilacji iteracyjnej	N	BLOCK_SIZE	Sposób przydziału iteracji pętli do wątków	λ	θ	Df	X1	X2	X3	X4	Yt	γ	Yt(per_thread)	γ(per_thread)	$\delta_{Yt(per\_thread)}$
1	267	16	domyślny	0,2709	0,0037	1 763 003 540,96	0,04	23 844 295,00	134	2	221 618,04	317 271,62	113 689,56	162 759,63	30,15
2	267	24	domyślny	0,2709	0,0037	1 786 386 776,31	0,04	23 774 615,00	134	2	222 086,20	293 638,66	113 929,73	150 635,98	24,37
3	267	16	domyślny	0,2709	0,0000	1 155 885 591,91	0,06	15 836 882,50	89	3	222 332,72	392 759,90	77 187,55	136 354,99	43,39
4	267	24	domyślny	0,2709	0,0000	1 168 455 115,83	0,06	15 790 602,50	89	3	222 645,35	341 997,56	77 296,09	118 731,76	34,90
5	267	16	domyślny	0,2709	0,0037	864 387 129,82	0,08	11 922 147,50	67	4	224 348,83	450 687,97	59 041,14	118 606,07	50,22
6	267	24	domyślny	0,2709	0,0037	871 454 571,36	0,08	11 887 307,50	67	4	224 485,02	382 231,58	59 076,98	100 590,63	41,27
													Średnia dla $\delta_{Yt(per\_thread)}$		**37,38**

Rysunek 58 *Wykresy sporządzone na podstawie danych z Tabeli 64*





4.3. Podsumowanie

W rozdziale 4 przedstawiono sposób przeprowadzenia oraz szczegółowe wyniki badań eksperymentalnych, których celem było:

- wyznaczenie wartości parametrów *a1, a2, a3, a4* dla przykładowych modeli szczególnych (tj. dla modelu (35) oraz dla modelu (36)),
- przeprowadzenie oceny jakości przykładowych modeli szczególnych (tj. modelu (35) oraz modelu (36)) w aspekcie jakościowym i ilościowym.

Aby wyznaczyć wartości parametrów *a1, a2, a3, a4* dla przykładowych modeli szczególnych, wykorzystano autorskie pętle wzorcowe: nonInterf oraz matmul. Natomiast ocena jakości przedmiotowych modeli szczególnych została przeprowadzona przy wykorzystaniu 10 różnych pętli programowych (określanych w dysertacji jako pętle niewzorcowe), wybranych ze zbioru testów referencyjnych NPB [40] [63], a jej zbiorcze wyniki przedstawiają się następująco:

- Dla wszystkich analizowanych pętli niewzorcowych, uzyskano pozytywną ocenę jakości odnośnych modeli szczególnych w aspekcie jakościowym.
- Dla wszystkich analizowanych pętli niewzorcowych:
  - ✓ Średnia z błędów względnych oszacowania dla wszystkich postaci kodu źródłowego, przyjętych dla danej pętli niewzorcowej i dla danego *N*, nie przekracza 55 punktów procentowych (największą średnią z błędów względnych oszacowania, tj. 53,34 punkty procentowe, uzyskano dla pętli FT_auxfnct_2 i *N* = 30).
  - ✓ Maksymalny błąd względny oszacowania dla wszystkich postaci kodu źródłowego, przyjętych dla danej pętli niewzorcowej i dla danego *N*, nie przekracza 65 punktów procentowych (największą wartość przedmiotowego błędu, tj. 60,75 punkty procentowe, uzyskano dla pętli FT_auxfnct_2 i *N* = 38).

Dodatkowo, dla wszystkich analizowanych pętli niewzorcowych, przeprowadzono weryfikację hipotezy o normalności rozkładu reszt (uzyskanych dla odnośnych modeli szczególnych), wykorzystując w tym celu test Kołmogorowa-Smirnowa. We wszystkich analizowanych przypadkach, brak jest podstaw do odrzucenia hipotezy zerowej zakładającej normalność wspomnianego rozkładu – co oznacza, że dla wszystkich analizowanych przypadków można przyjąć, że wspomniany rozkład jest rozkładem normalnym.
Uzyskane wyniki testów Kołmogorowa-Smirnowa są statystycznym potwierdzeniem pozytywnej oceny jakości przykładowych modeli szczególnych (tj. modelu (35) oraz modelu (36)) w aspekcie jakościowym i ilościowym.

5. PODSUMOWANIE

Cel niniejszej pracy to:

> *Opracowanie rodziny modeli statystycznych (przewidywanych do zastosowania w optymalizujących kompilatorach zrównoleglających) pozwalających na oszacowanie czasu wykonania aplikacji gruboziarnistych w standardzie OpenMP C/C++ przez komputery wieloprocesorowe z pamięcią dzieloną.*

Aby zrealizować powyższy cel, należało wykazać prawdziwość przyjętej hipotezy badawczej, zakładającej, że:

> *Istnieje możliwość opracowania rodziny modeli statystycznych do oszacowania czasu wykonania aplikacji gruboziarnistych w standardzie OpenMP C/C++, które pozwolą na osiągnięcie zadawalającej dokładności oszacowania.*

Aby zweryfikować prawdziwość przyjętej hipotezy badawczej, należało zrealizować zadania cząstkowe, wskazane w rozdziale 1.2. Wyniki realizacji poszczególnych zadań cząstkowych zostały udokumentowane w dysertacji – zgodnie z wyszczególnieniem, przedstawionym w Tabeli 65.

Tabela 65 Udokumentowanie w dysertacji wyników realizacji poszczególnych zadań cząstkowych

Nr zadania	Treść zadania	Odnośne fragmenty dysertacji
1/	Sformułowanie listy czynników, mających wpływ na czas wykonania gruboziarnistych, równoległych pętli programowych, zrównoleglonych w standardzie OpenMP	Rozdział 3.1, str. 18 ÷ 20
2/	Sformułowanie listy potencjalnych zmiennych objaśniających modelu (będącego protoplastą poszukiwanej rodziny modeli) wyrażających w sposób ilościowy czynniki, o których mowa w zadaniu 1/	Rozdział 3.1, str. 20 ÷ 23
3/	Określenie sposobu konstrukcji modelu	Rozdział 3.1, str. 23
4/	Określenie możliwych struktur modelu	Rozdział 3.1, str. 23 ÷ 25
5/	Wybór zmiennych objaśniających modelu, spośród potencjalnych zmiennych objaśniających, o których mowa w zadaniu 2/	Rozdział 3.1, str. 26 ÷ 29
6/	Wybór struktury modelu, spośród możliwych struktur modelu, o których mowa w zadaniu 4/	Rozdział 3.1, str. 25 ÷ 26, 30 ÷ 32
7/	Opracowanie sposobu oszacowania parametrów modelu	Rozdział 3.2, rozdział 3.3 str. 53 ÷ 64
8/	Opracowanie sposobu oceny jakości modelu	Rozdział 3.4 str. 64 ÷ 67

Nr zadania	Treść zadania	Odnośne fragmenty dysertacji
9/	Opracowanie sposobu wykorzystania modelu w kompilacji iteracyjnej	Rozdział 3.5 str. 68 ÷ 70
10/	Przeprowadzenie badań eksperymentalnych, mających na celu ocenę faktycznych możliwości zastosowania opracowanego modelu (rodziny modeli) w kompilacji iteracyjnej	Rozdział 4 str. 82 ÷ 140
11/	Sformułowanie wniosków z przeprowadzonych badań	Rozdział 5 str. 144 ÷ 152

W wyniku realizacji zadań cząstkowych 1/ ÷ 6/ oraz 9/ zaproponowano – dla systemu operacyjnego Linux i standardu OpenMP C/C++ – statystyczny model ogólny (16) (patrz strona 32) oraz wskazano sposób wykorzystania zaproponowanego modelu w kompilacji iteracyjnej.

W zaproponowanym modelu ogólnym (16) szacowany czas wykonania pętli programowej, wyrażony zmienną *Yt,* został uzależniony od następujących czynników:

- rozmiaru pamięci podręcznej L1 i L2, przypadającej na pojedynczy wątek OpenMP,
- odcisku danych dla pojedynczego wątku OpenMP,
- liczby i typu operacji wykonywanych przez pojedynczy wątek OpenMP,
- sposobu przydziału iteracji do wątków OpenMP,
- liczby wątków OpenMP wykonujących program,

wyrażonych za pośrednictwem zmiennych niezależnych *X1, X2, X3, X4*, gdzie:

X1 – stosunek łącznego rozmiaru pamięci podręcznej L1 i L2, przypadającej na pojedynczy wątek OpenMP, do odcisku danych dla pojedynczego wątku OpenMP;
X2 – łączna ważona liczba operacji przypadająca na pojedynczy wątek OpenMP,
X3 – maksymalna, dla danego sposobu przydziału iteracji do wątków OpenMP, wielkość pojedynczej porcji iteracji przydzielonej do wykonania przez wątek,
X4 – liczba wątków OpenMP wykonujących program.

W wyniku realizacji zadania cząstkowego 7/, zaproponowany model ogólny (16) skonkretyzowano dla przykładowego środowiska (przedstawionego w Tabeli 7 na stronie 48) – poprzez wyznaczenie wartości parametrów *a1, a2, a3* oraz *a4* stosując analizę regresji dla danych empirycznych zebranych dla dwóch przygotowanych w tym celu pętli wzorcowych. W ten sposób, opracowano dwa przykładowe modele szczególne:

- model szczególny (35) (patrz strona 60), wyznaczony dla pętli wzorcowej nonInterf,
- model szczególny (36) (patrz strona 61), wyznaczony dla pętli wzorcowej matmul.:

Dla przykładowych modeli szczególnych – tj. modelu (35) oraz modelu (36) – przeprowadzono ocenę ich jakości w aspekcie jakościowym i ilościowym, w oparciu o autorskie podejście opracowane w ramach realizacji zadania cząstkowego 8/.
Idea tego podejścia polegała na oszacowaniu czasów wykonania przykładowych pętli wybranych ze zbioru testów referencyjnych NPB [40] [63] (pętle te są określane mianem pętli niewzorcowych) w środowisku przedstawionym w Tabeli 7, a następnie porównaniu otrzymanych w ten sposób oszacowanych czasów wykonania przedmiotowych pętli z czasami ich wykonania zmierzonymi empirycznie w środowisku przedstawionym w Tabeli 7. Szczegółowe wyniki tego porównania, uzyskane w ramach realizacji zadania cząstkowego 10/ dla wszystkich, przeanalizowanych pętli niewzorcowych, zamieszczono w rozdziale 4.2.2. Podsumowanie i omówienie tych wyników (w ramach realizacji zadania cząstkowego 11/) przedstawiono poniżej.

Model szczególny (35), wyznaczony dla pętli wzorcowej nonInterf, został zastosowany do oszacowania czasu wykonania 6 różnych pętli niewzorcowych, mianowicie:

- CG_cg_3,
- CG_cg_4,
- FT_auxfnct_2,
- LU_HP_pintgr_11,
- MG_mg_3,
- UA_diffuse_2.

Dla ww. pętli niewzorcowych oszacowano przy pomocy modelu szczególnego (35) czas wykonania 133 różnych kodów źródłowych, dla 18 różnych przypadków (przez przypadek rozumie się tu kombinację pętli niewzorcowej i rozmiaru problemu *N*, rozwiązywanego w tej pętli).
W Tabeli 66 przedstawiono ocenę jakości uzyskanych oszacowań w aspekcie jakościowym oraz w aspekcie ilościowym.
W Tabelach 66 ÷ 68 przyjęto oznaczenia przedstawione w rozdziale 4.2.2 (patrz strona 91).

Model szczególny (36), wyznaczony dla pętli wzorcowej matmul, został zastosowany do oszacowania czasu wykonania 4 różnych pętli niewzorcowych, mianowicie:

- UA_diffuse_3,
- UA_diffuse_4,
- UA_transfer_11,
- UA_transfer_16.

Dla ww. pętli niewzorcowych oszacowano przy pomocy modelu szczególnego (36) czas wykonania 108 różnych kodów źródłowych, dla 12 różnych przypadków (przez przypadek rozumie się tu kombinację pętli niewzorcowej i rozmiaru problemu *N*, rozwiązywanego w tej pętli).

W Tabeli 67 przedstawiono ocenę jakości uzyskanych oszacowań w aspekcie jakościowym oraz w aspekcie ilościowym.

Model szczególny (36), wyznaczony dla pętli wzorcowej matmul, został również zastosowany do oszacowania czasu wykonania 4 różnych pętli niewzorcowych z blokowaniem pętli niewzorcowych, mianowicie:

- UA_diffuse_3 (z blokowaniem),
- UA_diffuse_4 (z blokowaniem),
- UA_transfer_11 (z blokowaniem),
- UA_transfer_16 (z blokowaniem).

Dla ww. pętli niewzorcowych oszacowano przy pomocy modelu szczególnego (36) czas wykonania 66 różnych kodów źródłowych, dla 9 różnych przypadków (przez przypadek rozumie się tu kombinację pętli niewzorcowej i rozmiaru problemu *N*, rozwiązywanego w tej pętli).
W Tabeli 68 przedstawiono ocenę jakości uzyskanych oszacowań w aspekcie jakościowym oraz w aspekcie ilościowym.

Tabela 66 Ocena jakości oszacowań, przeprowadzonych przy pomocy modelu szczególnego (35)

Przypadek	Pętla niewzorcowa	*N*	Liczba różnych postaci kodu źródłowego pętli, dla których przeprowadzono oszacowanie	Ocena jakości oszacowań		
				Aspekt jakościowy (spełnienie wymagań, podanych w rozdziale 3.4, punkt a))	Aspekt ilościowy	
					Uzyskana wartość średnia dla $\delta_{Yt(per\_thread)}$	Uzyskana wartość maksymalna dla $\delta_{Yt(per\_thread)}$
1	CG_cg_3	75 000	8	+	14,27	24,97
2	CG_cg_3	118 000	8	+	13,73	24,01
3	CG_cg_3	160 000	8	+	12,48	21,46
4	CG_cg_4	100 000	8	+	10,48	20,43
5	CG_cg_4	215 000	8	+	11,66	24,43
6	CG_cg_4	330 000	8	+	13,66	27,06
7	FT_auxfnct_2	30	8	+	53,34	60,43
8	FT_auxfnct_2	38	9	+	51,54	60,75
9	FT_auxfnct_2	45	8	+	53,05	60,43
10	LU_HP_pintgr_11	200	8	+	16,42	32,04
11	LU_HP_pintgr_11	265	8	+	16,47	32,85
12	LU_HP_pintgr_11	330	8	+	14,84	28,76
13	MG_mg_3	26 000	6	+	25,96	34,41
14	MG_mg_3	57 444	6	+	29,21	38,35
15	MG_mg_3	88 888	6	+	31,04	40,51
16	UA_diffuse_2	80 000	6	+	6,80	15,29
17	UA_diffuse_2	173 333	6	+	5,12	13,61
18	UA_diffuse_2	266 666	6	+	6,44	17,01
			Łączna liczba różnych postaci kodu źródłowego wszystkich pętli, dla których przeprowadzono oszacowanie		**Największa spośród uzyskanych wartości średnich dla $\delta_{Yt(per\_thread)}$**	**Największa spośród uzyskanych wartości maksymalnych dla $\delta_{Yt(per\_thread)}$**
			133		**53,34**	**60,75**

Tabela 67 Ocena jakości oszacowań, przeprowadzonych przy pomocy modelu szczególnego (36)

Przypadek	Pętla niewzorcowa	*N*	Liczba różnych postaci kodu źródłowego pętli, dla których przeprowadzono oszacowanie	Ocena jakości oszacowań		
				Aspekt jakościowy (spełnienie wymagań, podanych w rozdziale 3.4, punkt a))	Aspekt ilościowy	
					Uzyskana wartość średnia dla $\delta_{Yt(per\_thread)}$	Uzyskana wartość maksymalna dla $\delta_{Yt(per\_thread)}$
1	UA_diffuse_3	30	9	+	31,60	38,46
2	UA_diffuse_3	50	9	+	16,88	24,02
3	UA_diffuse_3	71	9	+	27,84	49,59
4	UA_diffuse_4	30	9	+	28,55	35,80
5	UA_diffuse_4	50	9	+	12,70	20,18
6	UA_diffuse_4	71	9	+	26,49	45,39
7	UA_transfer_11	100	9	+	10,49	20,44
8	UA_transfer_11	267	9	+	11,30	29,81
9	UA_transfer_11	433	9	+	14,86	37,89
10	UA_transfer_16	100	9	+	12,27	23,60
11	UA_transfer_16	267	9	+	11,10	29,05
12	UA_transfer_16	433	9	+	14,86	34,97
			Łączna liczba różnych postaci kodu źródłowego wszystkich pętli, dla których przeprowadzono oszacowanie		**Największa spośród uzyskanych wartości średnich dla $\delta_{Yt(per\_thread)}$**	**Największa spośród uzyskanych wartości maksymalnych dla $\delta_{Yt(per\_thread)}$**
			108		**31,60**	**49,59**

Tabela 68 Ocena jakości oszacowań, przeprowadzonych przy pomocy modelu szczególnego (36) dla pętli niewzorcowych z blokowaniem

Przypadek	Pętla niewzorcowa	*N*	Liczba różnych postaci kodu źródłowego pętli, dla których przeprowadzono oszacowanie	Ocena jakości oszacowań		
				Aspekt jakościowy (spełnienie wymagań, podanych w rozdziale 3.4, punkt a))	Aspekt ilościowy	
					Uzyskana wartość średnia dla $\delta_{Yt(per\_thread)}$	Uzyskana wartość maksymalna dla $\delta_{Yt(per\_thread)}$
1	UA_diffuse_3 (z blokowaniem)	30	6	+	48,12	53,95
2	UA_diffuse_3 (z blokowaniem)	50	9	+	34,59	40,19
3	UA_diffuse_4 (z blokowaniem)	30	6	+	46,61	51,39
4	UA_diffuse_4 (z blokowaniem)	50	9	+	27,41	35,65
5	UA_diffuse_4 (z blokowaniem)	66	6	+	26,46	29,30
6	UA_transfer_11 (z blokowaniem)	100	9	+	41,76	56,95
7	UA_transfer_11 (z blokowaniem)	267	6	+	39,94	52,75
8	UA_transfer_16 (z blokowaniem)	100	9	+	40,90	54,73
9	UA_transfer_16 (z blokowaniem)	267	6	+	37,38	50,22
			Łączna liczba różnych postaci kodu źródłowego wszystkich pętli, dla których przeprowadzono oszacowanie		**Największa spośród uzyskanych wartości średnich dla $\delta_{Yt(per\_thread)}$**	**Największa spośród uzyskanych wartości maksymalnych dla $\delta_{Yt(per\_thread)}$**
			66		**48,12**	**56,95**

Biorąc pod uwagę dane, zestawione w Tabelach 66 ÷ 68, należy podkreślić, że ocena jakości obu przykładowych modeli szczególnych, tj. modelu szczególnego (35) oraz modelu szczególnego (36), została przeprowadzona w oparciu o bardzo liczny zbiór kodów źródłowych pętli niewzorcowych, zróżnicowanych ze względu na:

- rodzaj ponownego użycia przetwarzanych danych,
- obecność lub brak interferencji,
- zastosowanie lub brak blokowania i wielkość bloku,
- liczbę wątków wykonujących pętlę,
- sposób przydziału iteracji pętli do wątków.

Oba oceniane modele szczególne zostały zastosowane do oszacowania czasu wykonania 10 różnych pętli niewzorcowych (w tym dla 4 pętli niewzorcowych – przeprowadzono oszacowanie niezależnie: dla blokowania oraz dla sytuacji gdy nie występuje blokowanie). Łącznie dla wszystkich pętli niewzorcowych oszacowano przy pomocy ocenianych modeli szczególnych czas wykonania 307 różnych kodów źródłowych dla 39 różnych przypadków (przez przypadek rozumie się tu kombinację pętli niewzorcowej i rozmiaru problemu *N*, rozwiązywanego w tej pętli). Dla wszystkich przypadków, uzyskano zadawalającą jakość oszacowania, co – biorąc pod uwagę liczbę przypadków i ich różnorodność – potwierdza, że opracowane, przykładowe modele szczególne, tj. model szczególny (35) oraz model szczególny (36), pozwalają na osiągnięcie zadawalającej dokładności oszacowania.

Stosując model szczególny (35) oraz model szczególny (36) zgodnie z procedurą przedstawioną na Rysunku 19 (patrz strona 70), można skrócić czas trwania kompilacji iteracyjnej – czyli, zrealizować usprawnienie kompilacji iteracyjnej, postulowane w poprzednich rozdziałach dysertacji.
W oparciu o dane zebrane dla analizowanych pętli niewzorcowych i przedstawione w Tabelach 17 ÷ 64 (patrz strony 93 ÷ 140), oszacowano zysk czasowy wynikający z zastosowania opracowanych modeli szczególnych. Wyniki oszacowania przedstawiono w Tabeli 69. Przy szacowaniu przedmiotowego zysku przyjęto, że:

- *It* oznacza liczbę wszystkich iteracji kompilacji iteracyjnej dla danej kombinacji pętli niewzorcowej i rozmiaru problemu *N*, rozwiązywanego w tej pętli. Natomiast T_{It} oznacza faktyczny czas wykonania wszystkich *It* iteracji kompilacji iteracyjnej w docelowym środowisku sprzętowym (przedstawionym w Tabeli 7 na stronie 48) dla danej kombinacji pętli niewzorcowej i rozmiaru problemu *N*, w przeliczeniu na jeden wątek. Czas T_{It} jest wyrażony liczbą taktów zegara procesora.
- Jeżeli stosowany jest któryś z opracowanych modeli szczególnych (tj. model szczególny (35) lub model szczególny (36)),
  to zgodnie z procedurą przedstawioną na Rysunku 19, dla każdej kombinacji pętli niewzorcowej i rozmiaru problemu *N*, w docelowym środowisku sprzętowym (przedstawionym w Tabeli 7 na stronie 48) zostanie wykonanych *k* wersji kodu

źródłowego pętli, gdzie $k = \lceil 0{,}5 \times It \rceil$. Czas wykonania wspomnianych *k* wersji kodu źródłowego pętli jest oznaczony jako T_k.

- Wartość $k = \lceil 0{,}5 \times It \rceil$ została wybrana arbitralnie, jako wartość, która daje bardzo wysokie (aczkolwiek niemożliwe do jednoznacznego określenia w sposób ilościowy) prawdopodobieństwo tego, że dla każdej kombinacji pętli niewzorcowej i rozmiaru problemu *N*, do ostatecznego stosowania zostanie wybrana postać kodu o faktycznie najkrótszym czasie wykonania w docelowym środowisku sprzętowym.
- Czas niezbędny do wykonania oszacowań wg opracowanych modeli szczególnych (dalej oznaczany jako T_{calc}) jest zależny od postaci kodu źródłowego, który podlega oszacowaniu. W związku z tym, arbitralnie przyjęto, że dla każdej kombinacji pętli niewzorcowej i rozmiaru problemu *N, czas* T_{calc} jest równy 10 % czasu T_{It}.
- Zysk (oznaczony jako *P(It,k)*) należy rozumieć jako względne skrócenie czasu kompilacji iteracyjnej (wyrażone procentowo), wyznaczone wg następującego równania:

$$P(It,k) = \frac{T_{It} - (T_k + T_{calc})}{T_{It}} \times 100\% \tag{44}$$

Tabela 69 Oszacowanie zysku czasowego, wynikającego z zastosowania opracowanych modeli szczególnych, dla $k = \lceil 0{,}5 \times It \rceil$ *i wybranych pętli niewzorcowych*

Pętla niewzorcowa	***N***	***It***	***k***	T_{It}	T_k	T_{calc}	***P(It,k)* [%]**
CG_cg_3	75 000	8	4	1 957,38	779,83	195,74	50,16
CG_cg_3	118 000	8	4	3 007,58	1 192,63	300,76	50,35
CG_cg_3	160 000	8	4	4 086,69	1 609,16	408,67	50,62
CG_cg_4	100 000	8	4	2 053,32	812,41	205,33	50,43
CG_cg_4	215 000	8	4	4 334,33	1 701,67	433,43	50,74
CG_cg_4	330 000	8	4	6 630,62	2 602,75	663,06	50,75
FT_auxfnct_2	30	8	4	1 970,33	785,05	197,03	50,16
FT_auxfnct_2	38	9	5	4 262,63	1 880,18	426,26	45,89
FT_auxfnct_2	45	8	4	6 449,17	2 530,30	644,92	50,77
LU_HP_pintgr_11	200	8	4	2 411,02	954,08	241,10	50,43
LU_HP_pintgr_11	265	8	4	4 189,57	1 651,59	418,96	50,58
LU_HP_pintgr_11	330	8	4	6 463,89	2 542,51	646,39	50,67
MG_mg_3	26 000	6	3	1 415,69	558,57	141,57	50,54
MG_mg_3	57 444	6	3	3 077,12	1 199,71	307,71	51,01
MG_mg_3	88 888	6	3	4 721,80	1 834,41	472,18	51,15
UA_diffuse_2	80 000	6	3	1 507,84	591,18	150,78	50,79
UA_diffuse_2	173 333	6	3	3 209,56	1 249,05	320,96	51,08
UA_diffuse_2	266 666	6	3	4 952,80	1 923,01	495,28	51,17
UA_diffuse_3	30	9	5	47 490,99	20 930,54	4 749,10	45,93
UA_diffuse_3	50	9	5	367 227,10	161 681,93	36 722,71	45,97
UA_diffuse_3	71	9	5	1 553 446,57	683 631,01	155 344,66	45,99

Pętla niewzorcowa	*N*	*It*	*k*	T_{It}	$T_{k\cdot}$	T_{calc}	*P(It,k)* [%]
UA_diffuse_3 (z blokowaniem)	30	6	3	32 717,19	12 716,47	3 271,72	51,13
UA_diffuse_3 (z blokowaniem)	50	9	5	372 406,66	163 958,58	37 240,67	45,97
UA_diffuse_4	30	9	5	45 461,97	20 039,73	4 546,20	45,92
UA_diffuse_4	50	9	5	349 379,17	153 904,50	34 937,92	45,95
UA_diffuse_4	71	9	5	1 436 886,56	632 683,16	143 688,66	45,97
UA_diffuse_4 (z blokowaniem)	30	6	3	30 950,68	12 081,59	3 095,07	50,97
UA_diffuse_4 (z blokowaniem)	50	9	5	355 263,02	156 631,16	35 526,30	45,91
UA_diffuse_4 (z blokowaniem)	66	6	3	585 501,19	252 862,42	58 550,12	46,81
UA_transfer_11	100	9	5	39 440,76	17 364,48	3 944,08	45,97
UA_transfer_11	267	9	5	754 399,16	332 266,44	75 439,92	45,96
UA_transfer_11	433	9	5	3 322 817,71	1 463 143,04	332 281,77	45,97
UA_transfer_11 (z blokowaniem)	100	9	5	50 544,85	22 741,93	5 054,48	45,01
UA_transfer_11 (z blokowaniem)	267	6	3	791 499,21	339 586,40	79 149,92	47,10
UA_transfer_16	100	9	5	39 222,20	17 284,04	3 922,22	45,93
UA_transfer_16	267	9	5	753 418,47	331 574,98	75 341,85	45,99
UA_transfer_16	433	9	5	3 219 799,16	1 417 474,08	321 979,92	45,98
UA_transfer_16 (z blokowaniem)	100	9	5	51 596,66	23 494,68	5 159,67	44,46
UA_transfer_16 (z blokowaniem)	267	6	3	787 679,05	337 928,45	78 767,91	47,10

Przy przyjętych założeniach, szacowany zysk czasowy wynikający z zastosowania opracowanych modeli szczególnych (czyli, innymi słowy, skrócenie czasu trwania kompilacji iteracyjnej) kształtował się na poziomie od 44,46 % do 51,17 %.

Należy tu podkreślić, że szacowany zysk czasowy na poziomie od 44,46 % do 51,17 % uzyskano przy założeniu, że $k = \lceil 0{,}5 \times It \rceil$ i T_{calc} = 0,1 x $T_{It.}$ Aby mieć pewność znalezienia optymalnej postaci kodu źródłowego w każdym z przypadków, ww. założone wartości *k* oraz T_{calc} zostały przeszacowane, tj. przyjęte z nadmiarem. Tak więc, zmniejszając wartość *k* lub T_{calc} można osiągnąć jeszcze większy zysk czasowy – należy jednak pamiętać o tym, by przyjęta wartość *k* gwarantowała wysokie prawdopodobieństwo znalezienia optymalnej postaci kodu źródłowego.

Podsumowując, należy stwierdzić, że przedstawione powyżej rezultaty realizacji zadań cząstkowych, zdefiniowanych w rozdziale 1.2 i przytoczonych w Tabeli 65, tj.:

- opracowanie statystycznego modelu ogólnego (16) (patrz strona 32) oraz wskazanie sposobu wykorzystania zaproponowanego modelu w kompilacji iteracyjnej,

- opracowanie dwóch przykładowych modeli szczególnych (model szczególny (35) – patrz strona 60 oraz model szczególny (36) – patrz strona 61), będących konkretyzacją modelu ogólnego (16) dla przykładowego środowiska (przedstawionego w Tabeli 7 na stronie 48),
- uzyskanie, przy pomocy ww. modeli szczególnych, potwierdzonej empirycznie, zadawalającej dokładności oszacowania dla czasów wykonania przykładowych pętli wybranych ze zbioru testów referencyjnych NPB

potwierdzają prawdziwość hipotezy badawczej przyjętej w pracy.

W związku z tym, cel niniejszej pracy został osiągnięty, ponieważ:

- zaproponowany model ogólny (16) oraz sposób wyznaczenia modeli szczególnych, będących konkretyzacją modelu ogólnego (16), dają możliwość wyznaczenia kolejnych (innych niż przykładowe modele (35) i (36)) modeli szczególnych, dostosowanych do konkretnych potrzeb i pozwalających na oszacowanie czasu wykonania pętli niewzorcowych z zadawalającą dokładnością,
- zastosowanie modeli szczególnych, będących konkretyzacją modelu ogólnego (16), w kompilacji iteracyjnej pozwala skrócić czas trwania kompilacji iteracyjnej – co potwierdza, że zaproponowane podejście jest adekwatne do stosowania w kompilacji iteracyjnej, w charakterze postulowanym w dysertacji.

Wykorzystanie pętli wzorcowych do wyznaczenia modeli szczególnych jest bardzo istotną zaletą zaproponowanego podejścia i, z uwagi na sposób konstrukcji pętli wzorcowych (jedyne charakterystyki, które są brane pod uwagę przy tworzeniu pętli wzorcowych, to obecność/brak obecności ponownego użycia danych i interferencji) stanowi zasadniczy element nowości zaproponowanego podejścia
Dzięki zastosowaniu koncepcji pętli wzorcowych „wyprowadzonych” z arbitralnie przyjętych założeń odnośnie ponownego użycia danych i interferencji, zaproponowane podejście cechuje się dużą uniwersalnością oraz łatwością adaptacji do konkretnych potrzeb. Daje to możliwość dalszego rozwoju zaproponowanego podejścia np. poprzez opracowanie innych konkretyzacji pętli wzorcowych, niż pętle nonInterf i matmul.

BIBLIOGRAFIA

[1] Aho A.V., Lam M.S., Sethi R., Ullman J.D. *Compilers: Principles, Techniques, and Tools (2nd Edition)*, Addison Wesley, 2006

[2] Altman, N.S. *An introduction to kernel and nearest-neighbor nonparametric regression.* The American Statistician, Volume 46, Issue 3, 1992, s. 175-185

[3] Amdahl G.M. *Validity of the single processor approach to achieving large scale computing capabilities*. Proceedings of the April 18-20, 1967, spring joint computer conference, ACM, 1967, s. 483-485

[4] Asanovic Krste, et al. *The landscape of parallel computing research: A view from Berkeley*. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 2006

[5] Ayguadé E., et al. *Is the schedule clause really necessary in OpenMP?* OpenMP Shared Memory Parallel Programming, Springer Berlin Heidelberg, 2003, s. 147-159

[6] Bacon D.F., Graham S.L., Sharp O.J. *Compiler transformations for high-performance computing.* ACM Computing Surveys (CSUR), Volume 26, Issue 4, 1994, s. 345-420

[7] Barney B. *Introduction to Parallel Computing*
[online] https://computing.llnl.gov/tutorials/parallel_comp/ [dostęp: 2013]

[8] Barros R.C., Basgalupp M.P., de Carvalho A.C.P.L.F., Freitas A.A. *A Survey of Evolutionary Algorithms for Decision-Tree Induction*. IEEE Transactions on Systems, Man, and Cybernetics. Part C: Applications and Reviews, Vol. 42, No. 3, 2012, s. 291-312

[9] Barthou D. *Contributions to code optimization and high performance library generation. Habilitation à diriger les recherches (HDR).* Université de Versailles St-Quentin, Versailles, 2008

[10] Bell G., Sites R., Dally W., Ditzel D., Patt Y. *Architects Look to Processors of Future*. Microprocessor Report, Microdesign Resources, Vol. 10, No. 10, 1996, s. 1-7

[11] Berlińska J. *Metody tworzenia modeli statystycznych charakteryzujących aplikacje równoległe i rozproszone. Rozprawa doktorska*. Politechnika Szczecińska, 2005

[12] Bielecki W. *Essentials of Parallel and Distributed Computing.* Informa, 2002

[13] Bielecki W. *Przetwarzanie równoległe i rozproszone. Część 1. Metody zrównoleglenia algorytmów i tworzenia aplikacji*. Wydawnictwo Uczelnianie Politechniki Szczecińskiej, 2007

[14] Bielecki W., Pałkowski M., Siedlecki K.. *Badania efektywności metod wyszukiwania gruboziarnistej równoległości w pętlach programowych*. PS, M XI SNI, 2006, s. 213-228

[15] Blikberg R., Sørevik T. *Nested parallelism: Allocation of threads to tasks and OpenMP implementation.* Scientific Programming, Vol. 9, No. 2, 2001, s. 185-194

[16] Bodin F., Kisuki T., Knijnenburg P., O'Boyle M., Rohou E. *Iterative compilation in a non-linear optimisation space*. Workshop on Profile and Feedback-Directed Compilation, 1998

[17] Breiman L. *Random forests*. Machine learning, Vol. 45, No. 1, 2001, s. 5-32

[18] Caruana R., Niculescu-Mizil A. *An empirical comparison of supervised learning algorithms.* Proceedings of the 23rd international conference on Machine learning, ACM, 2006, s. 161-168

[19] Cavazos J., Fursin G., Agakov F., Bonilla E., O'Boyle M.F., Temam O. *Rapidly selecting good compiler optimizations using performance counters*. International Symposium on Code Generation and Optimization, 2007 (CGO'07), IEEE, 2007, s. 185-197

[20] Cavazos J., O'Boyle M.F.P. *Method-Specific Dynamic Compilation using Logistic Regression.* ACM SIGPLAN Notices, Vol. 41, No. 10, 2006, s. 229-240

[21] Chen D-K., Su H-M., Yew P-C. *The impact of synchronization and granularity on parallel systems.* ACM SIGARCH Computer Architecture News – Special Issue: Proceedings of the 17th annual international symposium on Computer Architecture, Volume 18 Issue 2SI, June 1990, s. 239-248

[22] Chen N., Johnson R. *Patterns for cache optimizations on multi-processor machines.* Proceedings of the 2010 Workshop on Parallel Programming Patterns, Article No. 2, ACM, 2010

[23] Clark D. *OpenMP: A parallel standard for the masses.* IEEE Concurrency, Volume 6, Issue 1, 1998, s. 10-12

[24] Coleman S., McKinley K.S. *Tile Size Selection Using Cache Organization and Data Layout.* ACM SIGPLAN Notices, Volume 30, Issue 6, 1995, s. 279-290

[25] Cooper K.D., Subramanian D., Torczon L. *Adaptive optimizing compilers for the 21st century.* The Journal of Supercomputing, Volume 23, Issue 1, 2002, s. 7-22

[26] *CPU cache* [online] http://en.wikipedia.org/wiki/CPU_cache#Associativity [dostęp: 2013]

[27] *CPU time* [online] http://en.wikipedia.org/wiki/CPU_time [dostęp: 2013]

[28] Cristianini N., Shawe-Taylor J. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000

[29] Dagum L., Menon R. *OpenMP: an industry standard API for shared-memory programming*. IEEE Computational Science & Engineering, Volume 5, Issue 1, 1998, s. 46-55

[30] Demichowicz M., Mazur P. *OpenMP – środowisko programowania komputerów równoległych ze wspólną pamięcią.* Pro Dialog, nr 15, 2003

[31] Devijver P.A., Kittler J. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, 1982

[32] Esseghir K. *Improving data locality for caches*. Master's thesis, Department of Computer Science, Rice University, 1993

[33] Everitt B.S., Landau S., Leese M., Stahl D. *Miscellaneous Clustering Methods*. Cluster Analysis, 5th Edition, John Wiley & Sons, Ltd, 2011, s. 215-255.

[34] Fog A. *Instruction tables. Lists of instruction latencies, throughputs and microoperation breakdowns for Intel, AMD and VIA CPUs. Last updated 2012-02-29* [online] http://www.agner.org/optimize/instruction_tables.pdf [dostęp: 2013]

[35] Fraguela B.B., Carmueja M.G., Andrade D. *Optimal tile size selection guided by analytical models*. Parallel Computing: Current & Future Issues of High-End Computing, Proceedings of the International Conference ParCo 2005, John von Neumann Institute for Computing Series, Vol. 33, 2005, s. 565-572

[36] Fursin G.G., O'Boyle M.F.P., Knijnenburg P.M.W. *Evaluating iterative compilation*. Languages and Compilers for Parallel Computing, Springer Berlin Heidelberg, 2005, s. 362-376

[37] Geisser S. *Predictive Inference*. Chapman and Hall, 1993

[38] Ghosh S., Martonosi M., Malik S. *Cache miss equations: An analytical representation of cache misses*. Proceedings of the 11th international conference on Supercomputing, ACM, 1997, s. 317-324

[39] Grama A., Gupta A., Karypis G., Kumar V. *Introduction to Parallel Computing, Second Edition*. Addison-Wesley, 2003

[40] Haoqiang J., Frumkin M., Yan J. *The OpenMP implementation of NAS parallel benchmarks and its performance*. Technical Report NAS-99-011, NASA Ames Research Center, 1999

[41] Hennessy J.L., Patterson D.A. *Computer architecture: a quantitative approach*. Elsevier, 2007

[42] Ho T.K. *Random Decision Forest*. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Volume 1, 1995, s. 278-282

[43] *HPCToolkit* [online] http://hpctoolkit.org/ [dostęp: 2013]

[44] Huang Te-Ming, Kecman V., Kopriva I. *Kernel based algorithms for mining huge data sets: Supervised, semi-supervised, and unsupervised learning*. Springer, 2006

[45] *International vocabulary of metrology – Basic and general concepts and associated terms (VIM).* JCGM 200:2008
[online] http://www.bipm.org/utils/common/documents/jcgm/JCGM_200_2008.pdf [dostęp: 2013]

[46] Ishizaka K., Obata M., Kasahara H. *Coarse grain task parallel processing with cache optimization on shared memory multiprocessor*. Languages and Compilers for Parallel Computing. Springer Berlin Heidelberg, 2003, s. 352-365

[47] Kazi I.H., Lilja D.J. *Coarse-grained Speculative Execution in Shared-memory Multiprocessors*. Proceedings of the 1998 International Conference on Supercomputing – ICS '98, Melbourne, Australia. ACM, 1998, s. 93-100

[48] Kecman V. *Learning and Soft Computing: Support Vector Machines, Neural Networks, Fuzzy Logic Systems*. The MIT Press, Cambridge, MA, 2001

[49] Kirner R., Puschner P. *Consideration of optimizing compilers in the context of WCET analysis.* Proc. Deutsche Informatiktage 2000, Bad Schussenried, 2000, s. 123-126

[50] Kisuki T., Knijnenburg P.M., O'Boyle M.F., Bodin F., Wijshoff H. A. *A feasibility study in iterative compilation*. High Performance Computing, Springer Berlin Heidelberg, 1999, s. 121-132

[51] Kisuki T., Knijnenburg P.M.W., Gallivan K., O'Boyle M.F.P. *The effect of cache models on iterative compilation for combined tiling and unrolling.* Concurrency and Computation: Practice and Experience, Volume 16, Issue 2-3, 2004, s. 247-270

[52] Knijnenburg P.M.W., Kisuki T., O'Boyle M.F.P. *Iterative compilation*. Embedded processor design challenges, Springer Berlin Heidelberg, 2002, s. 171-187

[53] Kohavi R. *A study of cross-validation and bootstrap for accuracy estimation and model selection.* Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Vol. 14, No. 2, 1995, s. 1137-1145

[54] Koronacki J., Ćwik J. *Statystyczne systemy uczące się*. Wydawnictwa Naukowo-Techniczne, 2005

[55] Kuhn B., Petersen P., O'Toole E. *OpenMP versus Threading in C/C+*. Concurrency: Practice and Experience, Volume 12, 2000, s. 1165-1176

[56] Lam M.S., Rothberg E.E., Wolf M.E. *The Cache Performance and Optimization of Blocked Algorithms.* ACM SIGARCH Computer Architecture News, Vol. 19, No. 2, 1991, s. 63-74

[57] Lampson B.W. *Lazy and speculative execution in computer systems*. ACM Sigplan Notices, Volume 43, Issue 9, 2008, s. 1-2

[58] Lokuciejewski P., Stolpe M., Morik K., Marwedel P. *Automatic Selection of Machine Learning Models for WCET-aware Compiler Heuristic Generation.* Proceedings of the 4th Workshop on Statistical and Machine Learning Approaches to Architectures and Compilation (SMART), 2010, s. 3-17

[59] Lowenthal D.K., Freeh V.W., Andrews G.R. *Efficient support for fine-grain parallelism on shared-memory machines.* Concurrency – Practice and Experience, Volume 10, Issue 3, 1998, s. 157-173

[60] Mahapatra N.R., Balakrishna V. *The processor-memory bottleneck: problems and solutions.* Crossroads, Volume 5, Issue 3es, Article No. 2,1999

[61] McKee S.A. *Reflections on the memory wall.* Proceedings of the 1st conference on Computing frontiers, ACM, 2004, s. 162-167

[62] Mordechai Ben-Ari *Understanding Programming Languages.* John Wiley & Sons, 1996

[63] *NAS Parallel Benchmarks* [online] http://www.nas.nasa.gov/publications/npb.html [dostęp: 2013]

[64] Niechaj T., Stanisławski R. *Wpływ pamięci podręcznej (cache L2 i L3) na wydajność procesorów* [online] http://pclab.pl/art51758.html [dostęp: 2013]

[65] Park E., Kulkarni S., Cavazos J. *An Evaluation of Different Modeling Techniques for Iterative Compilation*. Proceedings of the 14th international conference on compilers, architectures and synthesis for embedded systems, ACM, 2011, s. 65-74

[66] Pekhimenko G., Brown A.D. *Efficient Program Compilation through Machine Learning Techniques.* Software Automatic Tuning, Springer New York, 2010, s. 335-351

[67] Peterson L.E. *K-nearest neighbor*
[online] http://www.scholarpedia.org/article/K-nearest_neighbor [dostęp: 2013]

[68] Ramanujam J., Sadayappan P. *Tiling of Iteration Spaces for Multicomputers.* ICPP Volume 2, 1990, s. 179-186

[69] Reinhard W. et al. *The worst-case execution-time problem – overview of methods and survey of tools*. ACM Transactions on Embedded Computing Systems (TECS), Volume 7, Issue 3, Article No. 36, 2008

[70] Rivera G., Tseng C.W. *Eliminating conflict misses for high performance architectures.* Proceedings of the 12th international conference on Supercomputing, ACM, 1998, s. 353-360

[71] Rivera G., Tseng C.W. *Locality optimizations for multi-level caches.* Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM), Article No. 2, ACM, 1999

[72] Rokach L.; Maimon O. *Data mining with decision trees: theory and applications*. World Scientific Publishing Co. Pte. Ltd., 2008

[73] Saavedra R.H., Smith A.J. *Performance characterization of optimizing compilers.* IEEE Transactions on Software Engineering, Volume 21, Issue 7, 1995, s. 615-628

[74] Schölkopf B.; Burges C.J.C., Smola A.J. *Advances in Kernel Methods: Support Vector Learning*. The MIT Press, Cambridge, MA, 1999

[75] Schölkopf B.; Smola A.J. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002

[76] Shanmugam K., Malony A.D., Mohr B. *Performance Extrapolation of Parallel Programs*. Technical Report CIS-TR-95-14, University of Oregon, Department of Computer and Information Science, 1995

[77] Stallings W. *Organizacja i architektura systemu komputerowego. Projektowanie systemu a jego wydajność.* Wydawnictwa Naukowo-Techniczne, 2004

[78] *Storage Hierarchy*

[online] http://www.ts.avnet.com/uk/products_and_solutions/storage/hierarchy.html [dostęp: 2013]

[79] Szydłowski H. *Niepewności w pomiarach. Międzynarodowe standardy w praktyce.* Wydawnictwo Naukowe UAM, 2001

[80] Tallent N., Mellor-Crummey J., Adhianto L., Fagan M., Krentel M. *HPCToolkit: performance tools for scientific computing.* Journal of Physics: Conference Series, Vol. 125, No. 1, Article No. 012088, 2008

[81] Temam O., Fricker C., Jalby W. *Cache interference phenomena.* ACM SIGMETRICS Performance Evaluation Review, Volume 22, Issue 1, 1994, s. 261-271.

[82] *The 2012 ACM Computing Classification System*

[online] http://www.acm.org/about/class/class/2012 [dostęp: 2013]

[83] *The OpenMP API specification for parallel programming* [online] http://www.openmp.org/ [dostęp: 2013]

[84] Touati Sid-Ahmed-Ali, Barthou D. *On the Decidability of Phase Ordering Problem in Optimizing Compilation.* Proceedings of the 3rd conference on Computing frontiers, ACM, 2006. s. 147-156

[85] Van Der Pas R. *Memory hierarchy in cache-based systems*. Sun Blueprints, 2002

[86] Wilson G.V., *Practical parallel programming*. The MIT Press, Cambridge, MA, 1995

[87] Wolf M.E., Lam M.S. *A Data Locality Optimizing Algorithm*. *ACM Sigplan Notices,* Volume 26, Issue 6, 1991, s. 30-44

[88] Wolfe M. *High Performance Compilers for Parallel Computing*. Addison-Wesley, 1996

[89] Wolfe M. *More iteration space tiling*. Proceedings of the 1989 ACM/IEEE conference on Supercomputing, ACM, 1989, s. 655-664

[90] Woodside M., Franks G., Petriu D.C. *The future of software performance engineering.* Future of Software Engineering, 2007 (FOSE'07) IEEE, 2007, s. 171-187

[91] Wulf W.A., McKee S.A. *Hitting the memory wall: implications of the obvious.* ACM SIGARCH computer architecture news, Volume 23, Issue 1, 1995, s. 20-24

[92] Yotov K., Roeder T., Pingali K., Gunnels J., Gustavson F. *An experimental comparison of cache-oblivious and cache-conscious programs*. Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures, ACM, 2007, s. 93-104

[93] Zhang H. *The optimality of naive Bayes*. Proceedings of the 17 FLAIRS Conference, 2004, s. 562-567

[94] Ziemba S., Upadhyaya G., Pai V.S. *Analyzing the effectiveness of multicore scheduling using performance counters.* Proceedings of Workshop on the Interaction between Operating Systems and Computer Architecture, 2008

PUBLIKACJE WŁASNE, ZWIĄZANE Z TEMATYKĄ PRACY

[95] Kamińska A. *Model obliczeniowego szacowania czasu wykonania programu.* Metody Informatyki Stosowanej, nr 4/2011 (29), s. 125-134

[96] Kamińska A. *Obliczeniowe szacowanie czasu wykonania programu.* PAK, nr 02/2012, s. 193-195

[97] Kamińska A., Bielecki W. *Estimation of the execution time of coarse-grained parallel program loops.* Praca przyjęta do publikacji w „Przeglądzie Elektrotechnicznym" (referat wygłoszony na konferencji SoftSec 2013)

[98] Kamińska A., Bielecki W. *Model for the estimation of the execution time of parallel program loops.* Praca przyjęta do publikacji w „Przeglądzie Elektrotechnicznym" (referat wygłoszony na konferencji SoftSec 2013)

[99] Kamińska A., Bielecki W. *Obliczeniowe szacowanie lokalności danych na poziomie pamięci podręcznej.* Metody Informatyki Stosowanej, nr 4/2011 (29), s. 33-44

[100] Kraska K., Kamińska A. *Koncepcja metody zwiększania lokalności danych na poziomie pamięci podręcznej oparta na transformacjach pętli programowych.* Metody Informatyki Stosowanej, nr 2/2010 (23), s.63-72

[101] Kraska K., Wierciński T., Kamińska A. *Obliczeniowe szacowanie lokalności danych dla programów ANSI-C.* PAK, nr 08/2011, s.951-953

SPIS RYSUNKÓW

Rys. 1 Przetwarzanie sekwencyjne realizowane w systemie komputerowym [7] 7

Rys. 2 Przetwarzanie równoległe realizowane w systemie komputerowym [7] 7

Rys. 3 Taksonomia Flynna [77] 9

Rys. 4 Granulacja/równoległość grubo- i drobnoziarnista (na podstawie [14]) 14

Rys. 5 Struktury pętli zagnieżdżonych, reprezentujących granulację gruboziarnistą oraz granulację drobnoziarnistą 14

Rys. 6 Kod źródłowy mnożenia macierzy – w wersji bez blokowania i z blokowaniem (na podstawie [56]) 15

Rys. 7 Sposób dostępu do danych dla mnożenia macierzy – w wersji bez blokowania i z blokowaniem (na podstawie [56]) 16

Rys. 8 Granulacja/równoległość gruboziarnista bez i z blokowaniem 17

Rys. 9 Rzeczywisty czas wykonania programu a czas procesora (na podstawie [27]) 19

Rys. 10 Wykresy reszt dla zmiennych niezależnych uliniowionego modelu wyrażonego równaniem (6) – dla pętli nonInterf 29

Rys. 11 Wykresy reszt dla zmiennych niezależnych uliniowionego modelu wyrażonego równaniem (6) – dla pętli matmul 29

Rys. 12 Hierarchia pamięci [78] 34

Rys. 13 Komunikacja pomiędzy procesorem a pamięcią główną (na podstawie [77]) 35

Rys. 14 Struktura pamięci podręcznej i pamięci głównej [77] 36

Rys. 15 Porównanie odwzorowania bezpośredniego i sekcyjno-skojarzeniowego [26] 37

Rys. 16 Wpływ blokowania na kolejność wykonywania iteracji pętli 46

Rys. 17 Wyznaczanie granicznego rozmiaru boku bloku kwadratowego przy blokowaniu – dla pamięci podręcznej z odwzorowaniem bezpośrednim [56] 47

Rys. 18 Algorytm wyboru modelu, adekwatnego dla danej postaci pętli niewzorcowej – na przykładzie modeli szczególnych (35) i (36) 69

Rys. 19 Przykład zastosowania modeli szczególnych w kompilacji iteracyjnej 70

Rys. 20 Wykresy sporządzone na podstawie danych z Tabeli 17 93

Rys. 21 Wykresy sporządzone na podstawie danych z Tabeli 18 94

Rys. 22 Wykresy sporządzone na podstawie danych z Tabeli 19 95

Rys. 23 Wykresy sporządzone na podstawie danych z Tabeli 21 97

Rys. 24 Wykresy sporządzone na podstawie danych z Tabeli 22 98

Rys. 25 Wykresy sporządzone na podstawie danych z Tabeli 23 99

Rys. 26 Wykresy sporządzone na podstawie danych z Tabeli 25 101

Rys. 27 Wykresy sporządzone na podstawie danych z Tabeli 26.....102
Rys. 28 Wykresy sporządzone na podstawie danych z Tabeli 27.....103
Rys. 29 Wykresy sporządzone na podstawie danych z Tabeli 29.....105
Rys. 30 Wykresy sporządzone na podstawie danych z Tabeli 30.....106
Rys. 31 Wykresy sporządzone na podstawie danych z Tabeli 31.....107
Rys. 32 Wykresy sporządzone na podstawie danych z Tabeli 33.....109
Rys. 33 Wykresy sporządzone na podstawie danych z Tabeli 34.....110
Rys. 34 Wykresy sporządzone na podstawie danych z Tabeli 35.....111
Rys. 35 Wykresy sporządzone na podstawie danych z Tabeli 37.....113
Rys. 36 Wykresy sporządzone na podstawie danych z Tabeli 38.....114
Rys. 37 Wykresy sporządzone na podstawie danych z Tabeli 39.....115
Rys. 38 Wykresy sporządzone na podstawie danych z Tabeli 41.....117
Rys. 39 Wykresy sporządzone na podstawie danych z Tabeli 42.....118
Rys. 40 Wykresy sporządzone na podstawie danych z Tabeli 43.....119
Rys. 41 Wykresy sporządzone na podstawie danych z Tabeli 44.....120
Rys. 42 Wykresy sporządzone na podstawie danych z Tabeli 45.....121
Rys. 43 Wykresy sporządzone na podstawie danych z Tabeli 47.....123
Rys. 44 Wykresy sporządzone na podstawie danych z Tabeli 48.....124
Rys. 45 Wykresy sporządzone na podstawie danych z Tabeli 49.....125
Rys. 46 Wykresy sporządzone na podstawie danych z Tabeli 50.....126
Rys. 47 Wykresy sporządzone na podstawie danych z Tabeli 51.....127
Rys. 48 Wykresy sporządzone na podstawie danych z Tabeli 52.....128
Rys. 49 Wykresy sporządzone na podstawie danych z Tabeli 54.....130
Rys. 50 Wykresy sporządzone na podstawie danych z Tabeli 55.....131
Rys. 51 Wykresy sporządzone na podstawie danych z Tabeli 56.....132
Rys. 52 Wykresy sporządzone na podstawie danych z Tabeli 57.....133
Rys. 53 Wykresy sporządzone na podstawie danych z Tabeli 58.....134
Rys. 54 Wykresy sporządzone na podstawie danych z Tabeli 60.....136
Rys. 55 Wykresy sporządzone na podstawie danych z Tabeli 61.....137
Rys. 56 Wykresy sporządzone na podstawie danych z Tabeli 62.....138
Rys. 57 Wykresy sporządzone na podstawie danych z Tabeli 63.....139
Rys. 58 Wykresy sporządzone na podstawie danych z Tabeli 64.....140

SPIS TABEL

Tab. 1 Wartości współczynników determinacji dla różnych możliwych postaci modelu ogólnego – dla pętli nonInterf 26

Tab. 2 Wartości współczynników determinacji dla różnych możliwych postaci modelu ogólnego – dla pętli matmul 26

Tab. 3 Wartości skorygowanych współczynników determinacji dla różnych możliwych kombinacji potencjalnych zmiennych niezależnych – dla pętli nonInterf i modelu potęgowego wyrażonego równaniem (6) 27

Tab. 4 Wartości skorygowanych współczynników determinacji dla różnych możliwych kombinacji potencjalnych zmiennych niezależnych – dla pętli matmul i modelu potęgowego wyrażonego równaniem (6) 27

Tab. 5 Wyniki weryfikacji hipotezy o istotności równania regresji (model potęgowy wyrażony równaniem (6), ze zmiennymi niezależnymi: X1, X2, X3, X4) – dla pętli nonInterf i matmul .30

Tab. 6 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu potęgowego wyrażonego równaniem (6), ze zmiennymi niezależnymi: X1, X2, X3, X4) – dla pętli nonInterf i matmul 31

Tab. 7 Specyfikacja środowiska badań eksperymentalnych 48

Tab. 8 Konkretyzacja pętli wzorcowych 54

Tab. 9 Zależność pomiędzy N, total_matrix_size(N) oraz λ dla pętli wzorcowych matmul oraz nonInterf 56

Tab. 10 Próba do wyznaczenia wartości parametrów a1, a2, a3, a4 – dla pętli wzorcowej nonInterf 58

Tab. 11 Próba do wyznaczenia wartości parametrów a1, a2, a3, a4 – dla pętli wzorcowej matmul ...58

Tab. 12 Zakres stosowalności modelu (35) 63

Tab. 13 Zakres stosowalności modelu (36) 64

Tab. 14 Wyniki uzyskane dla pętli wzorcowej nonInterf 87

Tab. 15 Wyniki uzyskane dla pętli wzorcowej matmul 88

Tab. 16 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (35)) – dla pętli CG_cg_3 92

Tab. 17 Wyniki uzyskane dla pętli niewzorcowej CG_cg_3, dla N = 75 000 93

Tab. 18 Wyniki uzyskane dla pętli niewzorcowej CG_cg_3, dla N = 118 000 94

Tab. 19 Wyniki uzyskane dla pętli niewzorcowej CG_cg_3, dla N = 160 000 95

Tab. 20 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (35)) – dla pętli CG_cg_4 96

Tab. 21 Wyniki uzyskane dla pętli niewzorcowej CG_cg_4, dla N = 100 000 97

Tab. 22 Wyniki uzyskane dla pętli niewzorcowej CG_cg_4, dla N = 215 000 98

Tab. 23 Wyniki uzyskane dla pętli niewzorcowej CG_cg_4, dla N = 330 000 99

Tab. 24 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (35)) – dla pętli FT_auxfnct_2 100

Tab. 25 Wyniki uzyskane dla pętli niewzorcowej FT_auxfnct_2, dla N = 30 101

Tab. 26 Wyniki uzyskane dla pętli niewzorcowej FT_auxfnct_2, dla N = 38 102

Tab. 27 Wyniki uzyskane dla pętli niewzorcowej FT_auxfnct_2, dla N = 45 103

Tab. 28 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (35)) – dla pętli LU_HP_pintgr_11 104

Tab. 29 Wyniki uzyskane dla pętli niewzorcowej LU_HP_pintgr_11, dla N = 200 105

Tab. 30 Wyniki uzyskane dla pętli niewzorcowej LU_HP_pintgr_11, dla N = 265 106

Tab. 31 Wyniki uzyskane dla pętli niewzorcowej LU_HP_pintgr_11, dla N = 330 107

Tab. 32 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (35)) – dla pętli MG_mg_3 108

Tab. 33 Wyniki uzyskane dla pętli niewzorcowej MG_mg_3, dla N = 26 000 109

Tab. 34 Wyniki uzyskane dla pętli niewzorcowej MG_mg_3, dla N = 57 444 110

Tab. 35 Wyniki uzyskane dla pętli niewzorcowej MG_mg_3, dla N = 88 888 111

Tab. 36 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (35)) – dla pętli UA_diffuse_2 112

Tab. 37 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_2, dla N = 80 000 113

Tab. 38 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_2, dla N = 173 333 114

Tab. 39 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_2, dla N = 266 666 115

Tab. 40 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (36)) dla pętli UA_diffuse_3 116

Tab. 41 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_3, dla N = 30 117

Tab. 42 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_3, dla N = 50 118

Tab. 43 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_3, dla N = 71 119

Tab. 44 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_3 z blokowaniem, dla N = 30 120

Tab. 45 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_3 z blokowaniem, dla N = 50 121

Tab. 46 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (36)) dla pętli UA_diffuse_4 122

Tab. 47 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_4, dla N = 30 123

Tab. 48 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_4, dla N = 50 124

Tab. 49 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_4, dla N = 71 125

Tab. 50 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_4 z blokowaniem, dla N = 30 126

Tab. 51 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_4 z blokowaniem, dla N = 50 127

Tab. 52 Wyniki uzyskane dla pętli niewzorcowej UA_diffuse_4 z blokowaniem, dla N = 66 128

Tab. 53 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (36)) dla pętli UA_transfer_11 129

Tab. 54 Wyniki uzyskane dla pętli niewzorcowej UA_transfer_11, dla N = 100 130

Tab. 55 Wyniki uzyskane dla pętli niewzorcowej UA_transfer_11, dla N = 267 131

Tab. 56 Wyniki uzyskane dla pętli niewzorcowej UA_transfer_11, dla N = 433 132

Tab. 57 Wyniki uzyskane dla pętli niewzorcowej UA_transfer_11 z blokowaniem, dla N = 100 133

Tab. 58 Wyniki uzyskane dla pętli niewzorcowej UA_transfer_11 z blokowaniem, dla N = 267 134

Tab. 59 Wyniki weryfikacji hipotezy o normalności rozkładu reszt (uzyskanych dla modelu (36)) dla pętli UA_transfer_16 135

Tab. 60 Wyniki uzyskane dla pętli niewzorcowej UA_transfer_16, dla N = 100 136

Tab. 61 Wyniki uzyskane dla pętli niewzorcowej UA_transfer_16, dla N = 267 137

Tab. 62 Wyniki uzyskane dla pętli niewzorcowej UA_transfer_16, dla N = 433 138

Tab. 63 Wyniki uzyskane dla pętli niewzorcowej UA_transfer_16 z blokowaniem, dla N = 100 139

Tab. 64 Wyniki uzyskane dla pętli niewzorcowej UA_transfer_16 z blokowaniem, dla N = 267 140

Tab. 65 Udokumentowanie w dysertacji wyników realizacji poszczególnych zadań cząstkowych ... 142

Tab. 66 Ocena jakości oszacowań, przeprowadzonych przy pomocy modelu szczególnego (35) ... 146

Tab. 67 Ocena jakości oszacowań, przeprowadzonych przy pomocy modelu szczególnego (36) ... 147

Tab. 68 Ocena jakości oszacowań, przeprowadzonych przy pomocy modelu szczególnego (36) dla pętli niewzorcowych z blokowaniem 148

Tab. 69 Oszacowanie zysku czasowego, wynikającego z zastosowania opracowanych modeli szczególnych, dla k = $\lceil 0{,}5 \times It \rceil$ *i wybranych pętli niewzorcowych* 150