НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра обчислювальної техніки
(повна назва кафедри, циклової комісії)

КУРСОВА РОБОТА

з дисципліни «Паралельні та розподілені обчислення»
(назва дисципліни)

на тему: «Розробка програмного забезпечення для паралельних комп'ютерних систем»

Студента 3 курсу ІО-21 групи
напряму підготовки 050102 «Комп'ютерна інженерія»
Кузьменка В. З.
(прізвище та ініціали)

Керівник доцент Корочкін О. В.

Національна оцінка _____

Кількість балів: _____

Оцінка: ECTS _____

Члени комісії _____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали

Київ – 2015 рік

Національний технічний університет України
“Київський політехнічний інститут”

Факультет (інститут) інформатики та обчислювальної техніки
(повна назва)

Кафедра обчислювальної техніки
(повна назва)

Освітньо-кваліфікаційний рівень бакалавр

Напрям підготовки 6.050102 «Комп’ютерна інженерія»
(шифр і назва)

З А В Д А Н Н Я

НА КУРСОВУ РОБОТУ СТУДЕНТУ

Кузьменку Володимиру Зіновійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи «Розробка програмного забезпечення для паралельних комп’ютерних систем»
керівник роботи Корочкін Олександр Володимирович к.т.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
2. Строк подання студентом роботи 18 травня 2015 р.
3. Вхідні дані до роботи
   - огляд шести ядерних процесорів фірми AMD
   - математична задача $A = B\cdot(MO\cdot MK)\cdot\alpha + \min(Z)\cdot E\cdot MR$
   - структури паралельної комп’ютерної системи з спільною пам’яттю (ПКС СП) та паралельної комп’ютерної системи з локальною пам’яттю (ПКС ЛП);
   - бібліотека програмування: OpenMP, Open MPI.
4. Зміст розрахунково-пояснювальної
   - огляд шести ядерних процесорів фірми AMD
   - розробка і тестування програми ПРГ1 для ПКС ОП

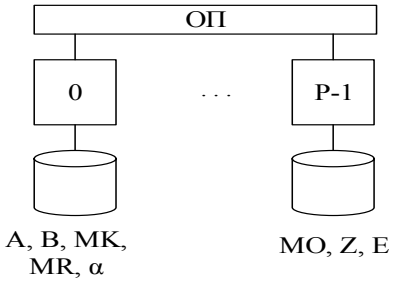

Рис 1.1. Структура ПКС СП

- розробка і тестування програми ПРГ2 для ПКС ЛП

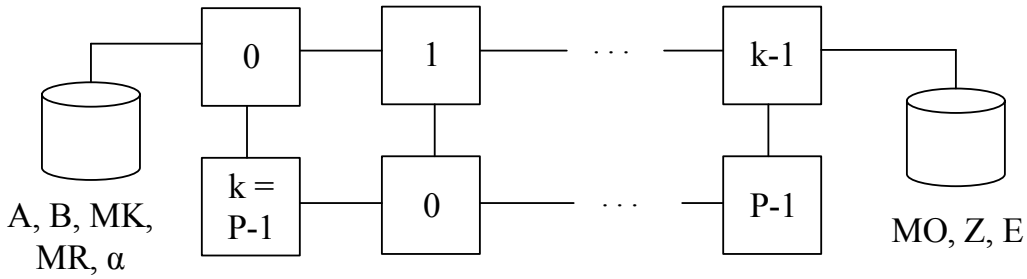

Рис 1.2. Структура ПКС ЛП

5. Перелік графічного матеріалу

- структурна схема ПКС ОП
- структурна схема ПКС ЛП
- схеми алгоритмів процесів і головної програми для ПРГ1
- схеми алгоритмів процесів і головної програми для ПРГ2.

6. Дата видачі завдання 02.02.2015

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання КР	Строк виконання етапів КР
1	Виконання Розділу 1	23.02.2015
2	Виконання Розділу 2	23.03.2015
3	Виконання Розділу 3	23.04.2015
4	Тестування програм ПРГ1 та ПРГ2	10.05.2015
5	Оформлення КР	17.05.2015
6	Захист КР	18.05.2015

Студент _____ Кузьменко В. З.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Корочкін О.В.
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

Для розв’язання багатьох задач необхідна висока продуктивність та висока швидкість передачі інформації по каналах зв’язку, великі об’єми оперативної і постійної пам’яті, які не можуть забезпечити типові обчислювальні засоби. Одним з шляхів забезпечення таких вимог є організація паралельних та розподілених обчислень і відповідних технічних засобів їх реалізації.

В першому розділі «Огляд шести ядерних процесорів компанії AMD», загальна характеристика процесорів AMD, їх архітектура, та наведені специфікації шести ядерних процесорів.

Другий та третій розділи присвячені розробці програми для обчислення математичної задачі в паралельній комп’ютерній системі зі спільною та локальною пам’яттю відповідно.

Для розробки програми обчислення математичної задачі в паралельній комп’ютерній системі зі спільною пам’яттю використана бібліотека OpenMP.

Для розробки програми обчислення математичної задачі в паралельній комп’ютерній системі із локальною пам’яттю використовується реалізація бібліотеки MPI для Java – MPJ Express. Топологія системи – маштабована решітка. Для пересилки даних у такій системи створюється буфер, у який можна помістити об’єкти вектора, матриці, числа, та пересилати їх як одне ціле. Для пересилки необхідно щоб JVM серіалізовувала об’єкти вектора і матриці, а також самого буфера.

Лістинги та алгоритми розроблених програм наведено у додатках.

РОЗДІЛ 1. ОГЛЯД ШЕСТИЯДЕРНИХ ПРОЦЕСОРІВ КОМПАНІЇ AMD

Advanced Micro Devices, Inc. (AMD) — компанія виробник інтегрованої електроніки [2]. Це другий найбільший постачальник x86сумісних процесорів і великий постачальник флеш-пам'яті. Через жорстку цінову конкуренцію з багаторічним суперником Intel фінансові показники не мають стабільності: благополучні періоди чергуються з періодами збитків. При цьому AMD, не економлячи, фінансує свої наукові дослідження (до 20% від обсягу продажу) та розширює свої виробничі потужності. У AMD одні з найсучасніших у галузі виробничі потужності, розташовані в США, Південно-Східній Азії. Що стосується азіатських виробництв AMD, то вони створені в Японії в рамках спільного з Fujitsu підприємства з виробництва модулів флеш-пам'яті на основі 0,35-мікронних технологій. Крім того, AMD має складальні і тестові майданчики у Сінгапурі і Таїланді. Стратегічними партнерами компанії AMD у виробництві персональних комп'ютерів є такі загальновідомі компанії, як Acer, Fujitsu/ICL, Hewlett-Packard і IBM. Їх супроводжують 3Com, Bay Networks, Cabletron і Cisco у мережевих продуктах і Alcatel, AT&T, Ericsson, NEC,Siemens і Sony на ринку телекомунікаційних систем.

1.1. Історія розвитку багатоядерних процесорів компанії AMD

1.1.1. Процесор Athlon 64 X2

Першим двохядерним мікропроцесором компанії AMD для персональних комп'ютерів став Athlon 64 X2. Цей процесор містить два ядра Athlon 64, розміщених на одному кристалі. Ядра мають у своєму розпорядженні загальний двоканальний контролер пам'яті, північний міст і додаткову логіку управління.

Початкові версії засновані на Athlon 64 степінга E і, залежно від моделі, мають 512 або 1024 КБ кешу другого рівня, а також 128 КБ кешу першого рівня

на кожне ядро. Техпроцес виробництва зберігся з попередніх версій і становить 65 нм.

Athlon 64 X2 підтримує набір інструкцій SSE3 (які раніше підтримувалися тільки процесорами компанії Intel), що дозволило запускати з максимальною продуктивністю код, оптимізований для процесорів Intel.[1] Ці поліпшення не унікальні для Athlon 64 X2 і також є в релізах процесорів Athlon 64, побудованих на ядрах Venice, San Diego і Newark.

Процесори Phenom

У 2007 році компанія випустила серію процесорів Phenom для персональних комп'ютерів. AMD Phenom — багатоядерний центральний процесор від компанії AMD. Має: два, три або чотири ядра (Phenom X2, Phenom X3, Phenom X4 відповідно).[2] Базується на архітектурі K10. Трьохядерні версії (кодова назва Toliman) Phenom відносяться до серії 8000, а чотирьохядерні (кодова назва Agena) до AMD Phenom X4 9000. В цій серії добавлено спільний кеш третього рівня, який дозволяє швидко обмінюватися інформацією між ядрами.

1.1.2. Процесори Phenom II

В 2010 році було представлено серію процесорів Phenom II, які виготовлені за 45 нм технологічним процесом, і базуються на AMD K10 мікроархітектурі, в свою чергу є наступниками процесорної лінійки Phenom.

Головною особливістю нової архітектури є розширення модельного ряду новими шестиядерними процесорами. У процесорах Phenom II збільшили втричі загальний розмір кеш-пам'яті L3 від 2 МБ (як в лінії Phenom) до 6 МБ, завдяки цьому приріст продуктивності збільшився до 30%. Ще одна відмінність від попередніх Phenom це те, що технологія Cool'n'Quiet тепер застосовується для процесора в цілому, а не для кожного окремого ядра. Це було зроблено з метою уникнення неправильної обробки потокових обчислень у Windows Vista, яка могла переводити однопотокові обрахунки на інші ядра, які працювали в режимі холостого ходу на частоті, зменшеній в два рази, в результаті продуктивність

процесу також зменшувалася вдвічі.

Нові шестиядерні та чотирьохядерні процесори AMD Phenom II сумісні з материнськими платами з Socket AM3 і AM2+, хоча доведеться оновити BIOS.

1.1.3. Архітектура Bulldozer

В 2011 році AMD перейшла на нову архітектуру з кодовою назвою Bulldozer. Процесори покоління AMD K11 виготовляються за 32-нм технологією і призначені для серверів і високопродуктивних ПК.

Процесори Bulldozer, за запевненням представників AMD, мають повністю нову архітектуру в порівнянні з попередніми поколіннями AMD K8 і AMD K10.

Тому компанія разом з процесорами презентувала новий сокет та чіпсет. Нові процесори сумісні з материнськими платами із Socket AM3+.

Основною особливістю цієї архітектури є те, що вона модульна. Модуль складається з двох фізичних ядер зі спільною кеш-пам'яттю другого рівня. У нових процесорах для персональних комп'ютерів може бути 2, 3, або 4 модуля, тобто 4, 6, або 8 ядер відповідно. З цього зрозуміло, що AMD випустила перший 8-ядерний процесор для ринку ПК, оскільки в процесорів Intel максимальна кількість ядер рівна шести. Процесори Bulldozer вперше підтримують нові інструкції x86 (SSE4.1, SSE4.2, CVT16, AVX і XOP, в тому числі 4-операндний модуль FMAC). Кожне ядро має 128-бітний модуль FPU з підтримкою FMA, при чому ці модулі можуть об'єднуватися в один загальний 256-бітний FPU між двома ядрами. Цю конструкцію супроводжують два модуля обчислень над цілими числами (по одному на кожне ядро) з 4-ма лініями зв'язку і можливістю спільної вибірки та декодування. Таким чином, один модуль з двома ядрами еквівалентний двохядерному процесору при операціях з цілими числами та одноядерному процесору при роботі над числами з плаваючою крапкою. Також кожен модуль має 2 МБ кешу другого рівня.

А кеш третього рівня загальний для всіх модулів. Його розмір залежить від кількості модулів і може займати 4, 6, або 8 МБ.

У новій архітектурі введена підтримка нової версії технології AMD Direct Connect і чотирьох каналів HyperTransport 3.1 на кожен процесор. Можливість

роботи з пам'яттю DDR3 і технологією розширення пам'яті AMD G3MX дозволить збільшити пропускну здатність процесора. Також покращено керування живленням.

Нові процесори отримають підтримку технології Turbo Core 2, яка дозволяє збільшити номінальну частоту всього процесора на 500МГц, або половину ядер на 1ГГц, і помітно підвищити продуктивність аналогічно технології Intel Turbo Boost.

1.1.4. Архітектура Piledriver

В 2012 році компанія переходить на нову мікроархітектуру Piledriver. Piledriver – мікроархітектура , розроблена AMD в якості наступника Bulldozer. Piledriver використовує той же модульний дизайн. Одними з головних відмінностей в оновленій модульній архітектурі є поліпшені модуль передбачення розгалужень і планування використання модулів цілих чисел і чисел з плаваючою крапкою, поряд з переходом на новий тип тригерів з поліпшеними показниками енергоспоживання . На практиці це призвело до зросту частоти на 8-10 % та збільшення продуктивності приблизно на 15 % з аналогічним енергоспоживанням.

Спочатку в продажу вийшли процесори типу AMD Accelerated Processing Unit (APU) з кодовою назвою Trinity та серія мобільних продуктів.Пізніше також розповсюдилися процесори FX-серії. Відмінністю APU від серії FX є наявність вбудованого графічного ядра.

Його потужності вистачає для роботи з мультимедіа, а розташування центрального і графічного процесорів на одному кристалі дешевше, ніж два елементи окремо. Графічний процесор займає близько половини площі кристалу, що значно більше, ніж в аналогів від Intel. У ГП відсутня внутрішня пам’ять, тому він резервує частину оперативної пам’яті для власних потреб. Через відведення великої частини площі кристалу на ГП, довелося змінити архітектурі центрального процесору, а саме: відмовитися від використання кешу третього рівня (в серії FX розподіл кешу залишився без змін).

З появою нової архітектури компанія AMD також націлилася на ринок мобільних комп'ютерів в низькому та середньому ціновому діапазоні. Тому були презентовані одно- та двомодульні APU з малим TDP (17-35 Вт).

Для встановлення процесорів FX-серії використовуються старий сокет AM3+, але для серії APU презентовано новий Socket FM2.

1.1.5. Архітектура Steamroller

На початку 2014 року компанія анонсувала нову архітектуру Steamroller. В Streamroller відсутня FX-серія, але APU продовжують використовувати модульні структуру, як і їх попередники, одночасно спрямовані на досягнення вищого рівня паралелізму.

Відмінністю нових модулів є розділені декодери інструкцій для кожного ядра в модулі. Також на 25% збільшено шину розсилки на ядро, оновлені планувальники інструкцій, більші і гнучкіші кеші (кеш другого рівня може динамічно змінювати розмір), додана черга мікрооперацій, на кристалі розміщено більше регістрових ресурсів, покращений контролер пам'яті.

Згідно з оцінками з AMD, ці поліпшення збільшать кількість виконуваних інструкцій за такт до 30 % в порівнянні з ядром першого покоління Bulldozer при збереженні високих тактових частот Piledriver, але із зменшеним енергоспоживання. В середньому нові процесори швидші за попередників на 9 % в однопоточних програмах та на 18% в багатопоточних.

1.1.6. Архітектура Excavator

В 2015 році запланований перехід на архітектуру Excavator. Зараз ця мікроархітектура на стадії розробки. Excavator APU будуть випускатися під кодовою назвою Carrizo. Очікується, підтримка нових інструкцій, таких як AVX2, BMI2 і RDRAND. Також очікується оновлення контролерів пам'яті для підтримки пам'яті стандартів DDR3 і DDR4.

1.2. Огляд шестиядерних процесорів AMD

1.2.1. Огляд шестиядерного процесора AMD Phenom II X6 1055T

AMD Phenom II X6 на ядрі Theban від AMD Phenom II X4 на ядрі Deneb, таких як поява ще двох обчислювальних ядер і реалізація технології Turbo Core, у вбудованого контролера пам'яті новинки офіційно з'явилася підтримка DDR3-1600, як показано у табл. 1.1

Таблиця 1.1 Специфікація AMD Phenom II X6 1055T

Параметр	Значення
Маркування	HDT55TFBK6DGR
Процесорний роз’єм	Socket AM3, AM2+
Тактова частота, МГц	2800
Множник	14
Частота шини HT, МГц	2000
Об'єм кеш-пам'яті L1, КБ	128 x 6
Об'єм кеш-пам'яті L2, КБ	512 x 6
Об'єм кеш-пам'яті L3, КБ	6144
Ядро	Thuban
Кількість ядер	6
Напруга живлення, В	1,125 – 1,40
Тепловий пакет, Вт	125
Тактова частота в режимі AMD Turbo Core, МГц	до 3300
Критична температура, °C	62
Техпроцес, нм	45
Підтримка технологій	AMD Turbo Core, Cool’n’Quiet 3.0, Coolcore Technology, Dual Dynamic Power, Management, Enhanced Virus Protection,, Virtualization Technology

Продовження таблиці 1.1

Параметр	Значення
Підтримка технологій	Core C1 and C1E states, Package S0, S1, S3, S4 and S5 states
Вбудований контролер пам'яті	
Типи пам'яті	DDR2-800/1066 DDR3-800/1066/1333/1600
Число каналів пам'яті	2
Максимальний об'єм пам'яті, ГБ	16
Максимальна пропускна здатність, ГБ/с	21,3
Підтримка ECC	немає

Використання більш швидшої оперативної пам'яті повинно трохи зменшити можливі затримки внаслідок збільшення кількості виконавчих блоків без розширення кеш-пам'яті третього рівня.

Маркування HDT55TFBK6DGR, яке можна розшифрувати так:

- HD – процесор AMD архітектури K10,5 для робочих станцій;
- T – процесор з фіксованим множником;
- 55T – модельним номер, що ідентифікує сам процесор та вказує на підтримку технології Turbo Core;
- FB – тепловий пакет процесора до 125 Вт при напрузі живлення до 1,4 В;
- K – впакований процесор у корпус 938 pin OμPGA (Socket AM3);
- 6 – загальна кількість активних ядер і відповідно об'єм кеш-пам'яті L2 6x512 КБ;
- DGR - ядро Thuban степпінгу E0.

Розподіл кеш-пам'яті процесора AMD Phenom II X6 залишився точно таким же, як і для інших двох-, трьох- і чотирьохядерних моделей з повним об'ємом кеш-пам'яті третього рівня. Так, AMD Phenom II X6 1055T має у своєму розпорядженні

128 КБ кеш-пам'яті першого рівня з дволінійною асоціативністю окремо для даних і інструкцій на кожне ядро, 512 КБ кеш-пам'яті другого рівня з шістнадцятилінійною асоціативністю також на кожне ядро та загальні 6 МБ кеш-пам'яті третього рівня з 48 ліній асоціативності[5].

1.2.2. Огляд шестиядерного процесора AMD Phenom II X6 1075T

Подібно іншим процесорам модель AMD Phenom II X6 1075T підтримує технологію Turbo Core, як показано у табл 1.2. Для того щоб дізнатися багато інформації про можливості процесора, достатньо лише глянути на його маркування на теплорозподільній кришці[12]. Маркування HDT75TFBK6DGR означає наступне:

- HD – процесор сімейства AMD Phenom архітектури K10,5 для робочих станцій;
- T75T – модельним номер, що ідентифікує сам процесор та вказує на підтримку технології Turbo Core (процесор з заблокованим множником);
- FB – тепловий пакет процесора 125 Вт;
- K – процесор упакований в корпус 938 pin OμPGA (Socket AM3);
- 6 – загальна кількість активних ядер;
- D – об'єм кеш-пам'яті L2 6x512 КБ і L3 6 МБ;
- GR – ядро степпінгу PH-E0.

Що ж стосується тильної сторони процесора, то перед нами з'являється знайоме 938-контактне пакування для роз’єму Socket AM3, який зворотно сумісний з Socket AM2+.

Таблиця 1.2 Специфікація AMD Phenom II X6 1075T

Параметр	Значення
Маркування	HDT75TFBK6DGR
Процесорний роз’єм	Socket AM3 (AM2+)
Тактова частота, МГц	3000

Продовження таблиці 1.2

Параметр	Значення
Тактова частота в режимі Turbo Core (спрацьовує для 3-х і менше ядер), МГц	3500
Множник	15
Частота шини HT, МГц	2000
Об'єм кеш-пам'яті L1, КБ	128 x 6
Об'єм кеш-пам'яті L2, КБ	512 x 6
Об'єм кеш-пам'яті L3, КБ	6144
Ядро	Thuban
Кількість ядер	6
Напруга живлення, В	1,15 – 1,475 (базова частота) 1,25 – 1,475 (Turbo Core mode) 1,00 – 1,225 (простій)
Напруга північного мосту, В	1,05 – 1,175
Тепловий пакет, Вт	95
Критична температура, °C	55 – 62
Техпроцес, нм	45
Підтримка технологій	Cool’n’Quiet 3.0, CoolCore Technology, Dual Dynamic Power Management, Enhanced Virus Protection, Virtualization Package S0, S1, S3, S4 and S5 states
Вбудований контролер пам'ятей	
Типи пам'яті	DDR2-667/800/1066 DDR3-800/1066/1333
Число каналів пам'яті	2
Максимальний об'єм пам'яті, ГБ	16

Продовження таблиці 1.2

Параметр	Значення
Максимальна пропускна здатність, ГБ/с	21,3
Підтримка ECC	є

Принципової відмінності моделі процесора AMD Phenom II X6 1075T від своїх «побратимів» практично ніякої немає, за винятком того, що множник заблокований, на відміну від старших моделей. Сам же множник зафіксований на 15, що, власне, і забезпечує частоту процесора 3,0 ГГц[4].

1.2.3. Огляд шестиядерного процесора AMD Phenom II X6 1100T

Процесор AMD Phenom II X6 1100T – вершина лінійки шестиядерних процесорів AMD на ядрі Thuban. Як показано у табл 1.3. даний процесор має три характерні риси: це найвища в лінійці шестиядерних процесорів AMD номінальна тактова частота, та статус самого потужного продукту.

На теплорозподільну кришку процесора нанесене спеціальне маркування – HDE00ZFBK6DGR, яка розшифровується таким чином:

- HD – процесор сімейства AMD Phenom архітектури K10,5 для робочих станцій;
- E00 – модельним номер, що ідентифікує сам процесор та вказує підтримку технології Turbo Core;
- Z – процесор з вільним множником;
- FB – тепловий пакет процесора 125 Вт;
- K –процесор упаковано в корпус 938 pin OμPGA (Socket AM3);
- 6 – загальна кількість активних ядер;
- D – об'єм кеш-пам'яті L2 6x512 КБ і L3 6 МБ;
- GR – ядро Thuban степпінга E0.

Таблиця 1.3 Специфікація AMD Phenom II X6 1100T

Параметр	Значення
Маркування	HDE00ZFBK6DGR
Процесорний роз’єм	Socket AM3, AM2+
Тактова частота, МГц	3300
Множник	16,5
Частота шини HT, МГц	2000
Об'єм кеш-пам'яті L1, КБ	128 x 6
Об'єм кеш-пам'яті L2, КБ	512 x 6
Об'єм кеш-пам'яті L3, КБ	6144
Ядро	Thuban
Кількість ядер	6
Напруга живлення, В	1,470
Тепловий пакет, Вт	125
Тактова частота в режимі AMD Turbo Core, МГц	до 3700
Критична температура, °C	62
Техпроцес, нм	45
Підтримка технологій	AMD Turbo Core, Cool’n’Quiet 3.0 CoolCore Technology, Dual Dynamic Power Management, Enhanced Virus Protection, Virtualization Technology Core C1 and C1E states, Package S0, S1,
Вбудований контролер пам'яті	
Типи пам'яті	DDR2-800/1066 DDR3-800/1066/1333/1600
Кількість каналів пам'яті	2

Продовження таблиці 1.3

Параметр	Значення
Максимальний об'єм пам'яті, ГБ	16
Максимальна пропускна здатність, ГБ/с	21,3
Підтримка ECC	є

Шестиядерний процесор AMD відрізняється від моделі на щабель нижче AMD Phenom II X6 1090T тільки збільшеним на половину одиниці множником, отже, збільшилася на 100 МГц тактова частота в номінальному режимі та режимі AMD Turbo Core.

При встановленні швидких модулів пам'яті DDR3-2000 1024 МБ Kingston Hyperx KHX16000D3T1K3 система автоматично розпізнає їх тільки як DDR3-1333, для використання пам'яті в більш швидшому режимі, ніж DDR3-1333, необхідно проводити її налаштування в BIOS.

Процесор AMD Phenom II X6 1100T вінчає вершину лінійки шестиядерних процесорів на ядрі Thuban. У плані функціональних можливостей даний процесор не відрізняється від молодших моделей і цікавий своїми характеристиками тільки при порівнянні з молодшими шести ядерними процесорами, що працюють на номінальних частотах[6].

1.2.4. Огляд шестиядерного процесора AMD FX-6100 на базі архітектури Bulldozer

Традиційне маркування на процесорній кришки повідомляє власнику досить багато інформації. У цьому випадку вона наступна – FD6100WMW6KGU:

- F – процесор належить до сімейства AMD FX;
- D – сфера застосування даного процесора – робочі станції;
- 6100 – модельним номер;
- WM – тепловий пакет процесора 95 W;

- W – упакований процесор у корпус 938 pin Socket AM3+;
- 6 – загальна кількість активних ядер;
- K – об'єм кеш-пам'яті L2 2 МБ на кожний модуль і 8 МБ кеш-пам'яті L3;
- GU - ядро процесора степпінга OR-B2.

На звороті є знайомі нам 938 контактів, однак у цьому випадку роз’єм для встановлення CPU використовує тільки Socket AM3+, що забезпечує підтримку лише DDR3.

Як показано у табл. 1.4 кеш-пам'ять новинки розподіляється таким чином. Кеш-пам'ять 1 рівня: по 16 КБ на кожне з 6 ядер виділяється для даних з чотирма каналами асоціативності, при цьому для інструкції є 64 КБна кожний 2-процесорний модуль з 2 каналами асоціативності. Кеш-пам'ять 2 рівня: по 2 МБ на кожний модуль процесора, яких як ви пам'ятаєте 3, з 16 каналами асоціативності. Кеш-пам'ять 3 рівня загальна для всього процесора і становить 8 МБ з 64 каналами асоціативності. Процесори даного сімейства оснащені технологією Turbo Core 2.0, яка дозволяє підвищувати частоту процесора при вирішенні ресурсномістких завдань[11].

Таблиця 1.4 Специфікація AMD FX-6100

Параметр	Значення
Маркування	FD6100WMW6KGU
Процесорний роз’єм	Socket AM3+
Тактова частота (номінальна), МГц	3300
Максимальна тактова частота з TC 2.0), МГц - для 6 ядер - для 3 ядер	 3600 3900
Множник (номінал)	16,5
Частота шини HT, МГц	2200

Продовження таблиці 1.4

Параметр	Значення
Об'єм кеш-пам'яті L1, КБ	3 x 64 (інструкції) 6 x 16 (дані)
Об'єм кеш-пам'яті L2, МБ	3 x 2
Об'єм кеш-пам'яті L3, МБ	8
Ядро	Zambezi
Кількість ядер	6
Тепловий пакет, Вт	95
Критична температура, °C	70
Техпроцес, нм	32
Підтримка технологій	Multiple low-power states Enhanced Virus Protection Advanced Power Management Virtualization Technology Hardware Thermal Control, Core C0, C1, C1E, C6, CC6, states. Package S0, S1, S3, S4 and S5. States, AMD Turbo CORE technology 2.0
Вбудований контролер пам'яті	
Типи пам'яті	DDR3-1066/1333/1600/1866
Кількість каналів пам'яті	2
Максимальний об'єм пам'яті, ГБ	16
Максимальна пропускна здатність, ГБ/с	21,3
Підтримка ECC	є

1.2.5. Огляд шестиядерного процесора FX-6200

Модель AMD FX-6200, анонсована наприкінці лютого 2012 року разом з AMD FX-4170. Даний ЦП виготовлений по 32 нм техпроцесу і складається з трьох обчислювальних модулів, кожний з яких містить по два ядра, що разом дає нам шість ядер[7]. Процесор також відрізняється підвищеною до 3,8 ГГц, але термопакет залишився без змін - 125 Вт.

Корпус процесора такий же, як і у інших представників сімейства AMD FX. Та і від більш старих Phenom II для Socket AM3 мало чим відрізняється. Масивна металева кришка захищає кремнієву основу процесора від пошкоджень при встановленні системи охолодження і одночасно забезпечує рівномірний розподіл тепла, запобігаючи перегріву окремих ділянок кристала. Маркування на кришці дає користувачу максимум інформації про процесор[11]. Для цієї моделі вона наступна - FD6200FRW6KGU:

- F – процесор належить до сімейства AMD FX;
- D – сфера застосування даного процесора – робочі станції;
- 6200 – модельним номер;
- FR – тепловий пакет процесора 125 W;
- W – впакований процесор у корпус 938-pin Socket AM3+;
- 6 – загальна кількість активних ядер;
- K – об’єм кеш-пам'яті L2 2 МБ на кожний модуль і 8 МБ кеш-пам'яті L3;
- GU - ядро процесора степінга OR-B2

Дана модель є найпотужнішим шестиядерним процесором на архітектурі Bulldozer. Детально ознайомитися із її характеристиками можна в табл. 1.5.

Таблиця 1.5 Специфікація AMD FX-6200

Параметр	Значення
Маркування	FD6100WMW6KGU
Процесорний роз’єм	Socket AM3+
Тактова частота (номінальна), МГц	3800

Продовження таблиці 1.5

Параметр	Значення
Максимальна тактова частота з TC 2.0, МГц (для всіх ядер)	4100
Множник (номінал)	16,5
Частота шини HT, МГц	2200
Об'єм кеш-пам'яті L1, КБ	3 x 64 (інструкції) 6 x 16 (дані)
Об'єм кеш-пам'яті L2, МБ	3 x 2
Об'єм кеш-пам'яті L3, МБ	8
Ядро	Zambezi
Кількість ядер	6
Напруга живлення, В	0,75-1,40
Тепловий пакет, Вт	125
Критична температура, °C	75
Техпроцес, нм	32
Підтримка технологій	Multiple low-power states Enhanced Virus Protection Advanced Power Management Virtualization Technology Hardware Thermal Control Core C0, C1, C1E, C6, CC6, states Package S0, S1, S3, S4 and S5 states AMD Turbo CORE technology 2.0
Вбудований контролер пам'яті	

Продовження таблиці 1.5

Параметр	Значення
Типи пам'яті	DDR3-1066/1333/1600/1866
Кількість каналів пам'яті	2
Максимальний об'єм пам'яті, ГБ	16
Максимальна пропускна здатність, ГБ/с	21,3
Підтримка ECC	є

1.2.6. Огляд шестиядерного процесора FX-6350

AMD FX-6350 є представником високопродуктивної лінійки компанії AMD. Він не обладнаний графічним прискорювачем, але при цьому має досить високу швидкодію та шість обчислювальних ядер. У лінійці AMD FX є два шестиядерні процесори, які основані на мікроархітектурі AMD Piledriver і входять у сімейство AMD Vishera.

Процесор AMD FX-6350 має класичний корпус, який за формою не відрізняється від інших моделей компанії AMD. Що стосується маркування CPU, то при його розшифруванні можна скласти досить детальну картину про даний пристрій. У нашому випадку FD6350FRW6KHK має таке позначення:

- F – сімейство AMD FX;
- D – сегмент робочих станцій;
- 6350 – номер моделі;
- FR – тепловий пакет 125 Вт;
- W – процесорний роз’єм Socket AM3+;
- 6 – загальна кількість активних ядер;
- K – обсяг кеш-пам'яті L2 – 2 МБ на кожний двоядерний модуль і 8 МБ загальної кеш-пам'яті L3;
- HK - ядро процесора степінга OR-C0.

Дана модель є найпотужнішим шестиядерним процесором на архітектурі Bulldozer[3]. Відмінністю від специфікацій FX-6200 є змінена частота, додаткові технології та відсутність критичної температури. Детально ознайомитися із її характеристиками можна в таблиці 1.6.

Таблиця 1.6 Специфікація AMD FX-6350

Параметр	Значення
Маркування	FD6100WMW6KGU
Процесорний роз’єм	Socket AM3+
Тактова частота (номінальна), МГц	3900
Максимальна тактова частота з TC 2.0, МГц (для всіх ядер)	4200
Множник (номінал)	16,5
Частота шини HT, МГц	2200
Об'єм кеш-пам'яті L1, КБ	3 x 64 (інструкції) 6 x 16 (дані)
Об'єм кеш-пам'яті L2, МБ	3 x 2
Об'єм кеш-пам'яті L3, МБ	8
Ядро	Zambezi
Кількість ядер	6
Напруга живлення, В	0,888-1,404
Тепловий пакет, Вт	125
Критична температура, °C	-
Техпроцес, нм	32

Продовження таблиці 1.6

Параметр	Значення
Підтримка технологій	Multiple low-power states Enhanced Virus Protection Advanced Power Management Virtualization Technology Hardware Thermal Control Core C0, C1, C1E, C6, CC6, states Package S0, S1, S3, S4 and S5 states AMD Turbo CORE technology 2.0
Вбудований контролер пам'яті	
Типи пам'яті	DDR3-1066/1333/1600/1866
Кількість каналів пам'яті	2
Максимальний об'єм пам'яті, ГБ	16
Максимальна пропускна здатність, ГБ/с	21,3
Підтримка ECC	є

1.3 Висновки до розділу 1

1. Копанія AMD за останні дев’ять років випустила 6 поколінь багатоядерних процесорів. Починаючи з двохядерних і закінчуючи восьмиядерними процесорами для ринку персональних комп’ютерів.
2. З точки зору архітектури ці покоління можна поділити на 2 частини. В першу входять Athlon 64 X2, Phenom та Phenom II. В другу – Bulldozer, Piledriver, Srteamroller. Останні відрізняються від перших наявністю модульної архітектури. Тобто, процесор поділяється не на ядра, а на модулі, кожен з яких містить по два ядра.

3. Процесори з архітектурами Phenom та Phenom II відрізняються від Athlon 64 X2 тим, що у них є кеш-пам’ять третього рівня. Ця пам’ять спільна для всіх ядер і дозволяє швидко обмінюватися даними.
4. Процесори з архітектурами Piledriver і Srteamroller, на відміну від попередніх поколінь, мають серії процесорів з вбудованим графічним прискорювачем. Огляд шестиядерних процесорів показав, що з кожним поколінням зростає частота процесора і на архітектурі Srteamroller перетнула межу в 4ГГц. В

РОЗДІЛ 2. РОЗРОБКА ПРОГРАМИ ПРГ1 ДЛЯ ПКС СП

У даному розділі розроблюється програма ПРГ1 для системи зі спільною пам’яттю, що відповідає технічному завданню, представленому на рис. 2.1.

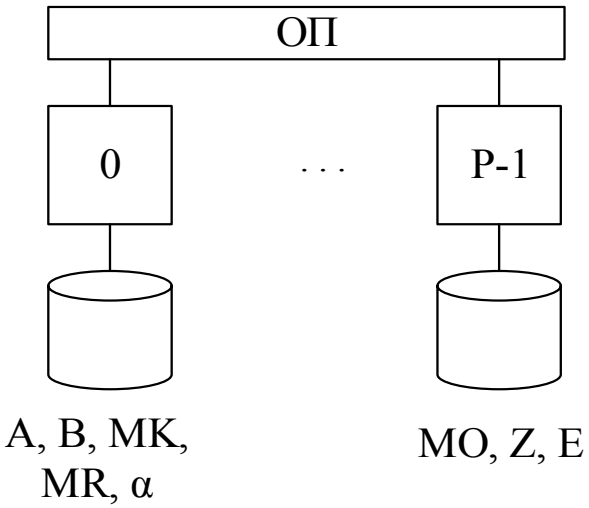

Рис 2.1 Структура ПКС СП

- Мова програмування: C++;
- Бібліотера: OpenMP;
- Математичний вираз: A = B·(MO·MK)·α + min(Z)·E·MR.

2.1 Розробка паралельного математичного алгоритму

Паралельний математичний алгоритм відповідно до рекомендованої методичної літератури [13] можна подати у вигляді наступних двох етапів:

1) $m_i = \min(Z_H)$, $i = \overline{(0, P-1)}$;

2) $m = \min(m, m_i)$, $i = \overline{(0, P-1)}$;

3) $A_H = \alpha \cdot B \cdot (MO \cdot MK_H) + m \cdot E \cdot MR_H$,

де:

H = N/P;

A_H – H рядків вектора A;

MK_H – H рядків матриці MK;

MR_H – H рядків матриці MR;

Спільні ресурси: α, B, E, MO

2.2 Розробка алгоритмів процесів

Оскільки розроблюване програмне забезпечення є масштабованим і працює на системі із кількістю процесорів P≥2, то для реалізації необхідно скласти єдиний алгоритм для всіх задач.

Задачі T(0) – T(P-1)

Крок алгоритму	КД, Бар'єри
1. Якщо tid = 0 Ввести α, B, MK, MR.	
2. Якщо tid = P-1 ввести Z, E, MO.	
3. **Бар'єр** для усіх задач. Синхронізація по вводу.	Бар'єр
4. Обчислення $minZ_i = min(Z)$, $i = \overline{0, P-1}$.	
5. Обчислення $minZ = min(minZ, minZ_i)$, $i = \overline{0, P-1}$.	КД
6. **Бар'єр** для усіх задач. Синхронізація обчислень minZ	Бар'єр
7. **Копіювати** MO_i=MO, $minZ_i = minZ$, $\alpha_i = \alpha$, $B_i = B$, $E_i = E$	КД
9. **Бар'єр** для усіх задач. Синхронізація обчислень A	Бар'єр
10. Якщо tid = 0 вивести результат A_H.	

2.3 Розробка схеми взаємодії процесів

На основі алгоритму для всіх задач, наведеному в попередньому розділі, було розроблена структурна схема взаємодії задач (рис. 2.2). Вона дозволяє наочно контролювати бар'єри та критичні ділянки. Крім того, на структурній схемі уводяться також замок, та критична секція, що будуть використовуватись в програмі.

Для демонстрації взаємодії між задачами вибрано три задачі: T(0), T(i), T(P-1) (i = $\overline{(0,P-1)}$). Задачі T(0) і T(P-1) вводять дані, тому з ними взаємодіють всі інші (синхронізація по вводу). Задачі T(i) (i = $\overline{(0,P-1)}$) виконують лише обчислення і синхронізуються зі всіма іншими задачами по обчисленню. Задача T(0) виводить результат обчислень.

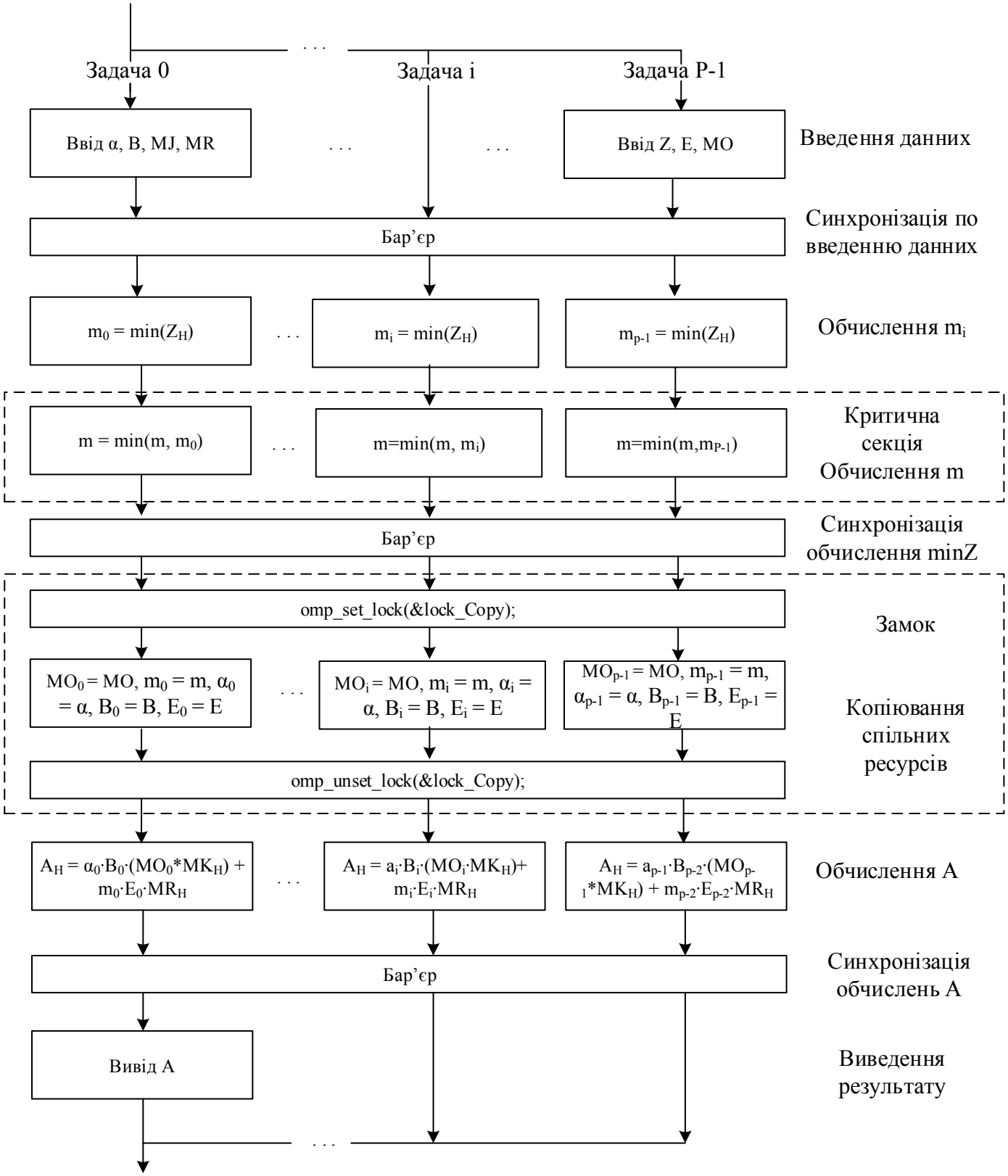

Рис 2.2 Структурна схема взаємодії задач

2.4 Розробка програми ПРГ2

Програма написана на мові C++ з використанням бібліотеки OpenMP[8] та складається з трьох модулів: основного pro2_course_work_prg1.cpp, та допоміжного operations.cpp і файлу заголовків operations.h.

Основний модуль pro2_course_work_prg1.cpp містить одну функцію:

main – точка входу в програму. Формує ідентифікатори tid задач, запускає задачі, а також вимірює час виконання elapsedTime програми ПРГ1. В цьому модулі об’явлена така змінна: P – налаштування кількості процесорів.

Файл заголовків operations.h. представляє заголовки службових функцій (таких як ввід/вивід, копіювання), та типи вектора і матриці. У ньому також об’явлена змінна N що визначає розмірність матриць і векторів.

Допоміжний модуль operations.cpp реалізовує функції об’явлені у файлі заголовків operations.h.

Лістинг програми ПРГ1 наведено у додатку Г.

2.5 Тестування ПРГ1

Для тестування використовувалась паралельна обчислювальна система з наступними апаратними характеристиками:

- процесор AMD Phenom II x6 1055T Processor 2.80 GHz
- оперативна пам’ять 3,25 ГБ

В якості програмного забезпечення виступали:

- операційна система: Microsoft Windows 7 Service Pack1.
- середовище розробки і компіляції C++ програми: Microsoft Visual Studio Premium 2013.

Для вимірювання часу виконання програми використовувався високоточний таймер з бібліотеки OpenMP.

Коефіцієнт прискорення K_n показує скорочення часу виконання паралельної програми в паралельній системі з P процесорами T_P в порівнянні з часом виконання послідовної програми в однопроцесорній системі T_1:

$$K_n = T_1 / T_P$$

Коефіцієнт ефективності K_e застосування комп'ютерної системи показує ступінь використання P процесорів системи:

$$K_e = T_1 / (T_P \cdot P) \cdot 100\% = K_n / P \cdot 100\%$$

Результати тестування і проведених досліджень ефективності розробленої програми наведено в таблицях 2.1 – 2.3.

Таблиця 2.1 Час виконання програми ПРГ1

N	T_1, мс	T_2, мс	T_3, мс	T_4, мс	T_5, мс	T_6, мс
900	3606	1808	1328	1045	851	786
1800	29169	14567	10407	7952	6383	5530
2400	68163	34222	24637	18724	14979	12548

На основі даних із таблиці 2.1 виконано розрахунок значень коефіцієнтів прискорення, які наведені в таблиці 2.2.

Таблиця 2.2 Коефіцієнти прискорення для програми ПРГ1

N	Кількість процесорів (P)					
	1	2	3	4	5	6
900	1,00	1,99	2,72	3,45	4,24	4,59
1800	1,00	2,00	2,80	3,67	4,57	5,27
2400	1,00	1,99	2,77	3,64	4,55	5,43

Коефіцієнти ефективності (таблиця 2.3) обчислено за даними таблиці 2.2.

Таблиця 2.3 Коефіцієнти ефективності для програми ПРГ1

N	Кількість процесорів (P)					
	1	2	3	4	5	6
900	100,00%	99,74%	90,52%	86,31%	84,75%	76,46%
1800	100,00%	100,12%	93,43%	91,70%	91,40%	87,92%
2400	100,00%	99,59%	92,22%	91,01%	91,01%	90,53%

Використовуючи таблиці 2.2-2.3 побудовано графік(рис. 2.2) зміни коефіцієнтів прискорення і ефективності в залежності від N і P.

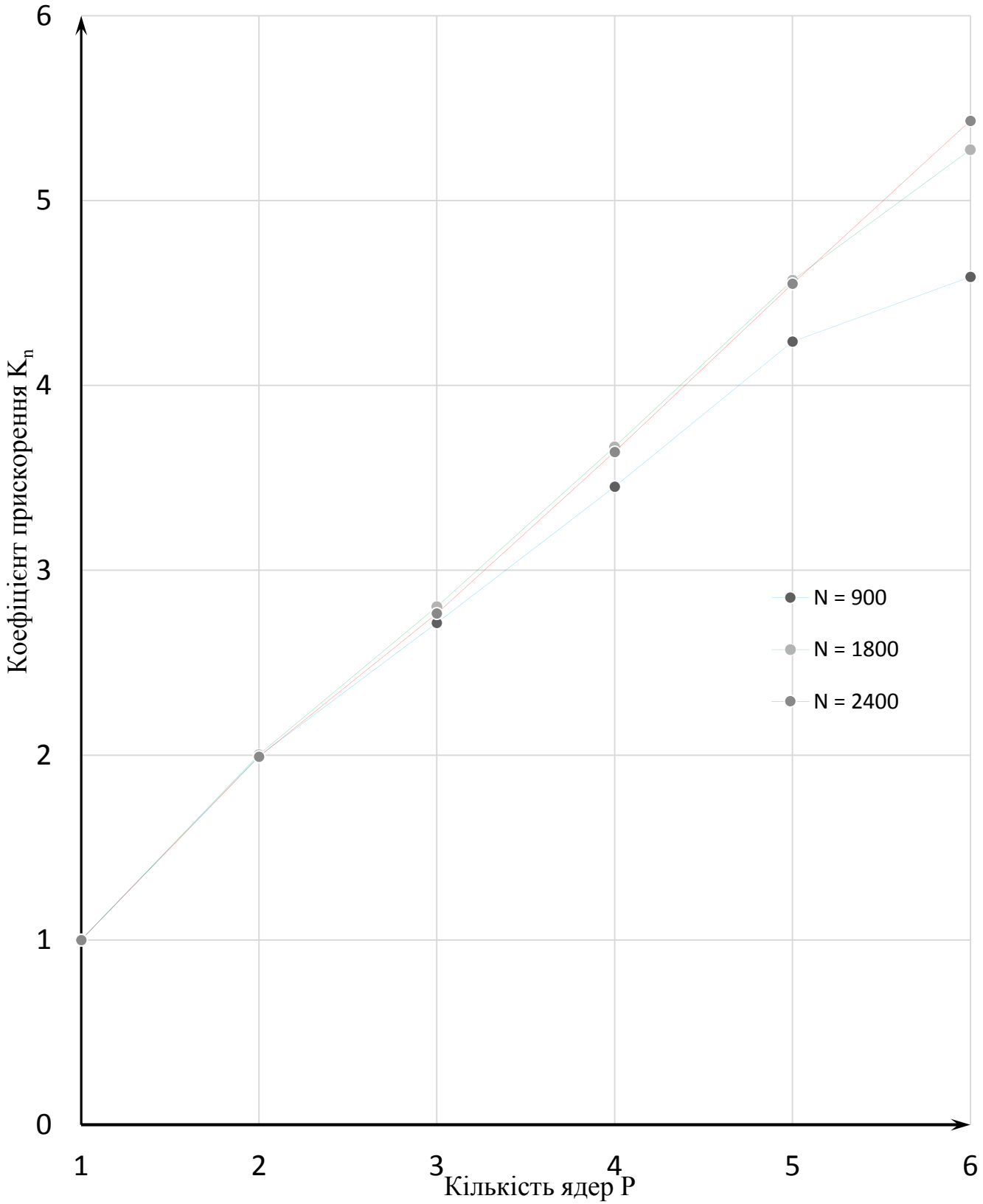

Рис. 2.2 – Графік зміни коефіцієнту прискорення програми ПРГ1 в залежності від кількості ядер

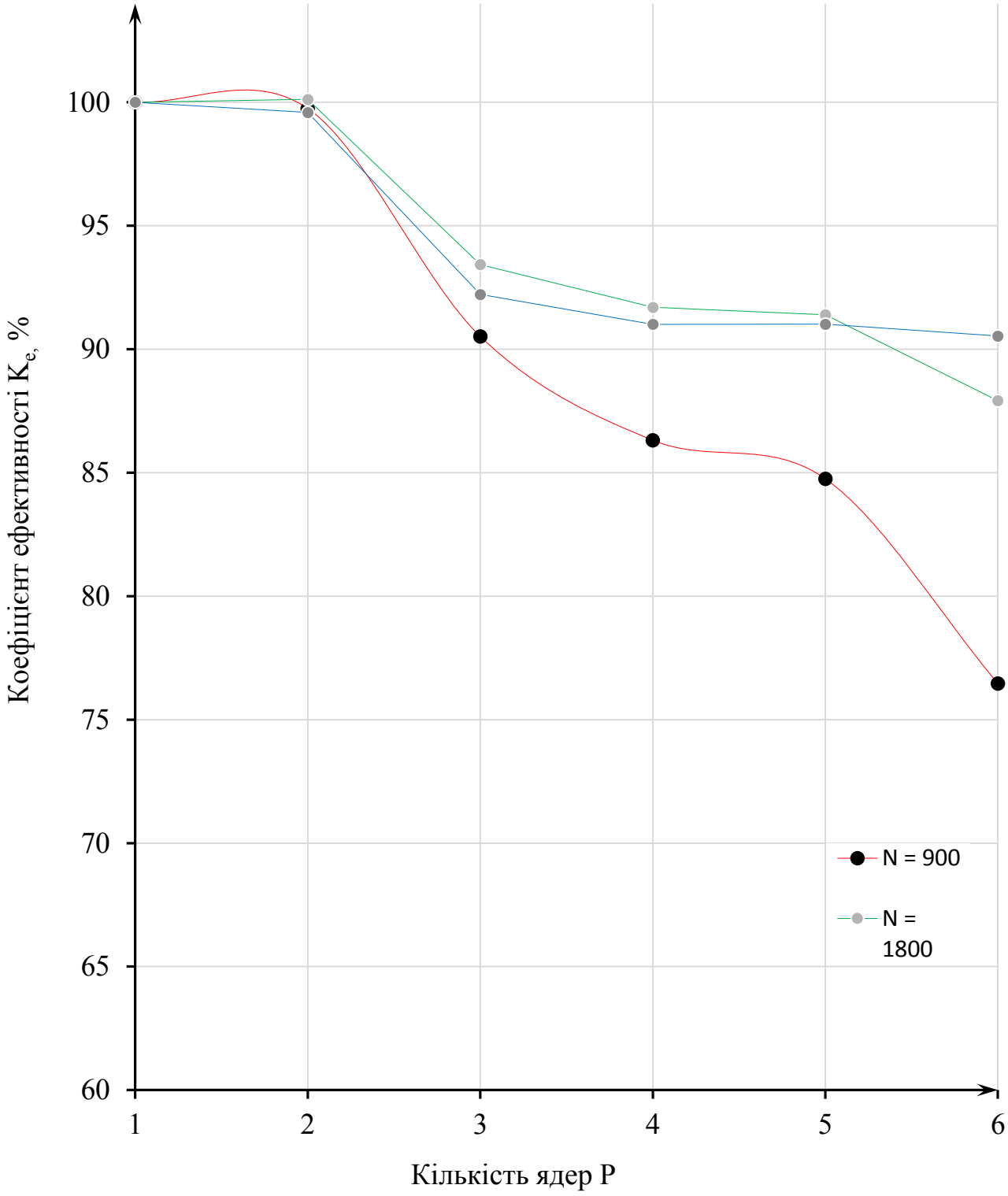

Рис. 2.3 – Графік зміни коефіцієнту ефективності програми ПРГ1 в залежності від кількості ядер

2.6 Висновки до розділу 2

Виконано розробку програми ПРГ1 для ПКС СП з використанням мови С++ і засобів синхронізації з бібліотеки OpenMP. Тестування програми ПРГ1 показало наступне:

- Використання багатоядерної ПКС та програми ПРГ1 забезпечує скорочення часу обчислення заданої математичної задачі при збільшенні P для всіх N, як показано у табл. 2.1. Значення K_n лежить в межах від 1,99 до 5,43, як показано у табл. 2.2.
- Максимальне значення $K_n = 5,43$ забезпечує ПКС з P = 6 та N = 2400.
- Мінімальне значення $K_n = 1,99$ виявлено у ПКС з P = 2 та N = 2400, N = 900.
- З ростом N для усіх значень P K_n лінійно зростає, як показано на рис. 2.2.
- Значення K_e змінюються від 76,46% до 100,12%, як показано у табл.2.3.
- Найефективніше програма ПРГ1 використовує ПКС з P = 2 та N = 1800 при цьому $K_e = 100,12\%$.
- Найнижча ефективність використання ПКС програмою ПРГ1 виявлена при P = 6, N = 900, як зображено на рис. 2.3.
- Зі зростанням P від 2 до 6 K_e падає від 99,74% до 76,46% при N = 900.
- Зі зростанням P від 2 до 6 K_e залишається практично однаковим, змінюючись лише від 93,43% до 87,92% для N = 1800 та від 92,22% до 90,53% для N = 2400.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ПКС ЛП

У даному розділі розроблюється програма ПРГ1 для системи зі спільною пам'яттю, що відповідає технічному завданню, структура представлена на рис. 3.1.

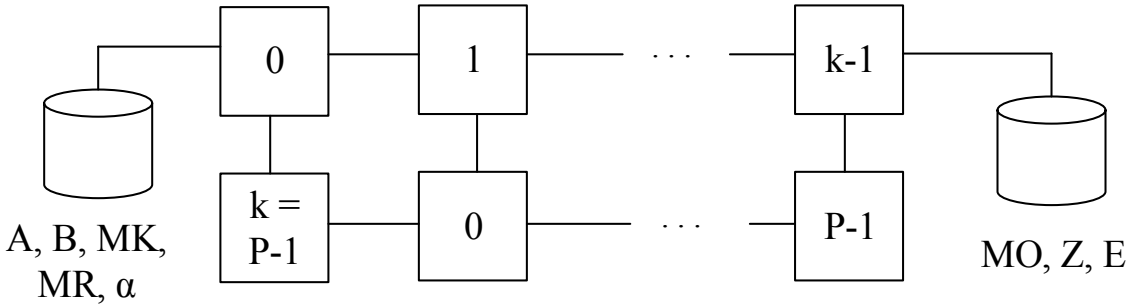

Рис. 3.1 Структура ПКС ЛП

Мова програмування: Java;

Бібліотека: MPI;

Математичний вираз: : $A = B \cdot (MO \cdot MK) \cdot \alpha + \min(Z) \cdot E \cdot MR$.

Оскільки математична задача співпадає тою, що розглядалась в розділі 2.1, то перший крок по рекомендованій методиці [13] не матиме відмінностей від уже виконаних. Тому розділ «Розробка паралельного математичного алгоритму» пропущений.

3.1 Розробка алгоритмів процесів

Для опису алгоритмів задач системи, розділимо всі задачі на 3 групи:

- задачі що належать до крайнього лівого стовпця (0, k);
- задачі що належать до крайнього правого стовпця (k-1, P-1);
- решта задач, що утворюють середні стовпці.

Задача крайнього лівого стовпця

1. Якщо rank = 0, то ввести B, MK, MR, α.
2. Якщо rank = 0, то передати B, $MK_{(k*H)}$, $MR_{(k*H)}$, α. Задачі знизу.
3. Якщо rank = k, то прийняти B, $MK_{(k*H)}$, $MR_{(k*H)}$, α, від задачі зверху.
4. Передати B, $MK_{(k*H)-H}$, $MR_{(k*H)-H}$, α задачі справа.

5. Прийняти MO, E, Z_H від задачі справа
6. Обчислити localMinZ = min(Z_H)
7. Якщо rank = k, то передати localMinZ задачі зверху
8. Якщо rank = 0, то прийняти minZ від задачі знизу
9. Якщо rank = 0, то обчислити localMinZ = min(minZ, localMinZ)
10. Якщо rank = 0, то передати localMinZ задачі з права
11. Якщо rank = 0, то Отримати minZ від задачі з ліва.
12. Якщо rank = 0, то Передати minZ задачі з низу
13. Якщо rank = k, то отримати minZ від задачі зверху
14. Обчислити A_H = B(MO MK_H)α + minZ·E·MR_H
15. Отримати A_{H*k-H} від задачі справа
16. Якщо rank = k, то передати A_{H*k} задачі зверху
17. Якщо rank = 0, то Отримати A_{H*k} від задачі знизу
18. Якщо rank = 0, то Вивести A

Задача крайнього правого стовпця

1. Якщо rank = k-1, то ввести MO, Z, E.
2. Якщо rank = k-1, то передати MO, $Z_{(k*H)}$, E задачі знизу.
3. Прийняти B, MK_H, MR_H, α від задачі зліва.
4. Якщо rank = P-1, то прийняти MO, $Z_{(k*H)}$, E від задачі зверху.
5. Передати MO, $Z_{(k*H)-H}$, E задачі зліва.
6. Обчислити localMinZ = min (Z_H)
7. Якщо rank = P-1, то передати localMinZ задачі зверху
8. Якщо rank = k-1, то прийняти minZ від задачі знизу
9. Якщо rank = k-1, то обчислити localMinZ = min(minZ, localMinZ)
10. Якщо rank = k-1, то передати localMinZ задачі з права
11. Якщо rank = k-1, то Отримати minZ від задачі з ліва.
12. Якщо rank = k-1, то Передати minZ задачі з низу
13. Якщо rank = P-1, то отримати minZ від задачі зверху
14. Обчислити A_H = B(MO MK_H)α + minZ·E·MR_H
15. Передати A_{H*k-H} задачі зліва

Задача середніх стовпців

1. Прийняти B, $MK_{(MK.size())*H}$ MR, α від задачі зліва.
2. Передати B, $MK_{(MK.size()-1)*H}$, MR, α задачі справа.
3. Прийняти MO, $Z_{(Z.size())*H}$, E від задачі справа.
4. Передати MO, $Z_{(Z.size()-1)*H}$, E задачі зліва.
5. Обчислити $m_i = \min(Z_H)$
6. Якщо rank >= k, то передати localMinZ задачі зверху
7. Якщо rank >= k, то прийняти m від задачі знизу
8. Якщо rank < k, то прийняти minZ від задачі знизу
9. Обчислити minZ = (localMinZ, minZ)
10. Якщо rank = >(k-1)/2 отримати minZ від задачі з справа
11. Якщо rank < (k-1)/2 отримати minZ від задачі зліва
12. Обчислити minZ = (localMinZ, minZ)
13. Якщо rank = >(k-1)/2, то передати minZ задачі зліва
14. Якщо rank < (k-1)/2, то передати minZ задачі справа
15. Якщо rank = (k-1)/2, то Отримати minZleft від задачі зліва
16. Якщо rank = (k-1)/2, то Отримати minZright від задачі справа
17. Якщо rank = (k-1)/2, то обчислити minZ = (minZ, minZleft, minZright)
18. Якщо rank = (k-1)/2, то передати minZ задачі зліва і справа
19. Якщо rank = >(k-1)/2 отримати minZ від задачі з зліва
20. Якщо rank < (k-1)/2 отримати minZ від задачі зліва
21. Якщо rank >k, то передати minZ задачі знизу
22. Якщо rank = > (k-1)/2, то передати minZ задачі справа
23. Якщо rank < (k-1)/2, то передати minZ задачі зліва
24. Обчислити $A_H = B(MO\ MK_H)\alpha + minZ \cdot E \cdot MR_H$
25. Отримати A (A.size()-1)*H від задачі справа
26. Передати A (A.size())*H задачі зліва

3.2 Розробка схеми взаємодії процесів ПРГ2

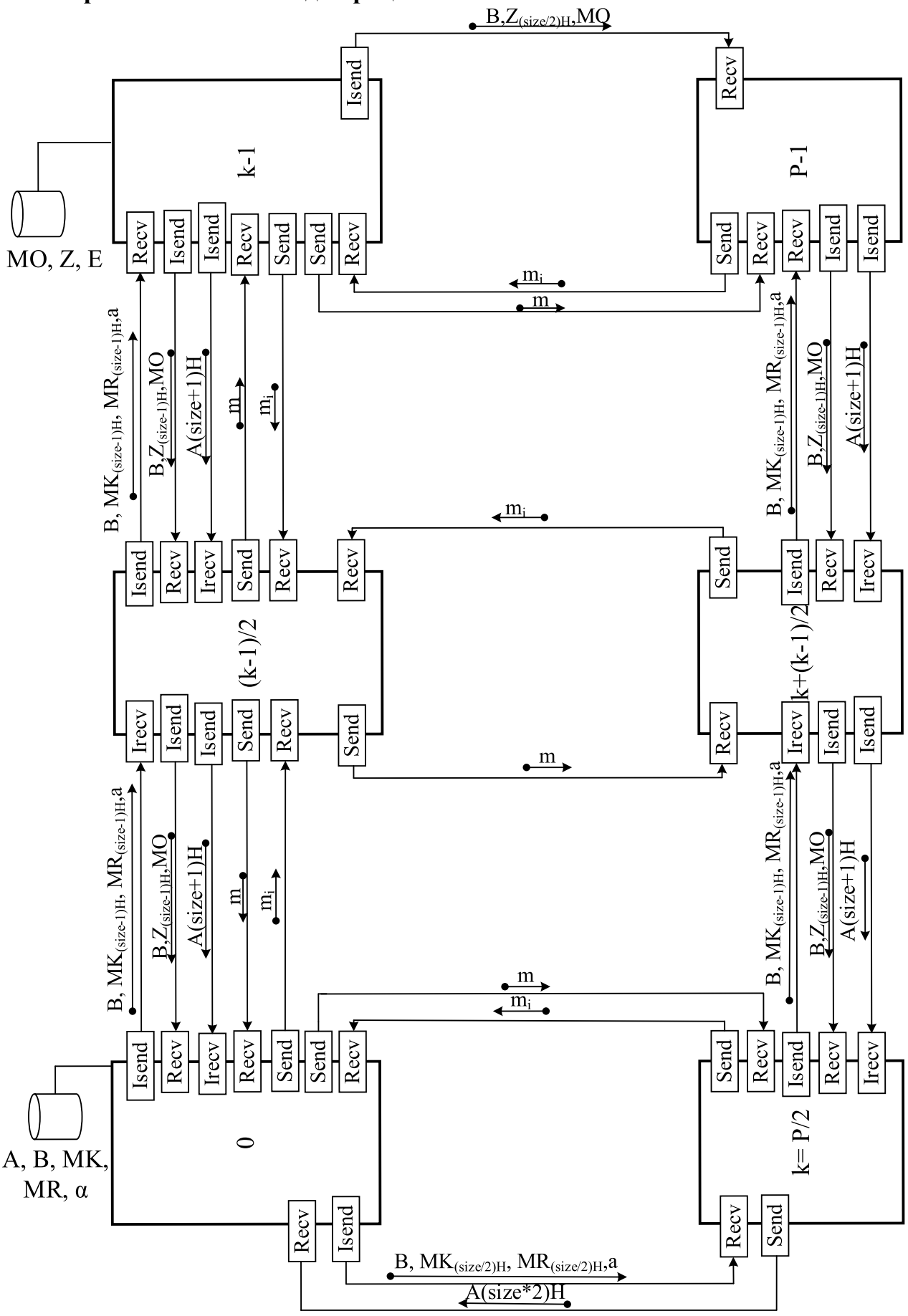

Рис 3.2 Схема взаємодії задач у ПРГ2

3.3 Розробка програми ПРГ2

Програма написана на мові Java [14] з використанням реалізації бібліотеки MPI для Java [10],[9] та складається з семи класів.

Розробка проведена згідно алгоритмів задач, які наведені у підрозділі 3.2, та згідно схеми взаємодії задач, яка зображена на рис. 3.2.

Класи Vector та Matrix інкапсулюють поняття вектору та матриці відповідно, та реалізую методи що дозволяють змінювати стан даних абстракцій.

Клас MessageBox інкапсулює повідомлення яке відправляється у програмі. За допомогою методів, реалізованих у цьому класі можна створювати повідомлення яке місить одночасно декілька матриць, векторів чи чисел. Це повідомлення далі процесори надсилають одне одному для обміну данними.

Клас CalculateUtils містить набір статичних методів що інкапсулюють математичні операції з векторами та матрицями. Також тут знаходяться методи для вводу та виводу данних.

Клас MessageKeys містить набір констант які використовуються для пересилки повідомлень між задачами.

Клас TaskPool інкапсулює задачі, він містить 3 клієнтських метода що відповідають 3 типам задач, алгоритми яких були розроблені у розділі 3.1.

Також клас містить клієнтський метод який виконує операцію, коли у нас лише одна задача.

Клас Executor є виконавчим, від запускає задачі на виконання, і обчислює їх час роботи.

3.4 Тестування ПРГ2

Для тестування ПРГ2 використовувалось те ж саме апаратне і програмне забезпечення, що і для програми ПРГ1.

Для обчислення часу роботи програми використовувалась функція MPI.COMM_WORLD.Reduce(), що знаходить максимальне значення часу роботи програми.

Результати тестування і проведених досліджень ефективності розробленої програми наведено в таблицях 3.1 – 3.3.

Таблиця 3.1 Час виконання програми ПРГ2

N	T_1, мс	T_2, мс	T_4, мс	T_6, мс
900	12870	6833	5039	3791
1800	161148	81432	49346	35163
2400	405016	207980	124052	91338

На основі даних із таблиці 3.1 виконано розрахунок значень коефіцієнтів прискорення, які наведені в таблиці 3.2.

Таблиця 3.2. Коефіцієнти прискорення для програми ПРГ2

	Кількість процесорів (P)			
N	1	2	4	6
900	1,00	1,88	2,55	3,39
1800	1,00	1,98	3,27	4,58
2400	1,00	1,95	3,26	4,43

Коефіцієнти ефективності (таблиця) обчислено за даними таблиці .

Таблиця 3.3 Коефіцієнти ефективності для програми ПРГ2

	Кількість процесорів			
N	1	2	4	6
900	100,00%	94,18%	63,85%	56,58%
1800	100,00%	98,95%	81,64%	76,38%
2400	100,00%	97,37%	81,62%	73,90%

Використовуючи таблиці 3.1 – 3.3 побудовано графіки зміни коефіцієнтів прискорення і ефективності в залежності від N і P.

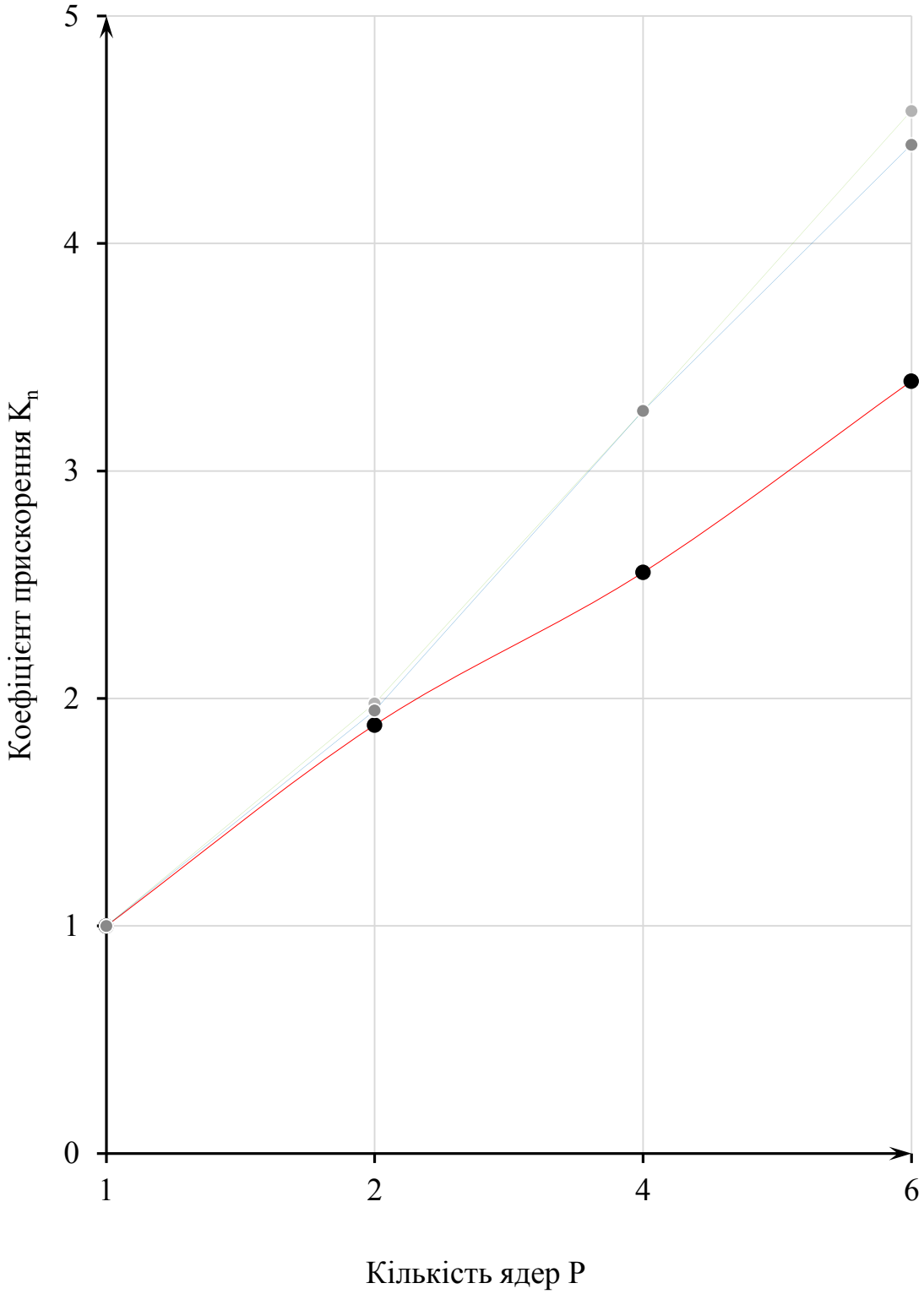

Рис. 3.2. Графік зміни коефіцієнту прискорення програми ПРГ2 в залежності від кількості ядер

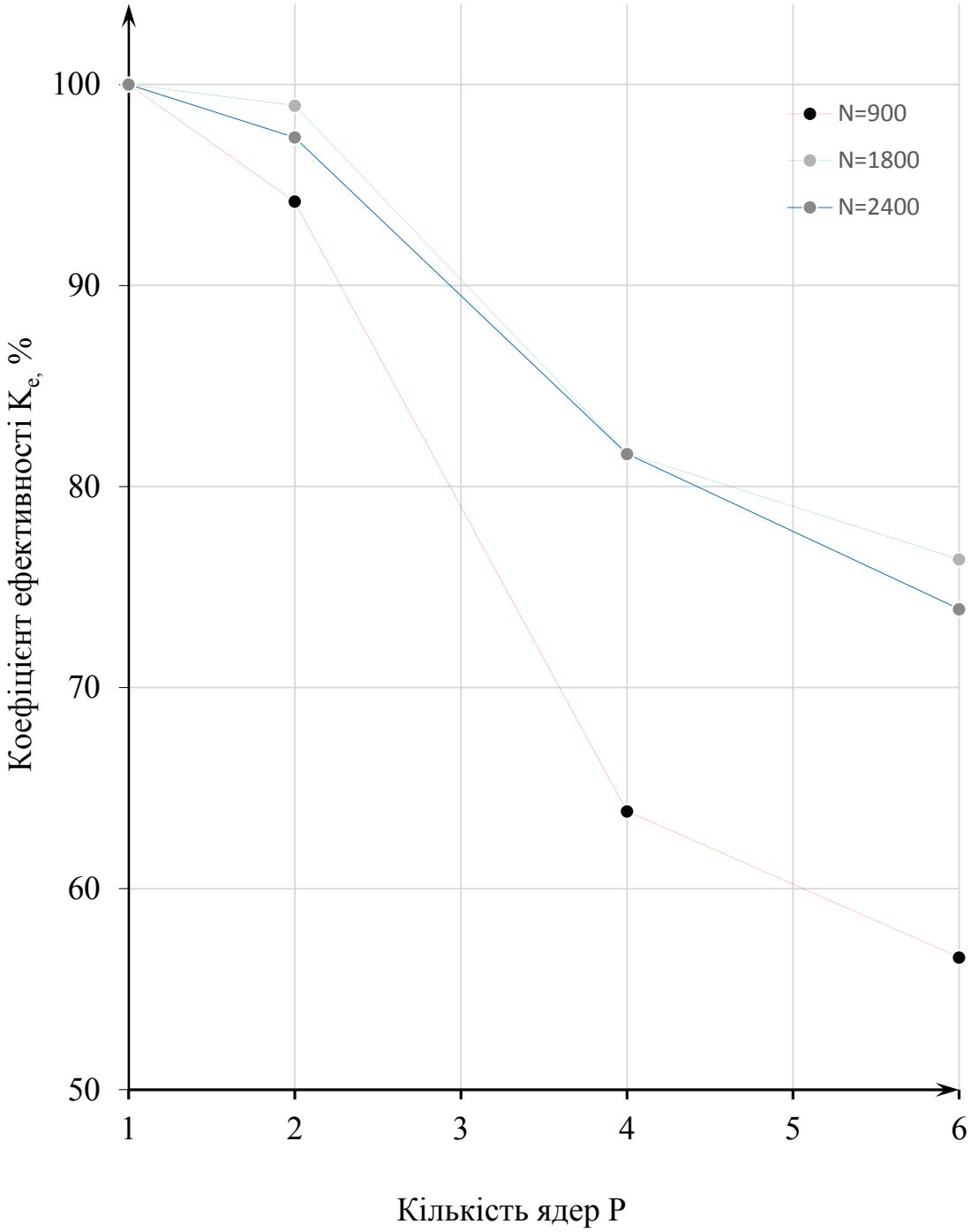

Рис. 3.3. Графік зміни коефіцієнту ефективності програми ПРГ2 в залежності від кількості ядер

3.5 Висновки до розділу 3

Виконано розробку програми ПРГ2 для ПКС ЛП з використанням мови програмування Java і бібліотеки реалізації бібліотеки MPI для цієї мови. Тестування програми показало наступне:

- Використання багатоядерної ПКС та програми ПРГ2 забезпечує скорочення часу обчислення заданої математичної задачі. Із збільшенням кількості процесорів у ПКС ЛП час роботи ПРГ2 зменшується, як показано у табл. 3.1. Значення K_n лежать в межах від 1,88 до 4,5, як показано у табл. 3.2.
- Максимальне значення K_n = 4,58 забезпечує ПКС з P = 6 та N = 2400. Динаміка зміни K_n для ПКС з такими параметрами зображена на рис. 3.4.
- Мінімальне значення K_n = 1,88 виявлено у ПКС з P = 2 та N = 900. Динаміка зміни K_n для ПКС з такими параметрами зображена на рис. 3.2
- З ростом N K_n лінійно зростає для всіх значень, як показано на рис.3.2.
- Значення K_e змінюється від 56,58% до 98,95%, як показано у табл. 3.3.
- Найефективніше програма ПРГ2 використовує ПКС з P = 2 та N = 1800, при цьому K_e = 98,95%.
- Найнижча ефективність використання ПКС програмою ПРГ2 виявлена при P = 6 та N = 900, як показано на рис. 3.5.
- Зі зростанням P від 1 до 6 K_e лінійно спадає, як показано на рис. 3.3.

ОСНОВНІ РЕЗУЛЬТАТИ НА ВИСНОВКИ

- Сучасним способом підвищення продуктивності мікропроцесора є збільшення кількості ядер в одному корпусі.
- Серед основних закономірностей спостерігаються збільшення частоти ядер, збільшення кількості ядер, збільшення кеш-пам’яті, зниження енергоспоживання.
- Починаючи з покоління Bulldozer компанія AMD почала випускати процесори з модульною архітектурою (по 2 ядра в модулі).
- Починаючи з архітектури Piledriver з’явилася нова серія процесорів APU. В них відсутня кеш-пам’ять третього рівня, зате присутній графічний прискорювач, який займає приблизно половину площі процесора.
- Різниця у часі виконання програм ПРГ1 та ПРГ2 пояснюється використанням абсолютно різних технологій. Оскільки ПРГ2 використовує віртуальну JVM, системі необхідний час на трансляцію комманд JVM.
- Значний вплив на різницю у часі роботи ПРГ1 і ПРГ2 впливає складна топологія системи, яку використовує ПРГ2. Це пояснюється тим, що збільшується час на передачу даних між процесорами, і як наслідок, це впливає на загальний час виконання ПРГ2.
- Використання неблокуючих пересилок у ПРГ2, де це дозволяє алгоритм ПРГ2 дещо зменшив час роботи ПРГ2.
- Використання механізму серіалізації об’єктів у Java, що необхідно для пересилки даних у ПРГ2 між процесорами, негативно впливає на час роботи програми.
- Алгоритм роботи для ПРГ1 значно легший і при проектуванні і при реалізації, ніж у ПРГ2. У випаду системи з локальною пам’яттю, велика кількість повідомлень різного розміру ускладнюють систему з пересилкою повідомлень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Знайомство з архітектурою шестиядерних процесорів AMD Phenom II X6 на ядрі Thuban [Електронний ресурс] .— Режим доступу: http://www.easycom.com.ua/data/cpu/1004272158/?lang=ukr
2. Огляд шестиядерного процесора AMD Phenom II X6 1075T[Електронний ресурс] .— Режим доступу: http://www.easycom.com.ua/cpu/amd_phenom_ii_x6_1075t/?lang=ukr
3. Офіційна презентація процесорів лінійки AMD FX [Електронний ресурс] .— Режим доступу: http://www.easycom.com.ua/ittech/ofitsialnaya_prezentatsiya_novyh_vysokoproizvoditelnyh_protsessorov_lineyki_amd_fx/?lang=ukr
4. Огляд шестиядерного процесора AMD Phenom II X6 1075T[Електронний ресурс] .— Режим доступу: http://www.easycom.com.ua/cpu/amd_phenom_ii_x6_1075t/?lang=ukr
5. Огляд шестиядерного процесора AMD Phenom II X6 1055T[Електронний ресурс] .— Режим доступу: http://www.easycom.com.ua/cpu/amd_phenom_ii_x6_1055t/?lang=ukr
6. Огляд шестиядерного процесора AMD Phenom II X6 1100T[Електронний ресурс] .— Режим доступу: http://www.easycom.com.ua/cpu/amd_phenom_ii_x6_1100t/?lang=ukr
7. Огляд шестиядерного процесора AMD FX-6100 на базі архітектури Bulldozer[Електронний ресурс] .— Режим доступу: http://www.easycom.com.ua/cpu/amd_fx-6100/?lang=ukr– Назва з екрану.
8. The OpenMP® API specification for parallel programming[Електронний ресурс] .— Режим доступу http://openmp.org/wp/openmp-specifications/
9. Open MPI v1.8.4 documentation [Електронний ресурс] .–Режим доступу: http://www.open-mpi.org/doc/v1.8/
10. MPJ Express Java-Docs[Електронний ресурс]. — Режим доступу http://mpj-express.org/docs/javadocs/index.html

11. Процессоры AMD FX[Електронний ресурс] .— Режим доступу: http://www.amd.com/ru/products/desktop/processors/amdfx/Pages/amdfx.aspx
12. Процессоры AMD Phenom™ II [Електронний ресурс] .— Режим доступу: http://www.amd.com/ru/products/desktop/processors/phenom-ii/Pages/phenom-ii.aspx
13. Жуков І.А., Корочкін О.В. Паралельні та розподілені обчислення: Навч. Посібник [Текст]. – К.: Корнійчук, 2005. – 226 с. – ISBN 996-7599-36-1.
14. Java™ Platform, Standard Edition 7 API Specification[Електронний ресурс] .— Режим доступу: http://docs.oracle.com/javase/7/docs/api/

ДОДАТКИ

Додаток А. Структурна схема ПКС СП

Додаток Б. Схема алгоритму головної програми зі вказанням паралельних ділянок для ПРГ1

Додаток В. Схема алгоритму процесів для програми ПРГ1

Додаток Г.

Лістинг програми ПРГ1 для ПКС СП

pro2_course_work_prg1.cpp:

```
/**
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                 *
*            Програмування паралельний комп'ютерних систем        *
*             Курсова робота. ПРГ1. Бібліотека OpenMP             *
*                                                                 *
* Завдання: A = B(MO*MK)*a + min(Z)*E*MR                          *
*                                                                 *
* Автор Кузьменко Володимир                                       *
* Група IO-21                                                     *
* Датае 23.03.15                                                  *
*                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*/
#include «stdafx.h»
#include «omp.h»
#include <windows.h>
#include <iostream>
#include «operations.h»
#include <ctime>
#include <clocale>
using namespace std;
#pragma comment(linker, «/stack:160000000»)
const int P = 5;
int main()
{
vector A = new int[N], B, E, Z;
matrix MO, MK, MR, MA;
int alfa;
int minZ = 9999999999;
const int H = N / P;
omp_lock_t lock_Copy;
double start_time = omp_get_wtime();
setlocale(LC_ALL, «Russian»);
omp_init_lock(&lock_Copy);
omp_set_num_threads(P);
#pragma omp parallel
{
                int tid = omp_get_thread_num();
                MA = initMatrix();
                cout << «Задача « << tid << « стартувала»<<endl;
                switch (tid)
                {
                      /*1. Якщо tid = 0 Ввести α, B, MK, MR.*/
                      case      0:
                              alfa = 1;
                              B = inputVector(1);
                              MK = inputMatrix(1);
```

```
49.                                 MR = inputMatrix(1);
50.                           break;
51.                           /*2. Якщо tid = P-1 ввести Z, E, MO.*/
52.                           case      P-1:
53.                                 Z = inputVector(1);
54.                                 E = inputVector(1);
55.                                 MO = inputMatrix(1);
56.                           break;
57.
58.                     }
59.                     /*3. Бар’єр для усіх задач. Синхронізація по вводу*/
60.       #pragma omp barrier
61.                     /*4. Обчислення minZi = min(Z), i = (0,P-1) */
62.                     int minZid = 9999999999;
63.                     for (int i = tid*H; i < (tid+1)*H; i++)
64.                     {
65.                           if (Z[i]< minZid)
66.                           {
67.                                 minZid = Z[i];
68.                           }
69.                     }
70.                     /*5. Обчислення minZ = min(minZ, minZi), i = (0,P-1) .*/
71.       #pragma omp critical
72.                     {
73.                           minZ = min(minZ, minZid);
74.                     }
75.                     /*6. Бар’єр для усіх задач. Синхронізація по обрахунку minZ*/
76.       #pragma omp barrier
77.                     matrix Moid;
78.                     vector Bid;
79.                     vector Eid;
80.                     int alfaId;
81.                     /*7. Копіювати Moi=MO, minZi = minZ, αi = α, Bi = B, Ei = E*/
82.                     omp_set_lock(&lock_Copy);
83.                     Moid = copyMatrix(MO);
84.                     alfaId = alfa;
85.                     minZid = minZ;
86.                     Bid = copyVector(B);
87.                     Eid = copyVector€;
88.                     omp_unset_lock(&lock_Copy);
89.
90.                     /*8. Обчислення AH = αi·Bi·(Moi·MKH) + minZi·Ei·MRH, i = (0,P-1)*/
91.                     int buf;
92.                     int sum;
93.                     for (int i = tid*H; i < (tid + 1)*H; i++)
94.                     {
95.                           A[i] = 0;
96.                           for (int j = 0; j < N; j++)
97.                           {
98.                                 sum = 0;
99.                                 for (int z = 0; z < N; z++) {
100.                                      sum += Moid[i][j] * MK[j][z];
```

```
                                    }
                                    A[i] += Bid[j] * sum;
                              }
                              A[i] = A[i] * alfaId;
                              buf = 0;
                              for (int j = 0; j < N; j++)
                              {
                                    buf += Eid[j] * MR[j][i];
                              }
                              A[i] += buf*minZid;
                        }
                        /*9. Бар’єр для усіх задач. Синхронізація по обчисленню A*/
      #pragma omp barrier
                        /*10. Якщо tid = 0 вивести результат AH.*/
                        if (tid==0)
                        {
                              output(A);
                        }

                        cout << «Задача « << tid << « завершилась» << endl;
      }
      double end_time = omp_get_wtime();
      double  elapsedTime = end_time – start_time;
      cout << «Час роботи: « << elapsedTime << endl;
      return 0;
      }
```

Файл operations.cpp:

```
/**
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                   *
*            Програмування паралельний комп’ютерних систем          *
*             Курсова робота. ПРГ1. Бібліотека OpenMP               *
*                                                                   *
* Завдання: A = B(MO*MK)*a + min(Z)*E*MR                           *
* Автор Кузьменко Володимир                                        *
* Група IO-21                                                      *
* Дата 23.03.15                                                    *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*/
#include «stdafx.h»
#include <windows.h>
using namespace std;
#include «operations.h»
#include <iostream>
vector inputVector(int value){
        vector result = new int[N];
        for (int i = 0; i < N; i++)
        {
               result[i] = value;
        }
        return result;
```

```
26.   }
27.   matrix inputMatrix(int value){
28.          matrix result = new vector[N];
29.          for (int i = 0; i < N; i++)
30.          {
31.                 result[i] = new int[N];
32.          }
33.          for (int i = 0; i < N; i++)
34.          {
35.                 for (int j = 0; j < N; j++)
36.                 {
37.                        result[i][j] = value;
38.                 }
39.          }
40.          return result;
41.   }
42.   void output(vector v) {
43.                 if (N <= 20) {
44.                     for (int i = 0; i < N; i++) {
45.                               cout << v[i] << « «;
46.                        }
47.                  cout << endl;
48.                 }
49.   }
50.   vector copyVector(vector v){
51.          vector result = new int[N];
52.          for (int i = 0; i < N; i++)
53.          {
54.                 result[i] = v[i];
55.          }
56.          return result;
57.   }
58.
59.   matrix copyMatrix(matrix m){
60.          matrix result = new vector[N];
61.          for (int i = 0; i < N; i++)
62.          {
63.                 result[i] = new int[N];
64.          }
65.

66.          for (int i = 0; i < N; i++)
67.          {
68.                 for (int j = 0; j < N; j++)
69.                 {
70.                        result[i][j] = m[i][j];
71.                 }
72.          }
73.          return result;
74.   }
75.
76.   matrix initMatrix(){
77.          matrix result = new vector[N];
```

```
78.          for (int i = 0; i < N; i++)
79.          {
80.                 result[i] = new int[N];
81.          }
82.          return result;
83.   }
```

Файл operations.h:

```
1.   /**
2.   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
3.   *                                                                   *
4.   *            Програмування паралельний комп’ютерних систем         *
5.   *             Курсова робота. ПРГ1. Бібліотека OpenMP              *
6.   *                                                                   *
7.   * Завдання: A = B(MO*MK)*a + min(Z)*E*MR                           *
8.   *                                                                   *
9.   * Автор Кузьменко Володимир                                         *
10.  * Група ІО-21                                                       *
11.  * Дата 23.03.15                                                    *
12.  *                                                                   *
13.  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
14.  */
15.

16.  typedef int* vector;
17.  typedef int** matrix;
18.

19.  const int N =10;
20.

21.  vector inputVector(int);
22.  matrix inputMatrix(int);
23.  void output(vector);
24.  void output(matrix);
25.  vector copyVector(vector);
26.  matrix copyMatrix(matrix);
27.  matrix initMatrix();
28.
```

Додаток Д. Структурна схема ПКС ЛП

Додаток Е. Схема алгоритму головної програми зі вказанням паралельних ділянок для ПРГ2

Додаток Ж. Схема алгоритму процесів для програми ПРГ2

Додаток З.

Лістинг програми ПРГ2

Файл CalculateUtils.java

```
1.  /**
2.  ****************************************************************
3.  *                                                              *
4.  *            Програмування паралельний комп'ютерних систем     *
5.  *             Курсова робота. ПРГ2. Бібліотека MPI             *
6.  *                                                              *
7.  * Завдання: A = B(MO*MK)*a + min(Z)*E*MR                       *
8.  *                                                              *
9.  * Файл: CalculateUtils.java                                    *
10. * Автор: Кузьменко Володимир                                  *
11. * Група: ІО-21                                                *
12. * Дата: 23.04.15                                              *
13. *                                                              *
14. ****************************************************************
15. */
16. public class CalculateUtils {
17.
18.         public static Vector inputVector(int value) {
19.                 Vector vector = new Vector(Executor.N);
20.                 for (int i = 0; i < vector.size(); i++) {
21.                         vector.set(i, value);
22.                 }
23.                 return vector;
24.         }
25.
26.         public static Matrix inputMatrix(int value) {
27.                 Matrix matrix = new Matrix(Executor.N);
28.                 for (int i = 0; i < matrix.size(); i++) {
29.                         for (int j = 0; j < matrix.size(); j++) {
30.                                 matrix.set(i, j, value);
31.                         }
32.                 }
33.                 return matrix;
34.         }
35.
36.         public static void outputVector(Vector vector) {
37.                 if (vector.size() <= 40) {
38.                         System.out.print(vector.toString());
39.                 }
40.         }
41.         public static void outputMatrix(Matrix matrix) {
42.                 if (matrix.size() <= 40) {
43.                         System.out.print(matrix.toString());
44.                 }
45.         }
46.         public static Vector add(final Vector left, final Vector right) {
47.
48.                 Vector result = new Vector(left.size());
```

```
49.                 for (int i = 0; i < left.size(); i++) {
50.                         result.set(i, left.get(i) + right.get(i));
51.                 }
52.                 return result;
53.         }
54.         public static Vector mult(final int left, final Vector right) {
55.                 Vector result = new Vector(right.size());
56.                 for (int i = 0; i < right.size(); i++) {
57.                         result.set(i, left * right.get(i));
58.                 }
59.                 return result;
60.         }
61.         public static Matrix mult(final Matrix left, final Matrix right) {
62.
63.                 Matrix result = new Matrix(right.size());
64.                 result = result.copy(0, left.size());
65.                 for (int i = 0; i < left.size(); i++) {
66.                         for (int j = 0; j < right.size(); j++) {
67.                                 result.set(i, j, 0);
68.                                 for (int y = 0; y < right.size(); y++) {
69.                                         result.set(i, j,
70.                                                         result.get(i, j) + left.get(i, y) *
    right.get(y, j));
71.                                 }
72.                         }
73.                 }
74.                 return result;
75.         }
76.         public static Vector mult(final Vector left, final Matrix right) {
77.
78.                 Vector result = new Vector(right.size());
79.                 for (int i = 0; i < right.size(); i++) {
80.                         result.set(i, 0);
81.                         for (int j = 0; j < left.size(); j++) {
82.                                 result.set(i, result.get(i) + left.get(j) * right.get(i, j));
83.                         }
84.                 }
85.                 return result;
86.         }
87.         public static int min(final Vector vector) {
88.                 int min = Integer.MAX_VALUE;
89.                 for (int i = 0; i < vector.size(); i++) {
90.                         if (vector.get(i) < min) {
91.                                 min = vector.get(i);
92.                         }
93.                 }
94.                 return min;
95.         }
96. }
```

Файл Executor.java:

```
1.  import mpi.*;
2.  /**
```

```
3.    ****************************************************************
4.    *                                                              *
5.    *            Програмування паралельний комп'ютерних систем     *
6.    *             Курсова робота. ПРГ2. Бібліотека MPI             *
7.    *                                                              *
8.    * Завдання: A = B(MO*MK)*a + min(Z)*E*MR                       *
9.    *                                                              *
10.   * Файл: Executor.java                                          *
11.   * Автор: Кузьменко Володимир                                   *
12.   * Група: ІО-21                                                 *
13.   * Дата: 23.04.15                                               *
14.   *                                                              *
15.   ****************************************************************
16.   */
17.
18.   public class Executor {
19.          public static int N;
20.          public static int P;
21.          public static int H;
22.          public static int k;
23.
24.          public static void main(String[] args) throws Exception {
25.                MPI.Init(args);
26.                System.out.println("Task " + MPI.COMM_WORLD.Rank() + " started");
27.                P = Integer.parseInt(args[1]);
28.                N = Integer.parseInt(args[3]);
29.                H = N / P;
30.                k = P / 2;
31.                long start;
32.                long[] timework = new long[1];
33.                long[] buf = new long[1];
34.                TaskPool pool = new TaskPool();
35.                start = System.currentTimeMillis();
36.                if (MPI.COMM_WORLD.Size() == 1) {
37.                      pool.singleThreadTask();
38.                } else {
39.                      pool.leftTaskGroup();
40.                      pool.middleTaskGroup();
41.                      pool.rightTaskGroup();
42.                }
43.                buf[0] = System.currentTimeMillis() - start;
44.                MPI.COMM_WORLD.Reduce(buf, 0, timework, 0, 1, MPI.LONG, MPI.MAX, 0);
45.                System.out.println("Task " + MPI.COMM_WORLD.Rank() + " finished");
46.                if (MPI.COMM_WORLD.Rank() == 0) {
47.                      System.out.println("Time work: " + timework[0]);
48.                }
49.
50.                MPI.Finalize();
51.
52.          }
53.   }
```

Файл Matrix.java:

```
1.    import java.io.Serializable;
2.
3.    /**
4.    ****************************************************************
5.    *                                                              *
6.    *            Програмування паралельний комп'ютерних систем     *
7.    *             Курсова робота. ПРГ2. Бібліотека MPI             *
8.    *                                                              *
9.    * Завдання: A = B(MO*MK)*a + min(Z)*E*MR                       *
10.   *                                                              *
11.   * Файл: Matrix.java                                           *
12.   * Автор: Кузьменко Володимир                                  *
13.   * Група: ІО-21                                                *
14.   * Дата: 23.04.15                                              *
15.   *                                                              *
16.   ****************************************************************
17.   */
18.   public class Matrix implements Serializable {
19.          private static final long serialVersionUID = 1L;
20.          private Vector[] array;
21.
22.          public Matrix(int n) {
23.                 array = new Vector[n];
24.                 for (int i = 0; i < array.length; i++) {
25.                        array[i] = new Vector(n);
26.                 }
27.          }
28.          public void set(int n, int m, int val) {
29.                 array[n].set(m, val);
30.          }
31.          public int get(int n, int m) {
32.                 return array[n].get(m);
33.          }
34.          public Vector get(int index) {
35.                 return array[index];
36.          }
37.
38.          public int size() {
39.                 return array.length;
40.          }
41.          public String toString() {
42.                 String res = "";
43.                 for (int i = 0; i < array.length; i++) {
44.                        res += array[i].toString();
45.                 }
46.                 return res;
47.          }
48.          public Matrix copy(int start, int size) {
49.                 Matrix result = new Matrix(size);
50.                 for (int i = 0; i < result.size(); i++) {
51.                        result.array[i] = array[i + start].copy(0, Executor.N);
52.                 }
53.                 return result;
```

```
54.          }
55.    }
```

Файл MessageBox.java:

```
1.     import java.io.Serializable;
2.     import java.util.ArrayList;
3.     /**
4.     ****************************************************************
5.     *                                                              *
6.     *            Програмування паралельний комп'ютерних систем     *
7.     *             Курсова робота. ПРГ2. Бібліотека MPI             *
8.     *                                                              *
9.     * Завдання: A = B(MO*MK)*a + min(Z)*E*MR                       *
10.    *                                                              *
11.    * Файл: MessageBox.java                                        *
12.    * Автор: Кузьменко Володимир                                   *
13.    * Група: ІО-21                                                 *
14.    * Дата: 23.04.15                                               *
15.    *                                                              *
16.    ****************************************************************
17.    */
18.    public class MessageBox implements Serializable {
19.           private static final long serialVersionUID = 1L;
20.           private ArrayList<Matrix> matrixs = new ArrayList<Matrix>(3);
21.           private ArrayList<Vector> vectors = new ArrayList<Vector>(3);
22.           private ArrayList<Integer> values = new ArrayList<>(2);
23.
24.           public void addMatrix(Matrix matrix) {
25.                  matrixs.add(matrix);
26.
27.           }
28.
29.           public Matrix setMatrix(int key, Matrix matrix) {
30.                  return matrixs.set(key, matrix);
31.           }
32.
33.           public void AddVector(Vector vector) {
34.                  vectors.add(vector);
35.           }
36.
37.           public Vector setVector(int key, Vector vector) {
38.                  return vectors.set(key, vector);
39.           }
40.
41.           public void AddValue(int value) {
42.                  values.add(value);
43.           }
44.
45.           public int setValue(int key, int value) {
46.                  return values.set(key, value);
47.           }
48.
49.           public Matrix getMatrix(int key) {
```

```
50.                 return matrixs.get(key);
51.          }
52.
53.          public Vector getVector(int key) {
54.                 return vectors.get(key);
55.          }
56.
57.          public int getValue(int key) {
58.                 return values.get(key);
59.          }
60.   }
```

Файл MessageKeys.java:

```
1.   /**
2.   ****************************************************************
3.   *                                                              *
4.   *            Програмування паралельний комп'ютерних систем     *
5.   *             Курсова робота. ПРГ2. Бібліотека MPI             *
6.   *                                                              *
7.   * Завдання: A = B(MO*MK)*a + min(Z)*E*MR                       *
8.   *                                                              *
9.   * Файл: MessageKeys.java                                       *
10.  * Автор: Кузьменко Володимир                                   *
11.  * Група: ІО-21                                                 *
12.  * Дата: 23.04.15                                              *
13.  *                                                              *
14.  ****************************************************************
15.  */
16.  public class MessageKeys {
17.         static int SEND_RECIVE_B_MK_MR_alfa = 1;
18.         static int SEND_RECIVE_Z_E_MO = 2;
19.         static int SEND_RECIVE_RESULT_A = 3;
20.         static int SEND_RECIVE_B_MK_MR_alfa_TO_k=4;
21.         static int SEND_RECIVE_MO_Z_E_TO_P_1=5;
22.         static int SEND_RECIVE_A_TO_0 = 6;
23.  }
```

Файл TaskPool.java:

```
1.     import mpi.MPI;
2.     import mpi.Request;
3.     /**
4.     ****************************************************************
5.     *                                                              *
6.     *            Програмування паралельний комп'ютерних систем     *
7.     *             Курсова робота. ПРГ2. Бібліотека MPI             *
8.     *                                                              *
9.     * Завдання: A = B(MO*MK)*a + min(Z)*E*MR                       *
10.    *                                                              *
11.    * Файл: TaskPool.java                                         *
12.    * Автор: Кузьменко Володимир                                   *
13.    * Група: ІО-21                                                 *
14.    * Дата: 23.04.15                                              *
15.    *                                                              *
```

```
16.     ****************************************************************
17.     */
18.     public class TaskPool {
19.             private int P = MPI.COMM_WORLD.Size();
20.             private int H = Executor.H;
21.             private int k = P / 2;
22.             private int rank;
23.             int leftRank;
24.             int rightRank;
25.             int minRank = (k - 1) / 2;
26.             int endFirsRowRank = k - 1;
27.             int middleRightRank = P - 1;
28.             private MessageKeys KEY;
29.             boolean notFourCore = true;
30.             boolean notTwoCore = true;
31.
32.             private Vector B, E, Z, A;
33.             private Matrix MK, MR, MO;
34.             private int alfa;
35.             private int localMin = Integer.MAX_VALUE;
36.
37.             private MessageBox[] box = new MessageBox[1];
38.             private MessageBox message = new MessageBox();
39.
40.             public TaskPool() {
41.                     rank = MPI.COMM_WORLD.Rank();
42.                     rightRank = rank + 1;
43.                     leftRank = rank - 1;
44.
45.                     if (P == 4) {
46.                             minRank = 1;
47.                             endFirsRowRank = 1;
48.                             notFourCore = false;
49.                     }
50.
51.                     if (P == 2) {
52.                             notTwoCore = false;
53.                             endFirsRowRank = 1;
54.                             minRank = 1;
55.
56.                     }
57.
58.             }
59.
60.             public void singleThreadTask() {
61.                     Vector B, E, Z, A;
62.                     Matrix MK, MR, MO;
63.                     int alfa = 1;
64.                     E = CalculateUtils.inputVector(1);
65.                     Z = CalculateUtils.inputVector(10);
66.                     Z.set(7, 1);
67.                     MO = CalculateUtils.inputMatrix(1);
68.                     B = CalculateUtils.inputVector(1);
```

```
69.                 MR = CalculateUtils.inputMatrix(1);
70.                 MK = CalculateUtils.inputMatrix(1);
71.
72.                 A = CalculateUtils.add(
73.                               CalculateUtils.mult(alfa,
74.                                             CalculateUtils.mult(B, CalculateUtils.mult(MK,
      MO))),
75.                               CalculateUtils.mult(CalculateUtils.min(Z),
76.                                             CalculateUtils.mult(E, MR)));
77.
78.                 CalculateUtils.outputVector(A);
79.
80.           }
81.
82.           @SuppressWarnings("static-access")
83.           public void leftTaskGroup() {
84.                 if (rank == 0 || (rank == k && notTwoCore)) {
85.
86.                       if (rank == 0) {
87.                             // 1. Якщо rank = 0, то ввести B, MK, MR, α.
88.                             alfa = 1;
89.
90.                             B = CalculateUtils.inputVector(1);
91.                             MR = CalculateUtils.inputMatrix(1);
92.                             MK = CalculateUtils.inputMatrix(1);
93.
94.                             message.AddValue(alfa);
95.                             message.AddVector(B);
96.                             if (notTwoCore) {
97.                                   message.addMatrix(MK.copy(MK.size() / 2, MK.size() /
      2));
98.                                   message.addMatrix(MR.copy(MR.size() / 2, MR.size() /
      2));
99.                                   MR = MR.copy(0, MK.size() / 2);
100.                                  MK = MK.copy(0, MK.size() / 2);
101.                                  box[0] = message;
102.                                  // 2. Якщо rank = 0, то передати B, MK(k*H), MR(k*H),
      α.
103.                                  // Задачі знизу.
104.                                  MPI.COMM_WORLD.Isend(box, 0, 1, MPI.OBJECT, k,
105.                                              KEY.SEND_RECIVE_B_MK_MR_alfa_TO_k);
106.                            } else {
107.                                  message.addMatrix(MK.copy(H, MR.size() - H));
108.                                  message.addMatrix(MR.copy(H, MR.size() - H));
109.                                  box[0] = message;
110.                            }
111.
112.                            message.setMatrix(0, MK.copy(H, MR.size() - H));
113.                            message.setMatrix(1, MR.copy(H, MR.size() - H));
114.                            // 4. Передати B, MK(k*H)-H, MR(k*H)-H, α задачі справа.
115.                            MPI.COMM_WORLD.Isend(box, 0, 1, MPI.OBJECT, rightRank,
116.                                        KEY.SEND_RECIVE_B_MK_MR_alfa);
117.
```

```
118.                                box[0] = null;
119.                        }
120.                        if (rank == k && notTwoCore) {
121.                                // 3. Якщо rank = k, то прийняти B, MK(k*H), MR(k*H), α, від
122.                                // задачі зверху.
123.                                MPI.COMM_WORLD.Recv(box, 0, 1, MPI.OBJECT, 0,
124.                                                KEY.SEND_RECIVE_B_MK_MR_alfa_TO_k);
125.                                MK = box[0].getMatrix(0);
126.                                MR = box[0].getMatrix(1);
127.
128.                                box[0].setMatrix(0, MK.copy(H, MK.size() - H));
129.                                box[0].setMatrix(1, MR.copy(H, MR.size() - H));
130.                                // 4. Передати B, MK(k*H)-H, MR(k*H)-H, α задачі справа.
131.                                MPI.COMM_WORLD.Isend(box, 0, 1, MPI.OBJECT, rightRank,
132.                                                KEY.SEND_RECIVE_B_MK_MR_alfa);
133.
134.                                alfa = box[0].getValue(0);
135.                                B = box[0].getVector(0);
136.                                box[0] = null;
137.
138.                        }
139.
140.                        MK = MK.copy(0, H);
141.                        MR = MR.copy(0, H);
142.                        // 5. Прийняти MO, E, ZH від задачі справа
143.                        MPI.COMM_WORLD.Recv(box, 0, 1, MPI.OBJECT, rightRank,
144.                                        KEY.SEND_RECIVE_Z_E_MO);
145.                        E = box[0].getVector(0);
146.                        Z = box[0].getVector(1);
147.                        calcMinButtonTask();
148.                        MO = box[0].getMatrix(0);
149.                        box[0] = null;
150.                        Request r = MPI.COMM_WORLD.Irecv(box, 0, 1, MPI.OBJECT, rightRank,
151.                                        KEY.SEND_RECIVE_RESULT_A);
152.                        // 14. Обчислити MAH = B(MO MKH)α + minZ·E·MRH
153.                        A = CalculateUtils.add(CalculateUtils.mult(alfa,
154.                                        CalculateUtils.mult(B, CalculateUtils.mult(MK, MO))),
155.                                        CalculateUtils.mult(localMin, CalculateUtils.mult(E,
      MR)));
156.
157.                        // 15. Отримати MAH*k-H від задачі справа
158.                        r.Wait();
159.                        A.merge(box[0].getVector(0));
160.                        if (rank == k && notTwoCore) {
161.                                box[0].setVector(0, A);
162.                                // 16. Якщо rank = k, то передати MAH*k задачі зверху
163.                                MPI.COMM_WORLD.Send(box, 0, 1, MPI.OBJECT, 0,
164.                                                KEY.SEND_RECIVE_A_TO_0);
165.                        }
166.                        if (rank == 0) {
167.                                if (notTwoCore) {
168.                                        box[0] = null;
```

```
169.                                    // 17. Якщо rank = 0, то Отримати МАН*k від задачі
      знизу
170.                                    MPI.COMM_WORLD.Recv(box, 0, 1, MPI.OBJECT, k,
171.                                                  KEY.SEND_RECIVE_A_TO_0);
172.                                    A.merge(box[0].getVector(0));
173.                             }
174.                             // 18. Якщо rank = 0, то Вивести МА
175.                             CalculateUtils.outputVector(A);
176.                      }
177.
178.               }
179.
180.         }
181.         @SuppressWarnings("static-access")
182.         public void middleTaskGroup() {
183.                if (rank != 0 && rank != k && rank != endFirsRowRank && rank != P - 1) {
184.                      Request[] inputRequest = new Request[2];
185.                      // 1. Прийняти B, MK(MK.size())*H MR, α від задачі зліва.
186.                      inputRequest[0] = MPI.COMM_WORLD.Irecv(box, 0, 1, MPI.OBJECT,
187.                                    leftRank, KEY.SEND_RECIVE_B_MK_MR_alfa);
188.                      inputRequest[0].Wait();
189.                      MK = box[0].getMatrix(0);
190.                      MR = box[0].getMatrix(1);
191.                      box[0].setMatrix(0, MK.copy(H, MK.size() - H));
192.                      box[0].setMatrix(1, MR.copy(H, MR.size() - H));
193.                      // 2. Передати B, MK(MK.size()-1)*H, MR, α задачі справа.
194.                      MPI.COMM_WORLD.Isend(box, 0, 1, MPI.OBJECT, rightRank,
195.                                    KEY.SEND_RECIVE_B_MK_MR_alfa);
196.                      B = box[0].getVector(0);
197.                      alfa = box[0].getValue(0);
198.                      box[0] = null;
199.                      // 3. Прийняти MO, Z(Z.size())*H, E від задачі справа.
200.                      inputRequest[1] = MPI.COMM_WORLD.Irecv(box, 0, 1, MPI.OBJECT,
201.                                    rightRank, KEY.SEND_RECIVE_Z_E_MO);
202.                      inputRequest[1].Wait();
203.                      Z = box[0].getVector(1);
204.                      box[0].setVector(1, Z.copy(H, Z.size() - H));
205.                      // 4. Передати MO, Z(Z.size()-1)*H, E задачі зліва.
206.                      MPI.COMM_WORLD.Isend(box, 0, 1, MPI.OBJECT, leftRank,
207.                                    KEY.SEND_RECIVE_Z_E_MO);
208.                      Z = Z.copy(0, H);
209.                      calcMinButtonTask();
210.                      E = box[0].getVector(0);
211.                      MO = box[0].getMatrix(0);
212.                      MK = MK.copy(0, H);
213.                      MR = MR.copy(0, H);
214.
215.                      Request r = MPI.COMM_WORLD.Irecv(box, 0, 1, MPI.OBJECT, rightRank,
216.                                    KEY.SEND_RECIVE_RESULT_A);
217.                      // 22. Обчислити МАН = B(MO MKH)α + minZ·E·MRH
218.                      A = CalculateUtils.add(CalculateUtils.mult(alfa,
219.                                    CalculateUtils.mult(B, CalculateUtils.mult(MK, MO))),
```

```
220.                                     CalculateUtils.mult(localMin, CalculateUtils.mult(E,
      MR)));
221.
222.                         // 23. Отримати MA (MA.size()-1)*H від задачі справа
223.                         r.Wait();
224.                         A.merge(box[0].getVector(0));
225.                         box[0].setVector(0, A);
226.                         // 24. Передати MA (MA.size())*H задачі зліва
227.                         MPI.COMM_WORLD.Isend(box, 0, 1, MPI.OBJECT, leftRank,
228.                                     KEY.SEND_RECIVE_RESULT_A);
229.
230.                 }
231.         }
232.
233.         @SuppressWarnings("static-access")
234.         public void rightTaskGroup() {
235.                 if (rank == P - 1 || (rank == endFirsRowRank && notTwoCore)) {
236.
237.                         if (rank == endFirsRowRank) {
238.                                 // 1. Якщо rank = P/2-1, то ввести MO, Z, E.
239.                                 E = CalculateUtils.inputVector(1);
240.                                 Z = CalculateUtils.inputVector(10);
241.                                 Z.set(7, 1);
242.                                 MO = CalculateUtils.inputMatrix(1);
243.                                 message.addMatrix(MO);
244.                                 message.AddVector(E);
245.                                 if (notTwoCore) {
246.                                         message.AddVector(Z.copy(k * H, Z.size() / 2));
247.                                         box[0] = message;
248.                                         // 2. Якщо rank = P/2-1, то передати MO, Z(k*H), E
      задачі
249.                                         // знизу.
250.                                         MPI.COMM_WORLD.Isend(box, 0, 1, MPI.OBJECT, P - 1,
251.                                                         KEY.SEND_RECIVE_MO_Z_E_TO_P_1);
252.                                         box[0].setVector(1, Z.copy(H, Z.size() / 2 - H));
253.                                         message = box[0];
254.                                 } else {
255.                                         message.AddVector(Z.copy(H, Z.size() / 2));
256.                                 }
257.                         }
258.
259.                         box[0] = null;
260.                         // 3. Прийняти B, MKH, MRH, α від задачі справа.
261.                         MPI.COMM_WORLD.Recv(box, 0, 1, MPI.OBJECT, leftRank,
262.                                     KEY.SEND_RECIVE_B_MK_MR_alfa);
263.
264.                         B = box[0].getVector(0);
265.                         MK = box[0].getMatrix(0);
266.                         MR = box[0].getMatrix(1);
267.                         alfa = box[0].getValue(0);
268.
269.                         if (rank == P - 1 && notTwoCore) {
270.                                 // 4. Якщо rank = P-1, то прийняти MO, Z(k*H), E від задачі
```

```
271.                                // зверху.
272.                                MPI.COMM_WORLD.Recv(box, 0, 1, MPI.OBJECT, k - 1,
273.                                              KEY.SEND_RECIVE_MO_Z_E_TO_P_1);
274.                                MO = box[0].getMatrix(0);
275.                                E = box[0].getVector(0);
276.                                Z = box[0].getVector(1);
277.                                box[0].setVector(1, Z.copy(H, Z.size() - H));
278.                                message = box[0];
279.                         }
280.
281.                         box[0] = message;
282.                         // 5. Передати MO, Z(k*H)-H, E задачі зліва.
283.                         MPI.COMM_WORLD.Isend(box, 0, 1, MPI.OBJECT, leftRank,
284.                                      KEY.SEND_RECIVE_Z_E_MO);
285.
286.                         Z = Z.copy(0, H);
287.                         calcMinButtonTask();
288.                         // 14. Обчислити MAH = B(MO MKH)α + minZ·E·MRH
289.                         A = CalculateUtils.add(CalculateUtils.mult(alfa,
290.                                      CalculateUtils.mult(B, CalculateUtils.mult(MK, MO))),
291.                                      CalculateUtils.mult(localMin, CalculateUtils.mult(E,
      MR)));
292.
293.                         box[0] = null;
294.                         box[0] = new MessageBox();
295.                         box[0].AddVector(A);
296.                         // 15. Передати MAH*k-H від задачі зліва
297.                         MPI.COMM_WORLD.Isend(box, 0, 1, MPI.OBJECT, leftRank,
298.                                      KEY.SEND_RECIVE_RESULT_A);
299.                 }
300.          }
301.          private void calcMinButtonTask() {
302.
303.                 if (rank > endFirsRowRank) {
304.                         // нижній рядок
305.                         int min[] = new int[] { CalculateUtils.min(Z) };
306.                         // передати localMinZ задачі зверху
307.                         MPI.COMM_WORLD.Send(min, 0, 1, MPI.INT, rank - k, 103);
308.                         min = new int[1];
309.                         MPI.COMM_WORLD.Recv(min, 0, 1, MPI.INT, rank - k, 130);
310.                         localMin = min[0];
311.                 } else {
312.
313.                         int min[] = new int[1];
314.                         if (notTwoCore) {
315.                                // прийняти minZ від задачі знизу
316.                                MPI.COMM_WORLD.Recv(min, 0, 1, MPI.INT, rank + k, 103);
317.                         } else {
318.                                min[0] = Integer.MAX_VALUE;
319.                         }
320.                         // Обчислити localMinZ = min(ZH)
321.                         // обчислити localMinZ = min(minZ, localMinZ)
322.
```

```
323.                         localMin = Math.min(min[0], CalculateUtils.min(Z));
324.                         if (rank > minRank) {
325.                                 int[] res = new int[1];
326.                                 int buf = localMin;
327.                                 if (rank != endFirsRowRank) {
328.                                         // отримати minZ від задачі з справа
329.                                         MPI.COMM_WORLD.Recv(res, 0, 1, MPI.INT, rank + 1,
      104);
330.                                         // minZ = (localMinZ, minZ)
331.                                         buf = Math.min(localMin, res[0]);
332.                                 }
333.                                 res[0] = buf;
334.                                 // передати localMinZ задачі зліва
335.                                 MPI.COMM_WORLD.Send(res, 0, 1, MPI.INT, rank - 1, 104);
336.                         } else {
337.                                 if (rank != minRank) {
338.
339.                                         int[] res = new int[1];
340.                                         res[0] = localMin;
341.                                         if (rank != 0) {
342.                                                 // отримати minZ від задачі зліва
343.                                                 MPI.COMM_WORLD.Recv(res, 0, 1, MPI.INT, rank -
      1, 104);
344.                                                 localMin = Math.min(localMin, res[0]);
345.                                                 res[0] = localMin;
346.                                         }
347.                                         // передати localMinZ задачі з права
348.                                         MPI.COMM_WORLD.Send(res, 0, 1, MPI.INT, rank + 1,
      104);
349.                                 }
350.                         }
351.                         if (rank == minRank) {
352.                                 int res[] = new int[1];
353.                                 if (notFourCore && notTwoCore) {
354.                                         // 14. Якщо rank = (k-1)/2, то Отримати minZright від
      задачі
355.                                         // справа
356.                                         MPI.COMM_WORLD.Recv(res, 0, 1, MPI.INT, rank + 1,
      104);
357.                                         // 15. Якщо rank = (k-1)/2, то обчислити minZ =
      (minZ,
358.                                         // minZleft, minZright)
359.                                         this.localMin = Math.min(this.localMin, res[0]);
360.                                         res = new int[1];
361.                                 }
362.                                 MPI.COMM_WORLD.Recv(res, 0, 1, MPI.INT, rank - 1, 104);
363.
364.                                 this.localMin = Math.min(this.localMin, res[0]);
365.                                 // знайшли мінімум
366.                                 res[0] = this.localMin;
367.                                 // 16. Якщо rank = (k-1)/2, то передати minZ задачі зліва
368.                                 MPI.COMM_WORLD.Send(res, 0, 1, MPI.INT, rank - 1, 122);
369.                                 if (notFourCore && notTwoCore)
```

```
370.                                    // 16. Якщо rank = (k-1)/2, то передати minZ задачі
      справа
371.                                    MPI.COMM_WORLD.Send(res, 0, 1, MPI.INT, rank + 1,
      122);
372.                        } else if (rank > 0 && rank < minRank) {
373.                                min = new int[1];
374.                                // 16. Якщо rank = (k-1)/2, то передати minZ задачі справа
375.                                MPI.COMM_WORLD.Recv(min, 0, 1, MPI.INT, rank + 1, 122);
376.                                this.localMin = min[0];
377.                                // 16. Якщо rank = (k-1)/2, то передати minZ задачі зліва
378.                                MPI.COMM_WORLD.Send(min, 0, 1, MPI.INT, rank - 1, 122);
379.                        } else if (rank != 0 && rank != endFirsRowRank) {
380.                                min = new int[1];
381.                                // 16. Якщо rank = (k-1)/2, то передати minZ задачі зліва
382.                                MPI.COMM_WORLD.Recv(min, 0, 1, MPI.INT, rank - 1, 122);
383.                                this.localMin = min[0];
384.                                // 16. Якщо rank = (k-1)/2, то передати minZ задачі справа
385.                                MPI.COMM_WORLD.Send(min, 0, 1, MPI.INT, rank + 1, 122);
386.                        } else if (rank == 0) {
387.                                min = new int[1];
388.                                // 16. Якщо rank = (k-1)/2, то передати minZ задачі справа
389.                                MPI.COMM_WORLD.Recv(min, 0, 1, MPI.INT, rank + 1, 122);
390.                                this.localMin = min[0];
391.                        } else {
392.                                min = new int[1];
393.                                // 16. Якщо rank = (k-1)/2, то передати minZ задачі зліва
394.                                MPI.COMM_WORLD.Recv(min, 0, 1, MPI.INT, rank - 1, 122);
395.                                localMin = min[0];
396.                        }
397.                        if (notTwoCore) {
398.                                min[0] = localMin;
399.                                // 19. Якщо rank >k, то передати minZ від задачі знизу
400.                                MPI.COMM_WORLD.Send(min, 0, 1, MPI.INT, rank + k, 130);
401.                        }
402.                }
403.        }
404.  }
```

Файл Vector.java:

```
1.    import java.io.Serializable;
2.
3.    /**
4.    ****************************************************************
5.    *                                                              *
6.    *            Програмування паралельний комп'ютерних систем     *
7.    *             Курсова робота. ПРГ2. Бібліотека MPI             *
8.    *                                                              *
9.    * Завдання: A = B(MO*MK)*a + min(Z)*E*MR                       *
10.   *                                                              *
11.   * Файл: Vector.java                                           *
12.   * Автор: Кузьменко Володимир                                  *
13.   * Група: ІО-21                                               *
14.   * Дата: 23.04.15                                             *
```

```
15.   *                                                                 *
16.   *******************************************************************
17.   */
18.   public class Vector implements Serializable {
19.          private static final long serialVersionUID = 1L;
20.          private int[] array;
21.
22.          public Vector(int n) {
23.                 array = new int[n];
24.          }
25.          public void set(int index, int value) {
26.                 array[index] = value;
27.          }
28.          public int get(int index) {
29.                 return array[index];
30.          }
31.          public int size() {
32.                 return array.length;
33.          }
34.          public String toString() {
35.                 String res = "";
36.                 for (int i = 0; i < array.length; i++) {
37.                        res += "   " + array[i];
38.                 }
39.                 res += "\n";
40.                 return res;
41.          }
42.          public Vector copy(int start, int size) {
43.                 Vector result = new Vector(size);
44.                 for (int i = 0; i < result.size(); i++) {
45.                        result.array[i] = this.array[i + start];
46.                 }
47.                 return result;
48.          }
49.          public void merge(Vector vector) {
50.                 int[] buf = array;
51.                 array = new int[buf.length + vector.size()];
52.                 System.arraycopy(buf, 0, array, 0, buf.length);
53.                 System.arraycopy(vector.array, 0, this.array, buf.length,
54.                               vector.array.length);
55.          }
56.          public void reverse() {
57.                 for (int i = 0; i < array.length / 2; i++) {
58.                        int buf = array[i];
59.                        array[i] = array[i + array.length / 2];
60.                        array[array.length / 2 + i] = buf;
61.                 }
62.          }
63.   }
```