Національний технічний університет України  
«Київський політехнічний інститут»

Кафедра обчислювальної техніки

(повна назва кафедри, циклової комісії)

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)

Студента 3 курсу групи ІО-01

напряму підготовки 050102

«Комп’ютерна інженерія»

Редько О.М.

(прізвище та ініціали)

Керівник доцент Корочкін О.В.

Національна оцінка

Кількість балів:

Оцінка: ECTS

**Курсова робота**

з дисципліни «Паралельні та розподілені обчислення»

(назва дисципліни)

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

Київ – 2013 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)

**Завдання**

на курсову роботу студенту

Редько Олександр Михайлович

(прізвище, ім’я, по батькові)

1. Тема роботи «Розробка програмного забезпечення для паралельних комп’ютерних систем».
2. Керівник роботи к. т. н., доцент Корочкін Олександр Володимирович.

(вчене звання, науковий ступінь, прізвище та ініціали)

1. Строк подання студентом роботи 14 травня 2013 р.
2. Вихідні дані для роботи:

* огляд восьмиядерних процесорів компанії Intel;
* математична задача ;
* структури паралельної комп’ютерної системи з спільною пам’яттю (ПКС СП) та паралельної комп’ютерної системи з локальною пам’яттю (ПКС ЛП);
* бібліотеки програмування: Win32, MPI.

1. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

* огляд восьмиядерних процесорів компанії Intel;
* розробка і тестування програми ПРГ1 для ПКС СП;
* розробка і тестування програми ПРГ2 для ПКС ЛП.



Рисунок . – Структура ПКС СП



Рисунок . – Структура ПКС ЛП

1. Перелік графічного матеріалу:

* структурна схема ПКС СП;
* структурна схема ПКС ЛП;
* схеми алгоритмів процесів і головної програми для ПРГ1;
* схеми алгоритмів процесів і головної програми для ПРГ2.

1. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |
| --- | --- | --- |
| № з/п | Назва етапів виконання КР | Строк виконання етапів КР |
|  | Виконання огляду для розділу 1 | 20.03.2013 |
|  | Розробка паралельного алгоритму рішення задачі | 01.04.2013 |
|  | Розробка алгоритмів процесів | 06.04.2013 |
|  | Розробка схем взаємодії процесів | 13.04.2013 |
|  | Розробка програм | 20.04.2013 |
|  | Тестування програм | 30.04.2013 |
|  | Оформлення КР | 10.05.2013 |
|  | Захист КР | 18.05.2013 |

Студент

(підпис) (прізвище та ініціали)

Керівник роботи

(підпис) (прізвище та ініціали)

**ПОЯСНЮВАЛЬНА  
ЗАПИСКА**

ЗМІСТ

[ВСТУП 6](#_Toc355801254)

[РОЗДІЛ 1. ОГЛЯД ВОСЬМИЯДЕРНИХ ПРОЦЕСОРІВ INTEL 7](#_Toc355801255)

[1.1 Багатоядерність як сучасний спосіб збільшення продуктивності обчислювальної системи 7](#_Toc355801256)

[1.2 Загальна характеристика багатоядерних процесорів Intel 8](#_Toc355801257)

[1.3 Восьмиядерні процесори Intel 9](#_Toc355801258)

[1.4 Загальна характеристика процесорів Itanium 9](#_Toc355801259)

[1.4.1 Огляд процесорів Intel Itanium 9500 Series [16] 10](#_Toc355801260)

[1.5 Загальна характеристика серверних процесорів Xeon 13](#_Toc355801261)

[1.5.1 Огляд мікропроцесорів Intel Xeon E7‑8800/4800/2800 [20] 15](#_Toc355801262)

[1.5.2 Огляд мікропроцесорів Intel Xeon E5‑4600/2600 [23] 16](#_Toc355801263)

[1.6 Висновки до розділу 1 18](#_Toc355801264)

[РОЗДІЛ 2. РОЗРОБКА ПРОГРАМИ ПРГ1 ДЛЯ ПКС СП 19](#_Toc355801265)

[2.1 Розробка паралельного математичного алгоритму 19](#_Toc355801266)

[2.2 Розробка алгоритмів процесів 20](#_Toc355801267)

[2.3 Розробка схеми взаємодії процесів 21](#_Toc355801268)

[2.4 Розробка програми ПРГ1 25](#_Toc355801269)

[2.5 Тестування програми ПРГ1 26](#_Toc355801270)

[2.6 Висновки до розділу 2 31](#_Toc355801271)

[РОЗДІЛ 3. РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ПКС ЛП 32](#_Toc355801272)

[3.1 Розробка алгоритмів процесів 32](#_Toc355801273)

[3.2 Розробка схеми взаємодії процесів 35](#_Toc355801274)

[3.3 Розробка програми ПРГ2 36](#_Toc355801275)

[3.4 Тестування програми ПРГ2 37](#_Toc355801276)

[3.5 Висновки до розділу 3 41](#_Toc355801277)

[ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ ПО РОБОТІ 42](#_Toc355801278)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 43](#_Toc355801279)

[ДОДАТКИ 46](#_Toc355801280)

[Додаток А. Структурна схема ПКС СП 47](#_Toc355801281)

[Додаток Б. Схема алгоритму головної програми зі вказанням паралельних ділянок для ПРГ1 48](#_Toc355801282)

[Додаток В. Схема алгоритму процесів для програми ПРГ1 49](#_Toc355801283)

[Додаток Г. Лістинг програми ПРГ1 53](#_Toc355801284)

[Додаток Д. Код скрипта для тестування ПРГ1 60](#_Toc355801285)

[Додаток Е. Структурна схема ПКС ЛП 51](#_Toc355801286)

[Додаток Ж. Схема алгоритму головної програми зі вказанням паралельних ділянок для ПРГ2 52](#_Toc355801287)

[Додаток З. Схема алгоритму процесів для програми ПРГ2 53](#_Toc355801288)

[Додаток К. Лістинг програми ПРГ2 69](#_Toc355801289)

[Додаток Л. Код скрипта для тестування ПРГ2 79](#_Toc355801290)

ВСТУП

Курсова робота по дисципліні «Паралельні і розподілені обчислення» складається з трьох розділів.

В першому розділі «Огляд восьмиядерних процесорів компанії Intel» описані передумови виникнення багатоядерності, загальна характеристика процесорів Intel та основні обчислювальні якості серверних процесорів Intel лінійок Itanium та Xeon.

Другий та третій розділи присвячені розробці програми для обчислення математичної задачі в паралельній комп’ютерній системі зі спільною та локальною пам’яттю відповідно. Програмне забезпечення для комп’ютерної системи зі спільною пам’яттю розроблено на мові C++ з використанням бібліотеки Win32, а для комп’ютерної системи з локальною пам’яттю – із застосуванням бібліотеки MPI. Проведено тестування отриманих програмних продуктів і зроблено висновки по їх ефективності.

Лістинги та алгоритми розроблених програм наведено у додатках.

# ОГЛЯД ВОСЬМИЯДЕРНИХ ПРОЦЕСОРІВ INTEL

## Багатоядерність як сучасний спосіб збільшення продуктивності обчислювальної системи

У 1965 році однин із засновників компанії Intel Гордон Мур висловив припущення, що кількість транзисторів на кристалі мікросхеми буде подвоюватись кожні 24 місяці [1, 2]. Пізніше Дейв Хаус доповнив, що при цьому продуктивність комп’ютера зростатиме в два рази кожні 18 місяців [3]. Закон Мура продовжує виконуватись [4]; наприклад, в 2010 році компанія Intel представила перший процесор з 2 мільярдами транзисторів “Tukwila” [5]. Одночасно зі збільшенням кількості транзисторів зростала частота роботи процесора. Таким чином нарощували продуктивність процесора. Проте при збільшенні тактової частоти зростає енергоспоживання процесора, що в свою чергу приводить до підвищення тепловиділення [6]. У вересні 2005 року в журналі IEE Review було надруковано, що «з підвищенням тактової частоти на 400 МГц споживання електроенергії процесором збільшується на 60%» [7]. Необмежене зростання продуктивності за рахунок збільшення частоти неможливе.

Слова Дейва Хауса все ще мають смисл. Для підвищення продуктивності одноядерних процесорів були винайдені такі методики, як модуль передбачення умовних переходів, кеш, конвеєр команд [8].

Хоча головним проривом у збільшенні продуктивності процесора без підвищення його тактової частоти стало розміщення кількох функціонально незалежних обчислювальних ядер разом з комунікаційним середовищем у спільному корпусі [9]. Такі процесори називаються багатоядерними.

Першим багатоядерним процесором в світі став розроблений компанією IBM POWER4 Processor [10]. Випущений у 2001 році, цей серверний процесор мав два ядра, тактову частоту в 1.1 ГГц і суперскалярну 64‑бітну архітектуру. Для архітектури x86 перші двоядерні процесори були представлені світу у 2005 році компаніями Intel та Advanced Micro Devices [11]. Деякі експерти говорять, що «до 2017 року вбудовані процесори матимуть 4096 ядер, серверні ЦП – 512 ядер, а в кристалі для мікропроцесорів для настільних комп’ютерів буде 128 ядер» [12].

Багатоядерні процесори виправляють недоліки одноядерних. Таблиця 1.1 показує відмінності між одно і восьмиядерним процесором, що використовується Технологічним Дослідницьким Центром у Технологічному Інституті Джорджії. Багатоядерний процесор працює на меншій тактовій частоті, ніж одноядерний, при цьому пропускна здатність більша в 10 разів, а споживання електроенергії нижче в 4 рази [13].

Таблиця . – Одне ядро проти багатьох

|  |  |  |
| --- | --- | --- |
|  | Одноядерний процесор (45 нм) | Багатоядерний процесор (45 нм) |
| Напруга електроживлення | 1.0 В | 1.0 В |
| Вхідні / вихідні піни | 1280 | 3000 |
| Робоча частота | 7.8 ГГц | 4 ГГц |
| Швидкість передачі даних у кристалі | 7.8 Гб/с | 4 Гб/с |
| Пропускна здатність | 125 ГБ/c | 1 ТБ/c |
| Електроспоживання | 429.78 Вт | 107.39 Вт |
| Загальна кількість пінів у кристалі | 3840 | 9000 |

## Загальна характеристика багатоядерних процесорів Intel

Компанія Intel ділить багатоядерні процесори на такі типи [14]:

* Незалежні обчислювальні ядра індивідуальною кеш-пам’яттю, розміщені на одному кристалі та під’єднані до системної шини. Процесор
* Незалежні обчислювальні ядра розміщені на різних кристалах.
* Обчислювальні ядра, що використовують спільні вузли, наприклад, кеш-пам’ять або системну шину.

## Восьмиядерні процесори Intel

Станом на початок 2013 року компанія Intel випускає близько 30 найменувань процесорів з вісьмома ядрами [15]. Це виключно серверні процесори серій Intel Itanium Processor 9500 Series, Intel Xeon E7‑8800/4800/2800, Intel Xeon E5‑2600, Intel Xeon X6000, Intel Xeon L7555. Мікропроцесори орієнтовані на використання в системах, що забезпечують швидку і точну обробку великої кількості інформації: аналітичні обчислення для бізнес моделей, робота з великими базами даних, розрахунок масиву астрономічних даних тощо.

## Загальна характеристика процесорів Itanium

Мікропроцесор з архітектурою IA-64 Itanium був спільно розроблений компаніями Intel та Hewlett-Packard. Вперше був представлений 29 травня 2001 року.

Itanium був спеціально розроблений для досягнення дуже високого рівня продуктивності в паралельних обчисленнях без збільшення частот. Ключові переваги архітектури Itanium:

* Виконання до 6 інструкцій за 1 цикл.
* Виконання двох SIMD операцій з плаваючою комою із 82‑бітними операндами за один цикл.
* Обчислювальні ресурси ядра із 256 регістрами і 64 предикативними регістрами.
* Великий кеш: 24 МБ у двоядерній версії, що надає дані кожному ядру зі швидкістю до 48 ГБ/с.
* За рахунок розміщення в кеш-пам’яті великих обсягів даних (задачі складного аналізу, Data Mining) дозволяє досягти приросту продуктивності при обробці.
* Великий адресний простір: 50‑бітна адресація фізичної пам’яті / 64‑бітна адресація віртуальної пам’яті.
* Маленьке, енергоефективне ядро за рахунок того, що функції розпаралелювання передали від Itanium до компілятора.
* Добре підходить для створення обчислювальних систем, що містять від 4 до 32 ЦП на сервері.

### Огляд процесорів Intel Itanium 9500 Series [16]

#### Загальна характеристика Intel Itanium Processor 9500 Series

У лінійці процесорів Intel Itanium Processor 9500 Series представлено чотири процесори, з них два – Itanium Processor 9560 та Itanium Processor 9540 – мають вісім ядер і відрізняються лише тактовою частотою та об’ємом кеш-пам’яті. Тому надалі приводиться характеристика всього модельного ряду Itanium 9500, а всі відмінності наведені в таблиці Таблиця 1.2.

Процесори Intel Itanium 9500 містять 3.1 мільярда транзисторів і підтримують мікропроцесорну архітектуру з явним паралелізмом команд (EPIC). У такому дизайні інтерфейс між апаратним і програмним забезпеченням спроектований для того, щоб використовувати весь наявний час компіляції та оптимально розподіляти інформацію між апаратними ресурсами. Архітектура EPIC дозволяє мікропроцесору виконувати інструкції паралельно спираючись тільки на роботу компілятора, а не виявляти можливість паралельної роботи інструкцій за допомогою спеціальних схем.

Для з’єднання з іншими системними компонентами Intel Itanium 9500 має послідовну кеш-когерентну шину типу точка-точка – Intel QuickPath Interconnect (QPI), тому процесор може використовуватись для побудови дуже великих обчислювальних систем. Збалансовані підсистеми ядер і пам’яті забезпечують велику продуктивність для широкого спектру задач: від комерційних розрахунків до складних науково-технічних обчислень.

На сьогодні процесори сімейства Itanium 9500 в основному використовуються тільки у серверах HP Integrity Servers [17].

#### Особливості процесорів Intel Itanium Processor 9500 Series

Процесори серії Intel Itanium 9500 оснащені 16‑32 МБ LLC кеш-пам’яттю, а також такими розробками компанії Intel, як Hyper-Threading Technology для підвищеної продуктивності, Virtualization Technology для удосконаленої віртуалізації, Cache Safe Technology для посиленої доступності, Turbo Boost Technology з підтримкою безперервного розгону. У Intel Itanium Processor 9500 використовується сучасний моніторинг потужності, яку споживає процесор, і система контролю за тим, щоб процесор працював весь час на найвищій тактовій частоті. Крім цього, процесор підтримує наскрізний паритетний захист із можливістю відновлення на всіх критичних шинах і каналах передачі даних. Блок обчислень з плаваючою точкою має радіаційно-стійкий захист.

Процесори Intel Itanium Processor 9500 Series складаються з восьми ядер і блоку системного інтерфейсу. Кожне процесорне ядро має 11-тирівневий конвеєр виконання команд. Обчислювальні ресурси складаються з шести цілочислових блоків, одного пристрою для множення цілих чисел, чотирьох блоків для мультимедіа, двох блоків завантаження / зберігання, трьох блоків розгалуження, двох пристроїв для обчислення операцій з плаваючою точкою для подвійної точності. Ядро забезпечує динамічну передвиборку команд, передбачення розгалужень, таблицю регістрів і неблокуючу кеш-пам’ять для оптимізації.

У процесорів Intel Itanium Processor 9500 Series присутні нові команди, які полегшують виконання типових задач: clz (порахувати кількість нулів перед першою значущою цифрою числа), mpy4 і mpyshl4 (беззнакове цілочислове множення і множення / зсув), mov‑to‑DAHR/mv‑from‑DAHR (для покращеної MLD/FLD передвиборки команд), hint@priority (використовується процесором для тимчасового розміщення більшої кількості ресурсів у потоці).

Intel Hyper-Threading Technology представлена у процесорах Intel Itanium Processor 9500 Series за допомогою подвійної багатопоточності. Ця технологія надає апаратну підтримку для двох потоків на одне ядро, разом із 96 потоковими елементами у потоковому буфері інструкцій. Три рівні кешу мінімізують загальну затримку пам’яті.

Процесори Intel Itanium Processor 9500 Series підтримують Intel Instruction Replay Technology [18]. Ця технологія дозволяє виявляти і виправляти програмні помилки у конвеєрі команд за 7 тактів, що надто швидко, щоб помітно затримати програму.

#### Архітектура процесорів Intel Itanium Processor 9500 Series

Окреме ядро процесора Intel Itanium Processor 9500 Series має свій власний кеш першого рівня (FLC) і кеш середнього рівня (MLC). Обидва кеші розділяються на кеш інструкцій (FLI, MLI) і кеш даних (FLD, MLD). Кеш останнього рівня (LLC) розділений між всіма ядрами і має 32 МБ пам’яті. Підтримуються наступні розміри сторінок пам’яті для вставки чи видалення: 4K, 8K, 16K, 64K, 256K, 1M, 4M, 16M, 64M, 256M, 1G, 4G.

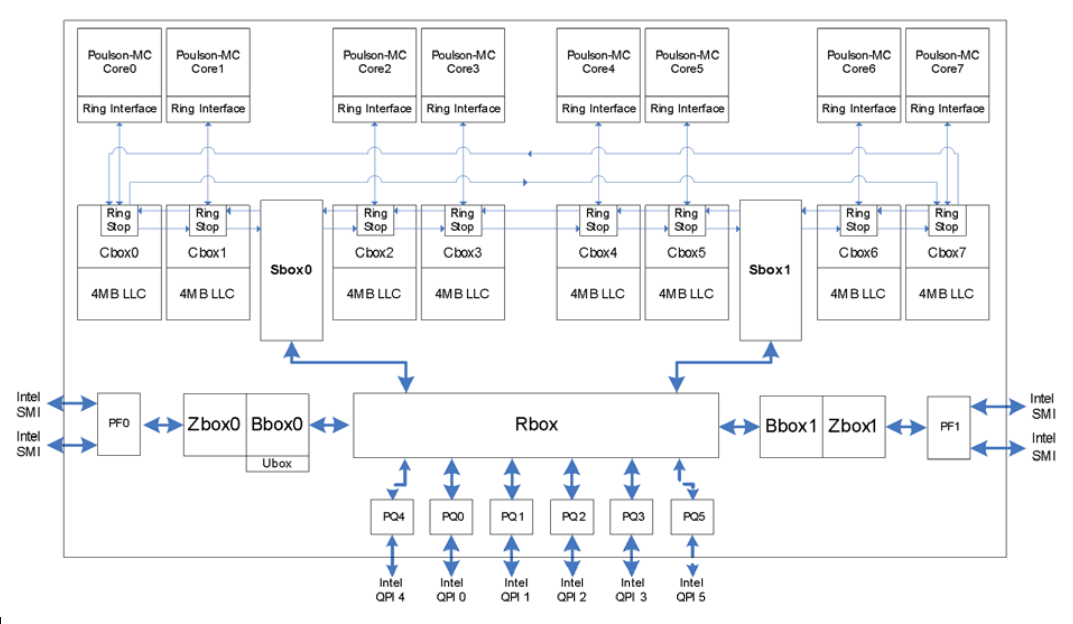


Рисунок . – Структурна схема Intel Itanium Processor 9500

Кожне ядро з’єднане з кільцем (Ring). Кільце надає доступ до LLC через контролери кешу (Cboxes), а також забезпечує з’єднуваність із Intel QPI через Sbox. Cache Agent управляє Cbox і Sbox. Процесор має два Home Agent (Bbox), що виступають як інтерфейси між контролером пам’яті та Intel QPI та підтримують каталог кешу. Кожен контролер пам’яті містить два Intel Scalable Memory Interconnects (SMI), що призначені для доступу до буферу пам’яті, який може розширюватись. Кожен Intel Itanium Processor 9500 підтримує шість Intel QPI у сокеті. Cache Agent, Home Agent і Intel QPI приєднані через 10-портовий Crossbar Router, кожен порт якого підтримує Intel QPI протокол. На рисунку Рисунок 1.1 показана структурна схема процесора.

Поля «вірус» і «отрута» в Intel QPI використовуються для позначення викривленого системного стану і дефектних даних відповідно. Якщо система «стала вірусною», то агент Intel QPI встановить поле «вірус» всередині усіх заголовків пакетів. Вірусний режим може встановитись трьома шляхами: отримання вірусного пакету, виявлення фатальної / критичної помилки, або коли прийнято глобальний вірусний сигнал (від Cboxes). Поле «отрута» встановлюється, коли дані ненадійні.

## Загальна характеристика серверних процесорів Xeon

Мікропроцесори Xeon були представлені світу у 1998 році [19]. Перші ЦП Xeon були одноядерними, але з часом кількість ядер зросла до десяти. Архітектура змінювалась таким чином: P6, NetBurst, Core, Nehalem, Westmere, Sandy Bridge. Серверні Xeon призначені для створення потужних суперкомп’ютерів і високоефективних центрів обробки даних.

Таблиця . – Порівняльна характеристика восьмиядерних процесорів виробництва компанії Intel [15]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва продукту | Intel Itanium Processor 9560 (32M Cache, 2.53 GHz) | Intel Itanium Processor 9540 (24M Cache, 2.13 GHz) | Intel Xeon Processor E7-8837 (24M Cache, 2.66 GHz, 6.40 GT/s Intel QPI) | Intel Xeon Processor E7-2830 (24M Cache, 2.13 GHz, 6.40 GT/s Intel QPI) | Intel Xeon Processor E7-4820 (18M Cache, 2.00 GHz, 5.86 GT/s Intel QPI) | Intel Xeon Processor E5-4650L (20M Cache, 2.60 GHz, 8.00 GT/s Intel QPI) | Intel Xeon Processor E5-2470 (20M Cache, 2.30 GHz, 8.00 GT/s Intel QPI) |
| Кодова назва | Poulson | Poulson | Westmere EX | Westmere EX | Westmere EX | Sandy Bridge-EP | Sandy Bridge-EP |
| Дата запуску на ринок | IV кв ’12 | IV кв ’12 | II кв ‘11 | II кв ‘11 | II кв ‘11 | II кв ’12 | II кв ’12 |
| Тактова частота | 2.53 Гц | 2.13 Гц | 2.66 Гц | 2.13 Гц | 2.00 Гц | 2.6 Гц | 2.3 Гц |
| Максимальна частота розгону |  |  | 2.8 Гц | 2.4 Гц | 2.266 Гц | 3.1 Гц | 3.1 Гц |
| Кеш-пам'ять | 32 МБ | 24 МБ | 18 МБ «розумного кешу» | 24 МБ «розумного кешу» | 18 МБ «розумного кешу» | 20 МБ «розумного кешу» | 20 МБ «розумного кешу» |
| Системна шина | 6.4 ГТ/с | 6.4 ГТ/с | 6.4 ГТ/с | 6.4 ГТ/с | 5.86 ГТ/с | 8 ГТ/с | 8 ГТ/с |
| Літографія | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм |
| Максимальне споживання енергії | 170 Вт | 170 Вт | 130 Вт | 105 Вт | 105 Вт | 115 Вт | 95 Вт |
| Максимальний розмір пам’яті |  |  | 4096 ГБ | 1024 ГБ | 1024 ГБ | 1500 ГБ | 375 ГБ |
| Тип пам’яті |  |  | DDR-3 800/978/1066/1333 | DDR-3 800/978/1066/1333 | DDR-3 800/978/1066/1333 | DDR-3 800/1066/1333/1600 | DDR-3 800/1066/1333/1600 |
| Підтримка пам’яті з корекцією помилок |  |  | Так | Так | Так | Так | Так |
| Максимальна температура корпусу |  |  | 69 °C | 64 °C | 64 °C | 68 °C | 78 °C |
| Ціна | 3880 у.о. | 4650 у.о. | 2280 у.о. | 1779 у.о. | 1446 у.о. | 3616 у.о. | 1440 у.о. |

### Огляд мікропроцесорів Intel Xeon E7‑8800/4800/2800 [20]

Сімейство процесорів Intel Xeon E7‑8800/4800/2800 є другим поколінням чіпів, які підтримують Intel QuickPath Interconnect (QPI) і можуть мати від 4 до 10 ядер. Великий, до 30 МБ, кеш 3‑го рівня доступний всім ядрам. У процесори Intel Xeon E7‑8800/4800/2800 інтегровано два контролери DDR3 пам’яті. Intel Xeon E7‑8800/4800/2800 побудовані за 32 нм технологією і підтримують архітектури x64, IA-32/IA-64. Структурна схема зображена на рис. Рисунок 1.2

ЦП Intel Xeon E7‑8800/4800/2800 базуються на мікроархітектурі Westmere [21] і містять 2.6 мільярда транзисторів. Детальніше про конкретні процесори лінійки подано в таблиці Таблиця 1.2.

Всі моделі підтримують MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, Enhanced Intel SpeedStep Technology (EIST), Intel 64, XD bit, TXT, Intel VT-x, Intel EPT, Intel VT-d, Intel VT-c, Intel DDDC, Hyper-threading, Turbo Boost, AES-NI, Smart Cache.

Варто зазначити, що процесори встановили 15 світових рекордів по продуктивності у 2011 році [22].

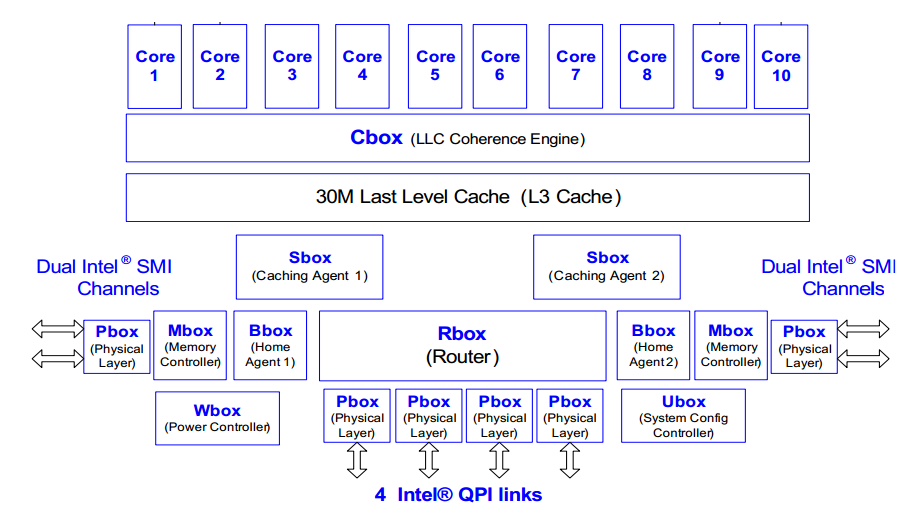


Рисунок . – Структурна схема процесорів Intel Xeon E7‑8800/4800/2800

### Огляд мікропроцесорів Intel Xeon E5‑4600/2600 [23]

Intel Xeon E5 – сімейство адаптивних процесорів для гнучких і ефективних обчислювальних центрів, що мають 64‑бітну архітектуру Sandy Bridge‑E [24]. Базуючись на 2‑му поколінні енергоефективній мікроархітектурі Intel Core Processor Family, процесор створений для двочіпової платформи, що складається з процесора і хаба контролера платформи (PCH). Сімейство Intel Xeon E5‑2600 спроектовані для ефективних і потужних сервера, робочої станції та HPC платформ. Сімейство Intel Xeon E5‑4600 підтримує сервери, що можуть масштабуватись і платформи HPC, які складаються з двох або більше процесорів.

У цих процесорів присутній міжпроцесорний канал високошвидкісної передачі даних до 8.0 ТБ/с Intel QPI. Intel Xeon E5‑4600/2600 підтримують 46‑бітну адресацію фізичної пам’яті та 58‑бітну віртуальну адресацію.

Сімейство процесорів підтримує технологію Intel Integrated I/O, що зменшує затримки в мережі та в пристроях зберігання даних у віртуальних системах. Для управління живленням передбачені технології Intel Power Tuning та Intel Intelligent Power, що допомагають оптимізувати ефективність і зменшити витрати шляхом розподілу і динамічного вирівнювання навантаження на рівні серверної стійки.

На рис. Рисунок 1.3 і рис. Рисунок 1.4 показані процесорна двох- і чотирьох- сокетна конфігурація платформи. “Legacy CPU” – це завантажувальний процесор, що з’єднаний до PCH. У 4-сокетній конфігурації “Remote CPU” – процесор, що під’єднаний до Legacy CPU.

Детальні відомості про конкретні процесори модельних рядів Intel Xeon E5‑2600/4600 подано в табл. Таблиця 1.2.

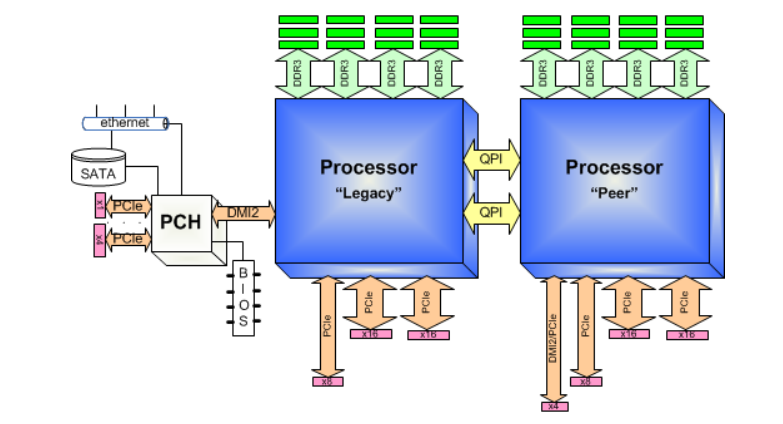


Рисунок . – Сімейство Intel Xeon E5‑2600 на 2-сокетній платформі

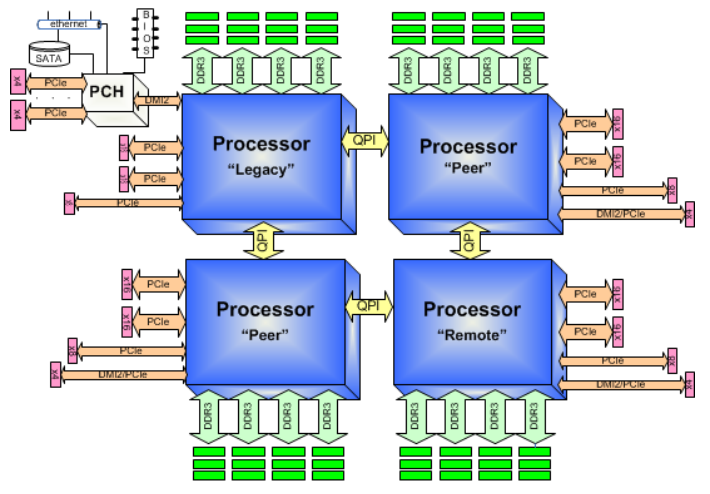


Рисунок . – Сімейство Intel Xeon E5‑4600 на 4-сокетній платформі

## Висновки до розділу 1

* Сучасним способом підвищення продуктивності мікропроцесора є збільшення кількості ядер в одному корпусі.
* Компанія Intel випускає восьмиядерні процесори тільки для серверів.
* Восьмиядерні процесори модельних рядів Intel Itanium та Intel Xeon орієнтовані на використання в системах, що забезпечують швидку і точну обробку великої кількості інформації: аналітичні обчислення для бізнес моделей, робота з великими базами даних, розрахунок масиву астрономічних даних тощо
* Мікропроцесори серії Intel Itanium 9500 хоч і мають достатній обчислювальний потенціал, але дуже дорогі в порівнянні з процесорами модельного ряду Intel Xeon E7‑4800/2800.
* Процесори Intel Itanium в сучасних системах реально використовуються тільки в HP Integrity Servers.
* Лінійка процесорів Intel Xeon E5 забезпечує серед усіх розглянутих модельних рядів найкраще поєднання продуктивності, вбудованих функцій та рентабельності, роблячи обчислювальні центри більш гнучкими та ефективними.

# РОЗРОБКА ПРОГРАМИ ПРГ1 ДЛЯ ПКС СП

## Розробка паралельного математичного алгоритму

Паралельний математичний алгоритм відповідно до рекомендованої методики [25] можна подати у вигляді наступних двох етапів:

1. ;

.

1. ,

де:

* ;
*  –  рядків матриці ;
*  –  рядків матриці ;
*  –  рядків матриці ;
*  –  рядків матриці ;

Спільні ресурси: .

Проаналізуємо розроблений паралельний математичний алгоритм із застосуванням концепції необмеженого паралелізму.

Для оцінки необхідного часу обчислень використаємо теорему Мунро-Петерсона, яка для компютерної системи з необмеженим числом процесорів формулюється так: якщо виконується обчислення скалярної величини, яке потребує ** бінарних операцій, то необхідний час обчислень :

.

Для обчислення  необхідно виконати  бінарних операцій порівняння. Тому час виконання буде:

.

Для операції  необхідно виконати  множення та  додавань на кожен елемент, що обчислюється. Тому час виконання буде:

.

Сумарний час виконання всіх двох етапів обчислень буде виражатись наступною формулою:

.

## Розробка алгоритмів процесів

Оскільки розроблюване програмне забезпечення є масштабованим і працює на системі із кількістю процесорів , то зручним варіантом реалізації є написання єдиного алгоритму для всіх задач.

|  |  |
| --- | --- |
| Задачі T(0) – T(P-1) | |
| Крок алгоритму | КД і ТС |
| Якщо tid = 1, ввести . |  |
| Якщо tid = 1, **сигнал** всім іншим задачам про завершення вводу . |  |
| Якщо tid = 2, ввести . |  |
| Якщо tid =2, **сигнал** всім іншим задачам про завершення вводу. |  |
| Якщо tid = 5, ввести . |  |
| Якщо tid = 5, **сигнал** всім іншим задачам про завершення вводу. |  |
| Якщо tid != 1 **чекати** на завершення вводу . |  |
| Якщо tid != 2 **чекати** на завершення вводу . |  |
| Якщо tid != 5 **чекати** на завершення вводу . |  |
| Обчислення . |  |
| Обчислення . | КД |
| **Сигнал** всім задачам про завершення обчислення . |  |
| **Чекати** на завершення обчислення  від всіх задач. |  |
| Копіювати , , , . | КД |
| Обчислення , . |  |
| Якщо tid != 1, **сигнал** задачі T(1) про завершення обчислення . |  |
| Якщо tid = 1, **чекати** на завершення обчислення  в усіх інших задачам. |  |
| Якщо tid = 1, вивести результат . |  |

## Розробка схеми взаємодії процесів

На основі алгоритму для всіх задач, наведеному в попередньому розділі, було розроблена структурна схема взаємодії задач (рис. Рисунок 2.1‑Рисунок 2.2). Схема розділена на два рисунки для зручного сприйняття. Вона дозволяє наочно контролювати зв’язки належних точок синхронізації (сигналів S та очікувань W). Графічне зображення взаємодії задач дозволяє виявити тупикові ситуації в програмі у випадку, коли точка синхронізації W не буде повязана з належною точкою синхронізації S. Крім того, на структурній схемі уводяться також семафори і події, що будуть використовуватись в програмі.

Для демонстрації взаємодій між задачами вибрано чотири задачі: T(1), T(2), T(5) та T(*i*) . Задачі T(1), T(2), T(5) вводять дані, тому з ними взаємодіють всі інші (синхронізація по вводу). Задачі T(*i*)  виконують лише обчислення і синхронізуються зі всіма задачами по обчисленню.

На структурній схемі взаємодії задач уведено такі засоби для вирішення задач взаємного виключення та синхронізації:

* масив подій evnInput[3] для синхронізації із завершенням вводу в T(1), T(2), T(5);
* масив подій evnMax[P] для синхронізації всіх задач із завершенням обчислення  в усіх задачах;
* критична секція csMaxVal для вирішення задачі взаємного виключення під час обчислення ;
* критична секція csResources для керування доступом до спільних ресурсів ;
* множинний семафор semCalcEnd для синхронізації задачі T(1) із завершенням всіх обчислень в інших задачах.



Рисунок . – Структурна схема взаємодії задач для ПРГ1 під час кроків алгоритму 1‑9



Рисунок . – Структурна схема взаємодії задач для ПРГ1 під час кроків алгоритму 10‑18

## Розробка програми ПРГ1

Програма написана на мові C++ з використанням бібліотеки Win32 та складається з трьох модулів: основного prg1.cpp та допоміжних matrix.cpp і operations.cpp.

Основний модуль prg1.cpp містить дві функції:

* main – точка входу в програму, формує ідентифікатори tid задач, запускає задачі, а також вимірює час виконання elapsedTime програми ПРГ1;
* threadProcedure – підпрограма, що запускається у кожній із задач. Ця функція реалізує алгоритм, описаний в розділі 2.2.

В основному модулі об’явлені такі змінні: N – установка розмірності матриць, P – налаштування кількості процесорів, fileName – ім’я текстового файлу, в якому будуть зберігатися дані про час виконання ПРГ1, affinityMask – маска для установки, на яких процесорах буде виконуватись програма. Ініціалізація цих змінних виконується при запуску користувачем програми через параметри командного рядка.

Допоміжний модуль matrix.cpp представляє матрицю і реалізований у вигляді класу. Клас Matrix містить конструктор для створення матриць з rows рядками та cols стовпчиками, конструктор для створення копії вже існуючого об’єкту матриці, функції для доступу get і установки set елементів матриці, а також процедуру fill для заповнення матриці цілим числом value.

Модуль operations.cpp містить три функції:

* maxValue – знаходження максимального елементу матриці;
* calculation – процедура обчислення матриці ;
* getCores – за заданою маскою affinityMask обчислює, на скількох ядрах буде виконуватись ПРГ1.

Лістинг програми ПРГ1 наведено у додатку Додаток Г. .

## Тестування програми ПРГ1

Для тестування використовувалась паралельна обчислювальна система з наступними апаратними характеристиками:

* процесор: AMD Phenom II X6 1055T Processor(6 ядер по 2.8 Ггц, 6 МБ кешу третього рівня);
* оперативна пам'ять: DDR3 1333 МГц, 3327 МБ.

В якості програмного забезпечення виступали:

* операційна система: Microsoft Windows 7 x86 SP1 (Version 6.7.7601);
* середовище розробки і компіляції C++ програми: Microsoft Visual Studio 2010 (Version 10.0.30319.1 RTMRel).

Для вимірювання часу виконання програми використовувався високоточний таймер з бібліотеки Win32.

Тестування проводилось в автоматичному режимі за допомогою bat‑скрипта, який запускав програму із заданою розмірністю матриці на заданій кількості процесорів. Результати тестування записувались у текстовий файл. Код bat‑файлу приведений у додатку Додаток Д. .

Для оцінки ефективності програми використовуються коефіцієнти прискорення та ефективності.

Коефіцієнт прискорення  показує скорочення часу виконання паралельної програми в паралельній системі з ** процесорами ** в порівнянні з часом виконання послідовної програми в однопроцесорній системі :



Коефіцієнт ефективності  застосування комп’ютерної системи показує ступінь використання ** процесорів системи:



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

Таблиця . – Час виконання програми ПРГ1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N |  |  |  |  |  |  |
| 800 | 2959 | 1483 | 1051 | 875 | 790 | 594 |
| 1600 | 50044 | 25076 | 17104 | 14863 | 14205 | 10166 |
| 2400 | 175628 | 88427 | 60164 | 52645 | 49203 | 34851 |

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

Таблиця . – Коефіцієнти прискорення для програми ПРГ1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Кількість процесорів (P) | | | | | |
| N | 1 | 2 | 3 | 4 | 5 | 6 |
| 800 | 1,00 | 2,00 | 2,82 | 3,38 | 3,75 | 4,98 |
| 1600 | 1,00 | 2,00 | 2,93 | 3,37 | 3,52 | 4,92 |
| 2400 | 1,00 | 1,99 | 2,92 | 3,34 | 3,57 | 5,04 |

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

Таблиця . – Коефіцієнти ефективності для програми ПРГ1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Кількість процесорів (P) | | | | | |
| N | 1 | 2 | 3 | 4 | 5 | 6 |
| 800 | 100,00% | 100,00% | 94,00% | 84,50% | 75,00% | 83,00% |
| 1600 | 100,00% | 100,00% | 97,67% | 84,25% | 70,40% | 82,00% |
| 2400 | 100,00% | 99,50% | 97,33% | 83,50% | 71,40% | 84,00% |

Використовуючи таблиці Таблиця 2.2‑Таблиця 2.3 побудовано графіки зміни коефіцієнтів прискорення і ефективності в залежності від *N* і .



Рисунок . – Графік зміни коефіцієнту прискорення програми ПРГ1 в залежності від кількості ядер при N=800



Рисунок . – Графік зміни коефіцієнту прискорення програми ПРГ1 в залежності від кількості ядер при N=1600



Рисунок . – Графік зміни коефіцієнту прискорення програми ПРГ1 в залежності від кількості ядер при N=2400



Рисунок . – Графік зміни коефіцієнту ефективності програми ПРГ1 в залежності від кількості ядер при N=800



Рисунок . – Графік зміни коефіцієнту ефективності програми ПРГ1 в залежності від кількості ядер при N=1600



Рисунок . – Графік зміни коефіцієнту ефективності програми ПРГ1 в залежності від кількості ядер при N=2400

## Висновки до розділу 2

Виконано розробку програми ПРГ1 для ПКС СП з використанням мови С++ і засобів синхронізації з бібліотеки Win32. Тестування програми показало наступне:

* Використання багатоядерної ПКС та програми ПРГ1 забезпечує скорочення часу обчислення заданої математичної задачі. Значення  лежать в межах від 1,99 до 5,04.
* Максимальне значення  забезпечує ПКС з  та .
* Мінімальне значення  виявлено у ПКС з  та .
* З ростом N для   збільшується, а для  зменшується.
* Значення  змінюються від 70,4% до 100,0%.
* Найефективніше програма ПРГ1 використовує ПКС з . При цьому .
* Зі зростанням  від 1 до 5  лінійно спадає від 100,0% до 70,4% при . Далі при  зростає до 82,0%.

# РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ПКС ЛП

Оскільки математична задача співпадає тою, що розглядалась в роз­ділі 2.1, то перший крок по рекомендованій методиці [25] не матиме відмінностей від уже виконаних. Тому розділ «Розробка паралельного математичного алгоритму» пропущений.

## Розробка алгоритмів процесів

Задача T0

1. **Прийняти**  від задачі T5.
2. **Прийняти**  від задачі T1.
3. **Передати** задачі T1 .
4. **Прийняти**  від задачі T1.
5. Обчислення .
6. **Прийняти**  від задачі T5.
7. Обчислення .
8. **Передати** задачі T1 .
9. **Прийняти**  від задачі T1.
10. **Передати** задачі T5 .
11. Обчислення .
12. **Передати** задачі T5 результат .

Задача T1

1. **Прийняти**  від задачі T4.
2. **Передати** задачі T0 .
3. **Передати** задачі T2 .
4. **Прийняти**  від задачі T0.
5. **Передати** задачі T2 .
6. **Прийняти**  від задачі T2.
7. **Передати** задачі T0 .
8. Обчислення .
9. **Прийняти**  від задачі T4.
10. Обчислення .
11. **Прийняти**  від задачі T0.
12. Обчислення .
13. **Прийняти**  від задачі T2.
14. Обчислення .
15. **Передати** задачі T0 .
16. **Передати** задачі T2 .
17. **Передати** задачі T4 .
18. Обчислення .
19. **Передати** задачі T4 результат .

Задача T2

1. **Прийняти**  від задачі T3.
2. **Прийняти**  від задачі T1.
3. **Прийняти**  від задачі T1.
4. **Передати** задачі T1 .
5. Обчислення .
6. **Прийняти**  від задачі T3.
7. Обчислення .
8. **Передати** задачі T1 .
9. **Прийняти**  від задачі T1.
10. **Передати** задачі T3 .
11. Обчислення.
12. **Передати** задачі T3 результат .

Задача T3

1. Введення .
2. **Передати** задачі T2 .
3. **Прийняти**  від задачі T4.
4. **Прийняти**  від задачі T4.
5. **Передати** задачі T4 .
6. Обчислення .
7. **Передати** задачі T2 .
8. **Прийняти**  від задачі T2.
9. Обчислення .
10. **Прийняти** результат  від задачі T2.
11. **Передати** задачі T4 результат .

Задача T4

1. **Введення .**
2. **Передати** задачі T1 .
3. **Передати** задачі T5 .
4. **Передати** задачі T3 .
5. **Прийняти**  від задачі T5.
6. **Передати** задачі T3 .
7. **Прийняти**  від задачі T3.
8. **Передати** задачі T5 .
9. Обчислення .
10. **Передати** задачі T1 .
11. **Прийняти**  від задачі T1.
12. Обчислення .
13. **Прийняти** результат  від задачі T1.
14. **Прийняти** результат  від задачі T3.
15. **Передати** задачі T5 результат .

Задача T5

1. **Введення .**
2. **Передати** задачі T0 .
3. **Прийняти**  від задачі T4.
4. **Передати** задачі T4 .
5. **Прийняти**  від задачі T4.
6. Обчислення .
7. **Передати** задачі T0 .
8. **Прийняти**  від задачі T0.
9. Обчислення .
10. **Прийняти** результат  від задачі T0.
11. **Прийняти** результат  від задачі T4.
12. Виведення результату .

## Розробка схеми взаємодії процесів

Схема взаємодії задач приведена на рисунку Рисунок 3.1.



Рисунок . – Структурна схема взаємодії задач для ПРГ2

## Розробка програми ПРГ2

Програма написана на мові C++ з використанням бібліотеки MPI та складається з трьох модулів: основного prg2.cpp та допоміжних matrix.cpp і operations.cpp.

Основний модуль prg2.cpp містить тільки функцію main, в якій і зосереджена основна логіка програми: пересилання і отримання повідомлень в залежності від номера задачі. Для передачі і приймання використовувались функції MPI\_Send і MPI\_Recv.

Допоміжні модулі аналогічні тим, які використовуються у програмі ПРГ1. Відмінність полягає лише в одній додатковій функції get в класі Matrix, яка використовується для визначення адреси початку будь-якого рядка об’єкту матриці.

Лістинг програми ПРГ2 наведено у додатку Додаток К. .

## Тестування програми ПРГ2

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

MPI‑програма запускалась за допомогою утиліти Microsoft HPC Pack 2008.

Код скрипта для автоматичного тестування програми ПРГ2 поданий у додатку Додаток Л. .

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

Таблиця . – Час виконання програми ПРГ2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N |  |  |  |  |  |  |
| 800 | 2921 | 1474 | 1030 | 921 | 879 | 633 |
| 1600 | 50421 | 25425 | 17188 | 14777 | 14204 | 10082 |
| 2400 | 177053 | 89314 | 59997 | 51551 | 49375 | 34780 |

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

Таблиця . – Коефіцієнти прискорення для програми ПРГ2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Кількість процесорів (P) | | | | | |
| N | 1 | 2 | 3 | 4 | 5 | 6 |
| 800 | 1,00 | 1,98 | 2,84 | 3,17 | 3,32 | 4,61 |
| 1600 | 1,00 | 1,98 | 2,93 | 3,41 | 3,55 | 5,00 |
| 2400 | 1,00 | 1,98 | 2,95 | 3,43 | 3,59 | 5,09 |

Коефіцієнти ефективності (таблиця Таблиця 3.3) обчислено за даними таблиці Таблиця 3.2.

Таблиця . – Коефіцієнти ефективності для програми ПРГ2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Кількість процесорів (P) | | | | | |
| N | 1 | 2 | 3 | 4 | 5 | 6 |
| 800 | 100,00% | 99,00% | 94,67% | 79,25% | 66,40% | 76,83% |
| 1600 | 100,00% | 99,00% | 97,67% | 85,25% | 71,00% | 83,33% |
| 2400 | 100,00% | 99,00% | 98,33% | 85,75% | 71,80% | 84,83% |

Використовуючи таблиці Таблиця 3.2‑Таблиця 3.3 побудовано графіки зміни коефіцієнтів прискорення і ефективності в залежності від *N* і .



Рисунок . – Графік зміни коефіцієнту прискорення програми ПРГ2 в залежності від кількості ядер при N=800



Рисунок . – Графік зміни коефіцієнту прискорення програми ПРГ2 в залежності від кількості ядер при N=1600



Рисунок . – Графік зміни коефіцієнту прискорення програми ПРГ2 в залежності від кількості ядер при N=2400



Рисунок . – Графік зміни коефіцієнту ефективності програми ПРГ2 в залежності від кількості ядер при N=800



Рисунок . – Графік зміни коефіцієнту ефективності програми ПРГ2 в залежності від кількості ядер при N=1600



Рисунок . – Графік зміни коефіцієнту ефективності програми ПРГ2 в залежності від кількості ядер при N=2400

## Висновки до розділу 3

Виконано розробку програми ПРГ2 для ПКС ЛП з використанням мови С++ і бібліотеки MPI. Тестування програми показало наступне:

* Використання багатоядерної ПКС та програми ПРГ2 забезпечує скорочення часу обчислення заданої математичної задачі. Значення  лежать в межах від 1,98 до 5,09.
* Максимальне значення  забезпечує ПКС з  та .
* Мінімальне значення  виявлено у ПКС з .
* З ростом N  лінійно збільшується.
* Значення  змінюються від 66,4% до 99,0%.
* Найефективніше програма ПРГ2 використовує ПКС з . При цьому .
* Зі зростанням  від 1 до 5  лінійно спадає від 99,0% до 71,0% при . Далі при  зростає до 83,33%.

ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ ПО РОБОТІ

* Подальше підвищення тактової частоти мікропроцесорів стало неможливим із‑за досягнення деяких фізичних обмежень. Випуск багатоядерних процесорів є природним рішенням цієї проблеми.
* Восьмиядерні процесори компанії Intel присутні тільки у лінійках серверних процесорів Itanium та Xeon.
* Процесори модельного ряду Intel Xeon в порівнянні з Intel Itanium більш дешевші і продуктивніші.
* З точки зору концепції необмеженого паралелізму, в системі з будь‑якою кількістю процесорів час виконання операції знаходження максимального елементу матриці більше за час виконання операції множення матриць.
* Падіння коефіцієнту прискорення у програмах ПРГ1 та ПРГ2 зі збільшенням кількості ядер пов’язано в основному з недостатньою пропускною здатністю пам’яті.
* Час виконання програм ПРГ1 та ПРГ2 майже не відрізняється. При  та  час роботи програм збігається з точністю до мікросекунди.
* При малій розмірності  програма для роботи з ПКС ЛП програє ПРГ1. Це пояснюється тим, що складна топологія системи збільшує час на передачу даних між процесорами і як наслідок впливає на загальний час виконання ПРГ2.
* Алгоритм роботи для ПРГ1 значно легший і при проектуванні, і при реалізації, ніж у ПРГ2. У випадку системи з локальною пам‘яттю, велика кількість повідомлень різного розміру ускладнюють систему з пересилкою повідомлень, а також підвищують вірогідність помилки при реалізації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Moore G.E. Cramming more components onto integrated circuits [Text] // Electronics. – 1965, – T.38, No 8.
2. Kanellos M. New Life for Moore’s Law. [Electronic resource]. – Access mode: <http://news.cnet.com/New-life-for-Moores-Law/2009-1006_3-5672485.html>. – Last access: 17.03.2013. – Title from the screen.
3. Excerpts from a Conversation with Gordon Moore: Moore’s Law. [Electronic resource] // Video Transcript. – Access mode: [http://download.intel.com/museum/  
   Moores\_law/Video-transcripts/excepts\_a\_Conversation\_with\_gordon\_Moore.pdf](http://download.intel.com/museum/Moores_law/Video-transcripts/excepts_a_Conversation_with_gordon_Moore.pdf). – Last access: 23.03.2013. – Title from the screen.
4. Moore’s Law: Raising the Bar. [Electronic resource]. – Access mode <http://download.intel.com/museum/Moores_Law/Printed_Materials/Moores_Law_Backgrounder.pdf> – Last access: 17.03.2013. – Title from screen.
5. Tukwila (processor). [Electronic resource]. – Access mode: http://en.wikipedia.org/wiki/Tukwila\_(processor). – Last access: 23.03.2013. – Title from the screen.
6. Пахомов С. Эра многоядерных энергоэффективных процессоров [Электронный ресурс] // КомпьютерПресс. – 2006. – №6. – Режим доступа: <http://www.compress.ru/article.aspx?id=16962&iid=786>. – Последнее обращение: 23.03.2013. – Название с экрана.
7. Knight W. Two Heads Are Better Than One [Text] // IEEE Review, –2005. –Vol. 51, no 9.
8. Таненбаум Э. Архитектура компьютера. – 5-е изд.. – СПб.: Питер, 2007. – С. 81-83. – 848 с. – (Классика Computer Science). – ISBN 5-469-01274-3.
9. Intel Multi-core Processors. Leading the Next Digital Revolution [Electronic resource]. – Access mode: [http://www.intel.com/content/dam/www/public/us/en/  
   documents/technology-briefs/intel-labs-multi-core-revolution-paper.pdf](http://www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/intel-labs-multi-core-revolution-paper.pdf). – Last access: 23.03.2013. – Title from the screen.
10. The POWER4 Processor Introduction and Tuning Guide [Electronic resource]. – Access mode: <http://www.redbooks.ibm.com/redbooks/pdfs/sg247041.pdf>. – Last access: 23.03.2013. – Title from the screen.
11. Intel Multi-Core Briefing [Electronic resource]. – April, 2005. – Access mode: <http://download.intel.com/pressroom/kits/pentiumee/20050418presentation.pdf>. – Last access: 23.03.2013. – Title from the screen.
12. Merrit R. CPU designers debate multi-core future [Electronic resource] // EETimes. – February 2008. – Access mode: <http://eetimes.com/electronics-news/4076123/CPU-designers-debate-multi-core-future>. – Last access: 23.03.2013. – Title from the screen.
13. Muthana P. Packaging of Multi-Core Microprocessors: Tradeoffs and Potential Solutions // 2005 Electronic Components and Technology Conference, 2005.
14. Двухъядерные процессоры Intel и AMD: теория, часть 1 [Электронный ресурс] / С. Озеров, А. Карабуто. – Режим доступа: [http://www.ferra.ru/ru/  
    system/s25922/](http://www.ferra.ru/ru/system/s25922/). – Последнее обращение: 23.03.2013.– Название с экрана.
15. Compare Intel products [Electronic resource]. – Access mode: [http://ark.intel.com/search/advanced/?s=t&CoreCountMin=8&  
    CoreCountMax=8](http://ark.intel.com/search/advanced/?s=t&CoreCountMin=8&CoreCountMax=8). – Last access: 27.03.2013. – Title from the screen.
16. Intel Itanium Processor 9300 Series and 9500 Series [Electronic resource]. – November 2012 – Access mode: [http://www.intel.com/content/dam/www/public  
    /us/en/documents/datasheets/itanium-9300-9500-datasheet.pdf](http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/itanium-9300-9500-datasheet.pdf). – Last access: 27.03.2013. – Title from the screen.
17. HP Integrity Servers [Electronic resource]. – Access mode: <http://h20341.www2.hp.com/integrity/us/en/systems/integrity-systems-overview.html>. – Last access: 27.03.2013. – Title from the screen.
18. Intel Instruction Replay Technology [Electronic resource]. – Access mode: <http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/itanium-9500-reliability-mission-critical-applications-paper.pdf>. – Last access: 27.03.2013. – Title from the screen.
19. Xeon [Electronic resource]. – Access mode: <http://ru.wikipedia.org/wiki/Xeon>. – Last access: 27.03.2013. – Title from the screen.
20. Intel Xeon Processor E7-8800/4800/2800 Families Datasheet [Electronic resource]. – April, 2011. – Vol. 2 – Access mode: [http://www.intel.com/content/  
    dam/www/public/us/en/documents/datasheets/xeon-e7-8800-4800-2800-families-vol-2-datasheet.pdf](http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/xeon-e7-8800-4800-2800-families-vol-2-datasheet.pdf). – Last access: 27.03.2013. – Title from the screen.
21. Westmere (microarchitecture) [Electronic resource]. – Access mode: <http://en.wikipedia.org/wiki/Westmere_(microarchitecture)>. – Last access: 27.03.2013. – Title from the screen.
22. Intel Xeon processor E7‑8800/4800/2800 product families overall summary [Electronic resource]. – Access mode: [http://www.intel.ru/content/www/ru/ru  
    /benchmarks/server/xeon-e7-summary.html](http://www.intel.ru/content/www/ru/ru/benchmarks/server/xeon-e7-summary.html). – Last access: 27.03.2013. – Title from the screen.
23. Intel Xeon Processor E5-1600/E5-2600/E5-4600 Product Families Datasheet [Electronic resource]. – May, 2012. – Vol. 1 – Access mode: <http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/xeon-e5-1600-2600-vol-1-datasheet.pdf> . – Last access: 27.03.2013. – Title from the screen.
24. Sandy Bridge‑E [Electronic resource]. – Access mode: [http://en.wikipedia.org/  
    wiki/Sandy\_Bridge-E](http://en.wikipedia.org/wiki/Sandy_Bridge-E) . – Last access: 27.03.2013. – Title from the screen.
25. Жуков І.А., Корочкін О.В. Паралельні та розподілені обчислення: Навч. посібник [Текст]. – К.: Корнійчук, 2005. – 226 с. – ISBN 996-7599-36-1.

ДОДАТКИ

1. Структурна схема ПКС СП
2. Схема алгоритму головної програми зі вказанням паралельних ділянок для ПРГ1
3. Схема алгоритму процесів для програми ПРГ1
4. Лістинг програми ПРГ1

1 /\*\*

2 \* -------------------------------------------------------------------------

3 \* Паралельні та розподілені обчислення

4 \* Курсова робота. ПРГ1. Бібліотека Win32

5 \*

6 \* Завдання: MA = max(MT)\*ME + alpha\*MD\*MC

7 \*

8 \* Файл: prg1.cpp

9 \* Виконав: Редько Олександр Михайлович

10 \* Група: ІО-01

11 \* Дата: 04.04.13

12 \* --------------------------------------------------------------------------

13 \*/

14

15 #include **<**fstream**>**

16 #include **<**iostream**>**

17 #include **<**Windows**.**h**>**

18

19 #include "operations.h"

20

21 **using** **namespace** std**;**

22

23

24 size\_t N **=** 6**,** // Розмірність матриць

25 P **=** 6**,** // Кількість процесорів

26 H**;**

27

28 CRITICAL\_SECTION csResources**,** csMaxVal**;**

29 HANDLE **\***evnInput**,** **\***evnMax**,** semCalcEnd**;**

30

31 Matrix **\***MA**,** **\***MC**,** **\***MD**,** **\***ME**,** **\***MT**;**

32

33 int alpha**,**

34 maxMT **=** INT\_MIN**;**

35

36 size\_t affinityMask **=** 63**;**

37 char**\*** fileName **=** "testPrg1.txt"**;**

38

39 DWORD WINAPI threadProcedure**(**void **\***lpParameter**);**

40

41 int main**(**int argc**,** char **\***argv**[])** **{**

42

43 // Ініціалізація змінних через параметри командного рядка

44 fileName **=** argv**[**1**];**

45 N **=** atoi**(**argv**[**2**]);**

46 P **=** atoi**(**argv**[**3**]);**

47 affinityMask **=** atoi**(**argv**[**4**]);**

48 H **=** N **/** P**;**

49

50 MA **=** **new** Matrix**(**N**);**

51 MC **=** **new** Matrix**(**N**);**

52 MD **=** **new** Matrix**(**N**);**

53 ME **=** **new** Matrix**(**N**);**

54 MT **=** **new** Matrix**(**N**);**

55

56 evnInput **=** **new** HANDLE**[**3**];**

57 evnMax **=** **new** HANDLE**[**P**];**

58

59 **for(**size\_t i **=** 0**;** i **<** P **-** 1**;** i**++)** **{**

60 evnMax**[**i**]** **=** CreateEvent**(NULL,** TRUE**,** FALSE**,** **NULL);**

61 **if(**i **<** 3**)** **{**

62 evnInput**[**i**]** **=** CreateEvent**(NULL,** TRUE**,** FALSE**,** **NULL);**

63 **}**

64 **}**

65

66 InitializeCriticalSection**(&**csMaxVal**);**

67 InitializeCriticalSection**(&**csResources**);**

68

69 semCalcEnd **=** CreateSemaphore**(NULL,** 0**,** P **-** 1**,** **NULL);**

70

71 LARGE\_INTEGER performanceFrequency**,** startTime**,** endTime**;**

72

73 QueryPerformanceFrequency**(&**performanceFrequency**);**

74 QueryPerformanceCounter**(&**startTime**);**

75

76 // Створення задач

77 HANDLE **\***Thread **=** **new** HANDLE**[**P**];**

78 **for(**size\_t i **=** 0**;** i **<** P**;** i**++)** **{**

79 Thread**[**i**]** **=** CreateThread**(NULL,** 0**,** threadProcedure**,** **(**void **\*)** i**,** 0**,** **NULL);**

80 **}**

81

82 WaitForMultipleObjects**(**P**,** Thread**,** TRUE**,** INFINITE**);**

83

84 QueryPerformanceCounter**(&**endTime**);**

85 double elapsedTime **=** **((**double**)** **(**endTime**.**QuadPart **-** startTime**.**QuadPart**))**

86 **/** **(**double**)** performanceFrequency**.**QuadPart**;**

87 cout **<<** "PRG1: computations took " **<<** **(**int**)(**elapsedTime **\*** 1000**)**

88 **<<** " ms" **<<** endl**;**

89

90 // Запис часу виконання програми в файл

91 ofstream os**(**fileName**,** ofstream**::**out **|** ofstream**::**app**);**

92 os **<<** "N: " **<<** N **<<** "\tProcess time: " **<<** **(**int**)(**elapsedTime **\*** 1000**)**

93 **<<** "\tms" **<<** "\tCores: " **<<** getCores**(**affinityMask**)** **<<** endl**;**

94 os**.**close**();**

95

96 **for(**size\_t i **=** 0**;** i **<** P**;** i**++)** **{**

97 CloseHandle**(**Thread**[**i**]);**

98 **}**

99 DeleteCriticalSection**(&**csMaxVal**);**

100 DeleteCriticalSection**(&**csResources**);**

101 **for(**size\_t i **=** 0**;** i **<** P **-** 1**;** i**++)** **{**

102 CloseHandle**(**evnMax**[**i**]);**

103 **if** **(**i **<** 3**)** **{**

104 CloseHandle**(**evnInput**[**i**]);**

105 **}**

106 **}**

107

108 **return** 0**;**

109 **}**

110

111 DWORD WINAPI threadProcedure**(**void **\***lpParameter**)** **{**

112

113 int tid **=** **(**int**)** lpParameter**;**

114

115 SetThreadAffinityMask**(**GetCurrentThread**(),** affinityMask**);**

116

117 int startIndex **=** tid **\*** H**;**

118 int endIndex **=** **(**tid **!=** P **-** 1**)** **?** **((**tid **+** 1**)** **\*** H**)** **:** N**;**

119

120 cout **<<** "Task " **<<** tid **+** 1 **<<** " started" **<<** endl**;**

121

122 // Введення даних

123 **switch** **(**tid**)** **{**

124 **case** 1 **:**

125 ME**->**fill**(**1**);**

126 SetEvent**(**evnInput**[**0**]);**

127 **break;**

128 **case** 2 **:**

129 MD**->**fill**(**1**);**

130 MT**->**fill**(**1**);**

131 SetEvent**(**evnInput**[**1**]);**

132 **break;**

133 **case** 5 **:**

134 MC**->**fill**(**1**);**

135 alpha **=** 1**;**

136 SetEvent**(**evnInput**[**2**]);**

137 **break;**

138 **}**

139

140 // Чекати на завершення вводу

141 WaitForMultipleObjects**(**3**,** evnInput**,** TRUE**,** INFINITE**);**

142

143 // Обчислення maxMT = max(MT)

144 int maxMtLocal **=** maxValue**(\***MT**,** startIndex**,** endIndex**);**

145 EnterCriticalSection**(&**csMaxVal**);**

146 **if(**maxMtLocal **>** maxMT**)** **{**

147 maxMT **=** maxMtLocal**;**

148 **}**

149 LeaveCriticalSection**(&**csMaxVal**);**

150 SetEvent**(**evnMax**[**tid**]);**

151

152 // Чекати на завершення обчислення maxMT

153 WaitForMultipleObjects**(**P**,** evnMax**,** TRUE**,** INFINITE**);**

154

155 // Копіювання спільних ресурсів

156 EnterCriticalSection**(&**csResources**);**

157 int maxMtCopy **=** maxMT**;**

158 int alphaCopy **=** alpha**;**

159 Matrix McCopy**(\***MC**);**

160 LeaveCriticalSection**(&**csResources**);**

161

162 // Обчислення MAH = maxMT \* MEH + alpha \* MDH \* MC

163 calculation**(**maxMtCopy**,** alphaCopy**,** McCopy**,** **\***MD**,** **\***ME**,**

164 startIndex**,** endIndex**,** **\***MA**);**

165

166 **if(**tid **==** 1**)** **{**

167 // Чекати на завершення обчислення MA

168 **for(**size\_t i **=** 0**;** i **<** P **-** 1**;** i**++)** **{**

169 WaitForSingleObject**(**semCalcEnd**,** INFINITE**);**

170 **}**

171 // Виведення результату MA

172 **if(**N **<** 15**)** **{**

173 cout **<<** **\***MA**;**

174 **}**

175 **}** **else** **{**

176 ReleaseSemaphore**(**semCalcEnd**,** 1**,** **NULL);**

177 **}**

178

179 cout **<<** "Task " **<<** tid **+** 1 **<<** " finished" **<<** endl**;**

180

181 **return** 0**;**

182 **}**

183

184

1 /\*\*

2 \* -------------------------------------------------------------------------

3 \* Паралельні та розподілені обчислення

4 \* Курсова робота. ПРГ1. Бібліотека Win32

5 \*

6 \* Завдання: MA = max(MT)\*ME + alpha\*MD\*MC

7 \*

8 \* Файл: operations.h

9 \* Виконав: Редько Олександр Михайлович

10 \* Група: ІО-01

11 \* Дата: 04.04.13

12 \* --------------------------------------------------------------------------

13 \*/

14

15 #ifndef \_OPERATIONS\_H\_

16 #define \_OPERATIONS\_H\_ 1

17

18 #include "matrix.h"

19

20

21 // Повертає max(MT)

22 int maxValue**(**

23 const Matrix **&**MT**,**

24 size\_t start**,**

25 size\_t end**);**

26

27 // Обчислення MAH = maxMT \* ME + alpha \* MDH \* MC

28 void calculation**(**

29 const int maxMT**,**

30 const int alpha**,**

31 const Matrix **&**ME**,**

32 const Matrix **&**MD**,**

33 const Matrix **&**MC**,**

34 size\_t start**,**

35 size\_t end**,**

36 Matrix **&**MA**);**

37

38 // Повертає кількість ядер за заданою маскою відповідності

39 size\_t getCores**(**size\_t affinityMask**);**

40

41 #endif

1 /\*\*

2 \* -------------------------------------------------------------------------

3 \* Паралельні та розподілені обчислення

4 \* Курсова робота. ПРГ1. Бібліотека Win32

5 \*

6 \* Завдання: MA = max(MT)\*ME + alpha\*MD\*MC

7 \*

8 \* Файл: operations.cpp

9 \* Виконав: Редько Олександр Михайлович

10 \* Група: ІО-01

11 \* Дата: 04.04.13

12 \* --------------------------------------------------------------------------

13 \*/

14

15 #include "operations.h"

16

17

18 int maxValue**(**

19 const Matrix **&**MT**,**

20 size\_t start**,**

21 size\_t end**)**

22 **{**

23 int result **=** INT\_MIN**;**

24 **for(**size\_t i **=** start**;** i **<** end**;** i**++)** **{**

25 **for(**size\_t j **=** 0**;** j **<** MT**.**cols**;** j**++)** **{**

26 int x **=** MT**.**get**(**i**,** j**);**

27 **if(**x **>** result**)** **{**

28 result **=** x**;**

29 **}**

30 **}**

31 **}**

32 **return** result**;**

33 **}**

34

35

36 void calculation**(**

37 const int maxMT**,**

38 const int alpha**,**

39 const Matrix **&**MC**,**

40 const Matrix **&**MD**,**

41 const Matrix **&**ME**,**

42 size\_t start**,**

43 size\_t end**,**

44 Matrix **&**MA**)**

45 **{**

46 **for(**size\_t i **=** start**;** i **<** end**;** i**++)** **{**

47 **for(**size\_t j **=** 0**;** j **<** MA**.**cols**;** j**++)** **{**

48 int sum **=** 0**;**

49 **for(**size\_t k **=** 0**;** k **<** MA**.**cols**;** k**++)** **{**

50 sum **+=** MD**.**get**(**i**,** k**)** **\*** MC**.**get**(**k**,** j**);**

51 **}**

52 sum **=** maxMT **\*** ME**.**get**(**i**,** j**)** **+** alpha **\*** sum**;**

53 MA**.**set**(**i**,** j**,** sum**);**

54 **}**

55 **}**

56 **}**

57

58

59 size\_t getCores**(**size\_t affinityMask**)** **{**

60 size\_t n **=** affinityMask**;**

61 int i **=** 0**,** r **=** 0**;**

62 **for(;** i **<** 8 **\*** **sizeof(**n**);** i**++,** n **>>=** 1**)**

63 **if(**n **&** 1**)**

64 r**++;**

65 **return** r**;**

66 **}**

1 /\*\*

2 \* -------------------------------------------------------------------------

3 \* Паралельні та розподілені обчислення

4 \* Курсова робота. ПРГ1. Бібліотека Win32

5 \*

6 \* Завдання: MA = max(MT)\*ME + alpha\*MD\*MC

7 \*

8 \* Файл: matrix.h

9 \* Виконав: Редько Олександр Михайлович

10 \* Група: ІО-01

11 \* Дата: 04.04.13

12 \* --------------------------------------------------------------------------

13 \*/

14

15 #ifndef \_MATRIX\_H\_

16 #define \_MATRIX\_H\_ 1

17

18 #include **<**assert**.**h**>**

19 #include **<**ostream**>**

20

21

22 class Matrix **{**

23 public**:**

24 Matrix**(**size\_t rows**,** size\_t cols**);**

25 Matrix**(**size\_t N**);**

26 Matrix**(**Matrix const **&**other**);**

27 **~**Matrix**();**

28

29 int get**(**size\_t i**,** size\_t j**)** const **{**

30 assert**(**i **<** **this->**rows**);**

31 assert**(**j **<** **this->**cols**);**

32 **return** **this->**data**[**i **\*** rows **+** j**];**

33 **}**

34

35 void set**(**size\_t i**,** size\_t j**,** int value**)** **{**

36 assert**(**i **<** **this->**rows**);**

37 assert**(**j **<** **this->**cols**);**

38 **this->**data**[**i **\*** rows **+** j**]** **=** value**;**

39 **}**

40

41 void fill**(**int value**);**

42

43 const size\_t rows**;**

44 const size\_t cols**;**

45

46 private**:**

47 int **\***data**;**

48 friend std**::**ostream **&operator<<(**std**::**ostream **&**ostr**,** const Matrix **&**m**);**

49

50 **};**

51

52

53 #endif

1 /\*\*

2 \* -------------------------------------------------------------------------

3 \* Паралельні та розподілені обчислення

4 \* Курсова робота. ПРГ1. Бібліотека Win32

5 \*

6 \* Завдання: MA = max(MT)\*ME + alpha\*MD\*MC

7 \*

8 \* Файл: matrix.cpp

9 \* Виконав: Редько Олександр Михайлович

10 \* Група: ІО-01

11 \* Дата: 04.04.13

12 \* --------------------------------------------------------------------------

13 \*/

14

15 #include "matrix.h"

16

17

18 Matrix**::**Matrix**(**size\_t rows**,** size\_t cols**)** **:**

19 rows**(**rows**),**

20 cols**(**cols**),**

21 data**(new** int**[**rows **\*** cols**])**

22 **{**

23 **}**

24

25 Matrix**::**Matrix**(**size\_t N**)** **:**

26 rows**(**N**),**

27 cols**(**N**),**

28 data**(new** int**[**N **\*** N**])**

29 **{**

30 **}**

31

32 Matrix**::**Matrix**(**const Matrix **&**other**)** **:**

33 rows**(**other**.**rows**),**

34 cols**(**other**.**cols**),**

35 data**(new** int**[**other**.**rows **\*** other**.**cols**])**

36 **{**

37 **for(**size\_t i **=** 0**;** i **<** **this->**rows **\*** **this->**cols**;** i**++)** **{**

38 **this->**data**[**i**]** **=** other**.**data**[**i**];**

39 **}**

40 **}**

41

42 Matrix**::~**Matrix**()** **{**

43 **delete[]** data**;**

44 **}**

45

46

47 void Matrix**::**fill**(**int value**)** **{**

48 **for(**size\_t i **=** 0**;** i **<** **this->**rows**;** i**++)** **{**

49 **for(**size\_t j **=** 0**;** j **<** **this->**cols**;** j**++)** **{**

50 **this->**data**[**i **\*** **this->**cols **+** j**]** **=** value**;**

51 **}**

52 **}**

53 **}**

54

55

56 std**::**ostream **&operator<<(**std**::**ostream **&**ostr**,** const Matrix **&**matrix**)** **{**

57 **for(**size\_t i **=** 0**;** i **<** matrix**.**rows**;** i**++)** **{**

58 **for(**size\_t j **=** 0**;** j **<** matrix**.**cols**;** j**++)** **{**

59 ostr **<<** matrix**.**data**[**i **\*** matrix**.**cols **+** j**]** **<<** " "**;**

60 **}**

61 ostr **<<** std**::**endl**;**

62 **}**

63 **return** ostr**;**

64 **}**

1. Код скрипта для тестування ПРГ1

1 **goto** next

2 /**\*\***

3 **\*** ------------------------------------------------------------------------

4 **\*** Паралельні та розподілені обчислення

5 **\*** Курсова робота. Bat-файл для тестування ПРГ1

6 **\***

7 **\*** Файл: prg1\_test.bat

8 **\*** Виконав: Редько Олександр Михайлович

9 **\*** Група: ІО-01

10 **\*** Дата: 05.04.13

11 **\*** ------------------------------------------------------------------------

12 **\***/:next

13

14 setlocal enabledelayedexpansion

15

16 **set** fileName**=**testResultsWin32.txt

17 **if not exist** **%fileName%** (

18 @echo Tests Win32 **%time:~0,8%** **%date:~-10%>** **%fileName%**

19 ) **else** (

20 @echo.**>>** **%fileName%**

21 @echo Tests Win32 **%time:~0,8%** **%date:~-10%>>** **%fileName%**

22 )

23

24 **set** afMask[1]**=**1

25 **set** afMask[2]**=**3

26 **set** afMask[3]**=**7

27 **set** afMask[4]**=**15

28 **set** afMask[5]**=**31

29 **set** afMask[6]**=**63

30

31 **set** P**=**6

32 **set** N**=**800

33

34 **FOR** /L **%%I IN** (1,1,6) **DO** (

35 start /wait "" pro2\_course\_work\_Win32.exe **%fileName%** **%N%** **%P%** **!afMask[%%I]!**

36 )

37 @echo.**>>** **%fileName%**

38

39 **set** N**=**1600

40 **FOR** /L **%%I IN** (1,1,6) **DO** (

41 start /wait "" pro2\_course\_work\_Win32.exe **%fileName%** **%N%** **%P%** **!afMask[%%I]!**

42 )

43 @echo.**>>** **%fileName%**

44

45 **set** N**=**2400

46 **FOR** /L **%%I IN** (1,1,6) **DO** (

47 start /wait "" pro2\_course\_work\_Win32.exe **%fileName%** **%N%** **%P%** **!afMask[%%I]!**

48 )

49

50 endlocal

1. Структурна схема ПКС ЛП
2. Схема алгоритму головної програми зі вказанням паралельних ділянок для ПРГ2
3. Схема алгоритму процесів для програми ПРГ2
4. Лістинг програми ПРГ2

1 /\*\*

2 \* -------------------------------------------------------------------------

3 \* Паралельні та розподілені обчислення

4 \* Курсова робота. ПРГ2. Бібліотека MPI

5 \*

6 \* Завдання: MA = max(MT)\*ME + alpha\*MD\*MC

7 \*

8 \* Файл: prg2.cpp

9 \* Виконав: Редько Олександр Михайлович

10 \* Група: ІО-01

11 \* Дата: 16.04.13

12 \* -------------------------------------------------------------------------

13 \*/

14

15 #include **<**fstream**>**

16 #include **<**iostream**>**

17 #include **<**mpi**.**h**>**

18

19 #include "operations.h"

20

21 **using** **namespace** std**;**

22

23 #define IGNORE MPI\_STATUS\_IGNORE

24 #define WORLD MPI\_COMM\_WORLD

25

26

27 size\_t N **=** 6**,** // Розмірність матриць

28 P **=** 6**,** // Кількість процесорів

29 H**;**

30

31 int alpha**,**

32 maxMT **=** INT\_MIN**,**

33 maxMtBuf**;**

34

35 Matrix **\***MA**,** **\***MC**,** **\***MD**,** **\***ME**,** **\***MT**;**

36

37 char**\*** fileName **=** "testPrg2.txt"**;**

38 double startTime**,** elapsedTime**;**

39

40 int main**(**int args**,** char**\*** argv**[])** **{**

41

42 // Ініціалізація змінних через параметри командного рядка

43 fileName **=** argv**[**1**];**

44 N **=** atoi**(**argv**[**2**]);**

45 P **=** atoi**(**argv**[**3**]);**

46 H **=** N **/** P**;**

47

48 MPI\_Init**(&**args**,** **&**argv**);**

49 int rank**;**

50 MPI\_Comm\_rank**(**WORLD**,** **&**rank**);**

51

52 size\_t rowMa **=** N**,**

53 rowMe **=** N**,**

54 rowMtd **=** N**;**

55

56 **switch** **(**rank**)** **{**

57 **case** 0**:**

58 rowMa **=** H**;**

59 rowMe **=** 3 **\*** H**;**

60 rowMtd **=** H**;**

61 **break;**

62 **case** 1**:**

63 rowMa **=** H**;**

64 rowMe **=** 2 **\*** H**;**

65 rowMtd **=** 3 **\*** H**;**

66 **break;**

67 **case** 2**:**

68 rowMa **=** H**;**

69 rowMe **=** H**;**

70 rowMtd **=** H**;**

71 **break;**

72 **case** 3**:**

73 rowMa **=** 2 **\*** H**;**

74 rowMe **=** H**;**

75 rowMtd **=** H**;**

76 **break;**

77 **case** 4**:**

78 rowMa **=** 4 **\*** H**;**

79 rowMe **=** 2 **\*** H**;**

80 **break;**

81 **case** 5**:**

82 rowMtd **=** H**;**

83 startTime **=** MPI\_Wtime**();**

84 **break;**

85 **}**

86

87 MA **=** **new** Matrix**(**rowMa**,** N**);**

88 MC **=** **new** Matrix**(**N**);**

89 MD **=** **new** Matrix**(**rowMtd**,** N**);**

90 ME **=** **new** Matrix**(**rowMe**,** N**);**

91 MT **=** **new** Matrix**(**rowMtd**,** N**);**

92

93

94 cout **<<** "Task " **<<** rank **+** 1 **<<** " started" **<<** endl**;**

95

96 **switch** **(**rank**)** **{**

97 **case** 0**:**

98 // 1. Прийняти ME3H від задачі T5

99 MPI\_Recv**(**ME**->**get**(**0**),** 3 **\*** H **\*** N**,** MPI\_INT**,** 5**,** 0**,** WORLD**,** IGNORE**);**

100

101 // 2. Прийняти MDH, MTH від задачі T1

102 MPI\_Recv**(**MD**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 1**,** 0**,** WORLD**,** IGNORE**);**

103 MPI\_Recv**(**MT**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 1**,** 0**,** WORLD**,** IGNORE**);**

104

105 // 3. Передати задачі T1 ME2H

106 MPI\_Send**(**ME**->**get**(**H **\*** N**),** 2 **\*** H **\*** N**,** MPI\_INT**,** 1**,** 0**,** WORLD**);**

107

108 // 4. Прийняти alpha, MC від задачі T1

109 MPI\_Recv**(&**alpha**,** 1**,** MPI\_INT**,** 1**,** 0**,** WORLD**,** IGNORE**);**

110 MPI\_Recv**(**MC**->**get**(**0**),** N **\*** N**,** MPI\_INT**,** 1**,** 0**,** WORLD**,** IGNORE**);**

111

112 // 5. Обчислення maxMT = max(MTH)

113 maxMT **=** maxValue**(\***MT**,** 0**,** H**);**

114

115 // 6. Прийняти maxMT5 від задачі T5

116 MPI\_Recv**(&**maxMtBuf**,** 1**,** MPI\_INT**,** 5**,** 0**,** WORLD**,** IGNORE**);**

117

118 // 7. Обчислення maxMT = max(maxMT, maxMT5)

119 maxMT **=** max**(**maxMT**,** maxMtBuf**);**

120

121 // 8. Передати задачі T1 maxMT

122 MPI\_Send**(&**maxMT**,** 1**,** MPI\_INT**,** 1**,** 0**,** WORLD**);**

123

124 // 9. Прийняти maxMT від задачі T1

125 MPI\_Recv**(&**maxMT**,** 1**,** MPI\_INT**,** 1**,** 0**,** WORLD**,** IGNORE**);**

126

127 // 10. Передати задачі T5 maxMT

128 MPI\_Send**(&**maxMT**,** 1**,** MPI\_INT**,** 5**,** 0**,** WORLD**);**

129

130 // 11. Обчислення MAH = maxMT\*MEH + alpha\*MDH\*MC

131 calculation**(**maxMT**,** alpha**,** **\***MC**,** **\***MD**,** **\***ME**,** H**,** 0**,** 0**,** 0**,** **\***MA**);**

132

133 // 12. Передати задачі T5 результат MAH

134 MPI\_Send**(**MA**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 5**,** 0**,** WORLD**);**

135

136 **break;**

137 **case** 1**:**

138 // 1. Прийняти MD3H, MT3H від задачі T4

139 MPI\_Recv**(**MD**->**get**(**0**),** 3 **\*** H **\*** N**,** MPI\_INT**,** 4**,** 0**,** WORLD**,** IGNORE**);**

140 MPI\_Recv**(**MT**->**get**(**0**),** 3 **\*** H **\*** N**,** MPI\_INT**,** 4**,** 0**,** WORLD**,** IGNORE**);**

141

142 // 2. Передати задачі T0 MDH, MTH

143 MPI\_Send**(**MD**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 0**,** 0**,** WORLD**);**

144 MPI\_Send**(**MT**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 0**,** 0**,** WORLD**);**

145

146 // 3. Передати задачі T2 MDH, MTH

147 MPI\_Send**(**MD**->**get**(**2 **\*** H **\*** N**),** H **\*** N**,** MPI\_INT**,** 2**,** 0**,** WORLD**);**

148 MPI\_Send**(**MT**->**get**(**2 **\*** H **\*** N**),** H **\*** N**,** MPI\_INT**,** 2**,** 0**,** WORLD**);**

149

150 // 4. Прийняти ME2H від задачі T0

151 MPI\_Recv**(**ME**->**get**(**0**),** 2 **\*** H **\*** N**,** MPI\_INT**,** 0**,** 0**,** WORLD**,** IGNORE**);**

152

153 // 5. Передати задачі T2 MEH

154 MPI\_Send**(**ME**->**get**(**H **\*** N**),** H **\*** N**,** MPI\_INT**,** 2**,** 0**,** WORLD**);**

155

156 // 6. Прийняти alpha, MC від задачі T2

157 MPI\_Recv**(&**alpha**,** 1**,** MPI\_INT**,** 2**,** 0**,** WORLD**,** IGNORE**);**

158 MPI\_Recv**(**MC**->**get**(**0**),** N **\*** N**,** MPI\_INT**,** 2**,** 0**,** WORLD**,** IGNORE**);**

159

160 // 7. Передати задачі T0 alpha, MC

161 MPI\_Send**(&**alpha**,** 1**,** MPI\_INT**,** 0**,** 0**,** WORLD**);**

162 MPI\_Send**(**MC**->**get**(**0**),** N **\*** N**,** MPI\_INT**,** 0**,** 0**,** WORLD**);**

163

164 // 8. Обчислення maxMT = max(MTH)

165 maxMT **=** maxValue**(\***MT**,** H**,** 2 **\*** H**);**

166

167 // 9. Прийняти maxMT4 від задачі T4

168 MPI\_Recv**(&**maxMtBuf**,** 1**,** MPI\_INT**,** 4**,** 0**,** WORLD**,** IGNORE**);**

169

170 // 10. Обчислення maxMT = max(maxMT, maxMt4)

171 maxMT **=** max**(**maxMT**,** maxMtBuf**);**

172

173 // 11. Прийняти maxMT0 від задачі T0

174 MPI\_Recv**(&**maxMtBuf**,** 1**,** MPI\_INT**,** 0**,** 0**,** WORLD**,** IGNORE**);**

175

176 // 12. Обчислення maxMT = max(maxMT, maxMT0)

177 maxMT **=** max**(**maxMT**,** maxMtBuf**);**

178

179 // 13. Прийняти maxMT2 від задачі T2

180 MPI\_Recv**(&**maxMtBuf**,** 1**,** MPI\_INT**,** 2**,** 0**,** WORLD**,** IGNORE**);**

181

182 // 14. Обчислення maxMT = max(maxMT, maxMT2)

183 maxMT **=** max**(**maxMT**,** maxMtBuf**);**

184

185 // 15. Передати задачі T0 maxMT

186 MPI\_Send**(&**maxMT**,** 1**,** MPI\_INT**,** 0**,** 0**,** WORLD**);**

187

188 // 16. Передати задачі T2 maxMT

189 MPI\_Send**(&**maxMT**,** 1**,** MPI\_INT**,** 2**,** 0**,** WORLD**);**

190

191 // 17. Передати задачі T4 maxMT

192 MPI\_Send**(&**maxMT**,** 1**,** MPI\_INT**,** 4**,** 0**,** WORLD**);**

193

194 // 18. Обчислення MAH = maxMT\*MEH + alpha\*MDH\*MC

195 calculation**(**maxMT**,** alpha**,** **\***MC**,** **\***MD**,** **\***ME**,** H**,** 0**,** H**,** 0**,** **\***MA**);**

196

197 // 19. Передати задачі T4 результат MAH

198 MPI\_Send**(**MA**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 4**,** 0**,** WORLD**);**

199

200 **break;**

201 **case** 2**:**

202 // 1. Прийняти alpha, MC від задачі T3

203 MPI\_Recv**(&**alpha**,** 1**,** MPI\_INT**,** 3**,** 0**,** WORLD**,** IGNORE**);**

204 MPI\_Recv**(**MC**->**get**(**0**),** N **\*** N**,** MPI\_INT**,** 3**,** 0**,** WORLD**,** IGNORE**);**

205

206 // 2. Прийняти MDH, MTH від задачі T1

207 MPI\_Recv**(**MD**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 1**,** 0**,** WORLD**,** IGNORE**);**

208 MPI\_Recv**(**MT**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 1**,** 0**,** WORLD**,** IGNORE**);**

209

210 // 3. Прийняти MEH від задачі T1

211 MPI\_Recv**(**ME**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 1**,** 0**,** WORLD**,** IGNORE**);**

212

213 // 4. Передати задачі T1 alpha, MC

214 MPI\_Send**(&**alpha**,** 1**,** MPI\_INT**,** 1**,** 0**,** WORLD**);**

215 MPI\_Send**(**MC**->**get**(**0**),** N **\*** N**,** MPI\_INT**,** 1**,** 0**,** WORLD**);**

216

217 // 5. Обчислення maxMT = max(MTH)

218 maxMT **=** maxValue**(\***MT**,** 0**,** H**);**

219

220 // 6. Прийняти maxMT3 від задачі T3

221 MPI\_Recv**(&**maxMtBuf**,** 1**,** MPI\_INT**,** 3**,** 0**,** WORLD**,** IGNORE**);**

222

223 // 7. Обчислення maxMT = max(maxMT, maxMT3)

224 maxMT **=** max**(**maxMT**,** maxMtBuf**);**

225

226 // 8. Передати задачі T1 maxMT

227 MPI\_Send**(&**maxMT**,** 1**,** MPI\_INT**,** 1**,** 0**,** WORLD**);**

228

229 // 9. Прийняти maxMT від задачі T1

230 MPI\_Recv**(&**maxMT**,** 1**,** MPI\_INT**,** 1**,** 0**,** WORLD**,** IGNORE**);**

231

232 // 10. Передати задачі T3 maxMT

233 MPI\_Send**(&**maxMT**,** 1**,** MPI\_INT**,** 3**,** 0**,** WORLD**);**

234

235 // 11. Обчислення MAH = maxMT\*MEH + alpha\*MDH\*MC

236 calculation**(**maxMT**,** alpha**,** **\***MC**,** **\***MD**,** **\***ME**,** H**,** 0**,** 0**,** 0**,** **\***MA**);**

237

238 // 12. Передати задачі T3 результат MAH

239 MPI\_Send**(**MA**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 3**,** 0**,** WORLD**);**

240

241 **break;**

242 **case** 3**:**

243 // 1. Введення alpha, MC

244 alpha **=** 1**;**

245 MC**->**fill**(**1**);**

246

247 // 2. Передати задачі T2 alpha, MC

248 MPI\_Send**(&**alpha**,** 1**,** MPI\_INT**,** 2**,** 0**,** WORLD**);**

249 MPI\_Send**(**MC**->**get**(**0**),** N **\*** N**,** MPI\_INT**,** 2**,** 0**,** WORLD**);**

250

251 // 3. Прийняти MDH, MTH від задачі T4

252 MPI\_Recv**(**MD**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 4**,** 0**,** WORLD**,** IGNORE**);**

253 MPI\_Recv**(**MT**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 4**,** 0**,** WORLD**,** IGNORE**);**

254

255 // 4. Прийняти MEH від задачі T4

256 MPI\_Recv**(**ME**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 4**,** 0**,** WORLD**,** IGNORE**);**

257

258 // 5. Передати задачі T4 alpha, MC

259 MPI\_Send**(&**alpha**,** 1**,** MPI\_INT**,** 4**,** 0**,** WORLD**);**

260 MPI\_Send**(**MC**->**get**(**0**),** N **\*** N**,** MPI\_INT**,** 4**,** 0**,** WORLD**);**

261

262 // 6. Обчислення maxMT = max(MTH)

263 maxMT **=** maxValue**(\***MT**,** 0**,** H**);**

264

265 // 7. Передати задачі T2 maxMT

266 MPI\_Send**(&**maxMT**,** 1**,** MPI\_INT**,** 2**,** 0**,** WORLD**);**

267

268 // 8. Прийняти maxMT від задачі T2

269 MPI\_Recv**(&**maxMT**,** 1**,** MPI\_INT**,** 2**,** 0**,** WORLD**,** IGNORE**);**

270

271 // 9. Обчислення MAH = maxMT\*MEH + alpha\*MDH\*MC

272 calculation**(**maxMT**,** alpha**,** **\***MC**,** **\***MD**,** **\***ME**,** H**,** H**,** 0**,** 0**,** **\***MA**);**

273

274 // 10. Прийняти результат MAH від задачі T2

275 MPI\_Recv**(**MA**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 2**,** 0**,** WORLD**,** IGNORE**);**

276

277 // 11. Передати задачі T4 результат MA2H

278 MPI\_Send**(**MA**->**get**(**0**),** 2 **\*** H **\*** N**,** MPI\_INT**,** 4**,** 0**,** WORLD**);**

279

280 **break;**

281 **case** 4**:**

282 // 1. Введення MD, MT

283 MD**->**fill**(**1**);**

284 MT**->**fill**(**1**);**

285

286 // 2. Передати задачі T1 MD3H, MD3H

287 MPI\_Send**(**MD**->**get**(**0**),** 3 **\*** H **\*** N**,** MPI\_INT**,** 1**,** 0**,** WORLD**);**

288 MPI\_Send**(**MT**->**get**(**0**),** 3 **\*** H **\*** N**,** MPI\_INT**,** 1**,** 0**,** WORLD**);**

289

290 // 3. Передати задачі T5 MDH, MTH

291 MPI\_Send**(**MD**->**get**(**5 **\*** H **\*** N**),** H **\*** N**,** MPI\_INT**,** 5**,** 0**,** WORLD**);**

292 MPI\_Send**(**MT**->**get**(**5 **\*** H **\*** N**),** H **\*** N**,** MPI\_INT**,** 5**,** 0**,** WORLD**);**

293

294 // 4. Передати задачі T3 MDH, MTH

295 MPI\_Send**(**MD**->**get**(**3 **\*** H **\*** N**),** H **\*** N**,** MPI\_INT**,** 3**,** 0**,** WORLD**);**

296 MPI\_Send**(**MT**->**get**(**3 **\*** H **\*** N**),** H **\*** N**,** MPI\_INT**,** 3**,** 0**,** WORLD**);**

297

298 // 5. Прийняти ME2H від задачі T5

299 MPI\_Recv**(**ME**->**get**(**0**),** 2 **\*** H **\*** N**,** MPI\_INT**,** 5**,** 0**,** WORLD**,** IGNORE**);**

300

301 // 6. Передати задачі T3 MEH

302 MPI\_Send**(**ME**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 3**,** 0**,** WORLD**);**

303

304 // 7. Прийняти alpha, MC від задачі T3

305 MPI\_Recv**(&**alpha**,** 1**,** MPI\_INT**,** 3**,** 0**,** WORLD**,** IGNORE**);**

306 MPI\_Recv**(**MC**->**get**(**0**),** N **\*** N**,** MPI\_INT**,** 3**,** 0**,** WORLD**,** IGNORE**);**

307

308 // 8. Передати задачі T5 alpha, MC

309 MPI\_Send**(&**alpha**,** 1**,** MPI\_INT**,** 5**,** 0**,** WORLD**);**

310 MPI\_Send**(**MC**->**get**(**0**),** N **\*** N**,** MPI\_INT**,** 5**,** 0**,** WORLD**);**

311

312 // 9. Обчислення maxMT = max(MTH)

313 maxMT **=** maxValue**(\***MT**,** 4 **\*** H**,** N**);**

314

315 // 10. Передати задачі T1 maxMT

316 MPI\_Send**(&**maxMT**,** 1**,** MPI\_INT**,** 1**,** 0**,** WORLD**);**

317

318 // 11. Прийняти від задачі T1 maxMT

319 MPI\_Recv**(&**maxMT**,** 1**,** MPI\_INT**,** 1**,** 0**,** WORLD**,** IGNORE**);**

320

321 // 12. Обчислення MAH = maxMT\*MEH + alpha\*MDH\*MC

322 calculation**(**maxMT**,** alpha**,** **\***MC**,** **\***MD**,** **\***ME**,** H**,** 3 **\*** H**,** 4 **\*** H**,** H**,** **\***MA**);**

323

324 // 13. Прийняти результат MAH від задачі T1

325 MPI\_Recv**(**MA**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 1**,** 0**,** WORLD**,** IGNORE**);**

326

327 // 14. Прийняти результат MA2H від задачі T3

328 MPI\_Recv**(**MA**->**get**(**H **\*** N**),** 2 **\*** H **\*** N**,** MPI\_INT**,** 3**,** 0**,** WORLD**,** IGNORE**);**

329

330 // 15. Передати задачі T5 результат MA4H

331 MPI\_Send**(**MA**->**get**(**0**),** 4 **\*** H **\*** N**,** MPI\_INT**,** 5**,** 0**,** WORLD**);**

332

333 **break;**

334 **case** 5**:**

335 // 1. Введення ME

336 ME**->**fill**(**1**);**

337

338 // 2. Передати задачі T0 ME3H

339 MPI\_Send**(**ME**->**get**(**0**),** 3 **\*** H **\*** N**,** MPI\_INT**,** 0**,** 0**,** WORLD**);**

340

341 // 3. Прийняти MDH, MTH від задачі T4

342 MPI\_Recv**(**MD**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 4**,** 0**,** WORLD**,** IGNORE**);**

343 MPI\_Recv**(**MT**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 4**,** 0**,** WORLD**,** IGNORE**);**

344

345 // 4. Передати ME2H задачі T4

346 MPI\_Send**(**ME**->**get**(**3 **\*** H **\*** N**),** 2 **\*** H **\*** N**,** MPI\_INT**,** 4**,** 0**,** WORLD**);**

347

348 // 5. Прийняти alpha, MC від задачі T4

349 MPI\_Recv**(&**alpha**,** 1**,** MPI\_INT**,** 4**,** 0**,** WORLD**,** IGNORE**);**

350 MPI\_Recv**(**MC**->**get**(**0**),** N **\*** N**,** MPI\_INT**,** 4**,** 0**,** WORLD**,** IGNORE**);**

351

352 // 6. Обчислення maxMT = max(MTH)

353 maxMT **=** maxValue**(\***MT**,** 0**,** H**);**

354

355 // 7. Передати задачі T0 maxMT

356 MPI\_Send**(&**maxMT**,** 1**,** MPI\_INT**,** 0**,** 0**,** WORLD**);**

357

358 // 8. Прийняти maxMT від задачі T0

359 MPI\_Recv**(&**maxMT**,** 1**,** MPI\_INT**,** 0**,** 0**,** WORLD**,** IGNORE**);**

360

361 // 9. Обчислення MAH = maxMT\*MEH + alpha\*MDH\*MC

362 calculation**(**maxMT**,** alpha**,** **\***MC**,** **\***MD**,** **\***ME**,** H**,** 5 **\*** H**,** 0**,** 5 **\*** H**,** **\***MA**);**

363

364 // 10. Прийняти результат MAH від задачі T0

365 MPI\_Recv**(**MA**->**get**(**0**),** H **\*** N**,** MPI\_INT**,** 0**,** 0**,** WORLD**,** IGNORE**);**

366

367 // 11. Прийняти результат MA4H від задачі T4

368 MPI\_Recv**(**MA**->**get**(**H**\***N**),** 4 **\*** H **\*** N**,** MPI\_INT**,** 4**,** 0**,** WORLD**,** IGNORE**);**

369

370 // 12. Виведення результату MA

371 **if** **(**MA**->**cols **<** 15**)** **{**

372 cout **<<** **\***MA **<<** endl**;**

373 **}**

374

375 **break;**

376 **}**

377

378 cout **<<** "Task " **<<** rank **+** 1 **<<** " finished" **<<** endl**;**

379

380 **if(**rank **==** 5**)** **{**

381 elapsedTime **=** MPI\_Wtime**()** **-** startTime**;**

382 cout **<<** "PRG2: computations took " **<<** **(**int**)** **(**elapsedTime **\*** 1000**)**

383 **<<** " ms" **<<**endl**;**

384

385 ofstream os**(**fileName**,** ofstream**::**out **|** ofstream**::**app**);**

386 os **<<** "N: " **<<** N **<<** "\tProcess time: "

387 **<<** **(**int**)(**elapsedTime **\*** 1000**)** **<<** "\tms "**;**

388 os**.**close**();**

389 **}**

390

391 MPI\_Finalize**();**

392 **if(**rank **==** 5**)** **{**

393 system**(**"pause"**);**

394 **}**

395

396 **return** 0**;**

397 **}**

1 /\*\*

2 \* -------------------------------------------------------------------------

3 \* Паралельні та розподілені обчислення

4 \* Курсова робота. ПРГ2. Бібліотека MPI

5 \*

6 \* Завдання: MA = max(MT)\*ME + alpha\*MD\*MC

7 \*

8 \* Файл: operations.h

9 \* Виконав: Редько Олександр Михайлович

10 \* Група: ІО-01

11 \* Дата: 16.04.13

12 \* --------------------------------------------------------------------------

13 \*/

14

15 #ifndef \_OPERATIONS\_H\_

16 #define \_OPERATIONS\_H\_ 1

17

18 #include "matrix.h"

19

20

21 // Повертає max(MT)

22 int maxValue**(**

23 const Matrix **&**MT**,**

24 size\_t start**,**

25 size\_t end**);**

26

27 // Обчислення MAH = maxMT \* ME + alpha \* MDH \* MC

28 void calculation**(**

29 const int maxMT**,**

30 const int alpha**,**

31 const Matrix **&**MC**,**

32 const Matrix **&**MD**,**

33 const Matrix **&**ME**,**

34 size\_t H**,**

35 size\_t offsMA**,**

36 size\_t offsMD**,**

37 size\_t offsME**,**

38 Matrix **&**MA**);**

39

40

41 #endif

1 /\*\*

2 \* -------------------------------------------------------------------------

3 \* Паралельні та розподілені обчислення

4 \* Курсова робота. ПРГ2. Бібліотека MPI

5 \*

6 \* Завдання: MA = max(MT)\*ME + alpha\*MD\*MC

7 \*

8 \* Файл: operations.cpp

9 \* Виконав: Редько Олександр Михайлович

10 \* Група: ІО-01

11 \* Дата: 16.04.13

12 \* --------------------------------------------------------------------------

13 \*/

14

15 #include "operations.h"

16

17

18 int maxValue**(**

19 const Matrix **&**MT**,**

20 size\_t start**,**

21 size\_t end**)**

22 **{**

23 int result **=** INT\_MIN**;**

24 **for(**size\_t i **=** start**;** i **<** end**;** i**++)** **{**

25 **for(**size\_t j **=** 0**;** j **<** MT**.**cols**;** j**++)** **{**

26 int x **=** MT**.**get**(**i**,** j**);**

27 **if(**x **>** result**)** **{**

28 result **=** x**;**

29 **}**

30 **}**

31 **}**

32 **return** result**;**

33 **}**

34

35 void calculation**(**

36 const int maxMT**,**

37 const int alpha**,**

38 const Matrix **&**MC**,**

39 const Matrix **&**MD**,**

40 const Matrix **&**ME**,**

41 size\_t H**,**

42 size\_t offsMA**,**

43 size\_t offsMD**,**

44 size\_t offsME**,**

45 Matrix **&**MA**)**

46 **{**

47 int sum**;**

48 **for(**size\_t i **=** 0**;** i **<** H**;** i**++)** **{**

49 **for(**size\_t j **=** 0**;** j **<** MA**.**cols**;** j**++)** **{**

50 sum **=** 0**;**

51 **for(**size\_t k **=** 0**;** k **<** MA**.**cols**;** k**++)** **{**

52 sum **+=** MD**.**get**(**i **+** offsMD**,** k**)** **\*** MC**.**get**(**k**,** j**);**

53 **}**

54 sum **=** maxMT **\*** ME**.**get**(**i **+** offsME**,** j**)** **+** alpha **\*** sum**;**

55 MA**.**set**(**i **+** offsMA**,** j**,** sum**);**

56 **}**

57 **}**

58 **}**

1 /\*\*

2 \* -------------------------------------------------------------------------

3 \* Паралельні та розподілені обчислення

4 \* Курсова робота. ПРГ2. Бібліотека MPI

5 \*

6 \* Завдання: MA = max(MT)\*ME + alpha\*MD\*MC

7 \*

8 \* Файл: matrix.h

9 \* Виконав: Редько Олександр Михайлович

10 \* Група: ІО-01

11 \* Дата: 16.04.13

12 \* --------------------------------------------------------------------------

13 \*/

14

15 #ifndef \_MATRIX\_H\_

16 #define \_MATRIX\_H\_ 1

17

18 #include **<**assert**.**h**>**

19 #include **<**ostream**>**

20

21

22 class Matrix **{**

23 public**:**

24 Matrix**(**size\_t rows**,** size\_t cols**);**

25 Matrix**(**size\_t N**);**

26 Matrix**(**Matrix const **&**other**);**

27 **~**Matrix**();**

28

29 int get**(**size\_t i**,** size\_t j**)** const **{**

30 assert**(**i **<** **this->**rows**);**

31 assert**(**j **<** **this->**cols**);**

32 **return** **this->**data**[**i **\*** cols **+** j**];**

33 **}**

34

35 void set**(**size\_t i**,** size\_t j**,** int value**)** **{**

36 assert**(**i **<** **this->**rows**);**

37 assert**(**j **<** **this->**cols**);**

38 **this->**data**[**i **\*** cols **+** j**]** **=** value**;**

39 **}**

40

41 void**\*** get**(**size\_t element**)** **{**

42 **return** **this->**data **+** element**;**

43 **}**

44

45 void fill**(**int value**);**

46

47 const size\_t rows**;**

48 const size\_t cols**;**

49

50 private**:**

51 int **\***data**;**

52 friend std**::**ostream **&operator<<(**std**::**ostream **&**ostr**,** const Matrix **&**m**);**

53

54 **};**

55

56 #endif

1 /\*\*

2 \* -------------------------------------------------------------------------

3 \* Паралельні та розподілені обчислення

4 \* Курсова робота. ПРГ2. Бібліотека MPI

5 \*

6 \* Завдання: MA = max(MT)\*ME + alpha\*MD\*MC

7 \*

8 \* Файл: matrix.cpp

9 \* Виконав: Редько Олександр Михайлович

10 \* Група: ІО-01

11 \* Дата: 16.04.13

12 \* --------------------------------------------------------------------------

13 \*/

14

15 #include "matrix.h"

16

17

18 Matrix**::**Matrix**(**size\_t N**)** **:**

19 rows**(**N**),**

20 cols**(**N**),**

21 data**(new** int**[**N **\*** N**])**

22 **{**

23 **}**

24

25 Matrix**::**Matrix**(**size\_t rows**,** size\_t cols**)** **:**

26 rows**(**rows**),**

27 cols**(**cols**),**

28 data**(new** int**[**rows **\*** cols**])**

29 **{**

30 **}**

31

32 Matrix**::**Matrix**(**const Matrix **&**other**)** **:**

33 rows**(**other**.**rows**),**

34 cols**(**other**.**cols**),**

35 data**(new** int**[**other**.**rows **\*** other**.**cols**])**

36 **{**

37 **for(**size\_t i **=** 0**;** i **<** **this->**rows **\*** **this->**cols**;** i**++)** **{**

38 **this->**data**[**i**]** **=** other**.**data**[**i**];**

39 **}**

40 **}**

41

42 Matrix**::~**Matrix**()** **{**

43 **delete[]** data**;**

44 **}**

45

46

47 void Matrix**::**fill**(**int value**)** **{**

48 **for(**size\_t i **=** 0**;** i **<** **this->**rows**;** i**++)** **{**

49 **for(**size\_t j **=** 0**;** j **<** **this->**cols**;** j**++)** **{**

50 **this->**data**[**i **\*** **this->**cols **+** j**]** **=** value**;**

51 **}**

52 **}**

53 **}**

54

55

56 std**::**ostream **&operator<<(**std**::**ostream **&**ostr**,** const Matrix **&**matrix**)** **{**

57 **for(**size\_t i **=** 0**;** i **<** matrix**.**rows**;** i**++)** **{**

58 **for(**size\_t j **=** 0**;** j **<** matrix**.**cols**;** j**++)** **{**

59 ostr **<<** matrix**.**data**[**i **\*** matrix**.**cols **+** j**]** **<<** " "**;**

60 **}**

61 ostr **<<** std**::**endl**;**

62 **}**

63 **return** ostr**;**

64 **}**

1. Код скрипта для тестування ПРГ2

1 **goto** next

2 /**\*\***

3 **\*** ------------------------------------------------------------------------

4 **\*** Паралельні та розподілені обчислення

5 **\*** Курсова робота. Bat-файл для тестування ПРГ2

6 **\***

7 **\*** Файл: prg2\_test.bat

8 **\*** Виконав: Редько Олександр Михайлович

9 **\*** Група: ІО-01

10 **\*** Дата: 20.04.13

11 **\*** ------------------------------------------------------------------------

12 **\***/

13 :next

14

15 setlocal enabledelayedexpansion

16

17 **set** fileName**=**testResultsMPI.txt

18 **if not exist** **%fileName%** (

19 @echo Tests MPI **%time:~0,8%** **%date:~-10%>** **%fileName%**

20 ) **else** (

21 @echo.**>>** **%fileName%**

22 @echo Tests MPI **%time:~0,8%** **%date:~-10%>>** **%fileName%**

23 )

24

25 rem Hexadecimal values

26 **set** afMask[1]**=**0x1

27 **set** afMask[2]**=**0x3

28 **set** afMask[3]**=**0x7

29 **set** afMask[4]**=**0xF

30 **set** afMask[5]**=**0x1F

31 **set** afMask[6]**=**0x3F

32

33 **set** P**=**6

34 **set** N**=**800

35

36 **FOR** /L **%%I IN** (1,1,6) **DO** (

37 start /wait /affinity **!afMask[%%I]!** mpiexec -n 6 pro2\_course\_work\_MPI.exe **%fileName%** **%N%** **%P%**

38 @echo Cores: **%%I** **>>%fileName%**

39 )

40 @echo.**>>** **%fileName%**

41

42 **set** N**=**1600

43 **FOR** /L **%%I IN** (1,1,6) **DO** (

44 start /wait /affinity **!afMask[%%I]!** mpiexec -n 6 pro2\_course\_work\_MPI.exe **%fileName%** **%N%** **%P%**

45 @echo Cores: **%%I** **>>%fileName%**

46 )

47 @echo.**>>** **%fileName%**

48

49 **set** N**=**2400

50 **FOR** /L **%%I IN** (1,1,6) **DO** (

51 start /wait /affinity **!afMask[%%I]!** mpiexec -n 6 pro2\_course\_work\_MPI.exe **%fileName%** **%N%** **%P%**

52 @echo Cores: **%%I** **>>%fileName%**

53 )

54

55 endlocal