Міністерство освіти і науки України

Криворізький національний університет

Факультет інформаційних технологій

Кафедра комп’ютерних систем та мереж

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Комп’ютерна система на основі масивно-паралельної архітектури типу SunwayTaihuLight»

Студентка 4-го курсу групи КІ-21

напряму підготовки

123 «Комп’ютерна інженерія»

Перекопська Ю. А.

Керівник

проф. каф. КСМ

Купін А.І.

м. Кривий Ріг – 2025 рік

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Кафедра: Комп’ютерних систем та мереж

Дисципліна: Комп’ютерні системи

Спеціальність: Комп’ютерні системи та мережі

Курс: 4 Група: КІ-21 Семестр: 8\_

**ЗАВДАННЯ**

НА КВАЛІФІКАЦІЙНУ РОБОТА СТУДЕНТА

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Перекопська Юлія Андріївна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема роботи (роботи):Архітектура на основі масивно-паралельної структури типуSunwayTaihuLight.
2. Термін здачі студентом закінченого роботи: 01.04.2025
3. Вихідні данні до проекту (роботи):Об’єкт розробки — програмне забезпечення для паралельної обробки елементів масиву. Базові технічні характеристики досліджуваної системи: процесори SW26010 (архітектура ShenWei), розподілена пам’ять, комунікаційна інфраструктура InfiniBand.Використовувані програмні інструменти: мова програмування C#, середовище розробки Visual Studio Community, візуалізація алгоритмів — Microsoft Visio.Платформа тестування — персональний комп’ютер користувача.Початкові параметри для експериментів: обсяг масиву: до 100 000 елементів, кількість потоків: від 1 до 16, діапазон значень: 1 – 100 000.
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити): Аналіз сучасних комп’ютерних систем із масивно-паралельною обробкою, дослідження архітектури системи типу *Sunway TaihuLight*,огляд технічних характеристик процесора SW26010 та принципів його функціонування, розгляд особливостей архітектури MPP (Massively Parallel Processing) і принципів побудови обчислювального модуля,постановка задачі на розробку програмного забезпечення для паралельної обробки масиву,розробка алгоритму паралельного сортування елементів масиву,реалізація програмного забезпечення мовою C# у середовищі Visual Studio,проведення експериментального тестування ПЗ з аналізом впливу,аналіз результатів тестування,формулювання висновків щодо доцільності використання обраного підходу.
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень):

Структурна схема MPP-блоку паралельної обчислювальної системи – загальна архітектура модуля з виділенням основних елементів (майстер-вузол, підлеглі вузли, комутатори, мережа InfiniBand).Блок-схема алгоритму паралельного сортування масиву – логічна структура розбиття масиву та виконання сортування в багатьох потоках.Блок-схема функціонування розробленого програмного забезпечення – етапи роботи з даними: введення, розподіл, сортування, об'єднання, вивід.Схема архітектури процесора SW26010 – кластерна структура з обчислювальними елементами CPE та керуючим MPE.Схема з'єднання вузлів у Sunway TaihuLight – логічне або фізичне представлення розподілу компонентів системи.

1. Дата видачі завдання: 05.02.2025

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
| 1. | Постановка завдання | 05.02.2025 – 15.02.2025 |  |
| 2. | Вступ | 18.02.2025 – 22.02.2025 |  |
| 3. | Аналіз комп’ютерної системи з масивно-паралельною обробкою | 25.03.2025 – 1.03.2025 |  |
| 4. | Розробка паралельного алгоритму та програмного забезпечення для обробки вектору | 1.03.2025 – 15.03.2025 |  |
| 5. | Аналіз показників розробленого алгоритму та програмного забезпечення | 18.03.2025 – 22.03.2025 |  |
| 6. | Повний варіант записки | 25.03.2025 – 30.03.2025 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_ Перекопська Ю. А.

Керівник роботи: \_\_\_\_\_\_\_\_\_\_\_\_\_ Купін А.І.

01.04.2025

РЕФЕРАТ

Пояснювальна записка: 43 сторінки, 26 рисунків, 3 таблиці, 7 використаних джерел, 1 додаток.

Об’єкт проектування – Програмне забезпечення паралельної обробки елементів масиву

Роботи складається з трьох основних розділів.

Перший розділ присвячений огляду та аналізу комп’ютерних систем з масивно-паралельною обробкою типу SunwayThaihuLight та MPP комп’ютерних систем.

У другому розділі виконується розробка алгоритму та програми паралельної обробки елементів масиву, описується алгоритм функціонування, описуються основні етапи розробки.

У третьому розділі виконується аналітична та розрахункова робота відносно продуктивності, надійності, відмовостійкості, показників прискорення.

КОМП’ЮТЕРНІ СИСТЕМИ З МАСИВНО-ПАРАЛЕЛЬНОЮ ОБРОБКОЮ, SUNWAYTHAIHULIGHT, АЛГОРИТМ РОЗРАХУНКУ ЕЛЕМЕНТІВ ВЕКТОРА

Explanatory note: 43 pages, 26 figures, 3 tables, 7 sources used, 1 application.

Design object - Software for parallel processing of array elements

The project consists of three main sections.

The first section is dedicated to the review and analysis of computer systems with massively parallel processing such as Sunway TaihuLight and MPP computer systems.

In the second section, the development of the algorithm and the program of parallel processing of elements of the array, describes the algorithm of operation, describes the main stages of development.

In the third section analytical and design work is carried out in relation to performance, reliability, fault-tolerance, acceleration indicators.

SUNWAY TAIHULIGHT COMPUTER SYSTEMS WITH MASSIVE-PARALLEL PROCESSING, VEGETABLE ELEMENTS CALCULATION ALGORITHM

ЗМІСТ

[ПЕРЕЛІК СКОРОЧЕНЬ 9](#_Toc2541598)

[ВСТУП 10](#_Toc2541599)

[1. АНАЛІЗ КОМП’ЮТЕРНОЇ СИСТЕМИ З МАСИВНО-ПАРАЛЕЛЬНОЮ ОБРОБКОЮ ТИПУ SUNWAYTAIHULIGHT 12](#_Toc2541600)

[1.1. Визначення поняття комп’ютерних систем 13](#_Toc2541601)

[1.2. Аналіз комп’ютерних багатопроцесорних систем, архітектура системи з масивно-паралельною обробкою MPP. 14](#_Toc2541602)

[1.3. Опис функціонування типового MPP блоку 16](#_Toc2541603)

[1.4 Аналіз архітектури комп’ютерної системи SunwayTaihuLight 20](#_Toc2541604)

[1.4.1 Огляд процесору SW26010 на базі архітектури ShenWei 22](#_Toc2541605)

[1.5 Висновок 23](#_Toc2541606)

[2. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПАРАЛЕЛЬНОГО СОРТУВАННЯ ЕЛЕМЕНТІВ МАСИВУ 24](#_Toc2541607)

[2.1. Постановка задачі 25](#_Toc2541608)

[2.2. Розробка алгоритму функціонування програмного забезпечення 25](#_Toc2541609)

[2.3. Розробка програмного забезпечення для паралельного сортування масиву 27](#_Toc2541610)

[2.4. Висновок 29](#_Toc2541611)

[3. ТЕСТУВАННЯ ПЗ ДЛЯ СОРТУВАННЯ ЕЛЕМЕНТІВ МАСИВУ 30](#_Toc2541612)

[3.1. Тестування розробленого програмного забезпечення 31](#_Toc2541613)

[3.2. Висновок 36](#_Toc2541614)

[ВИСНОВКИ 37](#_Toc2541615)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 39](#_Toc2541616)

[ДОДАТКИ 41](#_Toc2541617)

# ПЕРЕЛІК СКОРОЧЕНЬ

КС – комп’ютерна система

ПК – персональний комп’ютер;

ПЗ – програмне забезпечення;

DDR - double data rate;

RAM – random access memory;

MPP – massive parallel processing;

DMA –direct memory access;

# ВСТУП

Сьогодні у світі щосекунди обробляються та генеруються терабайти даних, особливо активно розвиваються такі напрямки, як штучний інтелект, системи навчання та обробки даних на основі нейронних мереж, військові технології, системи стратегічного планування, наукове моделювання та робота з об’ємними даними (Big Data). Усі ці сфери тією чи іншою мірою спираються на використання комп’ютерних систем.

Метою даної кваліфікаційної роботи є дослідження комп’ютерних систем із масивно-паралельною обробкою, зокрема системи Sunway TaihuLight.

Об’єкт дослідження — процес розробки відповідного програмного забезпечення.

Основні завдання роботи:

1. Провести аналіз MPP-системи Sunway TaihuLight.
2. Розробити алгоритм паралельної обробки векторів і створити програмне забезпечення для реалізації цього алгоритму.
3. Виконати аналіз та обчислення, що дозволяють оцінити продуктивність, швидкодію, відмовостійкість і надійність системи Sunway TaihuLight.

У межах кваліфікаційної роботи буде розглянуто базові терміни та поняття, пов’язані з ресурсами й архітектурою комп’ютерних систем масивно-паралельної обробки інформації. Також буде створено алгоритм та програму для паралельної обробки векторів, проведено відповідні обчислення для оцінки основних характеристик системи Sunway TaihuLight.

Отже, ця кваліфікаційна робота має дослідницький характер і включає розробку програмного забезпечення. Її основою стане теоретичний аналіз ресурсів комп’ютерних систем та принципів їх використання й розподілу під час експлуатації.

Програмна частина буде реалізована в середовищі Visual Studio Community мовою C#, блок-схеми та алгоритми — в Microsoft Visio, а тестування проводитиметься на персональному комп’ютері користувача.

Дану роботу можна вважати актуальною, оскільки MPP-системи активно застосовуються в різних галузях для досягнення практичних результатів.

# 

# 1. АНАЛІЗ КОМП’ЮТЕРНОЇ СИСТЕМИ З МАСИВНО-ПАРАЛЕЛЬНОЮ ОБРОБКОЮ ТИПУ SUNWAYTAIHULIGHT

## 1.1. Визначення поняття комп’ютерних систем

Сучасні комп’ютерні системи включають велику кількість різноманітних пристроїв, ефективне функціонування яких можливе лише за умови взаємодії з процесором, наявності доступу до оперативної пам’яті та можливості обміну даними з периферійними пристроями. Важливо також, щоб ці пристрої не конфліктували між собою, що забезпечується раціональним розподілом системних ресурсів. Загалом комп’ютерну систему можна охарактеризувати як інформаційно-технічний комплекс, основними функціями якого є обробка, зберігання, введення та виведення інформації. До її складу входять комп’ютери, сервери, принтери та інше обладнання разом із програмним забезпеченням. Обмін інформацією в такій системі відбувається через комп’ютерну мережу за допомогою локальних або глобальних каналів передачі даних.

Передусім потрібно з’ясувати, що саме входить до поняття системних ресурсів — це комунікаційні канали, сигнали та адреси, які використовуються окремими вузлами комп’ютера для обміну даними через шини. До основних системних ресурсів належать:

1. Адреси службових центрів пам’яті;
2. Канали запитів переривань (IRQ);
3. Канали прямого доступу до пам’яті (DMA);
4. Адреси портів введення/виведення.

Цей перелік упорядковано за ймовірністю виникнення конфліктів — найбільше проблем пов’язано саме з пам’яттю, і виявлення причин таких конфліктів часто є складним завданням.

Сьогодні зростає інтерес до створення паралельних багатопроцесорних обчислювальних систем (БОС), що базуються на використанні стандартних, загальнодоступних компонентів і технологій. Це обумовлено кількома чинниками. По-перше, відзначається стрімкий розвиток таких мережевих технологій, як Gigabit Ethernet (GE), Fibre Channel (FC) та InfiniBand (IB), що відповідає вимогам ринку до продуктивності. По-друге, важливу роль відіграє поширення безкоштовної операційної системи Linux. Спочатку вона розглядалася як UNIX-аналог для архітектури Intel, однак згодом з’явилися версії і для інших високопродуктивних мікропроцесорів, таких як Alpha. Враховуючи економічну ситуацію в країні, побудова систем на основі загальнодоступних рішень стає особливо доцільною, оскільки дозволяє створювати системи з різними конфігураціями залежно від конкретних завдань і бюджету.

Вузли кластера зазвичай з’єднуються за допомогою комутаторів Gigabit Ethernet або InfiniBand, при цьому їх кількість та конфігурація визначаються вимогами до обчислювальних потужностей та фінансовими можливостями користувача. В умовах конкуренції на ринку комплектуючих ціни змінюються досить швидко, особливо з появою нових моделей. Завдяки широкому вибору сучасної електроніки, можна оперативно створити високопродуктивні обчислювальні системи загального призначення, що задовольняють потреби найрізноманітніших користувачів.

Однією з сучасних форм організації є «блейд»-кластер — компактна та зручна у використанні система, яка лише незначно перевищує вартість звичайного багатопроцесорного комплексу. Її побудова полягає в об’єднанні «лез» в одну обчислювальну мережу через вмонтований комутатор. Для такої системи часто достатньо одного жорсткого диска з образом ОС, а завантаження здійснюється по мережі (Network Boot). Після вмикання головний вузол (Master) ініціалізує кластер, призначаючи IP-адреси всім вузлам — і система готова до роботи. Саме враховуючи ці технологічні досягнення, було розроблено модуль високоефективної багатопроцесорної системи з можливістю масштабування обчислювальних ресурсів.

## 1.2.Аналіз комп’ютерних багатопроцесорних систем, архітектура системи з масивно-паралельною обробкою MPP.

Ключовим поняттям, що лежить в основі побудови багатопроцесорних систем, є багатопроцесорність — це застосування двох або більше процесорів в одній комп’ютерній системі. Цей термін також охоплює здатність системи розподіляти навантаження між процесорами.

Один із перших значущих підходів до класифікації паралельних обчислювальних систем запропонував у 1966 році Майкл Флін із IBM. Його класифікація ґрунтується на аналізі потоків команд і даних між процесором і пам’яттю. Згідно з цим підходом, комп’ютерні системи поділяються на:

* **SISD** — одиничний потік команд, одиничний потік даних (Single Instruction Single Data);
* **MISD** — множинний потік команд, одиничний потік даних (Multiple Instruction Single Data);
* **SIMD** — одиничний потік команд, множинний потік даних (Single Instruction Multiple Data);
* **MIMD** — множинний потік команд, множинний потік даних (Multiple Instruction Multiple Data).

Архітектура комп’ютерної системи — це логічна структура, що визначає принципи обробки даних, взаємодію програмного забезпечення та технічних засобів. Архітектуру з масивно-паралельною обробкою (MPP), до якої належить система **Sunway TaihuLight**, класифікують за Фліном як **MIMD**.

Основна риса MPP-архітектури — фізично розподілена пам’ять. Такі системи складаються з окремих модулів, кожен із яких має процесор, локальну оперативну пам’ять, один або два комунікаційні процесори (роутери чи адаптери), а іноді й накопичувачі чи пристрої вводу/виводу. Один роутер відповідає за передавання команд, інший — за передавання даних. По суті, кожен модуль є самостійною обчислювальною одиницею.

Доступ до локальної пам’яті модуля мають тільки його власні процесори. Зв’язок між модулями забезпечується за допомогою спеціалізованих комунікаційних каналів. Користувач може задавати логічні адреси процесорів та здійснювати обмін повідомленнями між ними.

Існують два підходи до організації операційних систем у таких системах: у першому — повноцінна ОС функціонує лише на керуючому вузлі (front-end), а на інших модулях працює спрощена версія ОС, що забезпечує виконання паралельних процесів. У другому підході — на кожному модулі встановлюється окрема повнофункціональна UNIX-подібна операційна система.

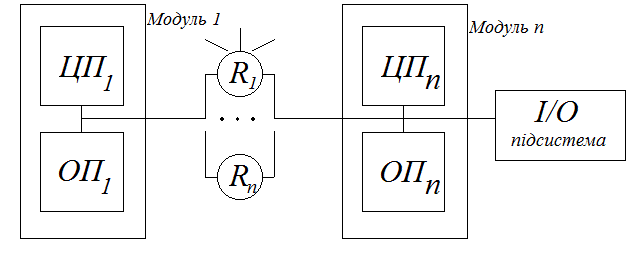


Рисунок 1.1 – Схематичний вид MPP-архітектури

Завдяки цьому підходу система уникає використання централізованої шини для обміну даними між процесорами, що характерно для симетричних багатопроцесорних (SMP) систем із єдиною спільною пам’яттю, а також позбавляється притаманних їм проблем масштабування.

Інструкції та дані можуть бути розміщені у локальних сегментах пам’яті кожного процесора, а для доступу одного процесора до пам’яті іншого, зазвичай, застосовується спеціальний механізм обміну повідомленнями (message passing). У таких системах спостерігається значна різниця у швидкості доступу до локальної та віддаленої пам’яті, тому конфігурація (топологія) зв’язків між обчислювальними модулями типу «процесор + локальна пам’ять» має вирішальне значення. Чим довший шлях між модулями, тим більше часу потрібно для обробки запитів, а відтак більша частина процесорного часу витрачається на обмін даними та міжпроцесорну взаємодію.

Повне з’єднання всіх вузлів між собою («кожен з кожним») використовується вкрай рідко через високу вартість і технічну складність при великій кількості модулів. Замість цього впроваджуються ефективніші топології, які забезпечують коротші шляхи між вузлами при мінімальних апаратних витратах — такі як топології тора, n-вимірні гіперкуби чи сітчасті структури. Однак високий рівень масштабованості таких систем має свою ціну: складність програмування. Розробнику доводиться вручну керувати розміщенням даних у локальній пам’яті та організовувати обмін інформацією між процесорами.

Серед ключових переваг архітектури MPP можна виокремити високу масштабованість завдяки розподіленій пам’яті та відсутність необхідності в синхронізації доступу до спільної пам’яті на кожному такті.

До недоліків MPP-архітектури належать: обмежений обсяг пам’яті, доступної кожному процесору, зниження швидкості міжпроцесорної взаємодії через відсутність спільної пам’яті, висока вартість спеціалізованого програмного забезпечення, а також складність реалізації механізмів обміну повідомленнями між процесорами.

## 1.3. Опис функціонування типового MPP блоку

Метою цього підрозділу є розкриття принципів функціонування модуля багатопроцесорної обчислювальної системи, побудованої на основі архітектури MPP. Така система має демонструвати максимальну ефективність і продуктивність під час розв’язання задач з високим рівнем взаємозв’язку, а також задач із масштабованими обчислювальними областями. Крім того, передбачається, що система повинна забезпечувати підвищену надійність і високу енергоефективність. Комплектування її блоків має здійснюватися з використанням компонентів обчислювальної техніки серійного виробництва. Структурна схема модуля БОС зображена на рисунку 1.2.

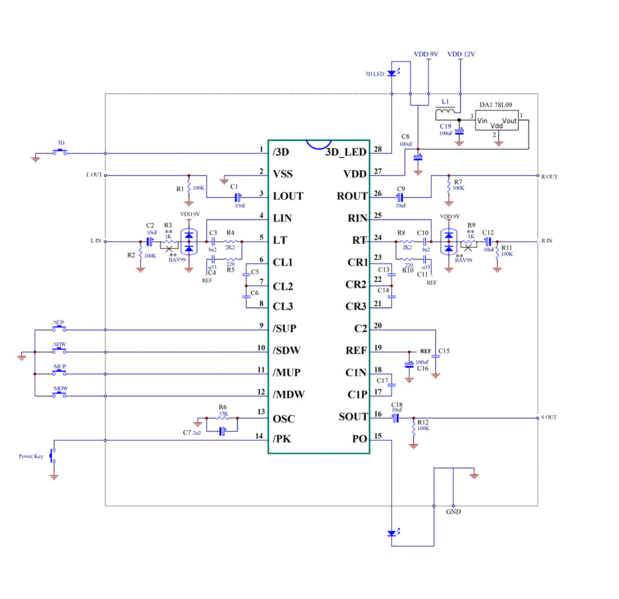


Рисунок 1.2 – схема MPPблоку паралельної обчислювальної системи

Модуль багатопроцесорної обчислювальної системи включає один головний (майстер) вузол (PM001) та N підпорядкованих обчислювальних вузлів (PN001, PN002, ..., PN00n), два керованих комутатори (SW1 та IB1), реконфігуровану мережу для обміну даними між обчислювальними елементами, систему локального зберігання проміжних і фінальних результатів (TCA Controller Storage System), а також механізм резервування критичних елементів. Завантаження вузлів здійснюється через мережу Gigabit Ethernet за допомогою комутатора SW1.

Усі вузли модуля — як головний, так і обчислювальні — побудовані на однакових апаратних компонентах: материнські плати, процесори, мережеві плати Gigabit Ethernet та зовнішні двопортові мережеві адаптери InfiniBand ConnectX-3 HCA (MCX354A-FCBT). Додатково головний вузол оснащений жорстким диском. Комутована мережа обчислювальної системи підтримує п’ять типів топологій: «зірка», «кільце», «лінія», «решітка» та «граф». Ці топології адаптовані до типових завдань, що характеризуються граничними обмінами даними. Налаштування і конфігурація комутатора IB1 здійснюються головним вузлом через два порти Gigabit Ethernet: один відповідає за керування (IB1GI.i1), інший — за масштабування (IB1GI.i2). Система зберігання даних TCA підключена до порту 16 комутатора IB1.

Вся система реалізована у вигляді єдиного корпусу, який слугує частиною обчислювальної шафи. Це забезпечує компактність, ефективне охолодження та зручний доступ до компонентів для обслуговування. Плати розташовані вертикально та паралельно одна одній, що відповідає концепції Blade-серверів.

Головний вузол (PM001) через комутатор SW1 виконує функції управління та діагностики. Підпорядковані вузли, згідно з алгоритмами задач, виконують відповідні обчислення. Дані між вузлами та умови задач передаються через окрему мережу, що працює на основі комутатора IB1. Для підвищення продуктивності використовується реконфігурована мережа з адаптерами InfiniBand (одно- або двопортовими), які адаптуються до характеру обчислювальних задач. Результати передаються на головний вузол через InfiniBand-комутатор.

Керування та передача даних від підпорядкованих вузлів здійснюється через HCA-адаптери, а зберігання — через адаптер TCA. Винесення обміну даними до окремої мережі на базі InfiniBand дозволило підвищити швидкодію та зменшити затримки в мережі. Завдяки архітектурі «точка-точка» у мережі, порівняно з шинною структурою, значно покращилася швидкість передачі даних. Всі передавання починаються й завершуються на HCA-адаптерах.

Використання режиму Quad Data Rate (QDR) у мережевих адаптерах MCX353A-FCBT із пропускною здатністю до 2 × 56 Гбіт/с, що відповідає FDR (Fourteen Data Rate), дало змогу узгодити обчислювальні та мережеві можливості системи через PCI Express (2 × 32 = 64 Гбіт/с). Принцип RDMA (Remote Direct Memory Access) дозволяє здійснювати передавання даних без участі ОС, буферизації та додаткових бібліотек, що знижує навантаження на процесори й підвищує ефективність.

Мережеве завантаження, резервування критичних компонентів і застосування індивідуальних блоків живлення для кожного леза забезпечує надійність модуля. Електроживлення подається через єдиний UPS, підключений до керуючого модуля П01, від якого через мережеві розгалужувачі струм надходить до живлення головного вузла та всіх підлеглих елементів. Це забезпечує стабільну напругу, оптимальне навантаження та зменшене енергоспоживання. Єдина архітектура енергопостачання спрощує систему живлення та знижує її вартість, дозволяючи обійтись без спеціалізованих систем охолодження.

Уніфікація компонентів, резервування живлення та винесення обміну даними в окрему InfiniBand-мережу значно підвищили продуктивність і доступність всієї системи. Застосування RDMA додатково зменшило латентність, а двопортові HCA-адаптери з агрегацією каналів дали можливість динамічної зміни мережевої конфігурації для досягнення максимальної пропускної здатності відповідно до характеру задач.

Таким чином, завдяки мережевому завантаженню, резервуванню ключових елементів, реконфігурованій мережі та оптимізованому енергозабезпеченню, досягнуто високої ефективності, гнучкості та надійності функціонування модуля багатопроцесорної системи.

## 1.4Аналіз архітектури комп’ютерної системи SunwayTaihuLight

SunwayTaihuLight — китайський суперкомп'ютер, котрий станом на листопаж 2018 року є третім у списку найпродуктивніших суперкомп'ютерів (рис. 1.3) у світі зі швидкістю обчислень 93 петафлопс згідно LINPACKbenchmarks.



Рисунок 1.3 – приміщення SunwayTaihuLight

Таблиця 1.1 – Характеристика SunwayTaihuLight

|  |  |
| --- | --- |
| Характеристика | Специфікація |
| Виробник | NRCPC(Sunway SPC) |
| Кіл-сть ядер | 10 649 600 |
| Обсяг пам'яті | 1310720 Гб |
| Пікова продуктивність | 93014,6 ПФлопс |
| Теоретично досягаєма продуктивність | 125436 ПФлопс |
| Потужність | 15371 кВ |
| Операційна система | Sunway RaiseOS 2.0.5 |

Цей суперкомп'ютер складається із понад 10,5 мільйонів процесорних ядер і працює під керуванням власної операційної системи SunwayRaiseOS 2.0.5 на основі Linux. У суперкомп'ютері використані багатоядерні 64-бітні RISC-процесори SW26010 архітектури ShenWei. Загальна кількість процесорів у системі — 40 960, кожен процесор містить 256 ядер загального призначення і 4 допоміжних ядра для керування, що загалом дає 10 649 600 ядер. Суперкомп'ютер SunwayTaihuLight призначений для складних розрахунків у виробництві, медицині, видобувній промисловості, для прогнозування погодних умов і аналізу «великих масивів даних».

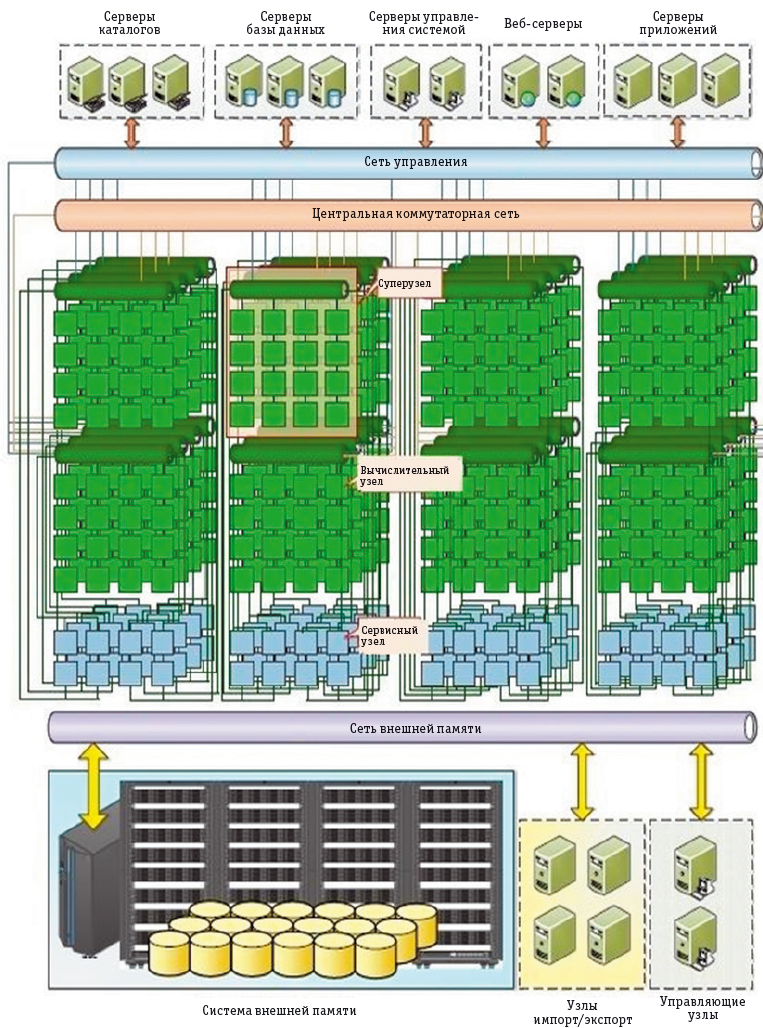


Рисунок 1.4 – Схематичне зображення структури SunwayTaihuLight

### 1.4.1 Огляд процесоруSW26010 на базі архітектуриShenWei

У процесорі комп’ютерної системи Sunway TaihuLight реалізовано 64-розрядну мікроархітектуру типу RISC, відому як ShenWei.  
Процесор SW26010, який містить 260 ядер (див. рис. 1.5), побудований із чотирьох кластерів, кожен з яких включає 64 обчислювально-процесорні елементи (CPE). Ці елементи організовані у вигляді матриці 8 × 8 (див. рис. 1.6).

CPE призначені для виконання SIMD-інструкцій (одна інструкція — багато даних) і здатні за один такт обробляти до 8 операцій з числами з плаваючою комою одинарної точності.  
Кожен кластер має у своєму складі окреме ядро загального призначення — так званий керуючий процесорний елемент, який відповідає за керування обчисленнями та моніторинг.

Кожен кластер також оснащено власним контролером пам’яті типу DDR3 SDRAM і незалежним банком пам’яті з окремим адресним простором.  
Процесор функціонує на тактовій частоті 1,45 ГГц.

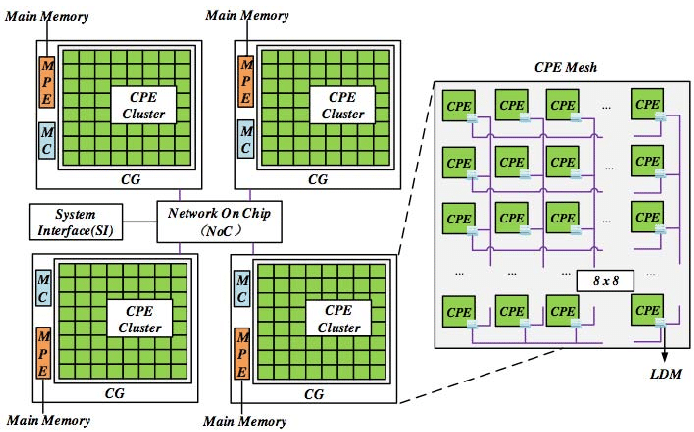


Рисунок 1.5 – будова процесора SW26010

Кожен обчислювальний процесорний елемент (CPE) оснащений вбудованою надоперативною пам’яттю обсягом 64 КБ для зберігання даних та 16 КБ для інструкцій. Ці елементи з’єднані між собою за допомогою мережі-на-кристалі (Network-on-Chip), яка замінює традиційну ієрархічну структуру кеш-пам’яті.

Керуючий процесорний елемент (MPE) використовує класичну кешову архітектуру, яка включає 32 КБ кешу першого рівня (L1) для даних та інструкцій, а також 256 КБ кешу другого рівня (L2).

Мережа-на-кристалі підключена до єдиного внутрішньосистемного інтерфейсу, що забезпечує взаємодію процесора з зовнішніми пристроями.



Рисунок 1.6 – зовнішній вигляд процесора SW26010

## 1.5Висновок

У цьому розділі було здійснено огляд паралельних багатопроцесорних комп’ютерних систем, визначено поняття системних ресурсів, проаналізовано будову та принцип роботи типового модуля архітектури MPP, а також розглянуто технічні особливості багатопроцесорної системи, побудованої на масивно-паралельній архітектурі Sunway TaihuLight.

# 

# 2. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПАРАЛЕЛЬНОГО СОРТУВАННЯ ЕЛЕМЕНТІВ МАСИВУ

## 2.1. Постановка задачі

В даному розділі необхідно розробити алгоритм та програмне забезпечення для сортування елементів масиву, серед основних вимог до даного ПЗ можна відмітити можливість регулювання кіл-сті потоків обробки, елементів масиву та стан заповнення масиву, а також додати можливість переглядати час виконання задачі.

## 2.2. Розробка алгоритмуфункціонування програмного забезпечення

Алгоритм функціонування програми з декількох етапів особливо важливими можна виділити наступні:

1. Ввід кількості елементів та кількості потоків обробки масиву;
2. Розбиття масиву на підмасиви при паралельній обробці;
3. Синхронізація отриманих даних в основному потоці.

Алгоритм роботи паралельного сортування елементів масиву має наступний вигляд (рисунок 2.1), також наведено алгоритм функціонування ПЗ (рисунок 2.2):



Рисунок 2.1 – Блок-схема алгоритмупаралельного сортування масиву



Рисунок 2.2 – Блок-схема алгоритмуфункціонування програмного забезпечення паралельного сортування масива

## 2.3. Розробка програмного забезпечення для паралельного сортування масиву

Програмне забезпечення, що реалізує паралельне сортування масиву, складається з двох основних функціональних частин. Перша з них відповідає за генерацію масиву, ініціалізацію змінних і запуск потоків. Друга частина містить безпосередньо алгоритм сортування, який виконується паралельно, відповідно до заданої кількості потоків.

Нижче представлено функцію, що виконує ініціалізацію змінних, а також реалізацію алгоритму сортування масиву, який полягає у перестановці пар елементів у разі, якщо перший елемент виявляється більшим за другий. Сортування здійснюється у порядку зростання значень елементів.

Для забезпечення роботи з введенням, виведенням та обробкою інформації використовуються директиви System та System.Diagnostics. Як показано в коді, залежно від заданої кількості потоків, елементи масиву рівномірно розподіляються між потоками та передаються у тимчасові змінні для подальшої обробки.

using System;

using System.Diagnostics;

class Program

{

static int n;

static int[] nums;

static int P;

public static void sort\_massiv(object param)

{

int nomerPotoka = (int)param;

int chunk = n / P;

int start = nomerPotoka \* chunk;

int temp;

for (int i = start; i < start + chunk; i++)

{

for (int j = i + 1; j < start + chunk; j++)

{

if (nums[i] > nums[j])

{

temp = nums[i];

nums[i] = nums[j];

nums[j] = temp;

}}}}

У цій частині програми реалізовано взаємодію з користувачем: передбачено введення кількості елементів масиву, кількості потоків, а також вказівку мінімального та максимального значень для генерації елементів масиву.

Для вимірювання часу виконання сортування використовується метод DateTime, який фіксує момент запуску програми. За допомогою меток екземпляра класу Stopwatch (методи Start і Stop) визначається проміжок часу від початку ініціалізації потоків і старту сортування до моменту завершення всього процесу.

Запуск потоків здійснюється методом Thread.Start, а для синхронізації з основним потоком після завершення сортування окремих частин масиву використовується метод Thread.Join.

static void Main(string[] args)

{

Console.WriteLine("Введiтькiлькiстьелементiвмасиву: ");

n = int.Parse(Console.ReadLine());

var random = new Random();

nums = new int[n];

Console.WriteLine("Введiтьмiнiмальнезначеннядiапазонузаповненнямасиву: ");

int m = int.Parse(Console.ReadLine());

Console.WriteLine("Введiтьмiнiмальнезначеннядiапазонузаповненнямасиву: ");

int l = int.Parse(Console.ReadLine());

for (var i = 0; i < nums.Length; i++)

nums[i] = random.Next(m, l);

Console.WriteLine("Введiтьчислопотокiв: ");

P = int.Parse(Console.ReadLine());

DateTime time = DateTime.Now;

var stopWatch = Stopwatch.StartNew();

System.Threading.Thread[] threads = new System.Threading.Thread[P];

for (int i = 0; i < P; i++)

{

threads[i] = new System.Threading.Thread(new System.Threading.ParameterizedThreadStart(sort\_massiv));

threads[i].Start(i);

}

for (int i = 0; i < P; i++)

{

threads[i].Join();

}

for (int i = 0; i < P; i++)

{

int nomerPotoka = i;

int chunk = n / P;

int start = nomerPotoka \* chunk;

Console.WriteLine(nomerPotoka.ToString() + "Вiдсортованоряд - поток " + nomerPotoka + " :");

for (int j = start; j < start + chunk; j++)

{

Console.Write(nums[j] + " ");

}

Console.WriteLine();

}

int temp;

for (int i = 0; i < nums.Length; i++)

{

for (int j = i + 1; j < nums.Length; j++)

{

if (nums[i] > nums[j])

{

temp = nums[i];

nums[i] = nums[j];

nums[j] = temp;

} } }

Console.WriteLine("Загальниймасив:");

for (int j = 0; j < nums.Length; j++)

{

Console.Write(nums[j] + " ");

}

Console.WriteLine();

stopWatch.Stop();

Console.WriteLine("Часвиконання: {0} ms", stopWatch.ElapsedMilliseconds);

Console.ReadLine();

}}

Оскільки розроблений алгоритм сортування ґрунтується на ітераційних операціях, досягти повного паралелізму виконання практично неможливо. Після сортування окремих частин масиву, їх об’єднання та подальша обробка виконуються вже в одному потоці. Для багатопроцесорних систем доцільніше застосовувати алгоритми швидкого сортування, однак у випадках, коли кількість обчислювальних елементів невелика, такі методи не забезпечують суттєвого приросту продуктивності. З огляду на це, для навчальних цілей було реалізовано простий ітераційний алгоритм сортування.

## 2.4. Висновок

У цьому розділі розроблено програмне забезпечення для сортування масиву, засноване на ітераційному методі порівняння. Представлено алгоритм роботи програмного забезпечення та алгоритм сортування. Програма дозволяє користувачеві задавати кількість потоків, розмір масиву та діапазон значень його елементів. Також реалізовано функціональність для фіксації часу виконання, що дає змогу проводити експериментальні дослідження та аналіз продуктивності.

Було створено програму, детально описано основні етапи її розробки й налаштування. Повний програмний код наведено у додатку А.

# 

# 3. ТЕСТУВАННЯ ПЗ ДЛЯ СОРТУВАННЯ ЕЛЕМЕНТІВ МАСИВУ

## 3.1. Тестування розробленого програмного забезпечення

Тестування буде проводитись шляхом запуску програми та сортування масивів з різними параметрами. У ході даного розділу буде проведено ряд експериментів зі зміною таких параметрів:

1. Зі зміною числа потоків;
2. Зі зміною кількості елементів масиву;
3. Зі зміною діапазону заповнення масиву.

Результат функціонування програмного забезпечення представлено на рисунку 3.1.

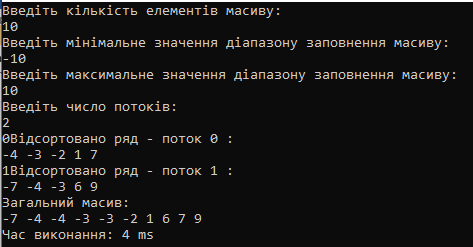


Рисунок 3.1 – Загальний вигляд програми та результат її функціонування

Експеримент №1 - Зі зміною числа потоків, в ході проведення даного експеримента буде встановлено залежність часу виконання від числа потоків, кількість елементів масиву обрано 10000, діапазон заповнення масиву цілочісельні значення від 1 до 100000, Результати виконання програми представлено на рисунках 3.2 – 3.6, усі дані отримані у ході експерименту занесено до таблиці 3.1

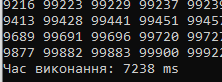


Рисунок 3.2 – Результат експерименту №1 з числом потоків рівному 1

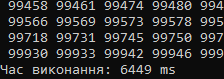


Рисунок 3.3 – Результат експерименту №1 з числом потоків рівному 2

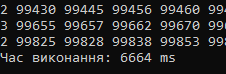


Рисунок 3.4 – Результат експерименту №1 з числом потоків рівному 4

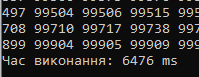


Рисунок 3.5 – Результат експерименту №1 з числом потоків рівному 8

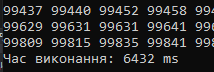


Рисунок 3.6 – Результат експерименту №1 з числом потоків рівному 16

Таблиця 3.1 – Результати та параметри експерименту №1

|  |  |  |  |
| --- | --- | --- | --- |
| Число потоків, N | Кількість елементів масиву, n | Діапазон заповнення масиву, f | Час виконання, мс |
| 1 | 10000 | 1-100000 | 7238 мс |
| 2 | 10000 | 1-100000 | 6449 мс |
| 4 | 10000 | 1-100000 | 6664 мс |
| 8 | 10000 | 1-100000 | 6476 мс |
| 16 | 10000 | 1-100000 | 6432 мс |

На рисунку 3.7 представлено часову діаграму залежності часу виконання сортування від числа потоків.

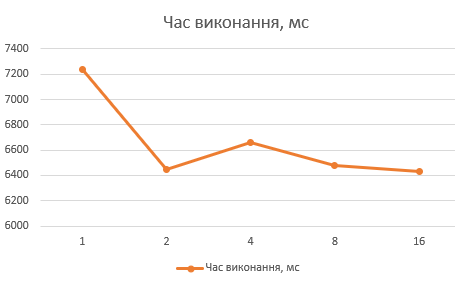


Рисунок 3.7 -часова діаграма залежності часу виконання сортування від числа потоків.

Експеримент №2 - Зі зміною кількості елементів масиву, в ході проведення даного експеримента буде встановлено залежність часу виконання від кількості елементів масиву, число потоків складає 16 на основі найкращого результату попереднього експерименту, діапазон заповнення масиву цілочісельні значення від 1 до 100000, Результати виконання програми представлено на рисунках 3.8 – 3.12, усі дані отримані у ході експерименту занесено до таблиці 3.2.

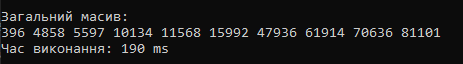


Рисунок 3.8 – Результат експерименту №2 з кількістю елементів масиву рівним 10



Рисунок 3.9 – Результат експерименту №2 з кількістю елементів масиву рівним 100

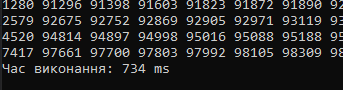


Рисунок 3.10 – Результат експерименту №2 з кількістю елементів масиву рівним 1000

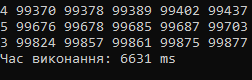


Рисунок 3.11 – Результат експерименту №2 з кількістю елементів масиву рівним 10000

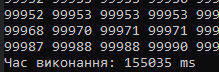


Рисунок 3.12 – Результат експерименту №2 з кількістю елементів масиву рівним 100000

Таблиця 3.2 – Результати та параметри експерименту №2

|  |  |  |  |
| --- | --- | --- | --- |
| Число потоків, N | Кількість елементів масиву, n | Діапазон заповнення масиву, f | Час виконання, мс |
| 16 | 10 | 1-100000 | 190 мс |
| 16 | 100 | 1-100000 | 230 мс |
| 16 | 1000 | 1-100000 | 734 мс |
| 16 | 10000 | 1-100000 | 6631 мс |
| 16 | 100000 | 1-100000 | 155035 мс |

На рисунку 3.13 представлено часову діаграму залежності часу виконання сортування від кількості елементів масиву.



Рисунок 3.13 -часова діаграма залежності часу виконання сортування від кількості елементів масиву

Експеримент №3- Зі зміною діапазону заповнення масиву, в ході проведення даного експеримента буде встановлено залежність часу виконання від діапазону заповнення масиву, число потоків складає 16, кількість елементів масивускладає 10000 елементів, Результати виконання програми представлено на рисунках 3.14 – 3.18, усі дані отримані у ході експерименту занесено до таблиці 3.3.



Рисунок 3.14 – Результат експерименту №3 з діапазоном заповнення масиву від 1 до 10

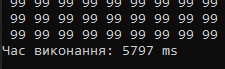


Рисунок 3.15 – Результат експерименту №3 з діапазоном заповнення масиву від 1 до 100

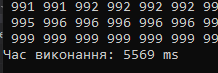


Рисунок 3.16 – Результат експерименту №3 з діапазоном заповнення масиву від 1 до 1000

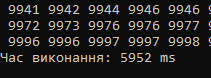


Рисунок 3.17 – Результат експерименту №3 з діапазоном заповнення масиву від 1 до 10000



Рисунок 3.18 – Результат експерименту №3 з діапазоном заповнення масиву від 1 до 100000

Таблиця 3.3 – Результати та параметри експерименту №3

|  |  |  |  |
| --- | --- | --- | --- |
| Число потоків, N | Кількість елементів масиву, n | Діапазон заповнення масиву, f | Час виконання, мс |
| 16 | 10000 | 1-10 | 4163 мс |
| 16 | 10000 | 1-100 | 5797 мс |
| 16 | 10000 | 1-1000 | 5569 мс |
| 16 | 10000 | 1-10000 | 5952 мс |
| 16 | 10000 | 1-100000 | 6484 мс |

На рисунку 3.18 представлено часову діаграму залежності часу виконання сортування від діапазону заповнення масиву.



Рисунок 3.18 -часова діаграма залежності часу виконання сортування віддіапазону заповнення масиву

## 3.2. Висновок

У цьому розділі було проведено тестування розробленого програмного забезпечення. У ході серії експериментів було встановлено залежність часу виконання програми від кількості елементів масиву, кількості потоків та обраного діапазону значень для заповнення масиву. На основі отриманих результатів можна сформулювати наступні висновки та припущення:

* Збільшення кількості потоків понад 8 може викликати надмірне навантаження та, як наслідок, призвести до зростання часу виконання програми як при малих, так і при великих обсягах даних. Це узгоджується із законом Амдала, який описує обмеження ефективності паралельних обчислень. У цьому випадку, незважаючи на розподіл обчислень між потоками, значна частина інструкцій виконується послідовно, що знижує загальну продуктивність;
* Ітераційна природа алгоритму сортування обмежує ефективність паралельного виконання, оскільки фінальний результат залежить від синхронізації потоків після завершення сортування окремих підмасивів;
* Діапазон значень у масиві також впливає на швидкодію: при повторенні значень, особливо у випадку дво- або трицифрових чисел, а також при роботі з великими числами, спостерігається зниження продуктивності на 500–1000 мс. Це пояснюється більшою складністю операцій порівняння та повільнішим зчитуванням великих чисел із пам’яті;
* Збільшення кількості елементів масиву призводить до прямопропорційного зростання часу виконання сортування, що є очікуваним для обраного підходу.

# 

# ВИСНОВКИ

У першому розділі було проведено огляд багатопроцесорних паралельних комп’ютерних систем, визначено термін системних ресурсів та проведено огляд структури та функціонування типового MPP-блоку, розглянуто характеристики багатопроцесорної комп’ютерної системи на базі масово-паралельної архітектури SunwayTaihuLight.

У другому розділі розроблено програмне забезпечення для сортування елементів масиву на основі ітераційного методу порівняння, наведено алгоритм функціонування програмного забезпечення та алгоритм сортування. Розроблено програмне забезпечення, описано основні етапи розробки та підготовки.

У третьому розділі було виконано тестування створеного програмного забезпечення, проведено ряд експериментів в ході яких було встановлено залежності часу виконання програми від кількості елементів масиву, числа потоків та діапазону заповнення масиву.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. МетодConsole.WriteMethod[Електронний ресурс] – Режим доступу:

<https://docs.microsoft.com/en-us/dotnet/api/system.console.write?view=netframework-4.7>.2

1. КласConsoleClass[Електронний ресурс] – Режим доступу:

<https://docs.microsoft.com/en-us/dotnet/api/system.console?view=netframework-4.7.2>

1. Клас ThreadClass[Електронний ресурс] – Режим доступу:

<https://docs.microsoft.com/en>us/dotnet/api/system.threading.thread?view=netframework-4.7.2

1. Офіційний сайт SunwayTaihulight[Електронний ресурс] – Режим доступу:<http://demo.wxmax.cn/wxc/index.php>
2. Сайт TOP500 ThelistNovember 2018[Електронний ресурс] – Режим доступу:<https://www.top500.org/lists/2018/11/>
3. Моделювання та комп’ютерна графіка : Матеріали 5-ї міжнародної науково - технічної конференції , м Донецьк , 24-27 вересня 2013 р . — Донецьк, ДонНТУ, Міністерство освіти та науки України , 2013. — 423 с.
4. Левитин А. В. Глава 3. Метод грубой силы: Сортировка выбором // Алгоритмы. Введение в разработку и анализ — М.: Вильямс, 2006. — С. 143–144. — 576 с.

# ДОДАТКИ

ДОДАТОК А

using System;

using System.Diagnostics;

classProgram

{

staticint n;

staticint[] nums;

staticint P;

publicstaticvoid sort\_massiv(object param)

{

int nomerPotoka = (int)param;

int chunk = n / P;

int start = nomerPotoka \* chunk;

int temp;

for (int i = start; i < start + chunk; i++)

{

for (int j = i + 1; j < start + chunk; j++)

{

if (nums[i] > nums[j])

{

temp = nums[i];

nums[i] = nums[j];

nums[j] = temp;

}

}

}

}

// Основнаяфункция

staticvoid Main(string[] args)

{

Console.WriteLine("Введiтькiлькiстьелементiвмасиву: ");

n = int.Parse(Console.ReadLine());

var random = new Random();

nums = newint[n];

Console.WriteLine("Введiтьмiнiмальнезначеннядiапазонузаповненнямасиву: ");

int m = int.Parse(Console.ReadLine());

Console.WriteLine("Введiтьмаксимальнезначеннядiапазонузаповненнямасиву: ");

int l = int.Parse(Console.ReadLine());

for (var i = 0; i < nums.Length; i++)

nums[i] = random.Next(m, l);

Console.WriteLine("Введiтьчислопотокiв: ");

P = int.Parse(Console.ReadLine());

DateTime time = DateTime.Now;

var stopWatch = Stopwatch.StartNew();

System.Threading.Thread[] threads = new System.Threading.Thread[P];

for (int i = 0; i < P; i++)

{

threads[i] = new System.Threading.Thread(new System.Threading.ParameterizedThreadStart(sort\_massiv));

threads[i].Start(i);

}

for (int i = 0; i < P; i++)

{

threads[i].Join();

}

for (int i = 0; i < P; i++)

{

int nomerPotoka = i;

int chunk = n / P;

int start = nomerPotoka \* chunk;

Console.WriteLine(nomerPotoka.ToString() + "Вiдсортованоряд - поток " + nomerPotoka + " :");

for (int j = start; j < start + chunk; j++)

{

Console.Write(nums[j] + " ");

}

Console.WriteLine();

}

int temp;

for (int i = 0; i < nums.Length; i++)

{

for (int j = i + 1; j < nums.Length; j++)

{

if (nums[i] > nums[j])

{

temp = nums[i];

nums[i] = nums[j];

nums[j] = temp;

}

}

}

Console.WriteLine("Загальниймасив:");

for (int j = 0; j < nums.Length; j++)

{

Console.Write(nums[j] + " ");

}

Console.WriteLine();

stopWatch.Stop();

Console.WriteLine("Часвиконання: {0} ms", stopWatch.ElapsedMilliseconds);

Console.ReadLine();

}

}