Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»
Кафедра комп’ютерних наук та інформаційних систем

Матеріали для самостійного вивчення дисципліни «Паралельні та розподілені обчислення» для студентів спеціальності 122 «Комп’ютерні науки та інформаційні технології»

Івано-Франківськ – 2020

Матеріали для самостійного вивчення дисципліни «Паралельні та розподілені обчислення» для студентів спеціальності 122 «Комп’ютерні науки та інформаційні технології» – 2020. – 18 с.

Розробник:

Ізмайлов Артем Вікторович, магістр, асистент кафедри комп’ютерних наук та інформаційних систем.

ВСТУП

Багатопотоковість – це властивість платформи або програми, яка полягає в тому, що процес, породжений операційною системою, може складатися із декількох потоків, які здатні виконуватися «паралельно».

При вирішенні деякого роду задач такий підхід дозволяє суттєво підвищити ефективність використання обчислювальних ресурсів. Ефективне використання обчислювальних ресурсів забезпечує позитивний економічний ефект при використанні відповідного програмного забезпечення.

Навчальна дисципліна «Паралельні та розподілені обчислення» є однією із центральних дисциплін, яка формує навички практичної та наукової діяльності бакалавра напряму підготовки «комп'ютерні науки». При вивченні цієї навчальної дисципліни використовуються поняття і методи функціонування операційних систем, теорії черг, оптимізації виконання операцій та архітектури обчислювальних систем. Дисципліна забезпечує практичне застосування знань, набутих при вивченні інших, у тому числі фундаментальних, дисциплін.

Мета викладання навчальної дисципліни полягає в:

- розкритті актуальних аспектів проведення високоефективних обчислень шляхом застосування алгоритмів багатопотокового, паралельного та розподіленого програмування;
- формуванні у студентів цілісної системи знань та вмінь з метою використання ними сучасних інформаційних технологій при розробці систем реального часу та систем математичної обробки інформаційних масивів;
- активному закріпленні, узагальненні та поглибленні знань, отриманих при вивченні базових загальнонаукових і загально технічних дисциплін;
- ознайомленні з основними методами імплементації багатопотоковості у прикладному програмному забезпеченні.

Завдання викладання навчальної дисципліни: ознайомити студентів із основними принципами та методами розпаралелювання та розподілення обчислень у прикладному програмному забезпеченні та спеціалізованих

системах; оволодінні навичками розв’язання практичних задач розпаралелювання та розподілення обчислень; навчити студентів здійснювати вибір типу обчислювального процесу (розподіленого, багатопотокового чи розпаралеленого), а також методології його реалізації.

У результаті вивчення навчальної дисципліни студент повинен

знати:

- основні концепції багатопотокового, паралельного та розподіленого програмування;
- методи програмування із розподіленими змінними;
- теорію розподіленого програмування;
- методи синхронного паралельного програмування.

вміти:

- здійснювати вибір методів реалізації високопродуктивних обчислень;
- будувати багатопоточні, паралельні та розподілені системи;
- використовувати блокування та бар’єри;
- використовувати м’ютекси та умовні змінні;
- проектувати систему взаємодії елементів систем розпаралелених обчислень;
- будувати моделі взаємодії процесів.

ЗМІСТ ЛЕКЦІЙНОГО КУРСУ

Тема 1. Загальні відомості з мультипроцесорних обчислювальних систем.

1) Області застосування і задачі паралельної обробки.
2) Мультипроцесорні обчислювальні системи.
3) Симетричні мультипроцесорні системи (SMP).
4) Системи з масовою паралельною обробкою (MPP).
5) Кластерні обчислювальні системи

Тема 2. Класифікації паралельних та розподілених систем.

1) Класифікація за типом апаратного з'єднання.
2) Класифікація за множинністю потоків команд і даних.
3) Класифікація за методами доступу до пам'яті.
4) Класифікація за структурою мережі зв'язків.
5) Класифікація за зв'язністю програмного забезпечення.
6) Класифікація за типом програмних зв'язків.

Тема 3. Операційні системи та їх розподіленість.

1) Види розподілених операційних систем.
2) Побудова розподілених операційних систем.
3) Програмне забезпечення проміжного рівня.
4) Прискорення при паралельних обчисленнях.

Тема 4. Реалізація багатопотоковості на рівні програмного забезпечення.

1) Управління потоками.
2) Розділення даних між потоками.
3) Синхронізація паралельних операцій.

Тема 5. Паралельні методи множення матриці на вектор.

1) Постановка задачі.
2) Метод смугового розбиття матриці по рядках.
3) Метод смугового розбиття матриці по стовпцях.
4) Метод розбиття матриці по блоках.

ЗМІСТ ТЕМ ЛАБОРАТОРНИХ ЗАНЯТЬ КУРСУ

Лабораторне заняття 1

Застосування м’ютексів у паралельних обчисленнях

***Мета роботи*:** Навчитись застосовувати м’ютекси для забезпечення ізолювання доступу потоків до розділюваних даних.

Хід роботи

1. Напишіть програму, яка для двох функцій: Thread1 – виводить у консольне вікно число 1 та Thread2 – виводить у консольне вікно число 2, створює окремі потоки після чого завершує свою роботу. Запустіть програму декілька разів, запишіть та проаналізуйте результати.
2. Для програми з пункту 1 застосуйте для кожного з потоків метод detach. Запустіть програму декілька разів, запишіть та проаналізуйте результати.
3. Напишіть програму, у якій здійснюється запуск потоків AddToList та ListContains для глобального списку *l*. У першому потоці здійснюється додавання елементу до кінця списку зі значенням, заданим у функції main, після чого, здійснюється додавання ще дев’яти елементів, кожен з яких на одиницю більший від попереднього. Після успішного додавання елемента, потік виводить відповідне повідомлення у консоль. У другому потоці десять разів здійснюється перевірка на входження елемента зі значенням, заданим у функції main, і результат виконання кожної спроби (входить / не входить) виводиться у консоль. Запустіть програму декілька разів, проаналізуйте результати, зробіть висновки.
4. За допомогою м’ютексів забезпечте інваріантний доступ потоків AddToList та ListContains з попереднього завдання до списку *l*. Запустіть програму декілька разів, проаналізуйте результати, зробіть висновки.
5. Запишіть лістинги розроблених програм у звіт. Зробіть висновки.

Контрольні запитання

1. Як забезпечується розпаралелювання задач у операційних системах?
2. Яким чином забезпечене управління потоками у мові C++?
3. Що таке м’ютекс та які основні принципи його роботи?
4. Яким чином створюється м’ютекс засобами мови C++?

Лабораторне заняття 2

Застосування шаблонів стандартної бібліотеки на основі м’ютексів для ізолювання доступу до даних

***Мета роботи*:** Навчитись застосовувати шаблони стандартної бібліотеки на основі м’ютексів для ізолювання доступу потоків до розділюваних даних.

Хід роботи

1. Внесіть наступні зміни до програми з пункту 4 попередньої лабораторної роботи:
   – Функції AddToList та ListContains повинні виконувати свої операції (додавання та перевірку) лише один раз;
   – У основному потоці програми (main) забезпечте запуск кожного з потоків 10 разів з наступним їх від’єднанням, за умови, що до списку додається кожного разу число більше на одиницю за попереднє, а перевірка на входження здійснюється кожного разу для однакового числа;
   – Замініть пряме застосування м’ютексів на застосування шаблону стандартної бібліотеки lock_guard.

   Запустіть програму декілька разів, проаналізуйте результати, зробіть висновки. Зробіть висновки стосовно різниці між прямим застосуванням м’ютексів та застосуванням шаблону lock_guard.
2. Створіть клас someData з полями рядкового типу «ім’я», «прізвище», «адреса» та полем цілочислового типу «вік». Створіть клас exchangePerson, полями якого є поле типу «someData» та м’ютекс. Для другого класу визначте статичні методи JohnDoe та JacobSmith, які змінюють інформацію у об’єктах someData (які є полями exchangePerson) відповідним чином (таблиця 1):

Таблиця 1 – Зміна інформації у об’єктах someData

Поле	JohnDoe	JacobSmith
ім’я	John	Jacob
прізвище	Doe	Smith
адреса	Unknown	Known
вік	120	1

Для класу exchangePerson визначте статичний метод Swap, який обмінює значення полів someData двох об’єктів класу exchangePerson, попередньо виключивши можливість роботи з об’єктами за однаковим посиланням. На стандартний пристрій вводу вивести дані про значення полів до і після обміну.

У головному потоці програми створити два об’єкти класу exchangePerson і запустити для кожного з них методи JohnDoe та JacobSmith у окремих потоках, які одразу ж від’єднуються від головного. Запустити у окремому потоці функцію Swap для двох створених об’єктів і гарантувати отримання результатів роботи цього потоку до завершення головного.

За допомогою шаблону std::lock_guard гарантувати ізольованість доступу до даних у методах JohnDoe та JacobSmith. У методі Swap за допомогою функції std::lock та шаблону std::lock_guard з параметром std::adopt_lock забезпечити захист від взаємного блокування під час здійснення операції обміну.

3. Змініть код програми з попереднього пункту замінивши застосування std::lock_guard і std::adopt_lock у функції Swap() застосуванням std::unique_lock і std::defer_lock. Зробіть висновки стосовно різниці між програмами та застосуванням шаблонів std::lock_guard та std::unique_lock.
4. Запишіть лістинги розроблених програм у звіт. Зробіть висновки загальні висновки щодо шаблонів стандартної бібліотеки на основі м’ютексів та способів уникнення взаємного блокування.

Контрольні запитання

1. Як забезпечується розпаралелювання задач у операційних системах?
2. Яким чином забезпечене управління потоками у мові C++?

3. Що таке м’ютекс та які основні принципи його роботи?
4. Яким чином створюється м’ютекс засобами мови C++?

Лабораторне заняття 3

Синхронізація паралельних операцій за допомогою механізму подій

***Мета роботи*:** Навчитись синхронізувати паралельні операції за допомогою механізму подій.

Хід роботи

1. Створити програму, у якій запускаються два потоки DataPreparation (від'єднується від головного потоку) та DataProcessing (головний потік очікує на його завершення). У потоці DataPreparation відбувається зчитування цілих чисел від користувача (через стандартний пристрій вводу), які поміщаються у глобальну чергу. У цей час потік DataProcessing очікує на завершення введення даних. Після завершення введення користувачем даних (передбачити можливість уведення довільного числа даних) потік DataPreparation встановлює прапорець для запуску потоку DataProcessing, який перевіряє усі числа у черзі на простоту та виводить прості.

   Для синхронізації потоків використати глобальний прапорець, об'єкт std::unique_lock та this_thread::sleep_for(chrono::milliseconds(100)).
2. Створити програму, яка запускає три потоки на основі функції Waits(), яка очікує на сигнал від потоку Awake() і перед початком очікування виводить повідомлення про те, що вона входить у стан очікування на стандартний пристрій вводу. Після отримання сигналу, виводиться повідомлення про те, що очікування завершене. У якості умови для функції wait() використати лямбда-функцію, яка перевіряє рівність значення глобальної змінної і одиниці (змінну і проініціалізувати нулем). Потік Awake очікує (спить) 100 мілісекунд, після чого повідомляє про те, що починає повідомляти решту (через стандартний пристрій вводу) і робить це за допомогою notify_all(), після чого переходить у «сон» на 100 мілісекунд. Далі (і лише зараз) він присвоює змінній і значення 1, видає повідомлення на стандартний пристрій вводу про повторне повідомлення і повторно викликає notify_all().

   Головний потік повинен дочекатись виконання потоку Awake.

Запустіть програму декілька разів, запишіть та проаналізуйте результати.

3. Створіть програму, яка створює три потоки Thread1, Thread2 та Thread3, кожен з яких очікує на сигнал аналогічним до попереднього завдання чином, і після сигналу виводить на стандартний пристрій вводу повідомлення: «Повідомлення з потоку…» із вказанням відповідного номеру (1, 2 або 3). Також, створюється потік Notify, який після очікування на протязі 100 мілісекунд пробуджує один з потоків за допомогою функції notify_one() і присвоює змінній і значення 1.

   Головний потік повинен дочекатись виконання потоку Notify.

   Запустіть програму декілька разів, запишіть та проаналізуйте результати.
4. Змініть код програми з пункту 1, щоб замінити синхронізацію на основі прапора синхронізацією на основі умовних змінних. У якості умови для функції wait() використати лямбда-функцію, яка перевіряє чергу на пустоту.
5. Запишіть лістинги розроблених програм у звіт. Зробіть висновки загальні висновки щодо шаблонів стандартної бібліотеки на основі м’ютексів та способів уникнення взаємного блокування.

Контрольні запитання

1. Як забезпечується розпаралелювання задач у операційних системах?
2. Яким чином забезпечене управління потоками у мові C++?
3. Що таке м’ютекс та які основні принципи його роботи?
4. Яким чином створюється м’ютекс засобами мови C++?
5. Що таке умовна змінна?
6. Яким чином створюються умовні змінні засобами мови C++?
7. Як змусити потік очікувати на зміну значення умовної змінної?

Лабораторне заняття 4

Синхронізація паралельних операцій за допомогою механізму майбутніх результатів

***Мета роботи*:** Навчитись синхронізувати паралельні операції за допомогою механізму майбутніх результатів.

Хід роботи

1. Створити програму, яка пропонує користувачеві дізнатися n-те просте число (n вводиться користувачем). Після цього, запустити на виконання обчислення цього числа за допомогою std::async. Далі, запропонувати користувачеві (поки йде обчислення) дізнатися, чому дорівнює значення однієї з функцій (корінь квадратний, синус, натуральний логарифм) від уведеного ним номера простого числа. Після цього, програма має вивести результат обчислень простого числа за заданим номером.

   У першому випадку створити об’єкт std::async з параметром std::launch::deferred, у другому – std::launch::async. Порівняти час очікування на результат обчислення (наприклад, мільйонного простого числа) у обох випадках. Зробити висновки про причини різниці у затримці.
2. Створити програму, яка у головному потоці пропонує користувачеві дізнатися довільну кількість n-тих простих чисел (n вводиться користувачем), тобто опитує користувача, доки ~~йому не набридне~~ не отримає стоп-символ. Створити два дека (структура даних), перший з яких містить об’єкти std::packaged_task<>, які відповідають завданням обчислення n-го простого числа, а другий – власне номери n. Створити окремий **незалежний** від головного потік, який виконує вказані завдання, що постачаються йому через деки.

   Забезпечити безпеку доступу до деків. Забезпечити очікування головним потоком завершення усіх обчислювальних завдань та коректне завершення роботи незалежного потоку. Вивід результатів обчислень здійснювати на стандартний пристрій вводу.

3. Створити програму, яка пропонує користувачеві дізнатися n-те просте число (n вводиться користувачем). Створити два незалежні від головного потоки. Перший з них повертає головному потокові для виводу на стандартний пристрій вводу n-те просте число за допомогою std::promise/std::future, встановлює булевий std::promise у значення true, чим сигналізує другому потокові, про початок його роботи, і повертає головному потокові для виводу на стандартний пристрій вводу (n*10)-те просте число за допомогою std::promise/std::future. Другий потік очікує на майбутній результат булевого типу (від першого потоку) і після його отримання, чекає дві секунди, після чого виводить на стандартний пристрій виводу значення квадратного кореня від числа n.

   Усі потоки, крім головного, повинні повертати тип void. У процесі написання програми не використовувати жодних глобальних змінних чи структур даних.

   Проаналізувати роботу одержаної програми та ефективність застосування пари std::promise/std::future.
4. Запишіть лістинги розроблених програм у звіт. Зробіть висновки щодо роботи шаблонів стандартної бібліотеки на основі об’єктів майбутніх подій.

Контрольні запитання

1. Як забезпечується розпаралелювання задач у операційних системах?
2. Яким чином забезпечене управління потоками у мові C++?
3. Що таке м’ютекс та які основні принципи його роботи?
4. Яким чином створюється м’ютекс засобами мови C++?
5. Що таке об’єкт майбутньої події?

Лабораторне заняття 5

Реалізація паралельних методів множення матриці на вектор

***Мета роботи*:** Навчитись реалізувати паралельні методи множення матриці на вектор.

Хід роботи

1. Визначити (і навести обчислення) свій варіант за формулою:

   num *mod* 4 + 1,

   де num – номер у журналі, *mod* – операція взяття остачі від ділення.
2. Засобами C++ Standard Template Library та C++ Thread Library створити програму, яка забезпечує реалізацію паралельного множення матриці на вектор методом смугового розбиття матриці по рядках.

   Кількість потоків у програмі визначити на основі номеру свого варіанту за наступною таблицею (Таблиця 1)

Таблиця 1 – Кількості потоків для різних варіантів для методу смугового розбиття матриці по рядках

Варіант	**Кількість потоків**
1	4
2	6
3	8
4	10

Розміри матриці увести з клавіатури (вважати, що кількість рядків матриці пропорційна значенню p – числу потоків, які здійснюють обчислення). Передбачити можливість зчитування матриці та вектора з, відповідно, файлів M1_VN.txt та V1_VN.txt (замість N вказати номер свого варіанту), з клавіатури або заповнення матриці та вектора випадковими числами з наступним їх виведенням на екран (для розмірів матриці <=12) або у файл із розширенням Input_Data.txt. Для результатів обчислення передбачити

аналогічні способи виведення (у випадку файлового виводу, файл із результатами повинен називатись Result.txt).

3. Засобами C++ Standard Template Library та C++ Thread Library створити програму, яка забезпечує реалізацію паралельного множення матриці на вектор методом смугового розбиття матриці по стовпцях.

   Кількість потоків у програмі визначити на основі номеру свого варіанту за наступною таблицею (Таблиця 2)

Таблиця 2 – Кількості потоків для різних варіантів для методу смугового розбиття матриці по стовпцях

Варіант	**Кількість потоків**
1	8
2	10
3	4
4	6

   Розміри матриці увести з клавіатури (вважати, що кількість стовпців матриці пропорційна значенню p – числу потоків, які здійснюють обчислення). Передбачити можливість зчитування матриці та вектора з, відповідно, файлів M2_VN.txt та V2_VN.txt (замість N вказати номер свого варіанту), з клавіатури або заповнення матриці та вектора випадковими числами з наступним їх виведенням на екран (для розмірів матриці <=12) або у файл із розширенням Input_Data.txt. Для результатів обчислення передбачити аналогічні способи виведення (у випадку файлового виводу, файл із результатами повинен називатись Result.txt).

4. Запишіть лістинги розроблених програм у звіт. Зробіть висновки щодо ефективності та зручності застосування паралельних методів множення матриці на вектор.

Контрольні запитання

1. Як забезпечується розпаралелювання задач у операційних системах?

2. Які є загальні способи розподілу даних, що представлені у вигляді матриць (масивів)?
3. Опишіть смугову схему розподілу матриці по рядкам для задачі матрично-векторного множення.
4. Опишіть смугову схему розподілу матриці по стовпцям задачі матрично-векторного множення.
5. Опишіть блочну схему задачі матрично-векторного множення.

Рекомендована література

Базова література

1. Основы многопоточного, паралельного и распределенного программирования.: Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 512 с.: ил. – Парал. тит. англ.
2. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608 с.: ил.
3. Уильямс Э. Параллельное программирование на C++ в действии. Практики разработки многопоточных программ.: Пер. с англ. – М.: ДМК-Пресс, 2016. – 672 с.

Допоміжна література

1. Пратт Т., Зелковиц М. Языки программирования: разработка и реализация / Под общей ред. А. Матросова.. – СПб.: Питер, 2002. – 688с. – ил.
2. Таненбаум Э. – Современные операционные системы. 3-е изд. СПб.: Питер, 2010. – 1120с. – ил. – (Серия «Класика Computer science»)
3. Gebali, Fayez. Algorithms and parallel computing/Fayez Gebali. p. cm.—(Wiley series on parallel and distributed computing ; 82) Includes bibliographical references and index. ISBN 978-0-470-90210-3 (hardback)

Інформаційні ресурси

www.scientific-library.net – Електронна бібліотека науково-технічної літератури
www.elibrary.ru – Наукова електронна бібліотека науково-технічної літератури
www.coursera.org – Електронна база курсів