НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Звіти до комп’ютерних практикумів з кредитного модуля

“Parallel computing”

**Виконав**

**Студенти групи ІТ-04**

**Гавриленко Я.С**

**Перевірив:**

Київ – 2022

**Комп‘ютерний практикум No 5**

**Застосування високорівневих засобів паралельного програмування для побудови алгоритмів імітації та дослідження їх ефективності**

ЗАВДАННЯ

![Зображення, що містить текст

Автоматично згенерований опис]()

**ВИКОНАННЯ**

Повний код лабораторної: <https://github.com/yan14171/parralel_calcullation_labs/tree/master/lab>5

**Завдання 1**

Роботу системи розділено на три основні класи: SMSQueue – уніфіковане представлення черги системи з можливістю забрати одиницю роботи, додати, порахувати середню кількість речей і статистичні дані, SMSProducer – єдиний відправник повідомлень до черги, SMSConsumer – клас, що представляє станцію обробки повідомлень на черзі, останніх є декілька. Обробка задачі є фіктивною затримкою.

Результат роботи можна побачити на рисунку 1.

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 1 Результат виконання СМО

**Завдання 2**

Для виконання завдання 2 у створений пул потоків додається декілька задач СМО, після чого результати роботи агрегуються і представляється статистичний аналіз.

Результати роботи можна побачити на рисунку 1.2.

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 1.2 Реалізація багатопоточного виконання СМО

**Завдання 3**

Реалізуємо додатковий потік, що додамо до внутрішнього пулу потоків СМО. Цей потік матиме доступ до критичних значень СМО та буде виводити їх на консоль кожну секунду.

Результати роботи можна побачити на рисунку 1.3.

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 1.3 Реалізація логування системи

На малюнку наведено момент виконання, в який системи вперше заповнюється повністю, після чого з часом починає збільшуватися відсоток необроблених повідомлень.

**Завдання 4**

Обчислимо показники ефективності для паралелізації алгоритму сортування бульбашкою. Для цього, спочатку визначимо час виконання з різною кількістю робочих процесорів. Результати вимірювань можна побачити в таблиці 1.

Таблиця 1. – Час виконання алгоритму

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Час виконання | | | | |
| ElementCnt/ThreadCnt | 10000 | 20000 | 50000 | 10000 |
| 1 | 682 | 2673 | 30506 | 66496 |
| 2 | 470 | 1918 | 12031 | 47423 |
| 6 | 164 | 671 | 4124 | 16169 |
| 12 | 71 | 261 | 1581 | 6366 |
| 24 | 45 | 167 | 983 | 3361 |

Далі обраховуємо показники ефективності. Прискорення за формулою

Дані можна побачити у таблиці 2.

Таблиця 2. – Прискорення алгоритму

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Прискорення | | | | |
| ElementCnt/ThreadCnt | 10000 | 20000 | 50000 | 10000 |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1.451064 | 1.393639 | 2.535616 | 1.402189 |
| 6 | 4.158537 | 3.983607 | 7.397187 | 4.112561 |
| 12 | 9.605634 | 10.24138 | 19.29538 | 10.44549 |
| 24 | 15.15556 | 16.00599 | 31.03357 | 19.78459 |

Ефективність за формулою

Дані можна побачити у таблиці 3.

Таблиця 3. –Ефективність алгоритму

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ефективність | | | | |
| ElementCnt/ThreadCnt | 10000 | 20000 | 50000 | 10000 |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 0.725532 | 0.69682 | 1.267808 | 0.701094 |
| 6 | 0.693089 | 0.663934 | 1.232865 | 0.685427 |
| 12 | 0.800469 | 0.853448 | 1.607949 | 0.870458 |
| 24 | 0.631481 | 0.666916 | 1.293065 | 0.824358 |

Вартість обчислень за формулою

Дані можна побачити у таблиці 4.

Таблиця 4. – Вартість обчислень алгоритму

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вартість обчислень | | | | |
| ElementCnt/ThreadCnt | 10000 | 20000 | 50000 | 10000 |
| 1 | 682 | 2673 | 30506 | 66496 |
| 2 | 940 | 3836 | 24062 | 94846 |
| 6 | 984 | 4026 | 24744 | 97014 |
| 12 | 852 | 3132 | 18972 | 76392 |
| 24 | 1080 | 4008 | 23592 | 80664 |

**ВИСНОВОК**

В рамках роботи, ми реалізували тестову систему МО за допомогою засобів багатопоточного програмування Java. Також навчилися проводити статистичні дослідження на таких системах, логувати поточні значення під час виконання.

В роботі обчислили показники ефективності для різної кількості потоків виконання при паралелізації алгоритму сортування бульбашкою.