Мета роботи: розробити паралельні реалізації алгоритмів сортування даних з допомогою технології MPI.

Теоретичні відомості

Сортування є однією з типових проблем оброблення даних, під яким зазвичай розуміють задачу розміщення елементів неупорядкованого набору значень

$$S = \{a_1, a_2, ..., a_n\}$$

у порядку монотонного зростання або спадання

$$S' = \{(a_1', a_2', ..., a_n') : a_1' \le a_2' \le ... \le a_n'\}$$

(тут і далі всі пояснення заради стислості викладення будуть даватися тільки на прикладі впорядкування даних за зростанням).

Можливі способи розв'язання цієї задачі широко обговорюються в літературі; один з найповніших оглядів алгоритмів сортування міститься в роботі [6], серед останніх видань може бути рекомендована робота [7].

Обчислювальна трудомісткість процедури впорядкування є досить високою. Так, для низки відомих простих методів (пузиркове сортування, сортування включенням тощо) кількість необхідних операцій визначається квадратичною залежністю від кількості значень, що впорядковуються, тобто $T_1 \sim n^2$.

Для ефективніших алгоритмів (сортування злиттям, сортування Шелла, швидке сортування) трудомісткість визначається величиною $T_1 \sim n \cdot \log_2 n$.

Цей вираз дає також нижню оцінку необхідної кількості операцій для впорядкування набору, що містить n значень; алгоритми з меншою трудомісткістю можуть бути отримані тільки в окремих випадках.

Прискорення процедури сортування може бути забезпечене в разі використання декількох $(p > 1)$ процесорів. Вихідний набір, що підлягає впорядкуванню, у цьому випадку поділяється між процесорами; у ході сортування дані пересилаються між процесорами й порівнюються між собою. Вислідний (упорядкований) набір, як правило, також поділений між процесорами; при цьому для систематизації такого поділу для процесорів запроваджують ту або іншу систему послідовної нумерації й зазвичай необхідно, щоб після завершення процедури сортування значення, розташовувані на процесорах з меншими номерами, не перевищували значень, що містяться на процесорах з більшими номерами.

Залишаючи докладний аналіз проблеми сортування для окремого розгляду, тут ми приділимо основну увагу вивченню паралельних способів виконання для низки широко відомих методів внутрішнього сортування, коли всі дані, що впорядковуються, можуть бути повністю розміщені в оперативній пам'яті системи.

Принципи розпаралелювання

Уважно розглянувши способи впорядкування даних, застосовувані в алгоритмах сортування, можна дійти висновку, що багато методів передбачають застосування тієї ж самої базової операції "порівняти й переставити" (compare-exchange), яка полягає в порівнянні тієї або іншої пари значень з набору даних, що сортується, і перестановці значень з цієї пари, якщо порядок їхнього розташування не відповідає умовам сортування.

```
// Лістинг 5.1. Базова операція сортування
void compare_exchange(double *x, double *y)
{
 double temp;
```

```
 if (*x > *y)
  {
   temp = *x;
   *x = *y;
   *y = temp;
  }
}
```

Цілеспрямоване застосування зазначеної операції дозволяє впорядкувати дані; у способах вибору пар значень для порівняння, власне, й проявляються відмінності між алгоритмами сортування.

Для паралельного узагальнення виділеної базової операції сортування розглянемо спочатку ситуацію, коли кількість процесорів збігається з кількістю значень даних, що сортуються, тобто $p = n$, і на кожному з процесорів міститься тільки по одному значенню вихідного набору даних. Тоді порівняння значень a_i і a_j, розташовуваних відповідно на процесорах p_i і p_j, можна організувати в такий спосіб (паралельне узагальнення базової операції сортування):

- виконати взаємообмін наявних на процесорах p_i і p_j значень (зі збереженням на цих процесорах вихідних елементів);
- порівняти на кожному процесорі p_i p_j отримані однакові пари значень $\left(a_i, a_j\right)$, результати порівняння використовуються для поділу даних між процесорами – на одному процесорі (наприклад, p_i) залишається менший елемент, інший процесор (тобто p_j) запам'ятовує для подальшого оброблення більше значення пари $a_i^{'} = \min\left(a_i, a_j\right)$, $a_j^{'} = \max\left(a_i, a_j\right)$.

Масштабування паралельних обчислень

Розглянуте паралельне узагальнення базової операції сортування може бути належним чином адаптоване й для випадку $p < n$, коли кількість процесорів є меншою ніж кількість значень, що впорядковуються. У такій ситуації кожний процесор буде містити вже не єдине значення, а частину (блок розміру n/p) набору даних, що сортується.

Визначимо як результат виконання паралельного алгоритму сортування такий стан набору даних, що впорядковується, за якого наявні на процесорах дані впорядковані, а порядок розподілу блоків по процесорах відповідає лінійному порядку нумерації (тобто значення останнього елемента на процесорі p_i є меншим або дорівнює значенню першого елемента на процесорі p_{i+1}, де $0 \le i < p-1$).

Блоки зазвичай впорядковуються на самому початку сортування на кожному процесорі окремо за допомогою якого-небудь швидкого алгоритму (початкова стадія паралельного сортування). Далі, дотримуючись схеми одноелементного порівняння, взаємодія пари процесорів p_i і p_{i+1} для спільного впорядкування вмісту блоків A_i і A_{i+1} може бути здійснена в такий спосіб:

- виконати взаємообмін блоків між процесорами p_i й p_{i+1};
- об'єднати блоки A_i й A_{i+1} на кожному процесорі в один відсортований блок подвійного розміру (за вихідної впорядкованості блоків A_i і A_{i+1} процедура їх об'єднання зводиться до швидкої операції злиття впорядкованих наборів даних);

♦ розділити отриманий подвійний блок на дві рівні частини й залишити одну із цих частин (наприклад, з меншими значеннями даних) на процесорі p_i, а іншу частину (з більшими значеннями відповідно) – на процесорі p_{i+1}.

Розглянуту процедуру в літературі зазвичай називають операцією "порівняти й розділити" (compare-split). Слід зазначити, що сформовані в результаті виконання такої процедури блоки на процесорах p_i і p_{i+1} збігаються за розміром з вихідними блоками A_i й A_{i+1}, і всі значення, розташовані на процесорі p_i, не перевищують значень, що містяться на процесорі p_{i+1}.

Визначена вище операція "порівняти й розділити" може бути використана як базова підзадача для організації паралельних обчислень. Слід зазначити, що кількість таких підзадач параметрично залежить від кількості наявних процесорів і, таким чином, проблема масштабування обчислень для паралельних алгоритмів сортування практично відсутня. Разом з тим слід зазначити, що вміст блоків даних, використовуваних підзадачами, зазвичай змінюється в ході виконання сортування. У простих випадках розмір блоків даних у підзадачах залишається незмінним. У складніших ситуаціях обсяг розташовуваних на процесорах даних може різнитися, що може призводити до порушення рівномірного обчислювального завантаження процесорів.

Контрольні запитання

1. У чому полягає постановка задачі сортування даних?
2. Наведіть кілька прикладів алгоритмів сортування. Яка обчислювальна складність наведених алгоритмів?
3. Яка операція є базовою для задачі сортування даних?
4. У чому суть паралельного узагальнення базової операції задачі сортування даних?
5. Чим є алгоритм парно-непарної перестановки?
6. У чому полягає паралельний варіант алгоритму Шелла? Які основні відмінності цього варіанта паралельного алгоритму сортування від методу парно-непарної перестановки?
7. Чим є паралельний варіант алгоритму швидкого сортування?
8. Що залежить від правильного вибору головного елемента для паралельного алгоритму швидкого сортування?
9. Які способи вибору головного елемента можуть бути запропоновані?
10. Для яких топологій можуть застосовуватися розглянуті алгоритми сортування?

Індивідуальні завдання

Розробити паралельну реалізацію одного з перерахованих в дужках алгоритмів сортування (Шелла, швидкого сортування, Бетчера) в середовищі MPI.