УДК 004.051

Є.В. Крилов, В.К. Анікін, В.І. Стельмах

РОЗРОБКА МОДЕЛІ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ НА PHP

Анотація: Розглядається розробка моделі розпаралелювання програмного коду на PHP для підвищення швидкої веб-додатків. Для реалізації паралельного виконання взято за основу модель передачі повідомлень та запропоновано метод її використання на PHP. Для розподілу навантаження по процесах, запропонована проста схема блочного розподілу.

Ключові слова: розпаралелювання, швидкодія веб-додатків, модель обміну повідомленнями, розподіл навантаження.

Вступ

На поточний момент однією з основних вимог до веб додатків є їх швидкодія. Адже час виконання розрахунків, і як наслідок час завантаження сторінки може скласти негативне враження у відвідувача.

При обробці великих масивів данних, більшість часу витрачається саме на пошук та доступ до конкретного елементу. Таким чином дістаючи данні паралельно можна збільшити швидкість розрахунку оскільки наступний запит в БД виконуватиметься не дочікуючись завершення попереднього та проведення обрахунку.

Стандартних засобів для виконання програмного коду паралельно PHP не має.

Огляд моделей розпаралелювання програмного коду

Для функціонування додатків з паралельними обчисленнями потрібно призначити виконавців та диригента (керівника) що керуватиме роботою виконавців та збиратиме результати.

Розглянемо існуючі програмні моделі для роботи на паралельних обчислювальних системах:

Послідовна модель. Припускає, написання звичайної послідовної програми в одній з послідовних моделей програмування для подальшого автоматичного її розпаралелювання компілятором або спеціальними програмними засобами.

Модель передачі повідомлень. Припускає, що додаток що виконується складається з набору процесів з різними адресними просторами, кожен з яких функціонує на своєму виконавці.

Модель розділеної пам'яті. Припускає, що додаток складається з набору потоків виконання (thread), що використовують колективні змінні і примітиви синхронізації.

Модель розділених даних. Припускає, що додаток складається з наборів процесів або потоків, кожен з яких працює зі своїм набором даних, обміну інформацією при роботі немає. Застосовна до обмеженого класу задач [1].

ISSN 1560-8956

Оцінивши існуючі методики розпаралелювання зроблено висновок що для використання в PHP підходить лише модель передачі повідомлень, оскільки вона реалізується безпосередньо на високому рівні, в той час як інші моделі потребують інтегрованих в компілятор методів або ж можливість безпосереднього доступу до ресурсів операційної системи, чого PHP не має. Тому для реалізації паралельного виконання скористаємося можливостями веб-сервера, який виділяє окремий потік для кожного нового запиту.

Постановка задачі

На сьогоднішній день модель обмін повідомленнями (message passing) є найбільш широко використовуваною моделлю паралельного програмування. Програми цієї моделі, створюють безліч процесів, з кожним з яких асоційовані локальні дані. Кожен процес ідентифікується унікальним ім'ям. Процеси взаємодіють, посилаючи й одержуючи повідомлення.

Модель обмін повідомленнями не накладає обмежень ні на динамічне створення процесів, ні на виконання декількох процесів одним процесором, ні на використання різних програм для різних процесів. Просто, формальні описи систем обміну повідомленнями не розглядають питання, пов'язані з маніпулюванням процесами, Однак, при реалізації таких систем доводиться приймати якесь рішення щодо цього. На практиці склалося так, що більшість систем обміну повідомленнями при запуску паралельної програми створює фіксоване число ідентичних процесів і не дозволяє створювати і руйнувати процеси протягом роботи програми.

В таких системах кожен процес виконує одну і ту ж програму, але працює з різними даними, тому про такі системи кажуть, що вони реалізують SPMD (single program multiple data - одна програма багато даних) модель програмування. SPMD модель прийнятна і досить зручна для широкого діапазону додатків паралельного програмування, але вона ускладнює розробку деяких типів паралельних алгоритмів.

Модель передачі повідомлень характеризується наступними властивостями:

1. Паралельне обчислення складається з одного або більше одночасно виконуються процесів, число яких може змінюватися протягом часу виконання програми.
2. Процес – це послідовна програма з локальними даними. Процес має вхідні і вихідні порти, які служить інтерфейсом до середовища процесу.
3. На додаток до звичайних операціях процес може виконувати наступні дії: послати повідомлення через вихідний порт, отримати повідомлення з вхідного порту, створити новий процес і завершити процес.

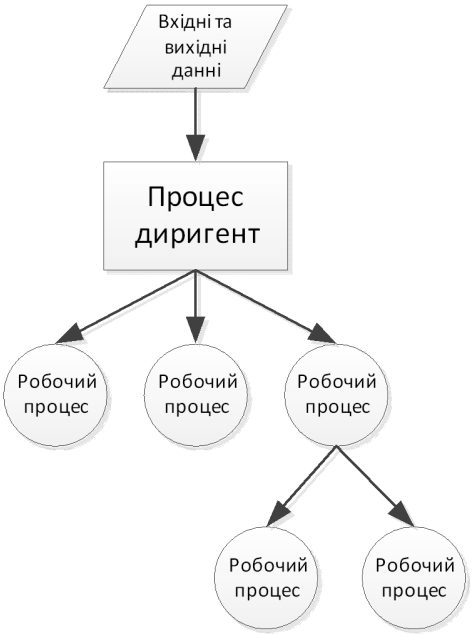

Рис. 1 – Структура моделі передачі повідомлень

4. Посилати операція асинхронна - вона завершується відразу, не чекаючи того, коли дані будуть отримані. Операція отримання синхронна: вона блокує процес до моменту надходження повідомлення.
5. Процеси можна розподіляти по фізичним процесорам довільним способами, причому використовуване відображення (розподіл) не впливає на семантику програми. Зокрема, безліч процесів можна відобразити на одиночний процесор [2].

Для розподілу навантаження по процесам, ми можемо скористатися наступною простою схемою обчислення кількості підзадач для блочного розподілу. Припустимо, є цикл в приблизно однакових за тривалістю ітерацій і потрібно розподілити його між N процесами, причому n не кратне N. Тоді на основі розподілу із залишком кількість ітерацій може бути представлено через цілі числа a, b в такій формі:

$$n = aN + b, 0 \leqslant b < N \tag{1}$$

В цьому випадку ширина інтервалу ітерацій для кожного процесу $n_i, i = 0, \ldots, \mathrm{N} - 1$ може бути обчислена таким чином:

$$n_i = \begin{cases} a+1, & i < b \\ a, & i \geqslant b \end{cases} \tag{2}$$

Ліва межа кожного інтервалу $n_i^l, i = 0, \ldots, N-1$ може бути обчислена на основі тих же співвідношень:

$$n_i^l = \sum_{k=0}^{i} n_k - n_i = ai + \begin{cases} i, & i < b \\ b, & i \geqslant b \end{cases} \tag{3}$$

При цьому на кожен процес розподіляється інтервал ітерацій [3]

$$n_i^l; n_i^l + n_i \tag{4}$$

Експериментальне дослідження

Для реалізації скрипта диригента можна використовувати стандартну функцію PHP – fsockopen(), яка ініціює поточное з'єднання, використовуючи TCP або UDP. Для домену Internet - відкриває сокет з'єднання TCP з *hostname* через порт *port*. *hostname* може бути в цьому випадку або повним кваліфікованим ім'ям домену, або IP-адресою. Для UDP необхідно явно специфікувати протокол шляхом додавання до hostname префікса 'udp: //'. fsockopen() повертає покажчик файлу, який може використовуватися іншими функціями (такими як fgets (), fgetss (), fputs (), fclose () і feof ()).

За замовчуванням з'єднання відкривається з блокуванням. Відповідно наступне з'єднання буде відкрито лише після завершення попереднього і програма виконуватиметься послідовно. Для можливості відкриття багатьох з'єднань у паралельному режимі використовується функція stream_set_blocking(), першим параметром вказавши *об'єкт з'єднання*, а другим параметром *false*, що вимкне блокування на вказаному потоці.

У якості виконавців можуть виступати окремі скрипти або скрипт запущений декілька разів але з різними вхідними параметрами.

Використовуючи функцію fsockopen скрипт диригент викликає виконавців, передає вхідні параметри та отримує результат виконання.

Таким чином диригент та кожен виконавець працюватимуть у різних потоках і операційна система, за наявності ресурсіі, зможе забезпечити їх паралельне виконання [4].

Для моделювання розрахунків можна обрати алгоритм розрахунку числа Пі, шляхом чисельного інтегрування.

$$\int_0^1 \frac{4}{1+x^2} dx = \pi \tag{5}$$

$$\int_0^1 \frac{4}{1+x^2} dx = 4 * arctg(x) = 4 * (\frac{\pi}{4} - 0) = \pi \quad (6)$$

Візьмемо найпростіший з методів чисельного інтегрування – метод прямокутників.

Оголосимо деяку кількість n прямокутників на які ми розіб'ємо нашу площу під інтегралом, порахуємо основу:

$$h = 1.0/n \quad (7)$$

і всю площу, порахувавши значення функції в кожному прямокутнику і помножимо на основу.

```
$sum = 0.0;
for( $i=1; $i <= $n ; $i++ ) {
  $x = $h * ( $i - 0.5 );
  $sum + = ( 4.0 / (1.0 + $x * $x ) ) ;
}
$pi = $h * $sum;
```

Виконавши обрахунок послідовно, отримали результат у 10.894656719318с.

Далі виконаємо декомпозицію алгоритму декомпозицію використовуючи метод програмної конвеєризації циклів, суть якого полягає в тому, що ітерації циклу поєднуються при виконанні з невеликим зрушенням по відношенню один до одного.

Відповідно попередній код зазнає наступних змін:

```
$h = 1.0 / $n;
$sum = 0.0;
for( $i = $rank + 1; $i <= $n; $i + = $size ) {
  $x = $h * ( $i - 0.5 ) ;
  $sum + = ( 4.0 / (1.0 + $x * $x ) ) ;
}
$pi = $h * $sum ;
```

З метою збільшення точності вимірів для кожної кількості виконавців виконаємо по 10 ітерацій і зафіксовано мінімальний час обрахунку. Мінімальний час використовується через те, що саме у поточний момент процесор був завантажений найменше сторонніми процесами.

Проведемо експериментальні виміри для обчислень з використанням від 1 до 15 виконавців, з проходом у 40,000,000 інтервалів.

Висновки

З результатів експерименту можна побачити спад часу виконання обрахунків зі збільшенням кількості виконавців. При розрахунку в 1 потік, задіяне одне ядро процесора і розрахунок триває 11.3 с.

Таблиця 1

Заміри часу виконання експерименту

Кількість виконавців	**Час виконання, с**
1	11.314646959305
2	6.0763478279114
3	4.1122350692749
4	3.936224937439
5	3.5412030220032
6	3.3061890602112
7	3.2371850013733
8	3.1852010917664
9	3.2951879501343
10	3.5212020874023
11	3.6431920814514
12	3.8301930046082
13	3.9851805842514
14	4.1301960468292
15	4.5512030124664

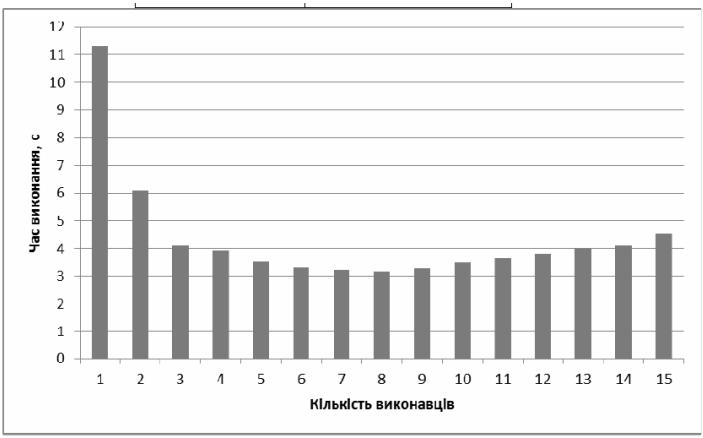

Рис. 2 – Діаграма замірів виконання експерименту

Зі збільшення потоків час обрахунку зменшується і на 8 потоках зменшується до 3.1 с, що дає виграш у швидкодії у 72% та доводить спроможність інтерпретатора PHP до вирішення обрахунків з розпаралелюванням виконання завдань. Але при перевищенні кількості ядер час обрахунку знову починає зростати, що доводить недоцільним використання розпаралелювання на одно процесорних (одноядерних) системах.

Перелік використаних джерел

1. Карпов В. Е. Введение в распараллеливание алгоритмов и программ / В. Е. Карпов // Компьютерные исследования и моделирование – 2010 - Т. 2 №3 С. 231–272.
2. Антонов А.С. Введение в паралельные вычисления / А. С. Антонов // Московский государственный университет им. Ломоносова–2002-70 с.
3. Крилов Є.В. Оптимізація роботи веб-сервера для взаємопов'язаних процесів / Є.В. Крилов, В.К. Анікін, А.О. Шумада // Міжвідомчий науково-технічний збірник "Адаптивні системи автоматичного управління". – 2014. - №1(24). – С.46-52.
4. Федотов И. Е. Некоторые приемы параллельного программирования / И. Е. Федотов // Московский государственный институт радиотехники –2008–188с.

Отримано 15.04.2015 р.