УДК 681.3

О.М. Данильченко, к.т.н.
С.М. Защипас, аспір.
Житомирський інженерно-технологічний інститут

ОПТИМІЗАЦІЯ ВИКОРИСТАННЯ КЛАСТЕРНИХ СИСТЕМ

Розпаралелювання та виконання алгоритмів в кластерних системах звичайно приводить до значного підвищення швидкості при вирішенні широкого класу задач. Але для різних задач та різних реалізацій алгоритмів їх вирішення ефективність використання кластерних систем залежить, як від кластерної системи, так і від параметрів задач, що на ній вирішуються. В даній доповіді розглядаються проблеми ефективності виконання програм у кластерних системах та варіанти методів їх вирішення.

1. Вступ

Останнього десятиріччя в усьому світі бурхливо розвивається відносно нова галузь обчислювальної індустрії: метакомп'ютинг. Поняття метакомп'ютинг – це абстракція за допомогою якої комп'ютерні кластери та окремі комп'ютери в розподіленій системі можуть бути представлені у вигляді єдиного віртуального комп'ютерного ресурсу [1-3]. В свою чергу комп'ютерний кластер є набором повноцінних комп'ютерів-вузлів, що зв'язані між собою мережею та надані кінцевому користувачу у вигляді єдиного обчислювального ресурсу [4,5].

Звичайно кластери будуються з метою створення високопродуктивної обчислювальної системи або створення високонадійної обчислювальної системи, звичайно не виключена можливість комбінованої мети.

Очевидно, що метакомп'ютер та кластер є подібними, один одному, абстракціями, хоча кластер може бути складовою частиною метакомп'ютера. Відповідно можна припустити, що існують певні закони, що однаково діють як на кластер, так і на метакомп'ютер.

Бурхливий розвиток метакомп'ютингу можна пов'язати перш за все з прогресуючими вимогами людства до обчислювальних систем. Причому з економічної точки зору метакомп'ютери або кластери часто є більш вигідними за спеціалізовані суперкомп'ютери, що створюються в одиничному екземплярі. Надалі в доповіді ми будемо розглядати кластер, як базову абстракцію метакомп'ютерних систем.

2. Підсистеми складання розкладу кластерних систем

Важливою та невід'ємною частиною кластерних системи виступають підсистеми складання розкладу виконання завдань системи.

Загальна задача складання розкладу оптимального виконання задач відповідно до набору певних умов є NP-повною. Звичайно для вирішення цієї задачі використовують наближені евристичні алгоритми.

Задача складання розкладу процесів у кластерній системі може бути спрямована на мету досягнення високої продуктивності системи за допомогою: мінімізації часу виконання окремої задачі, мінімізації пауз в обміні даними, максимізації утилізації системних ресурсів. В системах, що потребують обробки даних в реальному часі складання розкладу полягає в знаходженні оптимального розподілу ресурсів для задоволення вимог кожного запиту.

Можна виділити чотири основні рівні складання розкладів в кластерних системах:

1. Складання розкладу машинних команд, що виконуються в системі.
2. Складання розкладу команд на мовному рівні інтерпретатора або компілятора мови програмування, що використовується.
3. Складання розкладу ниток (threads) в рамках одної задачі.
4. Складання розкладу задач в рамках кластера.

Ми будемо розглядати складання розкладів на третьому та четвертому рівнях.

Існує три основні моделі складання розкладів в метакомп'ютерних системах та кластерах: статична, динамічна та гібридна (статично-динамічна).

Головною перевагою статичної моделі є простота програмування з точки зору побудови розкладу. При застосуванні цієї моделі досить легко стежити за процесом роботи програми, оцінювати вартість виконання роботи програми, при необхідності, досить просто, її перервати. Цю модель звичайно використовують при плануванні виділення певних ресурсів програмі користувача. До того як буде виконано програму, на вузол, що буде її виконувати, подається запит щодо отримання певних ресурсів, необхідних для виконання програми користувача. Статична модель є також найбільш ефективною для програмування багатоетапних задач, що

зв'язані між собою. При складанні статичного розкладу можна точно визначити послідовність та напрямки передачі даних між вузлами кластера. При використанні статичної моделі кожен вузол знає, звідки він має взяти вхідні дані та куди передати вихідні.

Основним недоліком статичної моделі є можливість збоїв та втрати даних у випадку втрати дієздатності одного або декількох вузлів системи внаслідок збою.

Динамічна модель складання розкладу, яку в літературі часто називають моделлю динамічного балансування, навантаження звичайно складається з двох рівнів: рівня локального розкладу та рівня розкладу навантаження кластера. Локальний розклад відповідає за оптимізацію черги завдань та використання ресурсів в середині конкретного вузла. Розклад навантаження кластера, визначає те, як програми будуть розподілені між вузлами, використовуючи інформацію про те, які дані з кожного вузла будуть збиратися в пул локального вузла, як часто буде відбуватися обмін інформацією між локальним вузлом та іншими вузлами кластера. При складанні розкладу також приймаються до уваги певні привілеї чи побажання кінцевих користувачів системи та побажання власника обчислювальних ресурсів.

При використанні динамічної моделі складання розкладу задачі, що містять паралельні або розподілені роботи, призначають на певні вузли кластера, базуючись на передбаченні того, чи може кожен з цих вузлів надати необхідні ресурси для вирішення відповідної задачі. Необхідні ресурси залежать від типу конкретного додатка, що виконується на вузлі кластера. Поняття ресурсу включає: на який максимальний термін може бути відкладений запуск, мінімальний час який ця програма зможе виконуватись на цьому вузлі без переривань, відносна швидкість вузла що до інших вузлів системи. Якщо на один процесор призначається занадто багато задач то в системі складання розкладу повинні бути певні правила перерозподілу задач між процесорами. Повинні бути також деякі локальні правила, що визначають, який процесор буде отримувати перерозподілені задачі. Залежно від того, чи вузол відправник особисто вибирає потенційні вузли для передачі задачі, чи вузол приймач особисто подає запит на прийом додаткових задач, розрізняють дві моделі подібного перерозподілу: перерозподіл ініційований передавачем та перерозподіл ініційований приймачем.

Перевагами динамічних систем складання розкладів перед статичними системами складання розкладів є те, що їм нічого не потрібно знати про властивості задачі до її запуску. Звичайно системи динамічного балансування навантаження використовуються в системах, де основною метою є максимальна утилізація ресурсів системи, на відміну від мети мінімізації часу виконання окремих задач. Досить часто подібні системи використовуються в обчислювальних мережах та кластерах.

Звичайно розглядаючи балансування навантаження в кластерній системі мають на увазі збалансоване розподілення ниток (thread) програми між вузлами. Балансування навантаження може бути проведене одним із двох методів: використовуючи глобальну чергу або використовуючи локальну чергу.

Наведемо для прикладу декілька методів, що можуть бути застосовані для призначення ниток на вузли кластерної системи за допомогою локальної черги:

1-й метод: Вибираємо вузол випадково та призначаємо нитку на цей вузол.

2-й метод: Вибираємо вузол випадково, досліджуємо його та його найближчих сусідів, призначаємо нитку на вузол, що є найменш завантаженим.

3-й метод: Перебираємо деяке обмежене число вузлів $N \in Z^+$, призначаємо нитку на найменш завантажений з цих вузлів.

Звичайно глобальні черги легко використовувати у системах з пам'яттю колективного користування (shared memory). Для систем з розподіленою моделлю пам'яті подібні черги досить складно побудувати через необхідність мати зв'язний список всіх доступних задач системи. В більшості систем, що використовують глобальні черги, нитки призначаються вузлам, виконуються на вузлі протягом певного часу, а після цього знову повертаються в чергу для перепризначення. Головною перевагою систем з глобальною чергою є рівномірне загальне навантаження всіх процесорів системи. Недоліком таких систем є суперечливість загальної черги та необхідність в пам'яті колективного користування.

Гібридне статично-динамічне складання розкладу виконання задач у кластерній системі використовується у тому випадку, коли обчислення задачі може бути розбите на деяку кількість послідовно-паралельних залежних частин. У такому випадку деякий вузол виконує свою задачу, після закінчення виконання якої він робить спробу визначити адресу вузла, який повинен виконувати наступний крок задачі та передає задачу на цей вузол. Такий підхід потребує щоб кожен вузол знав про інші вузли системи.

Продуктивність динамічного балансування навантаження в гетерогенних системах була досліджена в Chow та Kohler в [6]. Вони розглядали детерміністичну та не детерміністичну стратегії диспетчеризації задач. При не детерміністичній стратегії складання розкладу процес p_i розміщується на вузлі i довільно або відповідно до певної функції $F(S)$, що залежить від деяких параметрів системи. При детерміністичній стратегії диспетчер задач розміщує задачу на вузли керуючись певною політикою. Ця політика може прагнути мінімізації часу реагування системи, мінімізації часу обробки задач в системі, максимізації пропускної здатності системи. Chow та Kohler зробили висновок, що політика максимальної пропускної здатності, що використовує накопичувальну оцінку разом із оцінкою сервісу та станом системи, є більш оптимальною порівняно з іншими політиками, що базують свої рішення на оцінюванні сервісу та стану системи.

3. Застосування статичної та динамічної моделей складання розкладу в кластерних системах

Використання кластерних систем накладає певні вимоги на програмне забезпечення, що на них виконується. Ефективність використання таких систем залежить від технічних характеристик системи та задач, що на ній вирішуються.

Задачі, що вирішуються в кластерній системі можна умовно розділити на два типи:

1. Послідовні задачі, алгоритми яких слабо піддаються розпаралелюванню. Задача такого типу звичайно повністю виконується на якомусь одному вузлі кластера. В кластерній системі може одночасно (паралельно) виконуватися багато подібних задач. Для диспетчеризації таких задач, звичайно використовується модель динамічного балансування.
2. Задачі, що складаються з частин, які можна виконувати паралельно. Для програмування таких задач звичайно використовують спеціальні бібліотеки, які підтримують розпаралелювання даних (MPI,PVM [8,9]). Для диспетчеризації таких задач можна використовувати модель, як статичного, так і динамічного складання розкладів.

Розглянемо процес розв'язання задачі другого типу в кластерній системі.

1. Нехай існує множина даних A, яка повинна бути перетворена у множину даних A'. Для подібного перетворення необхідно використати алгоритм P, причому A може бути розбита певним чином на n підмножин: $A = A_1 \cup A_2 \cup ... \cup A_n$ так, що перетворення $P(A_i) = A_i'$, де $i \in [1;n]$ так, що $A_1' \cup A_2' \cup ... \cup A_n' = A'$
2. В кластерній системі перетворення $P(A_i) = A_i'$, складається з декількох етапів:

   1 етап: Формування завдання A_i

   2 етап: Передача A_i на вільний вузол кластера N_j, де $j \in [1;m]$, m - кількість вузлів кластера.

   3 етап: Виконання алгоритму P на вузлі N_j.

   4 етап: Повернення результату на ведучий вузол КС.

   5 етап: Формування загального результату.

Так як операції передачі інформації на вільний вузол кластера, виконання алгоритму P на вузлі N_i та повернення результату на ведучий вузол КС виконуються строго послідовно та повністю блокують канал передачі та вузол кластера, то ці операції можна об'єднати в одну з загальним часом виконання. Тоді весь процес перетворення даних в КС можна уявити трьома етапами.

1. Послідовність формування завдань A_i
2. Паралельна передача даних і виконання алгоритму P на вузлі N_i
3. Послідовне формування загального результату.

Час обробки даних на кожному етапі для кожної множини A_i позначимо через t_i^k, $k = 1,2,3$, $i \in [1;n]$. Тоді процес вирішення задачі оптимізації роботи КС зводиться до мінімізації загального часу виконання трьох етапних завдань Якщо, кількість вузлів кластера $m \geq n$ - кількості підмножин даних (для кожної підмножини завжди існує вільний вузол), то функцію цілі можна записати таким чином:

$$\max_{1\le p\le n}\left(\sum_{j=1}^{p} t_j^1 + t_p + \sum_{j=p}^{n} t_j^3\right) \to \min_{\pi_i}, \qquad (1)$$

де $\pi_i = (i_1,\ldots,i_n)$ - перестановка чисел від 1 до n.

В літературі подібна задача відома, як задача "Про редактора" [8] і досить добре досліджена. Оптимальний розв'язок досягається за допомогою такого алгоритму:

Алгоритм 1. Спочатку, на обробку в КС запускаються завдання (підмножини), для яких $t_i^1 \le t_i^3, i \in [1;n]$ в послідовності зростання величин t_i^2, а далі завдання, для яких $t_i^1 > t_i^3$ в послідовності зменшення величин t_i^2.

У випадку коли $n > m$ задача, що розглядається є NP-повною і для її рішення не існує ефективного (поліноміального алгоритму), що знаходить оптимальне рішення. Для вирішення цієї задачі застосовується евристичний алгоритму, в якому на першому етапі всі завдання впорядковуються по алгоритму 1, а у випадку, коли для певного завдання вільний вузол кластера відсутній, то завдання стає в чергу і запускається на першому вільному вузлі.

Слід відмітити, що для вирішення задачі не потрібно знання точного часу обробки завдань на етапах, а достатньо інформації про відносний час виконання (більше-менше).

При вирішенні цієї задачі динамічним алгоритмом необхідно, щоб бібліотека розпаралелювання дозволяла починати виконання обробки кожної підмножини незалежно від початку обробки інших підмножин. Подібна функціональність на сьогоднішній день існує лише в MPI-2.

ЛІТЕРАТУРА:

1. *Larry Smarr, Charles E. Catlett.* Metacomputing. Communications of the ACM, 35(6):44-52, June 1992.
2. *Geoffrey C. Fox, Roy D. Williams, Paul C. Messina.* Parallel Computing Works! Morgan Kaufmann Publishers, Inc., 1994.
3. *Что такое Мета-компьютеринг?* http://www.parallel.ru/computers/rewievs/meta-computing.html
4. *Что такое Beowulf?* http://www.parallel.ru/computers/rewievs/beowulf.html
5. *Савяк В.* Эффективные кластерные решения. http://www.ixbt.com/cpu/clustering.shtml
6. *Yuan Chieh Chow and Walter H. Kohler.* Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System. IEEE Trans. Computers, C-28(5):354-361, May 1979.
7. *Данильченко О.М., Защипас С.М.* Оптимізація розпаралелювання програм у кластерних системах. МКІ М – 2002, С-5:176-179
8. *Танаев В.С., Шкурба В.В.* Введение в теорию расписаний. М.: Наука, 1975, 256 с.
9. MPI: A Message-Passing Interface Standard – http://parallel.ru/docs/Parallel/mpi1.1/mpi-report.html
10. PVM: Parallel Virtual Machine – http://www.epm.ornl.gov/pvm/

ДАНИЛЬЧЕНКО Олександр Михайлович – кандидат технічних наук, доцент, завідувач кафедри програмного забезпечення обчислювальної техніки Житомирського інженерно-технологічного інституту.

Наукові інтереси:
- теорія розкладів;
- теорія складності екстремальних задач;
- паралельні обчислення.

ЗАЩИПАС Сергій Миколайович – аспірант Житомирського інженерно-технологічного інституту.

Наукові інтереси:
- теорія розкладів;
- паралельні обчислення;
- операційні системи.

Подано 21.08.2002