УДК 004.272.43(045) DOI: 10.18372/2073-4751.69.16811

Іванкевич О.В., к.т.н.,
orcid.org/0000-0002-7999-4970,
Мазур В.І.,
orcid.org/0000-0003-4498-7444

БАЛАНСУВАННЯ РОЗПОДІЛЕНОЇ ОБРОБКИ ВЕЛИКИХ МАСИВІВ ІНФОРМАЦІЇ У ДЕЯКИХ СУЧАСНИХ СИСТЕМАХ УПРАВЛІННЯ БАЗАМИ ДАНИХ

Національний авіаційний університет

ntb@npp.nau.edu.ua

Введення

При створенні сучасних інформаційних систем, що використовують бази даних (БД) великого обсягу для підвищення продуктивності додатків, забезпечення високої доступності даних, а також високої масштабованості обчислювальних систем використовуються кластерні системи. Кластерні технології можуть бути використані для побудови недорогих, але потужних систем БД (СБД).

Кластери складаються з серійних компонентів, що мають невелику вартість та є альтернативою монокорпусним суперкомп'ютерам з оригінальною закритою архітектурою. Такі системи широко застосовуються для виконання високопродуктивних обчислень, забезпечення доступності й масштабованості.

Мета статті

Метою статті є огляд системи, призначеної для балансування розподілу інформаційних файлів по вузлах комп'ютерної мережі при обробці великих обсягів даних. Досліджено особливості підвищення продуктивності комп'ютерних мереж при використанні розподілених баз даних.

Зазвичай кластером називається обчислювальна система, що складається з безлічі незалежних комп'ютерів, зв'язаних між собою каналами передачі даних. Всі його підсистеми можна «побачити» у єдиному адміністративному домені, і керування їм виконується, як єдиною обчислювальною системою. Вузли кластера – це серійні універсальні комп'ютери, що випускаються, здатні працювати самостійно. Вузли можуть бути одне- або мультипроцесорними. У класичній схемі всі вузли при роботі з додатками розділяють зовнішню пам'ять на масиві жорстких дисків, використовуючи внутрішні HDD для більше спеціальних функцій. Для межузлового взаємодії зазвичай застосовується стандартна мережева технологія, хоча це не виключає окремо розроблених каналів зв'язку. Кластерная мережа є відособленою, тобто вона ізольована від зовнішнього мережного середовища.

Більшість завдань по обробці великих обсягів даних можуть бути розділені на набір менших завдань, які можуть бути вирішені одночасно. Зазвичай паралельні кластерні обчислення вимагають координації дій. Такі обчислення можуть бути реалізовані в декількох формах: паралелізмі на рівні бітів, паралелізмі на рівні інструкцій, паралелізмі даних та паралелізмі завдань.

На сьогодні відомі два різновиди обробки даних: паралельна і конвеєрна обробка. Паралельна обробка: якщо деякий комп'ютер виконає одну операцію за одиницю часу, то тисячу таких операцій він виконає за тисячу одиниць часу. Якщо припустити що маємо п'ять комп'ютерів, які здатні працювати незалежно і одночасно, то ту ж тисячу операцій система виконає за двісті одиниць. Аналогічно система із n-комп'ютерів виконає ту ж саму роботу за 1000/n одиниць часу.

Ідея конвеєрної обробки полягає у виділенні окремих етапів виконання загальної операції, причому кожний етап, виконавши свою роботу, передає результат наступному, одночасно приймаючи нову

порцію вхідних даних. Отримуємо очевидну перевагу в швидкості обробки за рахунок суміщення раніше розміщених в часі операцій. Якщо є один неподільний пристрій, то 100 пар аргументів від опрацьовує за 500 одиниць. Коли кожну операцію виділити в окремий етап(ступінь) конвеєрного пристрою, то на п'ятій одиниці часу на різній стадії обробки такого пристрою будуть знаходитись перші п'ять пар аргументів, перший результат буде триманий через 5 одиниць часу, кожен наступний – через одну одиницю після попереднього, а весь набір із 100 пар буде оброблений за 5+99=104 одиниці часу, тобто буде отримано прискорення в порівнянні з послідовним пристроєм майже в 5 раз.

Приблизно так само буде і в загальному випадку. Якщо конвеєрний пристрій складається з l ступенів, а кожна ступень спрацьовує за одиницю часу, то час обробки n незалежних операцій цим пристроєм складе $l+n+1$ одиниць. Якщо ж пристрій використовувати в монопольному режимі, то час обробки буде складати $l \times n$. В результаті отримаємо прискорення майже в l разів за рахунок використання конвеєрної обробки даних. Здавалось би конвеєрну обробку можна з легкістю замінити звичайним паралелізмом, для чого можна продублювати основний пристрій стільки разів, скільки ступенів конвеєра необхідно виділити. Однак вартість і складність отриманої системи буде невідповідною по складності і вартості конвеєрного варіанта, а продуктивність буде такою ж.

При розробці паралельних алгоритмів вирішення складних обчислювальних задач і задач обробки даних, принциповим моментом є аналіз ефективності використання паралелізму, що полягає зазвичай в оцінці отримуваного прискорення процесу обчислень (скорочення часу вирішення задачі). Формування подібних оцінок прискорення може здійснюватися стосовно вибраного обчислювального алгоритму (оцінка ефективності розпаралелювання конкретного алгоритму). Інший важливий підхід полягає в побудові оцінок максимально можливого прискорення процесу рішення задачі конкретного типу (оцінка ефективності паралельного способу рішення задачі).

Ефективний розподіл процесу обчислень між багатьма процесорами й забезпечення їхнього рівномірного завантаження – основна складність паралельного програмування, яке справедливо вважається набагато більш складним, ніж послідовне. За умов нерівномірного завантаженні деякі процесори можуть проводити більшу частину часу чекаючи результату обчислень того з них, на який доводиться максимальне навантаження, й ефективність всієї системи виявляється вкрай низкою. Необхідно також відзначити, що паралельна програма досить тісно прив'язана до типу паралельної архітектури. Паралельні алгоритми дуже чутливі до нюансів тої архітектури, для якої вони реалізовані, тому необхідно ретельне узгодження структури програм і алгоритмів з особливостями конкретної паралельної обчислювальної системи.

Схема паралельного виконання обчислення на кластері.

Хай p – кількість процесорів, використовуваних для виконання алгоритму. Тоді для паралельного виконання обчислень необхідно задати множину (розклад):

$$H_p = \{(i, P_i, t_i) : i \in V\}$$

i – номер операції,
Pi – номер процесора,
ti – час початку виконання i-ї операції.

Повинні виконуватися умови:

- один і той же процесор не повинен призначатися різним операціям в один і той же момент часу: $\forall i, j \in V : t_i = t_j \Rightarrow P_i \neq P_j$;
- до моменту виконання операції, що призначається, всі необхідні дані вже повинні бути обчислені: $\forall (i, j) \in R \Rightarrow t_j \geq t_i + 1$.

Модель паралельного алгоритму:

$$A_p(G, H_p).$$

Обчислювальна схема алгоритму G спільно з розкладом Hp може розглядатися як модель паралельного алгоритму

Ap(G,Hp), що виконується з використанням p процесорів.

Час виконання паралельного алгоритму із заданим розкладом:

$$T_p(G, H_p) = \max_{i \in V}(t_i + 1)$$

Для обраної схеми обчислень бажане використання розкладу, що забезпечує мінімальний час виконання алгоритму

Час виконання паралельного алгоритму з оптимальним розкладом:

$$T_p(G) = \min_{H_p} T_p(G, H_p)$$

Мінімальний можливий час рішення задачі при заданій кількості процесорів (визначення якнайкращої обчислювальної схеми):

$$T_p = \min_{G} T_p(G)$$

Оцінка найбільш швидкого виконання алгоритму (при використанні паркомп'ютера – системи з необмеженим числом процесорів):

$$T_\infty = \min_{p \geq 1} T_p$$

Час виконання послідовного алгоритму для заданої обчислювальної схеми: $T_1(G) = |\overline{V}|$, де, $|\overline{V}|$ є кількість вершин обчислювальної схеми без вершин введення. Важливо відзначити, що якщо при визначенні оцінки обмежитися розглядом тільки одного вибраного алгоритму рішення задачі і використовувати величину

Час виконання послідовного алгоритму:

$$T_1 = \min_{G} T_1(G)$$

Тобто отримувані при такій оцінці показники прискорення характеризуватимуть ефективність розпаралелювання вибраного алгоритму. Для оцінки ефективності паралельного рішення досліджуваної обчислювальної задачі час послідовного рішення слід визначати з урахуванням різних послідовних алгоритмів, тобто використовувати величину $T_1^* = \min T_1$, де операція мінімуму береться по множині всіх можливих послідовних алгоритмів рішення даної задачі.

Мінімально можливий час виконання паралельного алгоритму визначається довжиною максимального шляху обчислювальної схеми алгоритму, тобто:

$T_\infty (G) = D (G)$.

Нехай для деякої вершини виведення в обчислювальній схемі алгоритму існує шлях з кожної вершини введення. Крім того, хай вхідна ступінь вершин схеми (кількість вхідних дуг) не перевищує 2. Тоді мінімально можливий час виконання паралельного алгоритму обмежена знизу значенням:

$T_\infty(G) = log_2 n$,

де N – є кількість вершин введення в схемі алгоритму.

При зменшенні кількості використовуваних процесорів час виконання алгоритму збільшується пропорційно величині зменшення кількості процесорів, тобто:

$Q = CP$, $0<C<1 T_{pc} T_q$.

Для будь-якої кількості використовуваних процесорів справедлива наступна верхня оцінка для часу виконання паралельного алгоритму:

$pT_p < T_\infty + T_1/p$.

Час виконання алгоритму, який можна порівняти з мінімально можливим часом T_∞, можна досягти при кількості процесорів порядку $P \sim T_1 / T_\infty$, а саме:

$pT_1 / T_\infty \; Tp_2 T_\infty$.

При меншій кількості процесорів час виконання алгоритму не може перевищувати більш ніж у 2 рази найкращий час обчислень при наявному числі процесорів.

Наведені твердження дозволяють дати наступні рекомендації з правилами формування паралельних алгоритмів:

$$p < T_1/T_\infty \Rightarrow \frac{T_1}{p} \leq T_p \leq 2\frac{T_1}{p}.$$

Під час вибору обчислювальної схеми алгоритму повинен використовуватися граф з мінімально можливим

діаметром. Для паралельного виконання доцільна кількість процесорів визначається величиною:

$P \sim T1 / T_\infty$.

Нехай H_∞ є розклад для досягнення мінімально можливого часу виконання T_∞. Для кожної ітерації τ, $0 < \tau < T_\infty$, виконання розкладу H_∞ позначимо через $n\tau$ кількість операцій, які виконуються в ході ітерації τ. Розклад виконання алгоритму з використанням P процесорів може бути побудовано таким чином. Виконання алгоритму розділимо на T_∞ кроків, на кожному кроці τ слід виконати всі $n\tau$ операцій, які виконувалися на ітерації τ розкладу H_∞. Ці операції можуть бути виконані не більше, ніж за $\lceil n\tau / p \rceil$ ітерацій при використанні P процесорів. Як результат, час виконання алгоритму TP може бути оцінений наступним чином:

$$T_p = \sum_{T=1}^{T_\infty} \left\lceil \frac{n_{tau}}{p} \right\rceil < \sum_{T=1}^{T_\infty} \left(\frac{n_{tau}}{p} + 1 \right) = \frac{T_1}{p} + T_\infty.$$

Доказ твердження дає практичний спосіб побудови розкладу паралельного алгоритму. Спочатку може бути побудовано розклад без урахування обмеженості числа використовуваних процесорів (розклад для паракомп'ютера). Потім, згідно з схемою виведення теореми, може бути побудовано розклад для конкретного кількості процесорів.

До показників ефективності паралельного алгоритму можна віднести прискорення (*speedup*), що отримується при використанні паралельного алгоритму для p процесорів, в порівнянні з послідовним варіантом виконання обчислень визначається величиною:

$S_p(n)=T_1(n)/T_p(n)$

Тобто як відношення часу рішення задач на скалярній ЕОМ до часу виконання паралельного алгоритму (величина n застосовується для параметризації обчислювальної складності вирішуваної задачі і може розумітися, наприклад, як кількість вхідних даних завдання). Ефективність (*efficiency*) використання паралельним алгоритмом процесорів при вирішенні задачі визначається співвідношенням:

$E_p(n)=T_1(n)/(pT_p(n))=S_p(n)/p$

Величина ефективності визначає середню частку часу виконання алгоритму, протягом якої процесори реально задіяні для вирішення завдання.

З приведених співвідношень можна стверджувати, що в якнайкращому випадку $S_p(n)=p$ і $E_p(n)=1$. При практичному застосуванні даних показників для оцінки ефективності паралельних обчислень слід враховувати два важливі моменти:

1. При певних обставинах прискорення може бути більше числа процесорів, що використовуються, $S_p(n)>p$ – в цьому випадку говорять про існування надлінійного (*superlinear*) прискорення. Не дивлячись на парадоксальність таких ситуацій (прискорення перевищує число процесорів), на практиці надлінійне прискорення може мати місце. Однією з причин такого явища може бути неоднаковість умов виконання послідовної і паралельної програм.

2. При детальному розгляді можна звернути увагу, що спроби підвищення якості паралельних обчислень по одному з показників (прискоренню або ефективності) можуть привести до погіршення ситуації по іншому показнику, бо показники якості паралельних обчислень є часто суперечливими. Так, наприклад, підвищення прискорення зазвичай може бути забезпечене за рахунок збільшення числа процесорів, що приводить, як правило, до падіння ефективності. І навпаки, підвищення ефективності досягається у багатьох випадках при зменшенні числа процесорів (у граничному випадку ідеальна ефективність $E_p(n)=1$ легко забезпечується при використанні одного процесора). Як результат, розробка методів паралельних обчислень часто припускає вибір деякого компромісного варіанту з урахуванням бажаних показників прискорення і ефективності.

Оцінка якості паралельних обчислень припускає знання якнайкращих

(максимально досяжних) значень показників прискорення і ефективності, проте отримання ідеальних величин $S_p=p$ для прискорення і $E_p=1$ для ефективності може бути забезпечене не для всіх обчислювально-трудомістких завдань.

Практично в будь-якій програмі є деякий відсоток операцій, що не допускають паралельного виконання. Позначимо його через α. Очевидно, відсоток операцій, що допускають паралельне виконання, дорівнює 1–α. Максимальний приріст продуктивності, який можна одержати від паралельного виконання програми з такими характеристиками на машині з N процесорами в порівнянні з однопроцесорної ЕОМ, виражається законом Амдала, який звучить наступним чином:

Досягненню максимального прискорення може перешкоджати існування у виконуваних обчисленнях послідовних розрахунків, які не можуть бути розпаралелюванні. Хай f є частка послідовних обчислень у вживаному алгоритмі обробки даних, тоді відповідно до закону Амдала (Amdahl) прискорення процесу обчислень при використанні процесорів обмежується величиною:

$$S_p \le \frac{1}{f+(1-f)/p} \le S^* = \frac{1}{f}.$$

Випадок $f = 0$ відповідає повністю паралельній програмі й ми одержуємо N-кратний приріст, випадок $f = 1$ – повністю послідовної, і в цьому випадку приросту немає. Закон Амдала деякою мірою допомагає відчути складність паралельного програмування: наприклад, для прискорення виконання програми в 100 разів, необхідно, щоб 99,99% операцій в програмі можливо було б виконувати з 100-кратним розпаралеленням.

Устремління числа процесорів N у нескінченність приводить до очевидного результату: $S(\infty,f)=1/f$, тобто принципово неможливо одержати прискорення більше $1/f$ при будь-якій кількості використовуваних процесорів. Втім, цей песимістичний прогноз на практиці часто не виправдується. Було помічено, що параметри N і f не є незалежними для багатьох задач. Іншими словами, багато класів обчислювальних задач є масштабованими: комп'ютер з більшою кількістю процесорів дозволяє реалізовувати більш детальні обчислення, збільшувати їхню точність тощо, що часто призводить до зменшення.

Висновки

На сьогодні існує широкий спектр кластерів, які можна використовувати під час виконання великих обсягів обчислень. Вони відрізняються типом і швидкодією процесорів, розміром поділюваної вузлами пам'яті, технологією взаємозв'язку вузлів, моделями й інтерфейсами програмування. Однак результат, що досягається з їх допомогою, значно залежить від особливостей додатків, які планується на них розгорнути. Існує можливість побудови на базі кластерних технологій СБД, які зможуть ефективно обробляти великі масиви інформації таких задач, як моделювання траєкторії літаків, обробка даних чорних ящиків, обчислення великих масивів економічної інформації тощо.

Література

1. *Жуков І.А., Іванкевич О.В.* Аналіз використання кластерних технологій у системах керування розподіленими базами даних на сучасних авіапідприємствах // Проблеми інформатизації та управління. – Вип. 2(26). – К.: НАУ, 2009. – С. 45-51.

2. *Иванкевич А.В.* Использование специализированных программных комплексов на базе распределенных хранилищ данных авиапредприятий Украины // Проблеми інформатизації та управління. – Вип. 1(23). – К.: НАУ, 2008. – С. 138-142.

3. *Иванкевич А.В., Аль Шибани Салим.* Метод повышения производительности серверов для работы с распределенными базами данных // Наука і молодь: Збірник наукових праць міжнародної наукової конференції "Політ-2007". – К.: НАУ, 2007. – С. 50.

4. *Жуков И.А., Гуменюк А.В.* Перспективы использования кластерных вычислительных систем в авиации // Вісник Київського міжнародного університету цивільної авіації. – К., 1999. – № 2. – С. 96-100.

5. *Креденцар С.М.* Перспективы применения параллельных вычислений и кластерных вычислительных систем в системах отображения воздушной обстановки// Матеріали VII міжнародної науково-технічної конференції "АВІА-2006", 25-27 вересня 2006. – К.:НАУ, 2006. – Т. 1.– С. 21.113-21.116.

6. *Корочкин С.* Организация вычислений в кластерных ситсемах с многоядерной архитектурою // Проблеми інформатизації та управління. – К.:НАУ, 2008. – Вип. 1(23). – С. 143-145.

7. *Гуменюк В.А.* Технології кластерних архітектур // Проблеми інформатизації та управління. – К.:НАУ, 2004. – Вип. 10. – С. 151–156.

8. *Иванкевич А.В., Аль-Сурики Ибрагим.* Использование распределенных баз данных в информационных системах авиапредприятий Украины // Труды восьмой международной научно-практической конференции “Современные информационные и электронные технологии”. – Одесса.: ОНПУ, 2007. – С. 42

9. *Жуков И.А., Иванкевич А.В., Салим Аль Шибани.* Средства повышения эффективности обработки баз данных большого объема в информационных системах авиапредприятий Украины. // Інформаційно-діагностичні системи: Матеріали IX Міжнародної науково-технічної конференції "АВІА-2007". – К.: НАУ, 2007. – Т. 1. – С. 13.37-13.40.

10. *Иванкевич А.В., Салим Аль Шибани.* Организация системы распределенной обработки запросов к серверам баз данных в компьютерных сетях // Збірник наукових праць за результатами міжнародної науково-практичної конференції "Мікропроцесорні пристрої та системи в автоматизації виробничих процесів". – № 3. – Хмельницький: Технологічний університет Поділля, 2007. – Т.1. – С. 82-85.

Іванкевич О.В., Мазур В.І.

БАЛАНСУВАННЯ РОЗПОДІЛЕНОЇ ОБРОБКИ ВЕЛИКИХ МАСИВІВ ІНФОРМАЦІЇ У ДЕЯКИХ СУЧАСНИХ СИСТЕМАХ УПРАВЛІННЯ БАЗАМИ ДАНИХ

Запропоновано алгоритм балансування розподіленої обробки великих масивів інформації за допомогою кластерної системи, що може використовуватися у деяких сучасних системах управління базами даних. Алгоритм призначений для рівномірного розподілу навантаження по вузлах кластера при обробці великих обсягів даних. Досліджено особливості підвищення продуктивності кластерних систем при вирішенні задач авіаційної галузі за допомогою розподілених баз даних.

Ivankevich O.V., Mazur V.I.

BALANCING OF DISTRIBUTED PROCESSING OF LARGE ARRAYS OF INFORMATION IN SOME MODERN DATABASE MANAGEMENT SYSTEMS

An algorithm for balancing distributed processing of large information arrays using a cluster system is proposed, which can be used in some modern database management systems. The algorithm is designed to evenly distribute the load on cluster nodes when processing large volumes of data. The peculiarities of increasing the productivity of cluster systems in solving problems of the aviation industry with the help of distributed databases have been studied.