УДК 004.272.43(045)

Жуков І.А., д.т.н.,
Іванкевич О.В., к.т.н.,
Єфимець В.М., к.т.н.

РОЗПОДІЛ НАВАНТАЖЕННЯ У СИСТЕМАХ КЕРУВАННЯ БАЗАМИ ДАНИХ ВЕЛИКИХ РОЗМІРІВ НА БАЗІ КЛАСТЕРІВ НА ПІДПРИЄМСТВАХ ЦИВІЛЬНОЇ АВІАЦІЇ

zhukov@nau.edu.ua, ntb@nau.edu.ua

Запропоновано алгоритм балансування кластерної системи, 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

Ключові слова: кластер, навантаження вузлів кластера, розподілена база даних, балансування навантаження, підвищення продуктивності системи, система баз даних, алгоритм балансування

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 порівняно дешевих, але потужних систем БД (СБД) [1-5].

Кластери складаються з серійних компонентів, що мають невелику вартість та є альтернативою монокорпусним суперкомп'ютерам з оригінальною закритою архітектурою. Такі системи часто застосовуються для виконання високопродуктивних обчислень, забезпечення доступності й масштабованості [1-7].

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 [4-5, 8-10].

Мета

Метою статті є огляд системи, призначеної для балансування розподілу інформаційних масивів даних по вузлах кластерної системи при обробці великих обсягів даних, дослідженню особливостей підвищення продуктивності кластерних систем при використанні великих БД.

Підсистеми кластера можна «побачити» у єдиному адміністративному домені, і керування ними виконується, як єдиною обчислювальною системою. Вузли кластера можуть бути одно- або мультипроцесорними [6-7]. У класичній схемі всі вузли при роботі з додатками розділяють зовнішню пам'ять на масиві жорстких дисків, використовуючи внутрішні диски для спеціальних функцій. Для міжвузлової взаємодії зазвичай застосовується стандартна мережева технологія, хоча це не виключає окремо розроблених каналів зв'язку. Кластерна мережа є відособленою, тобто вона ізольована від зовнішнього мережного середовища [4-7].

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.

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 приско-

рення процесу обчислень (скорочення часу вирішення задачі). Формування подібних оцінок прискорення може здійснюватися стосовно вибраного обчислювального алгоритму (оцінка ефективності розпаралелювання конкретного алгоритму). Інший 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 вузлам кластера й забезпечення їхнього рівномірного завантаження – основна складність паралельного програмування, яке справедливо вважається набагато більш складним, ніж послідовне [5-6]. За умов нерівномірного завантаження деякі вузли можуть проводити більшу частину часу чекаючи результату обчислень того з них, на який доводиться максимальне навантаження, тому ефективність всієї системи виявляється вкрай низькою. Необхідно також відзначити, що паралельна програма прив'язана до типу паралельної архітектури. Паралельні алгоритми чутливі до нюансів тієї 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

Позначимо через p кількість вузлів кластера, що використовуються для виконання алгоритму. Тоді для паралельного виконання обчислень необхідно задати множину (розклад):

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

де i – номер операції;

P_i – номер вузла;

t_i – час початку виконання i-ї операції.

Повинні виконуватися умови:

– один і той же вузол не повинен призначатися різним операціям одночасно:

$$\forall i, j \in V : t_i = t_j \Rightarrow P_i \neq P_j;$$

– до початку виконання операції, 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 H_p може розглядатися як модель паралельного алгоритму $A_p(G,H_p)$, що виконується з використанням p вузів.

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:

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

Мінімально можливий час рішення задачі при заданій кількості вузлів (визначення якнайкращої обчислювальної схеми):

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

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

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

Час виконання послідовного алгоритму для заданої обчислювальної схеми:

$T_1(G) = \left|\overline{V}\right|$, де $\left|\overline{V}\right|$ 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 часу виконання послідовного алгоритму:

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

то отриманні 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

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

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

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

де n – є кількість вершин введення в схемі алгоритму.

При зменшенні кількості вузлів, що використовуються, час виконання алгоритму збільшується пропорційно величині зменшення кількості вузлів, тобто:

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

Для кількості вузлів, що використовуються більше 4, 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 вузлів порядку $P \sim T_1 / T_\infty$, а саме:

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

При кількості вузлів 1-4 час виконання алгоритму не може перевищувати більш ніж у 2 рази найкращий час обчислень при наявному числі вузлів.

Наведені твердження дозволяють дати наступні рекомендації до правил формування паралельних алгоритмів:

$$p < T_1 / T_\infty \Rightarrow \frac{T_1}{p} \le T_p \le 2\frac{T_1}{p},$$

– при виборі схеми алгоритму повинен використовуватися граф з мінімально можливим діаметром;

– для паралельного виконання доцільна кількість вузлів визначається величиною.

$$P \sim T1 / T_\infty.$$

Припустимо, 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 вузлів 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 $[\ n\tau/p\ ]$ ітерацій при використанні P вузлів. 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 вузлів, потім, згідно з схемою виведення, може бути побудовано розклад для конкретної кількості вузлів.

До показників ефективності паралельного алгоритму можна віднести прискорення, що отримується при використанні паралельного алгоритму для p вузлів, в порівнянні з послідовним варіантом виконання обчислень:

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

Тобто як відношення часу рішення задач на скалярній електронній обчислювальній машині до часу виконання паралельного алгоритму (величина n застосовується для параметризації обчислювальної складності вирішуваної задачі і може

розумітися, наприклад, як кількість вхідних даних завдання). Ефективність використання паралельним алгоритмом вузлів 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 вузли реально задіяні для вирішення завдання.

З приведених співвідношень можна стверджувати, що в кращому 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:

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

– при детальному розгляді необхідно 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 вузлів, що приводить, як правило, до падіння ефективності. І навпаки, підвищення ефективності досягається у багатьох випадках при зменшенні числа вузлів (у граничному випадку ідеальна ефективність $E_p(n)=1$ легко забезпечується при використанні одного вузла). 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-якому алгоритмі 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 електронної машини, виражається законом Амдала, який звучить наступним чином.

Досягненню максимального прискорення може перешкоджати існування у виконуваних обчисленнях послідовних розрахунків, які не можуть бути розпаралелюванні. Нехай f є частка послідовних обчислень у алгоритмі обробки даних, тоді відповідно до закону Амдала прискорення процесу обчислень при використанні вузлів 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.

Збільшення числа вузлів N до нескінченності приводить до очевидного результату: $S(\infty, f)=1/f$, тобто принципово неможливо одержати прискорення більше $1/f$ при будь-якій кількості використовуваних вузлів. Втім, цей прогноз на практиці часто не виправдується. Було помічено, що параметри n і f не є неза-

лежними для багатьох задач. Іншими словами, багато класів обчислювальних задач є масштабованими. Кластер з більшою кількістю вузлів дозволяє реалізовувати більш детальні обчислення, збільшувати їхню точність тощо.

Висновки

У статті розглянуто можливість застосування кластерів різних класів з різною кількістю вузлів для рішення задач, що потребують великих обсягів обчислень. Проведено аналіз кластерної системи, призначеної для балансування розподілу інформаційних масивів між вузлами при обробці великих обсягів даних у таких задачах як моделювання траєкторій літаків, обробки даних чорних ящиків, обчислення великих масивів економічної інформації тощо. Досліджено особливості підвищення продуктивності кластер них систем. Запропоновано алгоритм балансування, призначений для рівномірного розподілу навантаження по вузлах кластера при обробці великих обсягів даних. Показано, що результат, що досягається з їх допомогою, суттєво залежить від особливостей додатків, які планується на них розгорнути.

Список літератури

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 КМУЦА. – 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 р.– К.: НАУ, 2006. – Т. 1.– С. 21.113-21.116.

6. Корочкин С. Организация вычислений в кластерных ситсемах с многоядерной архітектурою // Проблеми інформатизації та управління: збірник наукових праць. – 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., Аль Шибани Салим. 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., Аль Шибани Салим. 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.

Статтю подано до редакції 19.02.2014