УДК 004.7(045)

Кременецький Г.М., к.т.н.

МЕТОДИ ПІДВИЩЕННЯ ПРАЦЕЗДАТНОСТІ СПЕЦІАЛІЗОВАНИХ ОБЧИСЛЮВАЛЬНИХ МЕРЕЖ ПРИ ДИНАМІЧНІЙ КЛАСТЕРИЗАЦІЇ

Національного авіаційного університету

Розглянуто проблеми кластерізації спеціалізованих обчислювальних мереж і методи підвищення їх працездатності. Запропоновано метод розширення зони мінімальної взаємодії у поверхневому шарі з поступовою кластеризацією. Досліджено використання WEB-сервісів для побудови динамічної кластерної нейронної мережі. Проведено аналіз особливостей WEB-сервісів. Розглянуто їх позитивні та негативні боки щодо використання

Вступ

Однією з важливих проблем є об'єднання обчислювальних ресурсів. Не дивлячись на те, що мережі довели свою практичну корисність як засіб глобальної передачі різних форм інформації, їх потенціал використовується не повністю. Засобом ефективної організації обчислень можуть стати спеціалізовані мережі, які об'єднують високопродуктивні комп'ютерні системи для вирішення задач підвищеної складності або ресурсоємності – метакомп'ютинг.

Існує два основних напрями підвищення швидкодії обчислювальних систем:

1) створення продуктивних вузлів;

2) розпаралелювання виконуваного завдання між декількома обчислювальними вузлами.

Однак більш перспективним напрямом є використання систем балансування навантаження [1]. Балансування навантаження застосовується для *WEB*-серверів, *FTP*-серверів, серверів баз даних, а також в кластерних системах. Особливо важливим питання балансування навантаження постає в динамічних кластерних системах, коли на основі показників завантаженості вузлів може змінюватись структура обчислювальної системи. Тому основною метою даного дослідження є аналіз методів підвищення працездатності спеціалізованих обчислювальних мереж.

Необхідність застосування систем балансування навантаження обумовлена наявністю великої кількості легко розпаралелюючих завдань, що вимагають значної кількості обчислювальних ресурсів.

Використання нейронних мереж для розподілу обчислювального навантаження

Для задач, які розбиваються на елементарні операції, або для випадків обмеженого кола задач, для якого можна одного разу визначити функцію оптимізації, існують методики, що розв'язують дану проблему з певним рівнем точності. Балансування навантаження, для систем з динамічно змінною безліччю задач, що складно розбиваються на елементарні операції, існуючими методиками, або утруднена, або практично неможлива.

Метою роботи є визначення концепції застосування нейронних мереж (НС) для вирішення існуючої проблеми.

У загальному випадку задачу можна представити у вигляді суми підзадач:

$$T = \sum_{l=1}^{L} \sum_{i=1}^{N} b_l t_{li}, \qquad (1)$$

де L - кількість підзадач даної задачі; N - кількість обчислювальних вузлів; t_{li} - час виконання i-ої підзадачі на l-му комп'ютері; b - бітовий вектор в якому b_i=0, якщо завдання не виконуватиметься на i-му обчислювальному вузлі і b_i=1 інакше.

З формули видно, що якщо підібрати такий вектор, щоб були мінімальні, то і загальний час виконання завдання буде мінімальним.

Розглянемо особливості існуючих методів.

У разі, коли завдання легко розбивається на елементарні операції час рішення задачі легко

визначити по формулі:

$$t = \sum_{j=1}^{J} t_{op_j} + \sum_{j=1}^{J-1} t_{wait_j}, \qquad (2)$$

де t_{op_j} - час виконання j-ої операції: t_{wait_j} - час очікування на перемикання між операціями.

Існує декілька моделей попереднього визначення часу виконання підзадачі на основі статистичної інформації.

В [2] запропонована наступна модель. Вся безліч задач розбивається на підмножини по рівню навантаження, потім моделюється сумарна функція розподілу часу виконання для кожної підмножини, далі ця функція використовується для визначення часу рішення задачі. В результаті дослідження сумарної функції розподілу часу виконання завдань Доуней дійшов висновку, що вона має вигляд

$$P\{L < t\} = \beta_0 + \beta_1 \ln t. \qquad (3)$$

Після обчислення коефіцієнтів цієї функції використовується дві різні методики обчислення часу вирішення задачі. Перша з них грунтується на обліку середнього часу виконання і обчислюється за формулою:

$$\sqrt{ae^{\frac{1-\beta_0}{\beta_1}}}. \qquad (4)$$

Друга методика використовує умовний середній час життя

$$\frac{t_{\max} - a}{\log t_{\max} - \log a}, \qquad (5)$$

де

$$t_{\max} = e^{\frac{1-\beta_0}{\beta_1}}. \qquad (6)$$

Перевагою методики Доунея є її простота і висока швидкість виконання оцінки. Недоліком служить використання лише однієї статистичної характеристики для вже відомого набору завдань, що знижує достовірність отриманих результатів.

Відповідно до методу Гіббонса [3] зібрана інформація використовується для попереднього визначення часу виконання задачі. При цьому застосовується фіксований набір шаблонів і різні характеристики для вибору шаблону.

До переваг даної методики відноситься простота її реалізації і більш висока (у порівнянні з попереднім методом) точність. Недоліком є необхідність визначення шаблонів для фіксованого кола задач.

До складу системи входить кілька модулів: сервера, агентів і клієнтської частини. Ці модулі реалізовані із застосуванням мови програмування *Java*, що дозволяє використовувати їх на різних апаратних платформах і в різноманітних операційних системах.

Під час подачі на виконання деякої задачі клієнт посилає набір певних параметрів на сервер. Сервер подає на вхід нейронної мережі ці значення і дані про завантаження, одержані від агента з i-го комп'ютера. Цей процес повторюється для i=1..N, де N – кількість комп'ютерів в мережі. На виході нейронної мережі визначається час вирішення задачі для кожного з N комп'ютерів, з них вибирається мінімальне значення. Клієнт одержує інформацію про адресу комп'ютера, з мінімальним часом вирішення задачі, а також її унікальний номер. Після отримання цієї інформації від сервера клієнт посилає завдання на вказаний комп'ютер для виконання. Після закінчення рішення номер задачі, а також час виконання відправляються на сервер для подальшого навчання нейронної мережі.

Методи розподілу навантаження між серверами

Для розподілу навантаження між *WEB*-серверами використовуються три методи балансування: циклічної вибірки, тестового навантаження і метод, заснований на використанні спеціальних програм-агентів.

Метод циклічної вибірки є циклічним перенаправленням запитів клієнтів між декількома серверами за допомогою спеціального оголошення у файлі настройок *DNS* сервера. Запит клієнта спочатку поступає на *DNS*-сервер для дозволу імені комп'ютера в *IP* адресу. У конфігурації *DNS*-сервера цьому імені відповідають декілька *IP*-адрес, що призводить до того, що *IP*-адреси видаються по черзі при кожному новому запиті. Гідністю цього методу є простота реалізації, а також відсутність необхідності використання дорогого програмного забезпечення або

спеціалізованих апаратних систем. Методу характерні наступні недоліки: не враховується навантаження комп'ютерів, що приводить до нерівномірного її розподілу, а значить і до неефективного використання ресурсів, у випадку, якщо один з комп'ютерів взагалі не працює, то частина запитів залишиться не обслуженою, при зверненні клієнта по реальній *IP*-адресі не здійснюється балансування.

Метод тестового навантаження характеризується тим, що перед напрямом клієнта на один з серверів, посилається тестовий запит на сервера що беруть участь в процесі балансування, і той з них, який швидше відповість і буде визнаний менш завантаженим. Даний метод володіє перевагою найбільш точного визначення оптимального сервера для виконання запиту, а як недолік можна назвати найвищий рівень завантаження системи, в порівнянні з іншими.

Метод, заснований на використанні спеціальних програм-агентів, що запускаються на серверах, і що інформують «балансувальника» навантаження про завантаженість систем, представляє комбінацію двох попередніх методів. Тому цей метод володіє перевагами першого методу – не вимогливістю до ресурсів, і другого – достатньо високою точністю. Недоліком методу є складність розробки точних і невимагаючих великої кількості ресурсів алгоритмів визначення навантаження.

Конфігурація системи балансування представлена на рис. 1.

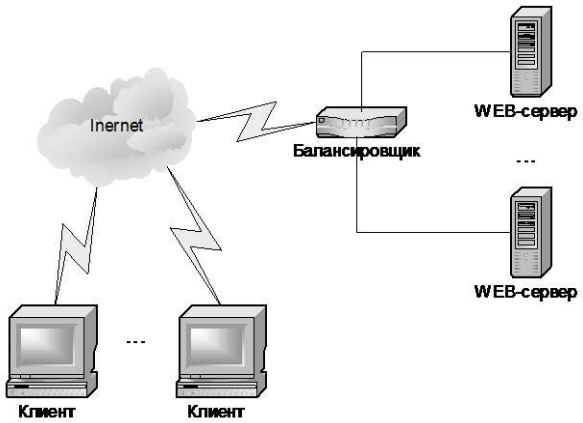

Рис. 1. Схема реалізації конфігурації системи з використанням «балансувальника»

Як «балансувальник» можна використовувати комп'ютер з мінімальною апаратною конфігурацією (i486, *RAM – 32Mb*, *HDD – 1Gb*), на якому встановлена операційна система *Linux*, *WEB*-сервер *Apache*, конфігурований з використанням модуля перезапису адрес (*mod_rewrite*). Як умова перенаправлення служить скрипт балансування навантаження, що реалізовує один з останніх двох методів.

Основними відмінностями реалізацій є протоколи передачі службової інформації, алгоритми визначення завантаження залучених в систему розподіленої обробки комп'ютерів, а також алгоритми розподілу навантаження між ними. Графік залежності середнього часу обслуговування T_c від кількості запитів N для *WEB*-сервера приведений на рис. 2.

Графік має вид експоненти для всіх трьох методів. На ділянці від 0 до N_{opt} запитів час обслуговування практично не зростає, оскільки при цьому навантаження невелике навіть для одного комп'ютера. Після N_{opt} запитів графік для першого методу починає зростати різкіше, ніж для інших у зв'язку з невисокою оптимальністю даної методики. Частина запитів перенаправляється на менш швидкодіючий комп'ютер, що приводить до збільшення середнього часу затримки всієї системи. З графіка виходить, що найбільш ефективним можна рахувати метод з використанням баласувальника.

Алгоритму обчислення рекомендованого завантаження можна представити наступною формулою:

$$F(t) = \sum_{i=0}^{N} a_i f_i(t), \qquad (7)$$

де a_i - коефіцієнти, що підбираються для конкретного завдання; $f_i(t)$ - функція визначення завантаження елементу комп'ютера; N – кількість елементів, що істотно впливають на виконання завдання; t - час виміру.

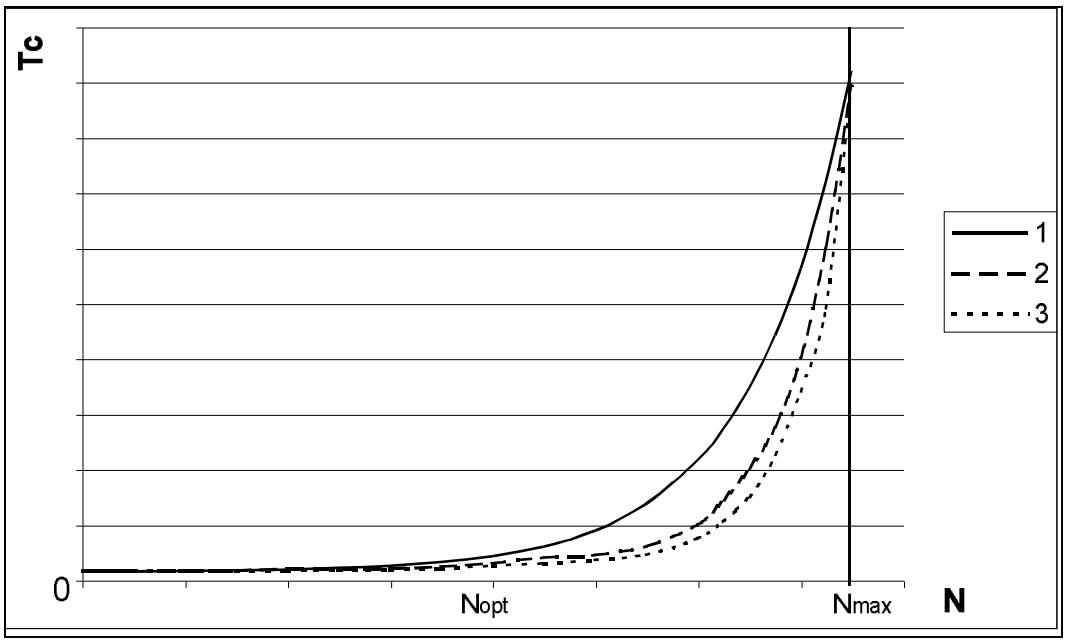

Рис. 2 Залежність середнього часу від кількості запитів
для різних методів розподілу навантаження
(1 – метод циклічної вибірки, 2 – метод з використанням тестового навантаження,
3 - метод з використанням «балансувальника»)

Для комп'ютера з максимальним значенням цієї формули виконання запиту буде оптимальним. Технологія пошуку об'єкту в мережі схожа з пошуком комп'ютера за допомогою *DNS*. Існує сервер імен об'єктів, до якого прямує запит клієнта на створення і роботу з якимсь об'єктом. В результаті запиту клієнт одержує об'єктне посилання (якщо об'єкт існує), що унікально ідентифікує об'єкт в мережі. Далі через заглушки (стаби), що згенеровані на основі інтерфейсу, здійснюється робота з об'єктом.

На даному етапі можна використовувати технології подібні тим, що описувалися вище для *WEB*-серверів, тобто так само застосовно вираз (7). На *Case* діаграмі, представленій на рис. 3, показана загальна схема функціонування даної моделі.

Припустимо, що в нашому розпорядженні *M*-комп'ютерів виділено для вирішення поставленого завдання. На кожному з них необхідно запустити агента завантаження і фабрику об'єктів. Користувач, посилаючи запит півночі імен на об'єкт певного типа, викликає наступну послідовність дій:

- на підставі інформації одержаною від агентів вибирається комп'ютер, на якому буде створений об'єкт необхідного типа;
- посилається запит на створення об'єкту відповідній фабриці;
- фабрика створює об'єкт;
- посилання на об'єкт відправляється клієнту;
- клієнт працює з об'єктом;
- після завершення роботи об'єкт руйнується.

Рішення задачі оптимального балансування навантаження об'єктів приведе до створення міжплатформеного розподіленого середовища розробки. У якій програміст при написанні програми може вирішувати задачі розпаралелювання між іншими машинами на рівні об'єктної моделі його мови програмування [4, 5].

Крім того, представляється можливим розробка методології і алгоритмів динамічної адаптації подібного роду систем на рівні кластерів.

У процесі розподілення обчислень між великою кількістю обчислювальних вузлів, можна зіткнутися із проблемою обмеженої швидкості передачі даних в комп`ютерних мережах.

Динамічний розподіл обчислень, а також, зміна алгоритмів навчання та передатних функцій нейронної мережі, дозволить створювати нейронні мережі, які будуть автоматично налаштовуватися на задачу, що вирішується, і розвиватися в процесі роботи без втручання людини, розподіляючись між вузлами обчислювальної мережі.

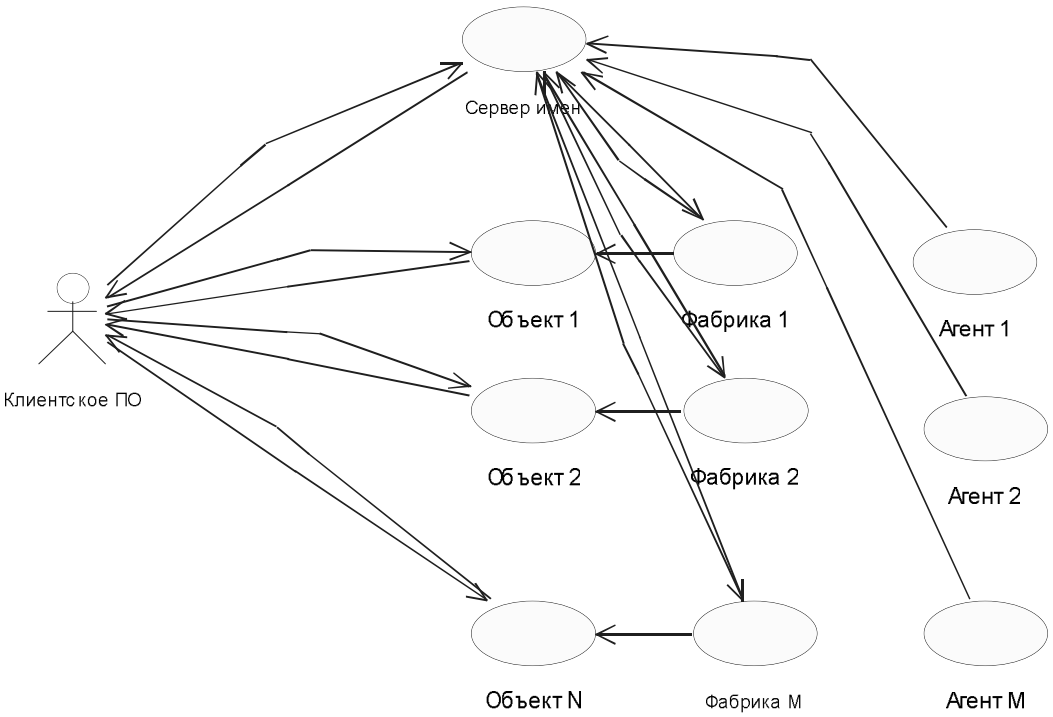

Рис. 3. *UML Case*-діаграма загальної схеми функціонування моделі з використання «балансувальника»

Підвищення працездатності обчислювальних систем з динамічною структурою

Проблему оптимізації навантаження вузлів обчислювальної мережі було вирішено в роботах [6, 7]. Дана задача вирішувалась за допомогою наступної функції оптимізації:

$$node = \begin{cases} \min\limits_{i=1}^{N^C}[L(c_i)], \\ \min\limits_{l=1}^{N^R}[V(r_l)], l = \{j,k\}, \end{cases}$$

де r_l – зв'язок між c_j та c_k кластерами, $L(c_j)$ – обчислювальне навантаження на *i*-й кластер, $V(r_l)$ – потрібна швидкість каналу для зв'язку r_l, N^C – кількість кластерів, N^R – кількість зв'язків між кластерами, $N^R \le \dfrac{N^C \cdot (N^C - 1)}{2}$.

Якщо $L^{c_i} \ge L^{Nd_k}$ (L^{c_i} – навантаження, яке потрібно для кластера, L^{Nd_k} – можливості вузла), то міграція частини обчислювань на вільний вузол Nd_m $(m \ne k)$.

Якщо $V^{r_{ij}} \ge V^{Ch_l}$, де $V^{r_{ij}}$ – необхідна пропускна здатність для існуваня зв'язків між класетрами, V^{Ch_l} – пропускна здатність каналa, то міграція частини обчислювань на вільний вузол Nd_o, де $Nd_o \notin \{Nd(Ch_p)\}$ тобто на вузол, що не має зв'язків, які використовують перевантажені канали V^{Ch_p}.

Також можна використовувати поперечне розділення у площині, що паралельна площині входів, яке відбувається формуванням передатних адаптерів в зоні прогнозованої стабілізації росту мережі, яка визначається або відсутністю росту взагалі, або ростом мережі нижче визначеного коефіцієнту.

Аналогічний процес відбувається і в перпендикулярних площинах мігруючої частини – повздовжнє розділення.

З огляду на [6] можна стверджувати про ефективність використання модифікованих алгоритмів динамічного зростання обчислювальних мереж, які передбачають динамічну кластеризацію вузлів обчислювальної мережі.

До даних алгоритмів можна віднести:

1) адаптований алгоритм розділення багатошарової нейронної мережі з урахуванням можливості динамічної кластеризації вузлів;

2) адаптований до багатошарової нейронної мережі алгоритм зростання.

Висновки

Для вирішення задачі підвищення працездатності роботи системи було запропоновано використовувати розпаралелювання виконання задач з подальшим динамічним розділенням спеціалізованої обчислювальної мережі на окремі кластери. Це дозволить мінімізувати потоки даних між кластерами при зростанні мережі за рахунок розроблених алгоритмів в межах *froth*-методу [7] динамічного зростання обчислювальної мережі.

В межах даного методу було побудовано два алгоритми:

1) адаптований до багатошарової нейронної мережі алгоритм зростання;

2) алгоритм моніторингу обмежень та міграції.

Алгоритм моніторингу обмежень та міграції у *froth*-методі призначений для використання у багатошарових штучних нейронних мережах. Він не порушує структуру мережі, тому після побудови кластерів залишається можливість використання алгоритмів навчання багатошарових мереж, наприклад, алгоритм зворотного розповсюдження помилки.

При вирішенні задачі динамічної кластерацації доведено ефективність використання програм-агентів для балансування навантаження вузлів мережі. Метод, заснований на використанні спеціальних програм-агентів, що запускаються на серверах, і що інформують «балансувальника» навантаження про завантаженість систем. Недоліком даного методу є складність розробки точних і невимагаючих великої кількості ресурсів алгоритмів визначення навантаження.

Список літератури

1. Chandra Kopparapu. Load Balancing Servers, Firewalls & Caches, John Wiley & Sons, – 2002. – 224 p.

2. Downey A.B. Predicting queue times on space-sharing parallel computers. – In: IPPS, Washington, USA. – April 1997. – P. 209-218.

3. Richard Gibbons. A Historical Application Profiler for Use by Parallel Schedulers.– Lecture Notes on Computer Science. – 1997. – P. 58-75.

4. Robert Orfali and Dan Harkey. Programming with Java and CORBA. – 1998. – 712 p.

5. Object Management Group The Common Object Request Broker: Architecture and Specification. – 2001.

6. Кременецкий Г.Н. Программно-апаратные средства организации сетей вычислительных ресурсов // Кременецкий Г.Н., Иванкевич А.В. - Проблеми інформатизації та управління: – Зб. наук. пр. – К.: НАУ, – 2002. – Вип. 5. – С. 124-127.

7. Кременецький Г.М. Модифікація алгоритмів росту для спеціалізованих обчислювальних мереж // Кременецький Г.М., Артамонов Є.Б. - Проблеми інформатизації та управління: – Зб. наук. пр. – К.: НАУ, – 2011. – Вип. 4 (36) – С. 19-23.