УДК 004.421

К.т.н., доцент Петрашенко А.В., студент Ісаков В.В.

Національний технічний університет України
«Київський політехнічний інститут»

ЦЕНТРАЛІЗОВАНИЙ АЛГОРИТМ БАЛАНСУВАННЯ ДЛЯ РОЗПОДІЛЕНОГО ДОДАТКУ

Abstract

Andrij V. Petrashenko, assoc. prof., PhD; Viktor Isakov, student
A centralized balancing algorithm for a distributed application

This article reveals the main features of the algorithm for balancing of a distributed application. The main difference of this algorithm is balancing a distributed system, which has a non-permanent structure of both client and server side. Therefore, the system always adapts its structure that enables it to optimally use the provided computing power.

Вступ

У зв'язку із широким використанням обчислювальної техніки для вирішення фундаментальних наукових та інженерних задач виникає необхідність у створенні високопродуктивних комп'ютерних систем. Одним із видів таких систем є розподілені комп'ютерні системи, що можуть складатися із великої кількості комп'ютерів відносно малої потужності, але загальна потужність таких систем може бути вищою, ніж у деяких суперкомп'ютерів [1,2]. Перевагою таких систем є відносно невеликі затрати на їх побудову та експлуатацію, а властивість масштабування дозволяє нарощувати потужність за рахунок приєднання нових обчислювальних машин.

Важливим елементом кожної розподіленої системи є балансувальник, що дозволяє ефективно використовувати усі обчислювальні потужності системи. В даній статті подано алгоритм централізованого динамічного балансування для розподіленої системи, що має нерегулярну структуру.

Постановка задачі

Задача полягає в розробці централізованого алгоритма динамічного балансування навантаження, який дозволить оптимально використовувати

обчислювальні потужності розподіленої системи за рахунок зміни структури як клієнтської, так і серверної сторони.

Задача динамічного балансування навантаження

Кожний метод динамічного балансування повинен періодично оцінювати інформаційну завантаженість кожного вузла системи. Тому головним завданням при розробці алгоритма є визначення критеріїв оцінки завантаженості [3]. В загальному випадку до критеріїв завантаження сервера та клієнта відносять: продуктивність використання процесору, час його простою, використання пам’яті, час пересилання даних між клієнтом та сервером тощо.

Метою розробників алгоритмів балансування є оптимальне використання наданих ресурсів для якнайшвидшого вирішення поставлених задач.

Короткий опис представленої розподіленої системи

Топологія розподіленої системи, для якої розроблюється алгоритм, має структуру, зображену на рис.1. Дана структура має 3 рівні: клієнтський рівень, серверний рівень та рівень централізованого динамічного балансувальника. Її особливістю є змінна кількість як клієнтів, так і серверів. Сервером у даному випадку вважається комп’ютер, на якому запущений серверний додаток, а не окремо виділена обчислювальна машина. У зв’язку з таким трактуванням поняття «сервер» стає можливим одночасне використання одного комп’ютера у вигляді як сервера, так і клієнта. Централізований динамічний балансувальник запускається на фізичному сервері та працює впродовж всієї роботи системи.

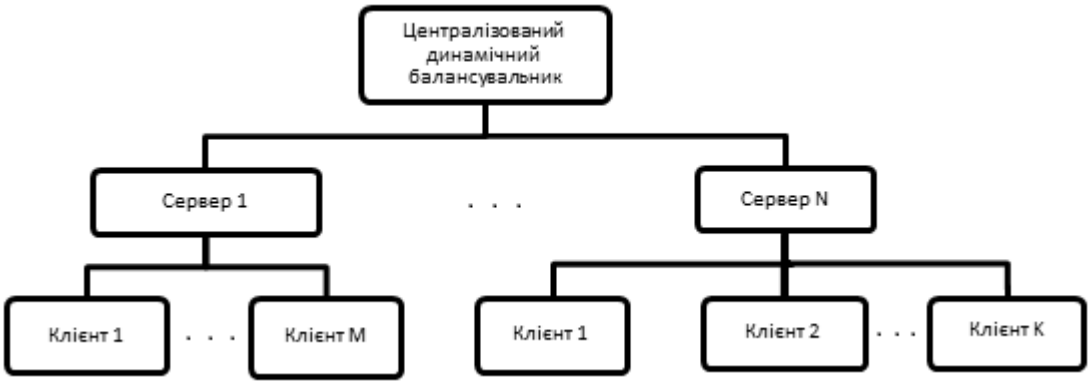

Рис.1. Топологія розподіленої системи

Опис розробленого алгоритма

Алгоритм централізованого динамічного балансування розроблявся спеціально для розподіленої веб-орієнтованої інформаційно-пошукової системи, вузлами якої є комп'ютери, що знаходяться в локальній мережі. Задачі, які вирішують сервери та клієнти є наступними: сервери завантажують із Інтернету веб-сторінки, віддають їх клієнтам для розбору та індексування. Вся система працює за принципом веб-павука (crawler), результатом її роботи є створення бази даних проіндексованих сторінок.

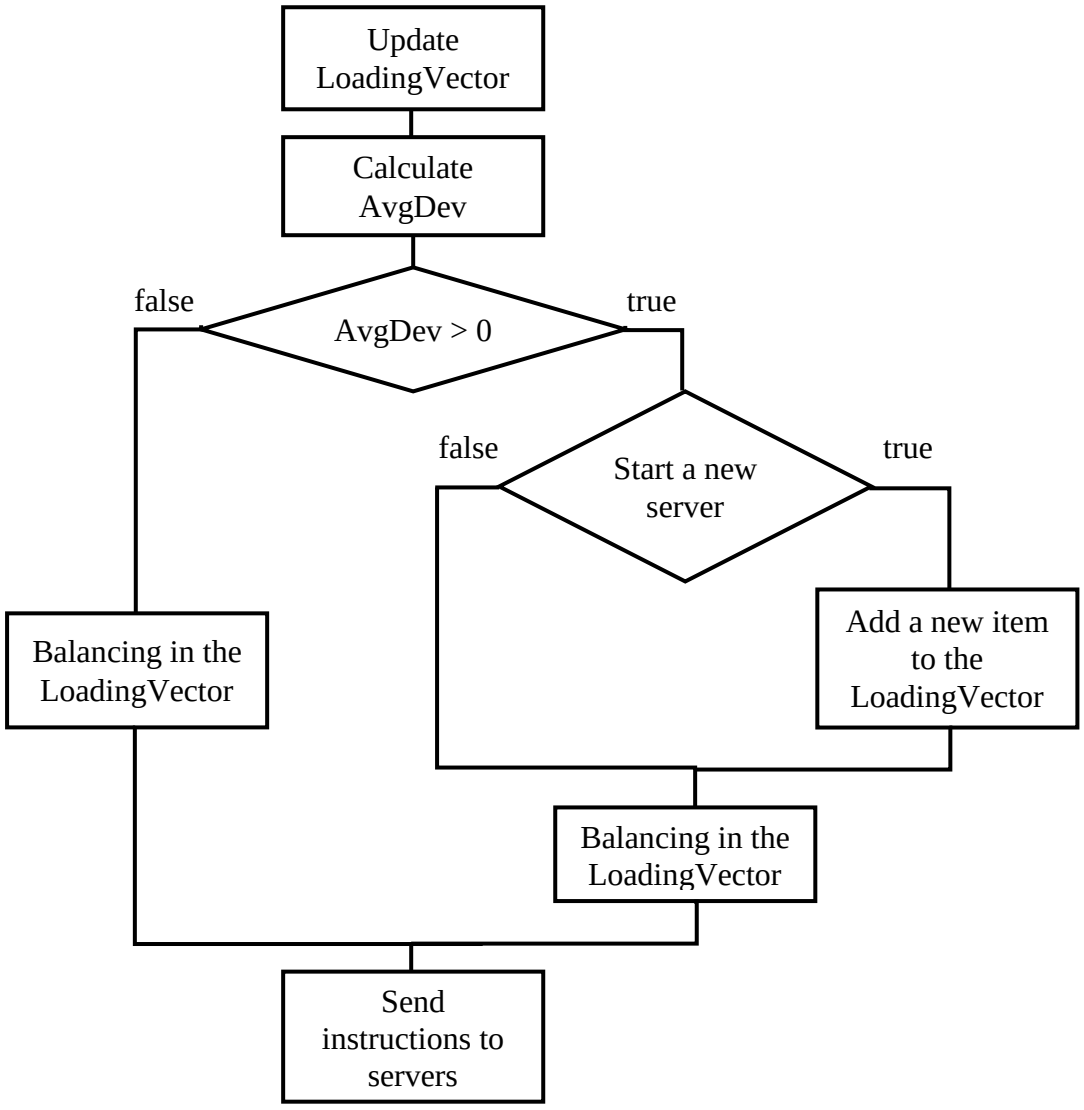

Рис. 2. Блок-схема алгоритму, що виконується балансувальником для збалансування системи

На рис.2 наведена блок-схема алгоритму, який періодично виконує балансувальник для визначення дисбалансу та його усунення. LoadingVector – це вектор, який складається із записів виду: [ClientsAmount, Deviation]. Кожний запис формується сервером за запитом від балансувальника. Позитивне значення в полі Deviation означає кількість клієнтів, переадресація яких на інші сервери необхідна для оптимальної роботи сервера. Негативне значення – кількість клієнтів, яку додатково може обслуговувати даний сервер. Балансування у векторі – це моделювання обміну клієнтами між серверами, яке б дозволило оптимально розподілити навантаження між ними. При виявленні перевантаження балансувальник намагається його компенсувати за рахунок введення нового сервера. Якщо запуск сервера був невдалий, то приймається рішення про відключення деяких клієнтів. Якщо ж система недовантажена, то після балансування можливе вимкнення деяких серверів. У результаті балансувальник відправляє інструкції на сервери про перенесення, відключення їхніх клієнтів чи самих серверів.

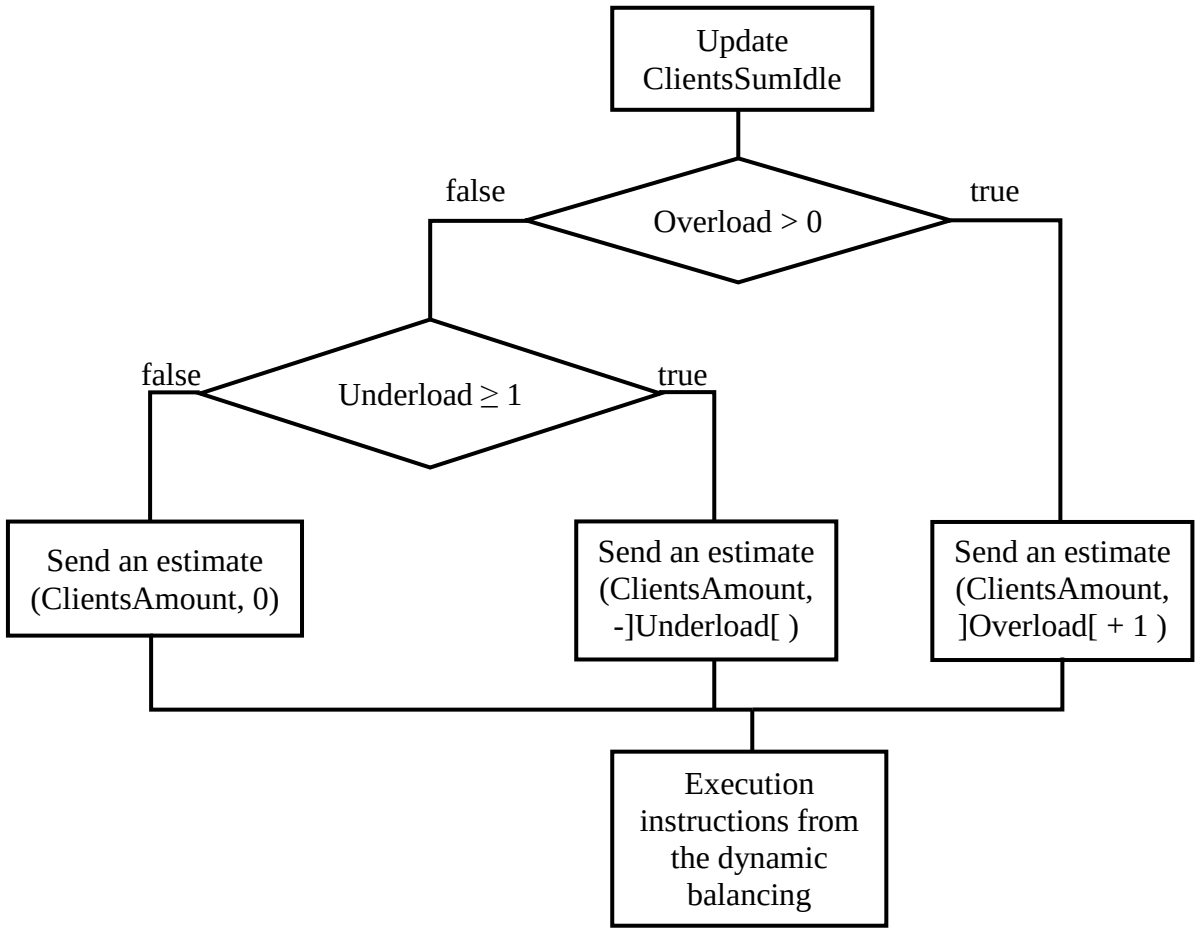

Рис. 3. Блок-схема алгоритма, що виконується на серверах по запитах від збалансування системи

На рис.3 зображена блок-схема алгоритма, що виконується на сервері по запиту від балансувальника. ClientsSumIdle – це сумарний час простою клієнтів, ServerIdle – час простою сервера, PureRunTime – час, витрачений сервером на обслуговування клієнтів. OverLoad та UnderLoad – це аналоги поняття Deviation із попереднього алгоритма, що визначаються наступним чином:

$\text{Overload} = \frac{\text{ClientsAmount} * \text{ClientsSumIdle}}{PureRunTime}$, $\text{Underload} = \frac{\text{ClientsAmount} * \text{ServerIdle}}{PureRunTime}$.

Дані беруться за останній проміжок часу, коли кількість клієнтів на сервері не змінювалась.

У випадку підключення нового клієнта до розподіленої системи балансувальник виконує наступні дії:

1. Виконується пошук по вектору LoadingVector сервера з від'ємним значенням Deviation.
2. Якщо такий сервер є, то його адреса відправляється клієнта. Після цього клієнт підключається до системи за цією адресою.
3. Якщо всі сервери зайняті, то клієнт отримує відмову у підключенні.

Висновки

Наведений алгоритм динамічного балансування дозволяє підвищити ефективність розподілення навантаження на сервери обчислювальної системи. Його відмінністю від подібних алгоритмів є робота зі змінною кількістю серверів, що використовуються у системі. Це дозволяє автоматично підлаштовуватись під зміну навантаження зі сторони клієнтів.

Для покращення алгоритма можливе використання більш точних критеріїв перевантаження чи недовантаження серверів, що опиратимуться на такі фізичні показники вузлів розподіленої системи, як потужність процесора чи ширина пропускного інтернет-канала.

Література

1. Вычислительная мощность распределенной системы BOINC. http://boinc.berkeley.edu
2. TOP500 List - November 2010. http://www.top500.org/list/2010/11/100
3. *Yagoubi B., Slimani Y.* Dymanic load balancing strategy for GRID computing.