УДК 004.33 (045)

Жуков І.А., д-р техн. наук
Кравець І. М.

МЕТОДИ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ ДЛЯ *WEB*-СЕРВЕРІВ

Інститут комп'ютерних технологій
Національного авіаційного університету

Розглянуто найбільш розповсюджені методи балансування навантаження для Web-серверів що підвищують їх швидкодію та продуктивність і забезпечують їх безперервну роботи. Кожен із методів було досліджено відносно: балансування, відмовостійкість та адміністрування. Також розглянуті змішані схеми балансування навантаження для Web-серверів.

Вступ

Web-сервери стали не тільки сховищем текстової і графічної інформації, але і місцем гігантських покладів відео- і ауді-оматеріалів, а також засобом проведення масштабних комерційних операцій [1]. На перший план виходить задача обслуговування запитів за гарантований час, що потребує удосконалених технічних, алгоритмічних і програмних засобів побудови розподілених *Web*-серверів [2-4].

Існує декілька способів підвищення швидкодії *Web*-сервера: можна збільшити полосу пропускання, встановити високопродуктивне мережеве обладнання, розробити ефективні додатки для *Web*, оптимізувати і модернізувати програмні і апаратні компоненти *Web*-сервера, а також взяти до уваги технологію «кешування» в середовищі *Web* [5].

Ще один спосіб підвищення продуктивності вузла *Web* полягає в тому, щоб збільшити кількість *Web*-серверів і розташовувати на них «зеркальні» копії матеріалів. Можна розбалансувати загальне навантаження на всі компоненти систем і зменшити час повернення інформації при виконанні сервером внутрішніх процедур оброблення клієнтських запитів. При цьому зберігаються і існуючі сервери, оскільки виводити їх із експлуатації і заміняти новими не має потреби [6].

Також слід відзначити програмні продукти, які вирівнюють навантаження, розподіляючи її на декілька серверів. Окрім цього, вони підвищують відмовостійкість *Web*-серверів: у випадку відмовлення однієї машини направляються пакети даних на інший сервер. Таким чином, час очікування зменшується, а число необроблених запитів зводиться до мінімуму. Системи балансування навантаження можна використовувати як при наявності лише одного *Web*-додатка, так і при роботі з цілим рядом вузлів. Отримавши представлення про те, що таке «системи балансування навантаження» і як вони працюють, можна визначити найбільш важливі їх характеристики, які слід враховувати при виборі засобу вирівнювання навантаження [7, 8].

Існує декілька розповсюджених методів балансування навантаження для *Web*-серверів. Це:

- **круговий *DNS* (*Round Robin DNS Load Balancing*).** Цей метод вважається одним із перших в основі якого знаходиться *DNS*-сервер, суть якого полягає в циклічному проходженні в списку *IP*-адрес серверів, що знаходяться у кластері;
- **апаратний розподіл навантаження (*Hardware Load Balancing*).** Метод полягає в направленні *TCP/IP* пакетів до різних серверів у кластері;
- **програмний розподіл навантаження (*Software Load Balancing*).** Процес балансування цим методом оснований на програмному забезпеченні. Зазвичай, це інтегровані компоненти у *Web*-сервер і серверні додатки у вигляді програмних пакетів;
- **змішані схеми**, коли використається комбінація апаратних і програмних засобів.

Круговий DNS

Круговий *DNS* – найпростіший спосіб перенаправлення *HTTP*-запитів на кілька серверів. Будь-який *DNS*-сервер зберігає пари "ім'я хоста/*IP*-адреса" для кожної машини в певному домені. Цей список виглядає приблизно так [9-11]:

- nau.edu.ua xxx.xxx.xxx.1
- www.nau.edu.ua xxx.xxx.xxx.2

Крім цього будь-якому імені можна призначити кілька *IP*-адрес. У результаті список буде виглядати так:

- nau.edu.ua xxx.xxx.xxx.1
- www.nau.edu.ua xxx.xxx.xxx.2
- www.nau.edu.ua xxx.xxx.xxx.3
- www.nau.edu.ua xxx.xxx.xxx.4
- www.nau.edu.ua xxx.xxx.xxx.5

DNS-сервер проходити колом по всіх записах у таблиці й віддає на кожен новий запит наступну *IP*-адресу. Так, на перший запит *DNS*-сервер видасть адресу "xxx.xxx.xxx.1", на другий – "xxx.xxx.xxx.2", а на третій – "xxx.xxx.xxx.3" і так далі. У результаті всі запити будуть однаково розділені між всіма *Web*-серверами.

Балансування

На перший погляд це рішення здається дешевим, але балансування навантаження полягає лише в тому, що кожен із серверів отримає рівну кількість запитів. Тобто, суть полягає в тому, що в кожного сервера є однаковий обсяг ресурсів (цілком реальне припущення), і що їм для виконання всіх операцій необхідна однакова кількість цих ресурсів (набагато менше реальне припущення). У цьому рішенні зовсім не враховується, наскільки завантажений процесор тієї або іншої машини. Хоч деякою мірою можна домогтися розподілу навантаження, але цей спосіб надзвичайно неефективний для серйозних додатків. Уявимо собі з'єднання із двох серверів (рис.1).

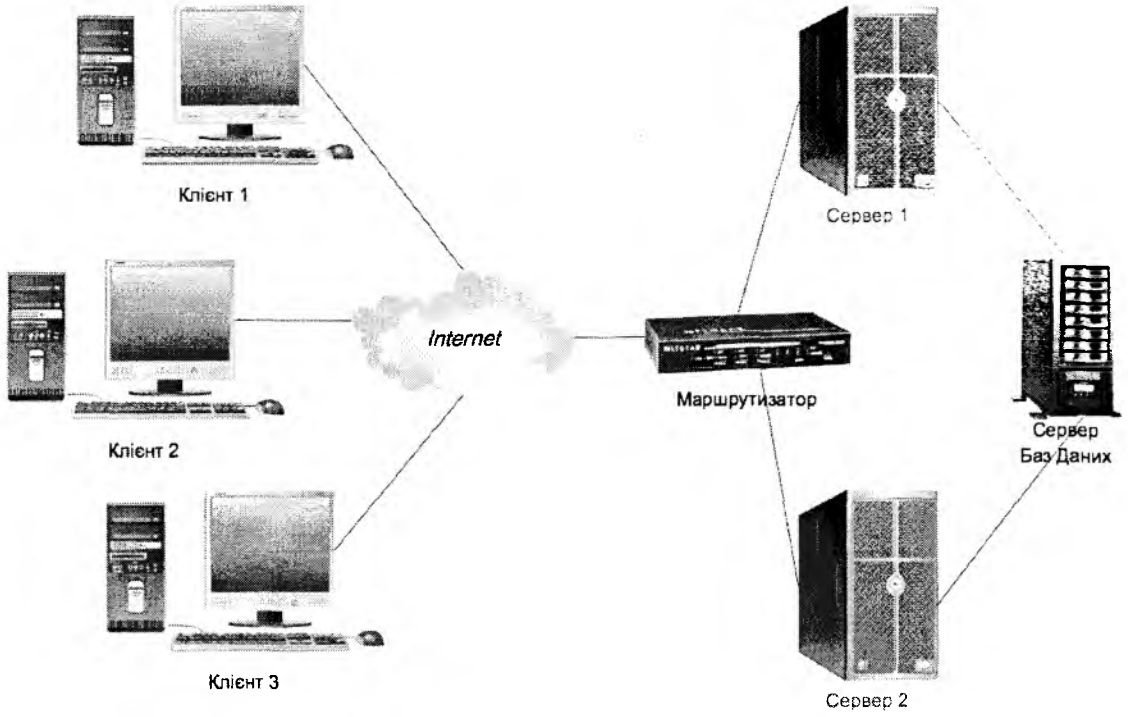

Рис.1. Приклад кругового *DNS* на базі двох серверів

Наприклад, використання процесора на «Сервер 1» вже досягло границі в 100%, через характер виконуваного *HTTP*-запиту. У цей же час «Сервер 2» завантажений, наприклад, лише на 10-15%. Схема кругового *DNS* як і раніше буде направляти половину всіх запитів на сервер А, що вже перевантажений. Це не тільки неефективно, але й приводить до того, що користувачі будуть натискати кнопку "*Refresh*" знову й знову, бачачи що сайт повільно відповідає, а в дійсності вони будуть генерувати усе більше й більше запитів. Тривати це буде доти, доки користувач не одержить повідомлення "*Server too busy*" [12].

Відмовостійкість

Крім того, ця схема не допоможе у випадку виходу будь-якої машини з ладу. Звернемося знову до прикладу, що розглядався вище: якщо «Сервер 1» виробить свій ресурс й "звалиться", половина *HTTP*-запитів будуть перенаправлятися на недоступну машину. У результаті користувач одержить ще більш цікаве повідомлення "*Server Unavailable*".

Багато *DNS*-серверів кешують у себе таблиці відповідностей. Кешування таблиць *DNS* може привести до того, що схема кругового *DNS* взагалі втрачає свою суть. Тому що локальний *DNS*-сервер клієнта запише в себе тільки одну відповідність "доменне ім'я/*IP*-адреса", всі клієнти цього локального *DNS*-сервера будуть весь час звертатися на той самий *IP*-адрес. Із часом, навантаження на машини, які були запущені раніше, буде збільшуватися до того часу, поки вони не "упадуть".

У радикальному випадку, при виході з ладу всього лише однієї машини у тисячі клієнтів, через кешування зав'язаного на *IP*-адресу «впавшої» машини, створиться враження що *Web*-сервер упав, хоча при цьому десять інших машин будуть прекрасно працювати й простоювати.

В разі зупинки шини на профілактику, потрібно внести зміни в таблицю *DNS*, а зміни в ній поширюються вкрай повільно. Відновлення у всій мережі може відбутися лише через кілька днів. А це значить, що до цього часу неможливо виключити машину, не вплинувши на роботу *Web*-сервера [13].

Крім цього, при такій схемі виникнуть проблеми:

1) з *Java*-аплетами, тому що вони в принципі можуть обмінюватися даними;
2) *SSL*-з'єднаннями;
3) з підтримкою сесій.

Адміністрування

З погляду адміністратора – ця схема ідеальна. Для її роботи не потрібне додаткове обладнання й перебудова мережі й серверів. На внесення змін у таблицю *DNS* не потрібно багато часу й сил.

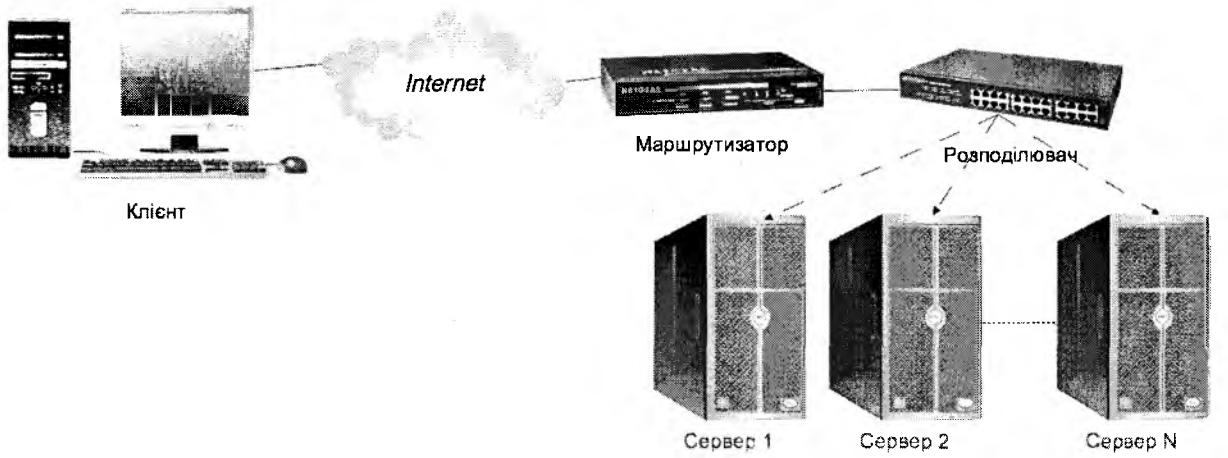

Рис.2. Приклад апаратного розподілу навантаження

Апаратний розподіл навантаження

Використання для розподілу навантаження спеціальних пристроїв (рис.2) не є альтернативою круговому *DNS*.

У мережу, між іншим, мережним обладнанням (мережним екраном, маршрутизатором і т. д.) і *Web*-серверами вставляються новий прилад, що називається апаратним розподілювачем навантаження (*Hardware Load Balancer*). Всі запити користувачів, адресовані на певний *URL* проходять через нього. До розподілювача підключається група *Web*-серверів, які поводяться, як один сервер. Ця конфігурація називається кластером (*cluster*). Для всього зовнішнього світу (*Internet*) весь кластер серверів має одну *IP*-адресу.

При отриманні *TCP/IP* пакетів призначених для кластера, розподілювач виконує наступні дії:

– приймає рішення, якому із серверів кластера варто направити наступний запит;

– опитує всі сервера й додатки (тобто певний *TCP/IP* порт) – чи доступні вони;

– у деяких випадках перевіряє, чи повертає сервер коректні дані. Це особливо критично в тих випадках, коли сервер відповідає, але у відповідь він увесь час видає "*HTTP Error 404*";

– переробляє *IP*-заголовок пакета так, що він іде певному серверу. Ця переробка називається "перетворенням мережної адреси" (*network address translation*);

– відправляє пакет на сервер;

– коли сервер відповідає клієнтові, розподілювач робить таке ж перетворення всіх пакетів і повертає їхньому клієнтові. У результаті цього другого перетворення клієнт одержує *TCP/IP*-пакети в такому виді, в якому вони були б отримані від певної *IP*-адреси, що закріплена за кластером.

У такий спосіб *Web*-сервер буде «живий» до того часу, доки буде «жива» хоча б одна машина в кластері. В якості додаткової функції: якщо виявляється, що якийсь із серверів не відгукується, розподілювач відправляє повідомлення на «пейджер» операторові.

Балансування

Як визначає розподілювач, якому із серверів направити запит? Кожного разу це залежить від розподілювача, хоча алгоритми майже завжди однакові. Розподільник навантаження збирає величезну кількість інформації про активність у мережі. У тому числі обсяг трафіку, що іде до сервера або від нього, швидкість, з якою відповідає сервер на *TCP/IP* запити, кількість з'єднань, що підтримує в цей момент часу кожен сервер, історія відповідей на попередні запити. У розподілювача навантаження закладено кілька алгоритмів, з яких адміністратор системи може вибирати один із них. Алгоритми містять у собі й кругове перемикання адрес і пропорційний перебір (круговий перебір адрес із коефіцієнтами). Завдяки цим алгоритмам розподільник може прийняти "розумне" й ефективне рішення щодо розподілу навантаження.

Використання процесора на «Сервер 1» досягло 100% через виконання якогось особливого запиту (рис.2). Процесор «Сервер 2» використовується тільки на 10-15%. Розподілювач навантаження помітить, що «Сервер 1» занадто повільно відповідає на запити, і перенаправляє всі наступні запити на «Сервер 2». Він буде продовжувати це робити доти, доки рівень завантаження «Сервер 1» не повернеться на нормалізований рівень. У такий спосіб отримуємо більш ефективний результат, ніж при використанні кругового *DNS*.

Відмовостійкість

Звернемося до прикладу кластера із двох серверів. Якщо сервер А "падає", апаратний розподілювач навантаження не буде посилати йому дані. Машина сама по собі може працювати, але якщо *Web*-сервер на цій машині не відповідає (скажімо його зупинили вручну), то розподілювач видалить «Сервер 1» зі свого списку серверів і направить весь трафік на «Сервер 2». У результаті стійкість, як бачимо, теж набагато вища, ніж при використанні кругового *DNS*.

Також слід звернути увагу на те, що сам розподілювач теж може вийти з ладу. Для рішення цієї проблеми звичайно встановлюють другий розподілювач, [14] що працює як "завжди готовий резерв" (*hot spare*) (рис.3).

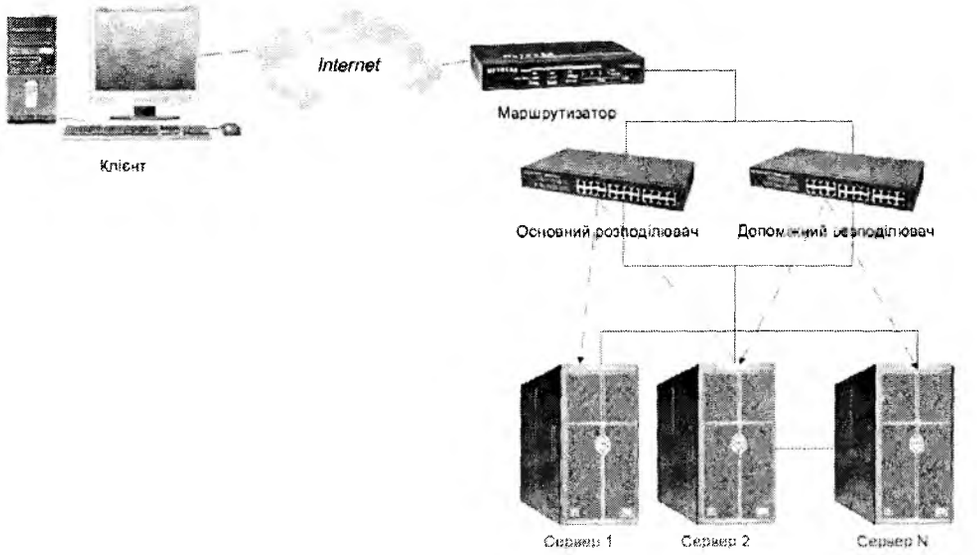

Рис.3. Приклад апаратного розподілу навантаження з допоміжним розподілювачем

Ця конфігурація з надлишковою надійністю як правило дуже стійка. Розподілювачі з'єднуються звичайно через послідовне з'єднання, наприклад через кабель *RS*-232. У кожному із приладів є контрольний пристрій (*watchdog processor*), які постійно підтримують зв'язок один з одним. У конкретний момент працює тільки один розподілювач, а другий вступає в лад, якщо основний "впаде". У випадку виходу приладу з ладу або його незвичайного поводження, один контрольний пристрій "жаліється" іншому й приймається рішення, хто візьме на себе операції з розподілу навантаження. Навіть після виходу з ладу першого розподілювача, другий, виконуючи роботу, постійно перевіряє перший. Якщо перший розподілювач повернувся в лад, другий повертає всі обов'язки першому й переходить у режим очікування.

Адміністрування

В адмініструванні апаратний розподілювач навантаження трохи складніший, ніж схема із круговим *DNS*. Встановити й налаштувати розподілювач не складніше, ніж звичайний маршрутизатор, і знань адміністратора мережі буде досить для цього. Адміністрування виконується тільки на самому розподілювачі й ніяк не впливає на самі серверні машини. Це велика перевага, тому що адміністраторам не потрібно займатися конкретно кожним сервером.

Апаратні розподілювачі навантаження – високоінтелектуальні прилади. Коштують вони приблизно скільки, скільки коштує гарний маршрутизатор або 10 гарних серверів. Через те, що робота розподілювача пов'язана з перетворенням заголовків всіх *TCP/IP* пакетів, пропускна здатність системи зменшується. Межа швидкості роботи – 45-50 Мб/с. Здавалося б, швидкість неймовірна, але великі сайти можуть перевищити (і перевищують) цю межу в період пікового навантаження. При цьому додаткові пакети просто ігноруються. Неприємно, звичайно, але більшості *Web*-серверам це не грозить.

Розподілювачі навантаження використовують алгоритми, які базуються на спостереженнях за роботою мережі. Час відповіді сервера може бути більшим через те, що він виконує який-небудь запит (наприклад, очікує відповіді від бази даних на сусідній машині). У цьому випадку розподілювач не буде звертатися до *Web*-сервера, хоча насправді в того буде ще повно ресурсів.

Звичайно, в набір функцій розподілювача входить можливість приписувати одному серверу одну певну *IP*-адресу або цілий простір адрес класу С. Ця функція називається "прив'язкою" ("*affinity*"). Також серед функцій може бути: віддалене адміністрування за допомогою *Web*-інтерфейсу, оповіщення адміністратора про неполадки по пейджеру, генерація звітів про роботу мережі в реальному часі й багато чого іншого.

У якості більш економічного, але менш функціонального рішення можна скористатися "врізаним" варіантом розподілювача, що називається "перемикачем навантаження" (*load director*). Його інтелектуальність трохи нижча (наприклад у нього немає підтримки "прив'язки" (*affinity*), але коштує він дешевше.

Незважаючи на недоліки, апаратний спосіб розподілу навантаження є без сумніву перевіреним і реальним методом, завдяки якому досягається збалансована робота *Web*-проекту. Компанії, що надають хостінг, а також багато комерційних і торговельних *Web*-проектів цілком покладаються на це рішення.

Програмний розподіл навантаження

Для ознайомлення з роботою даного методу, візьмемо до уваги програмний продукт, що базується на описі програмного розподілювача навантаження від Microsoft [8]. На кожен сервер встановлюється специфічне програмне забезпечення, що поєднує сервери в єдиний кластер з єдиною *IP*-адресою. Ця *IP*-адреса присвоюється доменному імені проекту. Кожній машині присвоюється свій ідентифікатор у кластері від 1 до 32. Тобто, максимум у кластері може бути 32 машини. Крім того, призначається ваговий коефіцієнт, тому що машини можуть мати різний обсяг ресурсів, встановлюється набір правил.

Балансування

Звісно, виникає запитання, якщо 32 машини в кластері будуть приймати пакети для однієї і тієї *IP*-адреси, то клієнт повинен отримувати 32 відповіді на свій запит? Справа в тому, що при встановленні розподілювача, між *TCP/IP* стеком і драйвером мережної карти впроваджується фільтр, що визначає, який із серверів повинен обробити запит. Всі сервера в кластері фільтрують трафік, і тільки певний сервер відповідає на запит.

Так як цей розподілювач є програмою, у нього більше засобів об'єктивно оцінювати здатності машини обробити запит. Йому доступний відсоток завантаження процесора, обсяг вільної пам'яті, і обсяг вільного місця на диску та інше. Тому що принцип роботи побудований на фільтрації пакетів, а не на перетворенні їхніх заголовків, програмний розподільник працює швидше, ніж апаратний.

На кожну машину встановлюють ще одну мережну плату, через яку машини в кластері спілкуються між собою й з базою даних. Тоді весь прихожий трафік іде через перший мережний інтерфейс (на якому знаходиться розподілювач), а весь міжмашинний трафік іде через другий інтерфейс, так що адміністрування машин та інші сервіси не заважають роботі *Web*-серверів.

Відмовостійкість

Розподілювачі на всіх машинах періодично розсилають у мережі особливі повідомлення, за допомогою яких визначається стан усього кластера. Коли додається або видаляється машина, програмний розподільник починає процес, що називається "відомість" (*convergence*). Під час відомості машини, оцінюється новий стан кластера й відповідно міняється алгоритм. Зазвичай, на це витрачається 10-13 сек.

Серед повідомлень, які генерує розподілювач є повідомлення яке називається "пульс" (*heartbeat*). У розсиланні цих повідомлень бере участь кожна машина. Частоту пульсу можна міняти, а за замовчуванням вона дорівнює 1 пульсу в секунду. Машина вважається недоступною, якщо вона не змогла взяти участь в 5 "ударах пульсу" підряд. Після цього машини, що залишилися, починають процес "відомості".

Адміністрування

На відміну від попередніх розглянутих рішень, програмний розподіл навантаження турбує кожен сервер. Неминуче на кожну машину в кластері буде потрібно встановити програмне забезпечення й налаштувати його, встановити додаткові мережні карти, тому потрібен ще один концентратор (*hub*) або комутатор (*switch*) і багато кабелів. Крім цього, конфігурація кожної машини буде відрізнятися, тому що в кожної машини буде своє унікальне ім'я й свій ваговий коефіцієнт.

Позитивна сторона полягає в тому, що програмним розподілом навантаження можна управляти з будь-якого комп'ютера мережі. Як тільки нова машина налаштована й на ній установлений розподілювач, її можна включати в мережу й віддалено добавити її в кластер чи видалити її звідти.

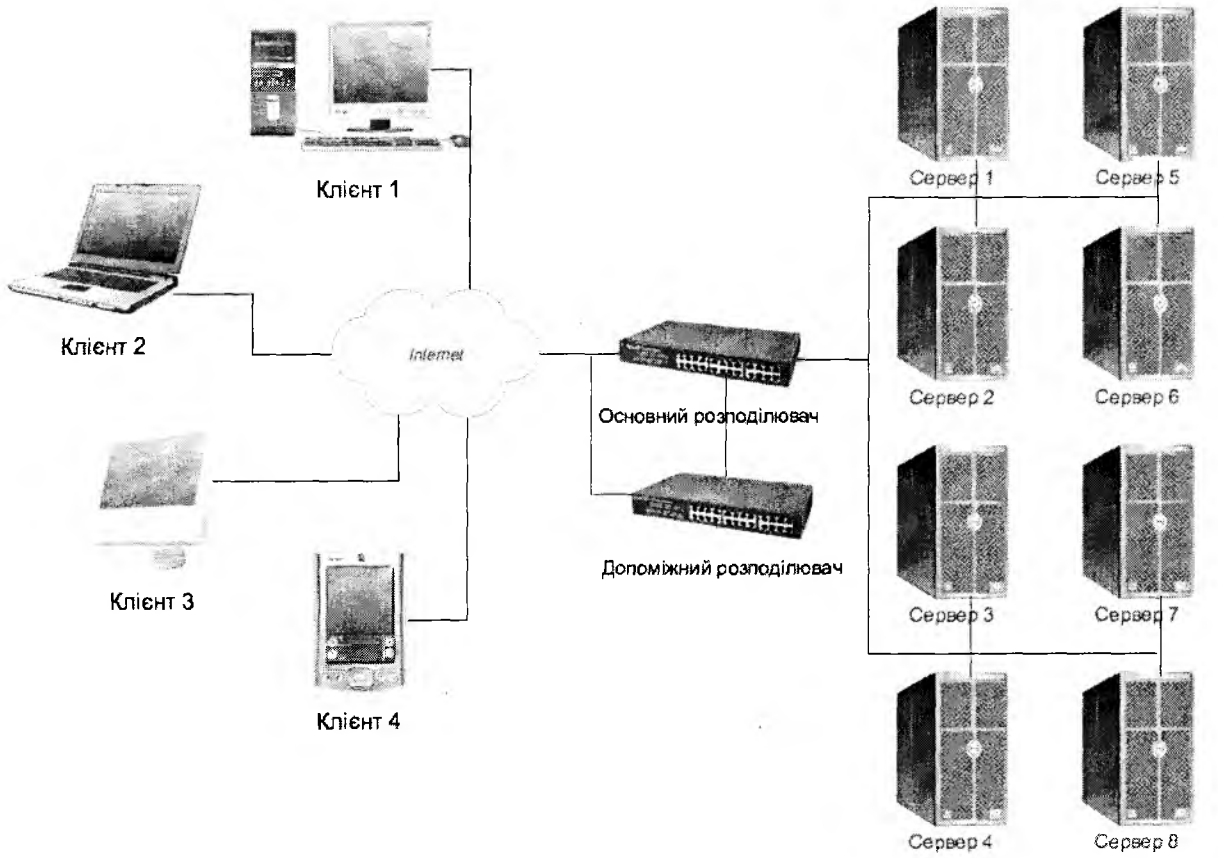

Рис.4. Приклад змішаної схеми розподілу навантаження

Змішані схеми

Існують також змішані схеми, коли працюють разом апаратні й програмні розподілювачі навантаження [15–19]. Розглянемо наступну конфігурацію (рис.4). У системі використовується один апаратний розподілювач навантаження (і ще один запасний) і два кластери серверів, у яких використаються програмні розподілювачі навантаження. Апаратному розподілювачу невідомо про те, що за двома *IP*-адресами знаходиться цілий «загін» машин. Так само, кожному із кластерів невідомо про існування свого сусіда.

Балансування

Використовуючи змішану схему можна перевищити межу в 32 машини, що встановлена програмним розподілювачем. По суті, якщо припустити, що апаратний розподілювач здатний працювати з 256 окремими вузлами (цілком реальне припущення), тоді можна розширювати потужність проекту аж до величезної кількості машин: 8192 серверів.

Відмовостійкість

Завдяки такій схемі можна будувати з'єднання серверів, що повністю ізольовані одне від одного не тільки через мережу, але й географічно. У такий спосіб можна отримати додатковий засіб захисту від різних примх природи (повеней, пожеж і т. д.) Якщо безпека якого-небудь із серверів буде порушена, його завжди можна відключити від мережі, не вплинувши на роботу *Web*-проекту.

Адміністрування

Як було підкреслено вище, апаратні розподілювачі не значно збільшують турботу адміністратора. Реалізація й використання змішаної схеми не відрізняється від реалізації й використання схеми із програмним розподілом навантаження.

Висновки

Таким чином, найпростішим способом зменшити навантаження на *Web*-серверах є вибір кругового *DNS*. Але, роблячи конкретизовані висновки, бачимо, що цей метод не є ефективним для великих проектів, так як у ньому кожен сервер отримує однакову кількість запитів в незалежності від навантаження всієї кластерної системи.

Але якщо питання у фінансовому плані не стоїть, то варто використовувати апаратний розподіл навантаження. Встановлення апаратного розподілювача навантаження дає йому змогу аналізувати кластерну систему за всіма параметрам:

- завантаження серверів (час відклику);
- стан сервера: відключений чи підключений;
- та ряд інших.

В адмініструванні апаратний розподілювач складніший, а ніж в схемі із круговим *DNS*.

Програмний розподіл навантаження, дешевшим у фінансовому плані за апаратний. Оскільки, в його основу лягає програмний продукт, що базується на описі програмного розподілювача навантаження. Недоліком цього методу є те, що на кожен із серверів в кластерній системі треба буде встановити цей програмний комплекс, налаштувати його, а також встановити додаткові мережеві карти, що за собою буде вимагати встановлення додаткового комутатора та багато кабелів. Окрім цього, конфігурація кожної машини буде відрізнятися, тому що кожен сервер буде мати своє унікальне ім'я та ваговий коефіцієнт.

Позитивна сторона цього методу полягає в тому, що програмним розподілом навантаження можна керувати з будь-якого комп'ютера в мережі та не буде проблем із додаванням нового сервера в кластер.

Проаналізувавши основні методи розподілення навантаження для *Web*-серверів, та маючи представлення розгалуження мережі на сьогодні, можна чітко аргументувати, що найефективнішим методом є поєднання апаратних і програмних розподілювачів навантаження, або іншими словами – змішані схеми. Завдяки таким схемам можна будувати з'єднання серверів, що повністю ізольовані один від одного не тільки через мережу, але й географічно. Тому, цей варіант дає змогу повноцінно масштабувати кластерну систему не тільки у відношенні до загального навантаження *Web*-серверів, а й прийняти і впровадити концептуальні рішення, щодо розгалуження кластерної системи в тому чи іншому географічному регіоні, у мірі його завантаження.

Наукові основи оптимального вибору того чи іншого методу балансування навантаження для *Web*-серверів ще не склались, причому, це відноситься до *Web*-кластерів. Коли мова іде про вибір методу балансування, кожне рішення має вузьку область застосування, межі якої можуть стати ще вужчими, адже сьогоднішні параметри навантаження на сервер – це не ті ж самі параметри навантаження, що будуть завтра. Різновидність підходів, методів, оцінок – потребує систематизації, що було першим кроком зроблено в даній статті.

Список літератури

1. *Paxson V., Floyd S.* Wide-area Traffic: The Failure of Poisson Modeling. - IEEE/ACM Transactions on Networking. 1995.

2. *Олифер В.Г., Олифер Н.А.* Компьютерные сети. Принципы, технологи и протоколы. – СПб.: Питер, 2001. – 672 с.

3. *Столлингс В.* Современные компьютерные сети. 2-е издание. – СПб.: Питер, 2003. – 783 с.

4. *Таненбаум Э.* Компьютерные сети. – СПб.: Питер, 2008. – 992 с.

5. *Duane Wessels.* Web caching. 2001. – 318 p.

6. *Киллелиа П.* Тюнинг веб-сервера. – DJVU: Питер, 2003. – 530 с.

7. *Либман Л.* Философия распределения нагрузки // Журнал сетевых решений LAN. – № 5. – 2000.

8. *Microsoft Corporation.* Балансировка нагрузки сети, http://www.microsoft.com/Rus/Business/Infrastructure/Unified/Scalability/LoadBalance.mspx

9. *Shaikh A., Tewari R., Agrawal M.* On the Effectiveness of DNS-based Server Selection. - Proceedings of IEEE INFOCOM '01, Anchorage, Alaska, 2001.

10. *Springer Berlin / Heidelberg.* A Load Balancing Method Using Ring Network in the Grid Database. 2007.

11. *Tony Bourke.* Server Load Balancing. 2001.

12. *Hunt G., Nahum E., Tracey J.* Enabling content-based load distribution for scalable services. - Technical report, IBM T.J. Watson Research Center. 1997.

13. *Arlitt M.F., Williamson C.L.* Web Server Workload Characterization: The Search for Invariants. - In Proceedings of the ACM SIGMETRICS '96 Conference, Philadelphia, PA, 1996.

14. *Cardellini V., Colajanni M., Yu P.S.* High-Performance Web-server Systems. - IEEE, Internet Computing, 1999.

15. *Жуков І.А., Мартинова О.П.* Метод побудови паралельних структур для пошуку альтернативних маршрутів у комп'ютерних мережах // Вісник НАУ. – К.: НАУ, 2004. – №1. – С. 14-17.

16. *Cardellini V., Casalicchio E., Colajanni M., Yu P.S.* The State of the Art in Locally Distributed Web-server Systems. - IBM Research Report, RC22209 (W0110-048), 2001.

17. *Crovella M. E., Taqqu M.S., Bestavros A.* Heavy-Tailed Probability Distributions in the World Wide Web. - In A Practical Guide To Heavy Tails, chapter 1, Chapman & Hall, New York.

18. *Harchol-Balter M., Crovella M., Murta C.* To queue or not to queue?: When FCFS is better than PS in a distributed system. - Technical Report, CS Department, Boston University, Number 1997-017, 1997.

19. *Hunt G., Goldszmidt G., King R., Mukherjee R.* Network Dispatcher: a connection router for scalable Internet service. - Computer Networks and ISDN Systems, Vol. 30, 1998.