МАТЕРІАЛИ МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ АСПІРАНТІВ І СТУДЕНТІВ «ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 2010»

ПРИКЛАДНІ ДОМЕНИ І ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

УДК 621.797:621.664

І.С. Шаповал

АДАПТИВНИЙ АЛГОРИТМ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ В КОМП'ЮТЕРНИХ МЕРЕЖАХ

Кременчуцький національний університет імені Михайла Остроградського

кафедра комп'ютерних та інформаційних систем

Науковий керівник
Сисюк Г.Ю.
(к.т.н., доцент)

Розглянуто нечіткий алгоритм, що реалізує політику участі вузлів мережі в балансуванні навантаження і передбачає можливість адаптивного керування трафіком в мережі.

Рассмотрен нечеткий алгоритм, реализующий политику участия узлов сети в балансировке загрузки и предусматривающий адаптивное управление трафиком в сети.

Considered a fuzzy logic algorithm that implements the of the network nodes involvement policy in load balancing and providing adaptive control of traffic in the network.,

Ключові слова: динамічне балансування навантаження, комп'ютерні мережі,розподілені обчислення, нечіткі алгоритми

Вступ

Балансування навантаження в розподілених системах – це процес розподілу робочого навантаження системи між обчислювальними ресурсами, що розділяються. З появою високошвидкісних каналів зв'язку все частіше в якості вузлів розподілених систем використовуються автономні комп'ютерів, об'єднаних високошвидкісними лініями зв'язку. Основною перевагою таких систем є висока продуктивність та доступність при низьких витратах. Внаслідок цього розподілені обчислення набувають все більшого значення як переважний метод обчислень порівняно з централізованою обробкою даних. .

Оскільки означена система є сукупністю автономних комп'ютерів, об'єднаних в мережу, між якими розподілені програми, що виконуються, користувачі не в змозі однаково використовувати комп'ютери протягом всього часу роботи. Однак в будь-який момент часу користувачу може знадобитися більше ресурсів, ніж у змозі забезпечити його робоча станція. Тому окремі вузли в процесі роботи можуть бути перевантаженими, а інші навпаки – знаходитись в режимі очікування, внаслідок чого продуктивність системи в цілому на деяких інтервалах може суттєво знижуватись. Підвищення продуктивності обчислень до прийнятного рівня часто можливо досягнути шляхом перерозподілу обчислювальних ресурсів між задачами.

В багатьох дослідженнях пропонуються різні підходи до проблеми балансування навантаження. [1, 2, 3, 4, 5, 6, 7]. Концептуально схеми балансування навантаження можна розділити на два типи – статичні та динамічні. В ста-

тичних схемах розподіл навантаження здійснюється на етапі проектування розподіленої програми. Однак дуже часто статичне балансування навантаження не є достатнім для досягнення продуктивності системи, оскільки:
- може змінитися обчислювальне середовище, в якому відбувається виконання програми, або обчислювальний вузол може вийти з ладу;
- обчислювальний вузол, на якому виконується розподілена програма, зайнятий додатково іншими обчисленнями, доля яких з часом може змінюватись.

Перша з перерахованих причин є характерною для паралельних алгоритмів, які реалізують чисельні методи, зокрема, методу скінчених елементів розв'язку систем диференційних рівнянь в часткових похідних, другий – коли мережа використовується як паралельний сервер

Внаслідок цього схеми динамічного перерозподілу навантаження, які перерозподіляють задачі відповідно до поточного завантаження вузла, забезпечують більш високу продуктивність системи.

Аналіз досліджень і публікацій

При розробці схеми динамічного балансування навантаження вирішуються три зв'язані між собою питання: 1) збирання інформації, 2) вибору вузлів-кандидатів на передачу навантаження, 3) власне передавання інформації з перевантаженого вузла [1].

На сьогодні існує багато різноманітних підходів до проектування балансувальиків, які розрізняються за методами вибіру комп'ютерів, між якими повинне бути передане навантаження. Найчастіше використовуються наступні чотири підходи.

Централізована політика передбачає використання одного агента, відповідального за прийняття рішень про комп'ютери джерела та адресату, якому повідомляються системні значення завантаження для визначення найбільш та найменш завантажених комп'ютерів та ініціювання передачі навантаження між ними [8]. Основним недоліком централізованого підходу є суттєве збільшення трафіку і можливі колізії в мережі за рахунок широкомовних запитів, які використовуються для збирання інформації про поточний стан та завантаження обчислювального середовища, що може бути причиною зменшення продуктивності системи в цілому.

При децентралізованому підході до балансування навантаження найчастіше використовуються наступні децентралізовані політики розподілу навантаження.

Перерозподіл за ініціативи відправника [9, 10, 11], коли відправник навантаження відомий, а завданням є визначення комп'ютера- адресата, який буде приймати участь у перерозподілі навантаження.

Перерозподіл за ініціативи отримувача [11], коли комп'ютер вирішує приймати участь у розподілі навантаження, якщо його навантаження менше за граничний поріг, а політика місце знаходження використовується для того, щоб знайти комп'ютер, від якого може бути передане навантаження. В цих випадках вибір місця розташування повністю залежить від механізму комунікацій, а політика повинна вибрати найбільш завантажений або випадковий комп'ютер.

Симетрична ініціалізація. В роботі [12] наводяться результати досліджень, які показали, що політика ініціалізованого відправника більш ефективна, в той час, коли сиситемне навантаження середнє, а політика ініціалізованого отримувача - коли навантаження високе. Ці дві політики можуть бути поєднані наступним чином [13].

Якщо завантаження головного комп'ютера вище за деякий поріг *T*, ініціюється режим відправника, і запити надсилаються на випадкові комп'ютери. Якщо отримане повідомлення, відправник передає завдання відповідному комп'ютеру. Якщо навантаження нижче порогу T, ініціюється режим отримувача, і запит надсилається на випадковий комп'ютер. Якщо комп'ютер, з яким входять в контакт, перевантажений, то він повертає завдання отримувачу. Якщо не було ніякої відповіді на запит, алгоритм переходить у режим очікування. Подібний пороговий підхід запропонований також в роботі [14], за виключенням того, що виконується не випадкове опитування, а відправники та отримувачі використовують списки комп'ютерів, які ідентифікували себе як зайняті або такі, що простоюють.

Аналіз рішень, які використовуються при проектуванні програм-балансувальників навантаження, дозволяє визначити їх наступні недоліки.

1. Однопорогові алгоритми, які використовуються для визначення ступеню завантаженості вузла, у більшості випадків не забезпечують достатню стійкість процедури балансування. Наприклад, у випадках, коли завантаження пари вузлів, які приймають участь у балансуван-

ні, близьке до порогового, за час від моменту прийняття рішення про передачу навантаження до початку його передачі, воно може змінитися (наприклад, завантаження вузла-приймача може зрости до значення, що перевищує порогове), внаслідок чого мета балансування не досягається. Для зменшення нестабільності в роботі [15] авторами запропоновний двопороговий алгоритм, який ділить шкалу завантаження на три частини – мале, середнє та високе. Однак означений алгоритм використовує фіксовані наперед визначені значення порогових рівнів, в той час як бажаною рисою сучасних розподілених систем повинна бути можливість динамічної зміни стану вузлів і трафіку.

2. Будь-яка політика вибору місця розташування передбачає попереднє опитування стану вузлів, які розглядаються балансувальником як вузли-кандидати на прийом чи передавання навантаження. В мережах, які мають велике число вузлів, наслідком може бути суттєве збільшення мережного трафіку, що зменшує продуктивність обчислень. Окрім цього, якщо використовуються алгоритми, які передбачають визначений час очкування відповідей від вузлів (наприклад, як в алгоритмах BID та PID [16]), то в умовах високого трафіку балансування навантаження може виявитись взагалі неможливим, оскільки відповіді від потенційних вузлів-кандидатів прийдуть запізно. Деякі алгоритми передбачають управління перевантаженням, але при цьому вони також використовують наперед визначені порогові рівні [17, 18].

3. Для прийняття рішення про необхідність розпочати процедуру балансування використовують локальну інформацію про рівень завантаження лише «найближчих» вузлів. Це штучно обмежує множину вузлів-кандидатів на передавання навантаження з вузла.

В запропонованому нами адаптивному алгоритмі вирішуються наступні проблеми класичних систем динамічного балансування:
- розширений пошук можливих вузлів-кандидатів на передачу навантаження без суттєвого збільшення трафіку в мережі за рахунок використання при пощуку вузлів-кандидатів на передавання навантаження ієрархічної доменної моделі мережі;
- підвищення стійкісті рішень щодо розподілу навантаження за рахунок використання фаззі-алгоритму вибору кандидатів;
- керування мережним трафіком та запобігання колізіям за рахунок зміни параметрів функції належності нечітких змінних;

Постановка завдання

Мета балансування завантаження може бути сформульована наступним чином. Виходячи з набору завдань, що включають обчислення і передачу даних в мережі комп'ютерів певної топології, знайти такий розподіл завдань між комп'ютерами, який забезпечує приблизно рівне обчислювальне завантаження комп'ютерів і мінімальні витрати на передачу даних між ними.

Звичайне практичне і повне рішення задачі балансування завантаження складається з чотирьох кроків:
- оцінка завантаження обчислювальних вузлів, результати якої використовуються для визначення виникнення дисбалансу та для обчислення об'єму робіт, необхідних для переміщення об'єктів;
- ініціація балансування завантаження, яка включає визначення моменту виникнення дисбалансу завантаження та міри необхідності балансування, тобто порівняння можливої користі від його проведення і втрат на нього;
- ухвалення рішень про балансування, яке визначається архітектурою балансувальника (централізованою чи розподіленою) та характером інформації, що використовується балансувальником (глобальною, яка враховує завантаження всіх, чи тільки частини вузлів комп’ютерної системи;
- переміщення програмних об'єктів між процесорами для досягнення нового балансу завантаження.

Модель мережі

Модель обміну даними, необхідними для балансування навантаження, використовує концепцію активних мереж, запропонованих в роботах [19, 20, 21]. Основна ідея активних мереж полягає в тому, що деякі елементи мережної інфраструктури обробляють дані в процесі передавання їх між кінцевими елементами системи. Ефективність такої архітектури полягає в тому, що порівняно з традиційною кінцеві елементи при обміні даних породжують інформаційні потоки меншої інтенсивності, що дозволяє запобігати колізіям..

Модель, яку ми використовуємо при пошуку вузлів-кандидатів на передачу навантаження, має наступні особливості.

1. Групи вузлів у мережі утворюють домени, в кожному з яких існує виділений вузол (контролер домену (КД), який взаємодіє з КД інших доменів.

2. Домени можуть бути як фізичними, так і логічними. Кожен вузол в логічному домені також є логічним вузлом і виступає як представник домену, що знаходиться на попередньому рівні ієрархії. Самі ж логічні вузли представлені фізичними вузлами. що знаходяться на нижніх рівнях ієрархії. Це проілюстровано на рис. 1.

Тут на першому рівні ієрархії визначені три фізичні домени (1, 2, 3), до яких входять фізичні вузли. Другий рівень –логічний домен 4, що складається з двох логічних вузлів, які представляють на поточному рівні домени 1 та 2. В домен 5 входять 2 логічні вузли, що представляють домени 3 та 4. З метою зниження накладних витрат на міжвузлові інформаційні обміни До одного домену включаються вузли, які знаходяться на близькій відстані., що потенційно дозволяє знизити накладні витрати на міжвузлові інформаційні обміни при балансуванні навантаження.

Фаззі-алгоритм вибору кандидатів

Балансування навантаження в системі здійснюється фаззі-контролером (рис.2) і передбачає виконання наступних кроків.

Фаззіфікація. Виходячи з обсягу використання ресурсів кожного вузла, розраховується значення індексу завантаження вузла, які фаззіфікуються з використанням функції належності (рис.3).

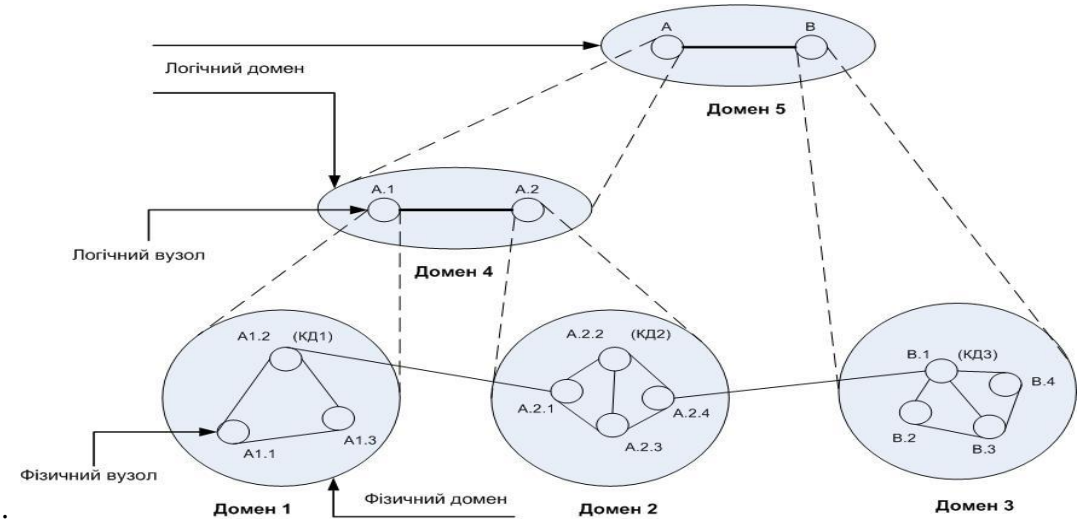


Рис.1 Ієрархічна модель мережі

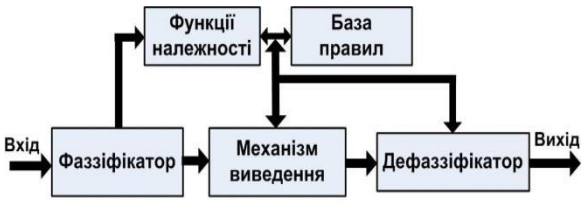


Рис. 2 – Архітектура фаззі-контролера

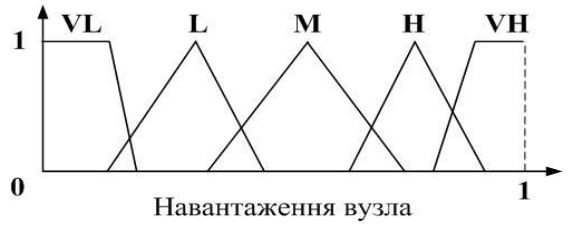


Рис. 3 Функції належності для вхідних змінних

В функції належності для індексу завантаження вузла визначені п’ять термів - дуже слабке (VL), слабке (L), помірне (M), сильне (H) та дуже сильне завантаження (VH) .

Застосування правил нечіткого виведення. Вихідною змінною фаззі-контролера в розробленому алгоритмі є ймовірність балансування навантаження між двома вузлами мережі, яка визначається за відомими значеннями їх індексів завантаження відповідно з правилами виведення В функції належності вихідної змінної визначені п’ять термів - дуже мала, мала, середня, висока та дуже висока ймовірність балансування навантаження між двома вузлами. Вигляд функції належності для виходу - аналогічний наведеному на рис.3. База правил виведення, яка використовується для балансування завантаження показана в табл.1.

Табл.1

База правил виведення

Завантаження передавача

Завантаження приймача	*VL*	*L*	*M*	*H*	*VH*
VL	VL	L	M	H	VH
L	VL	VL	L	M	H
M	VL	VL	VL	L	M
H	VL	VL	VL	VL	L
VH	VL	VL	VL	VL	VL

Розглянемо приклад застосування запропонованого алгоритму. Нехай навантаження вузла-передавача знаходиться в точці A, а вузла-приймача – в точці B, як це показано на рис.3. та 4.

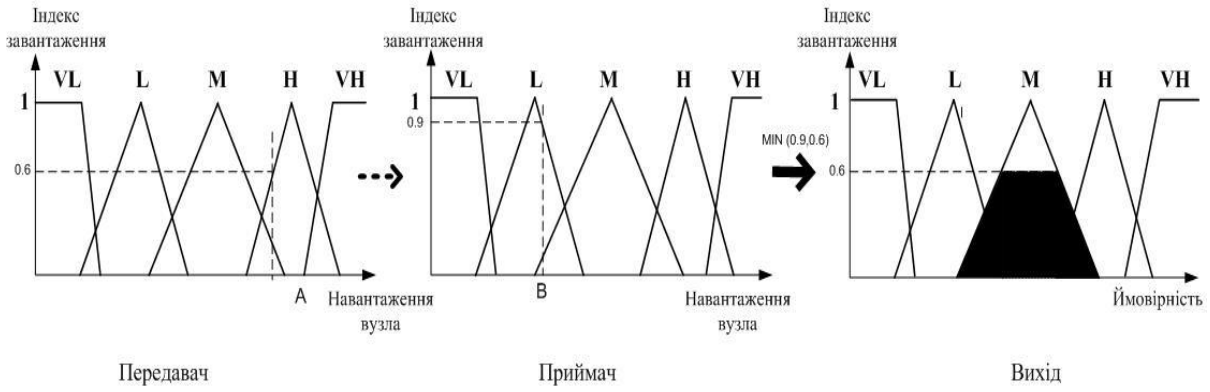
Тоді можна сформувати нечітку множину для індексу навантаження за наступними правилами.

Правило 1. Якщо вузол-передавач важко завантажений (H), а вузол-приймач легко завантажений (L), то вихід – середня ймовірність передачі навантаження приймачу помірна (M).

Правило 2. Якщо вузол-передавач помірно завантажений (M), і вузол-приймач помірно завантажений (M), то вихід – дуже низька ймовірність передачі навантаження приймачу помірна (VL).У результаті застосування означених правил отримуємо нечіткі множини індексу завантаження, показані на рис.4а . та 4б відповідно.

Комбінування. На рис.5 проілюстровано отримання композиції термів вихідної змінної у результаті застосування правил 1 та 2.

а)



б)

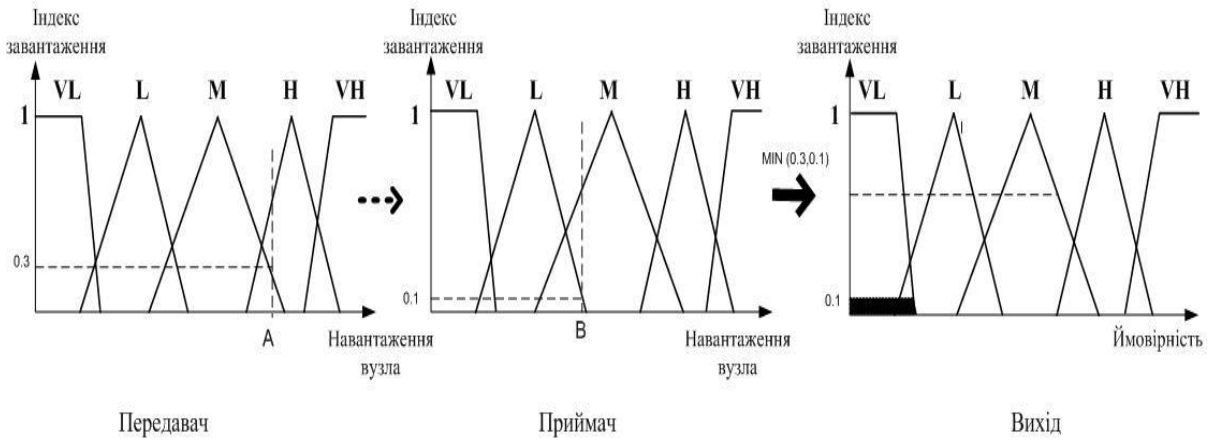


Рис.4 Приклад застосування фаззі-контролера

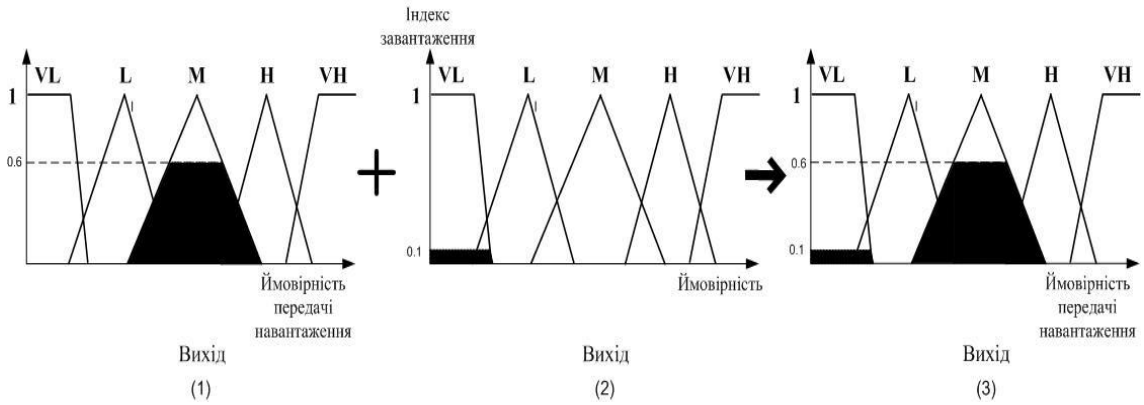


Рис.5 Вихід після композиції Правил 1 та 2

Дефазифікація – визначення точного значення виходу, тобто ймовірності передавання навантаження між вузлами A та B. Для цього застосовується метод повної інтерпретації, згідно до якого відшукується центр ваги відповідної функції належності, отриманої після композиції.

Очевидно, що запропонований алгоритм може використовуватись як для схем з ініційованим відправником, так і схем з ініційованим отримувачем.

Загальна схема балансування навантаження у випадку ініційованого відправника наступна. Фізичний вузол, приймаючи навантаження, перевіряє свій індекс завантаження, використовуючи фаззі-контролер і приймає рішення про необхідність передавання навантаження. В гомогенних мережах всі вузли можуть мати однакові функції належності, а в гетерогенних - різнитись між собою. В останньому випадку для визначення ймовірності передавання навантаження приймач повинен додатково відправити передавачеві параметри своєї функції належності.

Послідовність пошуку вузлів-кандидатів визначається ієрархічною доменною моделлю мережі. Спочатку для пошуку вузлів-кандидатів в своєму домені вузом-відправником використовується груповий запит доступним йому вузлам. Якщо серед них легко завантажені вузли відсутні, відсилається запит контролеру домену для здійснення повного пошуку легко завантажених вузлів в домені. Якщо і в цьому такі вузли не знайдені, контролер домену передає груповий запит на пошук іншим вузлам (контролерам домену) свого рівня ієрархії і очікує відповідь. Якщо і в цьому випадку легко завантажені вузли не знайдені, надсилається запит контролеру домену для пошуку на вищому рівні і т.д. Пошук продовжується доки не будуть знайдені вузли-кандидати для передавання навантаження, або пошук по всій ієрархічній структурі не буде виконаний повністю. На рис. 6 наведені формальні алгоритми пошуку вузлів кандидатів для передавача , приймача та контролеру домену.

```
// G - номер домену, R – множина доступних вузлів, DR – номер домену
// N1 – номер вузла для передачі навантаження
// M – обсяг навантаження, що передається на вузол N1
//
1: if is_heavily_loaded (sender_node) {
multicast ("LOAD-TRANSFER", G – {sender_node})
wait_for_response (t, R);                    // очікування відповіді протягом часу t
if (nodes are found and stored in set R) {   // якщо множина R не пуста
N1 = choose_best(R);                         //  вибір вузла для перредачі навантаження
send_message ("FOUND", R – {N1});            //  відсилка повідомлення "Вибір здійснений"
M = find_amount(N1);                         //  розрахунок обсягу навантаження для передачі
transfer_load(N1, M);                        // передача навантаження вузлу-кандидату
}
else if (R is empty) {                       // якщо множина R пуста
// відсилання КД запиту на пошук вузлів-кандидатів
send_request_to_DR("LOAD-TRANSFER", DR);
wait_for_reponse(t2, R);                     // очікування відповіді протягом часу t
if (set R is not empty) {                    // якщо множина R не пуста
N1 = choose_best(R);                         //  вибір вузла для перредачі навантаження
Send_message ("FOUND", DR);                  //  відсилка повідомлення"Вибір здійснений"до КД
M = find_amount(N1);                         //  розрахунок обсягу навантаження для передачі
transfer_load(N1, M);                        // передача навантаження вузлу-кандидату
}
else {                                       // множина R пуста
wait(S);                                     // пауза протягом часу S
Go To 1;
}
}
}
```

Рис.6 Формальний алгоритм пошуку адресата для ініційованого передавача

```
//  отримати запит на передавання навантаження від вузла або КД
if get_message_for_load_transfer (sender_node, DR) {
// якщо вузол не первантажений
if is_lightly_loaded(receiver_node) {
//
// відправити передавачу параметри власної функції належності
//
send_ack (sender_node, membership_function_parameters);
//
//  відправити КД повідмлення про те, що вузол доступний
send_ack_to_DR(msg, DR);
wait_for_load_transfer(t);          //  очікувати початку передавання навантаження
if (t > Max. Delay Time in the network) {      //  якщо перевищений час очікування
//
// відмовитись ві прийняття навантаження
stop_load_transfer(sender_node, stop_msg);}
else
start_load_transfer(sender_node); // розпочати прийом навантаження
}
else                                  // якщо навантаження вузла надмірне
ignore(sender_node);                //  ігнорувати повідомлення
```

Рис.7 Формальний алгоритм пошуку адресата для приймача

```
//  отримати запит на пошук кандидата від логічного чи фізичного вузла
get_message_from_node("LOAD-TRANSFER", N);
//
// якщо запит від фізичного вузла
if load_transfer_request(request_msg, N) {
// пошук по всьому дереву в глибину
//
Multicast(request_msg, G – {DR});
wait_for_response(t);
//
// якщо вузол-кандидат протягом заданого часу не знайдений
if ( (a lightly loaded node is not found) and
(t > Max. delay time in the network) ) {
//
// опитування КД  поточного логічного домену
// та відсилання контролеру домену запиту на пошук вузлів-кандидатів
//
poll another DR in the same logical group and
send a message to that DR for load transfer;
}
}
```

Рис.8 Формальний алгоритм пошуку адресата для контролера домена

Як адресат для надлишкового навантаження з відправника з поточного відправника використовується вузол, для якого ймовірність передавання навантаження максимальна.

Адаптивне керування трафіком

Затори в мережі можуть призводити до збільшення навантаження вузлів. Наприклад, у випадках, коли затор має місце на шляху до вищого рівня ієрархії, він звужує поле пошуку кандидатів на передавання навантаження і частина легко завантажених вузлів, що не можуть передати інформацію про параметри свого завантаження, не мають змоги прийняти його. В нашому алгоритмі з метою зменшення заторів застосований фаззі-алгоритм керування, в якому функції належності можуть змінюватись.

Роутер може змінити функції належності залежно від індексу завантаженості мережі. Нами застосовується функція належності, для якої визначені три нечіткі терми – низький, середній та високий трафік. (рис.9).

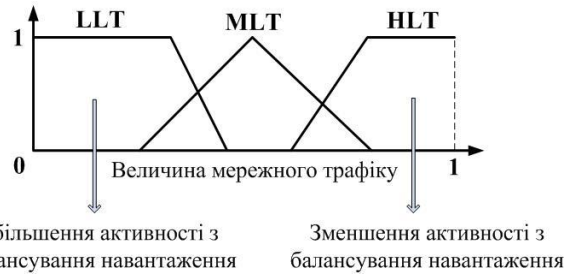


Рис.9 Функція належності для мережевого трафіку

У випадку легкого завантаження мережі вузол може підвищити активність виконання балансування навантаження, наприклад, приймаючи більше запитів на балансування, а якщо навантаження мережі високе - зменшити її, прийнявши менше пакетів від інших вузлів, і таким чином впливати на рівень трафіку в інших доменах. На рис.10 наведений приклад такої зміни функції належності у випадку високого трафіку.

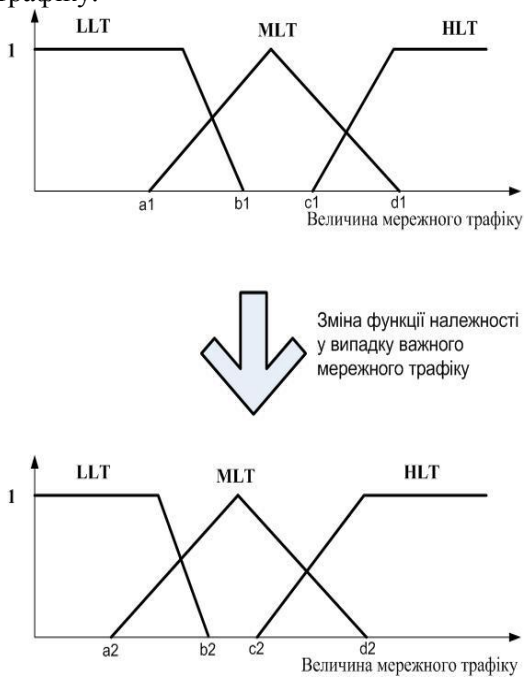


Рис.10 Приклад зміни функції належності для важного мережного трафіку
(*a2<a1, b2<b1, c2<c1, d2<d1)*

Методи визначення поточного стану мережі

Іншою проблемою алгоритмів динамічного балансування навантаження є обґрунтований вибір моделі для обчислення індексу завантаження. Роботи, присвячені цьому питанню, зокрема [3, 4], вказують на те, що використання індексу завантаження, який обчислюється як зважена сума деякого набору визначених параметрів, не завжди достатньо об'єктивно відображає рівень завантаження внаслідок обмеженням на число параметрів, які використовуються в моделі та фіксованим значень вагових коефіцієнтів. В той же час найбільше впливають на продуктивність комп'ютерної системи завантаження процесора та рівень використання оперативної пам'яті. Виходячи з цього, завантаження вузла розраховується за формулою:

$$Node\_Load = K_1 * CPU\_Load + \\ + K_2 * Mem\_Load \quad , \qquad (1)$$

де CPU_Load, Mem_Load - відповідно поточне завантаження процесору (%) та рівень використання оперативної пам'яті (%) ; K_1, K_2 - вагові коефіцієнти.

В якості метрики трафіку в мережі використовується величина часу між відправкою запиту на вузол та отриманням відповіді від нього (RTT - Round Trip Time).

Виміряні значення метрик у подальшом підлягають статистичній обробці з наступною агрегацією, що обумовлено наступними причинами:

- неможливістю здійснення у великих мережах моніторингу стану всіх з'єднань та вузлів мережі;
- використання усереднених зважених значень метрик окремих з'єднань для визначення відповідних оцінок метрик для мережі в цілому;
- доцільністю використання для оцінки завантаження мережі, трендів метрик мережі у часі.

З викладених вище міркувань доцільним є агрегація отриманих у результаті вимірювань даних за часом.

Отримані в результаті вимірювань на інтервалі ΔT значення метрики розглядаються як вибірка, елементи якої мають два параметри <час, метрика> з відповідними значеннями $<T_i, M_i>$, що належать інтервалу часу (T_0, T_f).

Будемо розглядати інтервал (T_0, T_f) як N послідовних підінтервалів ($T_{a,k-1}, T_{a,k}$). Тоді підмножини значень $<T_i, M_i>$, згруповані на підінтервалах, формують вибірки S_k, які є підмножинами вибірки S.

Для кожного $<T_i, M_i>$ визначимо значення агрегації на інтервалі ($T_{a,k-1}$, $T_{a,k}$) як $V_k = F(M_i),\ T_{a,k-1} \le T_i < T_{a,k}$, де F – деяка статистична функція. В результаті для кожного підінтервалу отримуємо ΔT метрику <

$(T_{ak-1}, T_{ak}), V_k >$. Розмір ΔT може бути фіксованим, або плаваючим й різнитись за довжиною залежно від метрики та мети агрегування. Статистичними функціями, що найчастіше застосовуються, є середнє значення, медіана та *n*-відсотковий інтервал вибірки з ΔT_i

У табл.2 наведені основні параметри системи визначення поточного стану мережі.

Табл. 2

Параметри системи визначення поточного стану мережі

Метрика	Шлях отримання	Протокол	Період	Статистична функція
RTT	Періодичні ping-запити	ICMP	< 1мс	- медіана - середнє - 2.5% інтервал - 97.5 % інтервал
Поточний стан елементів мережі та з'єднання	Періодичні запити до MIB. Обробка внутрішніх переривань (SNMP traps) Періодичні ping-запити	SNMP ICMP	Визначається необхідним рівнем точності	-
Використання процесору вузла	SNMP-запити до робочих станцій вузла CLI- утиліти	SNMP	Визначається необхідним рівнем точності	- медіана - середнє - 2.5% інтервал - 97.5 % інтервал
Об'єм доступної оперативної пам'яті на вузлі	SNMP-запити до робочих станцій вузла CLI- утиліти	SNMP	Визначається необхідним рівнем точності	- медіана - середнє - 2.5% інтервал - 97.5 % інтервал

Нами було виконане порівняння запропонованого алгоритму з двома відомими алгоритмами – кільцевим та випадковим шляхом імітаційного моделювання завантаження мережі, топологія якої наведена на рис.1. Основні параметри моделі та методи їх отримання наведені в табл. 3.

Табл.3

Основні параметри імітаційної моделі

Параметр	Значення
Максимальне завантаження вузла , задач	10
Час виконання задачі, кроків моделювання	Рівномірний розподіл на інтервалі [1..4]
Потік запитів на виконання задач	Розподіл Пуассона $\tilde{\lambda} = 3$

Показники якості, отримані в результаті 1000 прогонів моделі, наведені в табл. 4.

Табл.4

Порівняні показники якості алгоритмів динамічного балансування навантаження за результатами моделювання

Критерій якості	Характер задач, що виконуються в мережі					
	Розподілені обчислення			Розподілений сервер		
	RR	*Random*	*Fuzzy*	*RR*	*Random*	*Fuzzy*
Середня кількість балансувань	156,4	287,8	98,1	1000	1000	1000
Максимальне середнє абсолютне відхилення завантаження	0,911	1,02	1,2	2,17	3,45	1,13

Висновки

Дослідження показали, що використання фаззі-логіки дозволяє підвищити показників якості балансування навантаження і запобігти недоліків традиційних алгоритмів динамічного балансування.

Список літератури

1. *K. Abini*. Fuzzy Decision Making for Load Balancing in a Distributed System, Proceedings of the 36th Midwest Symposium Circuits and Systems, 1993, pp. 500–502.
2. *C.W. Cheong, V. Ramachandran*. Genetic Based Web Cluster Dynamic Load Balancing in Fuzzy Environment, Proceedings of the Fourth International Conference on High Performance Computing in the Asia-Pacific Region, Beijing, China, Vol. 2, 2000, pp. 714–719.
3. *P. Chulhye, J.G. Kuhl* A fuzzy-based distributed load balancing algorithm for large distributed systems, Proceedings of the Second International Symposium on Autonomous Decentralized Systems, April 1995, pp. 266–273.
4. *A. Corradi, L. Leonardi, F. Zambonelli*. Diffusive load-balancing policies for dynamic applications, Concurrency, 7 (1) (January – March 1999) 22–31.
5. *G.Cybenko*. Dynamic load balancing for distributed memory multiprocessors, J. Parallel Distributed Computsng, 7 (1989) 279–301.
6. *E. Damiani*. An intelligent load distribution system for CORBAcompliant distributed environments, Proceedings of IEEE International Conference on Fuzzy Systems, Vol. 1, Seoul, South Korea, 1999, pp. 331–336.
7. *M.V. Devarakonda, R.K. Iyer*. Predictability of process resource usage: a measurement-based study on UNIX, IEEE Trans. Software Eng. 15 (12) (December 1989) 1579–1586.
8. *David A. Nichols.* Using Idle Workstations in a Shared Computing Environment. In Proceedings of the Eleventh ACM Symposium on Operating Systems Principles, pages 5-12. ACM, November 1987
9. *Fred Doughs and John Ousterhout*. Transparent Process Migration: Design Alternatives and the Sprite Implementation. Software-Practice and Experience, 21(8):757-785, August 1991.
10. *Brian Bershad*. Load Balancing with Maitre d'. Technical Report CSD-85-276, University of California at Berkeley, December 1985.
11. *Harry I. Rubin.* The Design of a Load Balancing Mechanism for Distributed Computer Systems. Technical Report CSD-87-362, University of California at Berkeley, July 1987.
12. *Miron Livny and Myron Melman*. Load Balancing in Homogeneous Broadcast Distributed Systems. *Proceedings of the ACM Computer Network Performance Symposium*, pages 47-55, April 1982.
13. *K. Benmohammed-Mahieddine and P. M. Dew*. A Periodic Symmetrically-Initiated Load Balancing Algorithm for Distributed Systems. *SIGOPS*, 28(l):66-77, January 1994.
14. P. Krueger and N. Shivaratri. Adaptive Location Polices for Global Scheduling. *IEEE Transactions on Software Engineering*, 20(6):432-444, June 1994.
15. *Alonso, R., and Cova, L. L.* "Sharing Jobs Among Independently Owned Processors," In: Proceedings of the 8th International Conference on Distributed Computing Systems, IEEE, New York, pp. 282-288, June 1988.
16. Harry I. Rubin. The Design of a Load Balancing Mechanism for Distributed Computer Systems. Technical Report CSD-87-362, University of California at Berkeley, July 1987.
17. *Ramakrishnan, K., and R. Jain.* A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer. ACM Transactions on Computer Systems 8(2):158-181, May 1990.
18. *Kremien, O., and Kramer*, "Methodical Analysis of Adaptive Load Sharing Algorithms," IEEE Trans. Parallel and Distributed Systems, Vol. 3, N0. 6, Nov.
1992, pp 747-760.
19. *Theodore Faber.* "ACC: Using Active Networking to Enhance Feedback Congestion Control Mechanisms", IEEE network, pp 61-65, May/June 1998.
20. *Tennenhouse, D. and D. Wetherall.*. Towards an Active Network Architecture. In Multimedia Computing and Networking (MMCN 96), Jan 1996. San Jose, CA: SPIE. A revised version of this paper appears in Computer Communication Review, Vol. 26, No. 2 (April 96).
21. *Tennenhouse, D. J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden.* A Survey of Active Network Research, IEEE Communications Magazine, Vol 35, No. 1,pp 80-86, January 1997

Відомості про авторів:

Шаповал Інна Сергіївна , студентка, Кременчуцький національний університет імені Михайла Остроградського, наукові інтереси – паралельні комп'ютерні системи, e-mail: innesa-s@rambler.ru

Сисюк Геннадій Юрійович, доцент, Кременчуцький національний університет імені Михайла Остроградського, кафедра комп'ютерних та інформаційних систем ; кандидат технічних наук , наукові напрями – гетерогенні кластерні системи, паралельні та розподілені обчислення,
e-mail: velsinil@gmail.com