УДК 004.724.2, 004.75

АЛГОРИТМИ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ В ГРІД-СИСТЕМАХ

А.І. ПЕТРЕНКО, С.Я. СВІСТУНОВ, П.В. СВІРІН

Проведено аналіз стратегій балансування навантаження в Грід-системах та порівняльний аналіз наявних алгоритмів розподілу потоку завдань між обчислювальними ресурсами Грід-середовища.

ВСТУП

Необхідність застосування особливих механізмів управління ресурсами в *локально* і *глобально* розподілених середовищах є актуальною. До ресурсів відноситься все, що так чи інакше бере участь в обробці даних: обчислювальні кластери; сховища даних; файлові системи; програмне забезпечення (ПЗ); мережеве устаткування, яке забезпечує з'єднання ресурсів в єдину систему.

Векторні — паралельні комп'ютери, масивно-паралельні комп'ютери з розподіленою пам'яттю, паралельні комп'ютери із загальною пам'яттю та обчислювальні кластери з певною умовністю можна віднести до локальних розподілених обчислювальних середовищ «замкненої» конфігурації. Глобальне середовище метакомп'ютингу на сьогодні асоціюється з Грід-технологіями. Нинішня інтерпретація концепції Грід передбачає об'єднання в єдину обчислювальну систему як високопродуктивних суперкомп'ютерів, так і використання в якості обчислювальних ресурсів звичайних персональних комп'ютерів.

Для забезпечення вимог користувачів з продуктивності та ефективності виконання завдань Грід-система має реалізувати ефективний алгоритм розподілу завдань між доступними на даний час обчислювальними ресурсами. Основна мета такого балансування навантаження в Грід-системі — скоротити час виконання завдання користувача та забезпечити ефективність використання обчислювальних ресурсів для виключення ситуації у випадку, коли одні ресурси простоюють, а інші перевантажені виконанням завдань користувачів [1].

Незважаючи на те, що алгоритми балансування навантаження обчислювальних ресурсів у розподіленої системі досліджуються вже досить давно й існує багато готових алгоритмічних рішень, і програмних реалізацій, ін-

тенсивний розвиток Грід-технологій та вдосконалення проміжного ПЗ робить проблему балансування навантаження постійно актуальною, а інтерес до дослідження в цій галузі не зменшується, особливо для гетерогенних (неоднорідних) систем [2]. У цій роботі алгоритми балансування навантаження розглядаються з точки зору можливості застосування в українській Грід-інфраструктурі [3, 4, 5].

МОДЕЛЬ І СТРАТЕГІЇ РОЗПОДІЛУ НАВАНТАЖЕННЯ

Розглянемо загальну модель розподілу навантаження в Грід-системі G (рис. 1), яка складається з набору Грід-сайтів S_j, з'єднаних між собою звичайними каналами Інтернет [6]. Ресурси Грід-сайта включають у себе: обчислювальні ресурси, ПЗ, ресурси збереження даних і мережеву інфраструктуру. Кожен Грід-сайт S_j є обчислювальним кластером C_k з додатковими керуючими серверами із встановленим проміжним ПЗ. Це може бути проміжне ПЗ ARC проекту NorduGrid [7], gLite проекту EGI [8], UNICORE [9].

Обчислювальний кластер C_k є певною кількістю обчислювальних серверів CM_{jk}, з'єднаних комутаційним обладнанням, керуючого сервера з встановленою власної системою керування розподілом навантаження між обчислювальними серверами.

Цю модель, згідно з [11], назвемо *G/ S /C / M* моделлю, де G — Грід-система; S — кількість Грід-сайтів, що входять в Грід-систему; C — кількість кластерів, що входять у Грід-сайт; M — кількість обчислювальних серверів, що входять у кластер.

Ця загальна модель характеризується *неоднорідністю ресурсів* (як мережевих, так і обчислювальних) [12]; *автономністю ресурсів*, що належать окремим організаціям [13]; *динамічністю ресурсів* — коли обслуговуються як зовнішні, так і локальні користувачі, запити яких вишукуються в черги [14]; *спільним використанням ресурсів*, які можуть бути підключеними до багатьох Грід-систем одночасно та використовувати різне проміжне ПЗ; *різноманітністю додатків,* які можуть виконуватися паралельно чи послідовно. З урахуванням усіх аспектів завдань реалізація загальної мети балансування навантаження системи є вкрай складною [15].

У загальному вигляді стратегія балансування навантаження в Грід-системі має деревоподібну ієрархічну структуру (рис. 1), де введено такі рівні [11].

Рівень 0. Балансування навантаження між Грід-системами, які працюють під керуванням різного проміжного ПЗ. На даний час проблема функціональної сумісності та взаємодії Грід-систем не вирішена остаточно. Існуючі мета-планувальники забезпечують розподіл завдань користувачів у менш завантажену Грід-систему.

Рівень 1: Балансування навантаження всередині Грід-системи, коли планувальник збирає інформацію про поточне завантаження обчислювальних ресурсів у Грід-системі під час надходження завдань користувачів приймає рішення про розподіл завдань між Грід-сайтами.

Рівень 2: Балансування навантаження всередині Грід-сайту, коли розподілення навантаження виконується лише для завдань користувачів, які надійшли на цей Грід-сайт. Якщо в Грід-сайт входять декілька кластерів, то виконується розподіл завдань між цими обчислювальними кластерами.

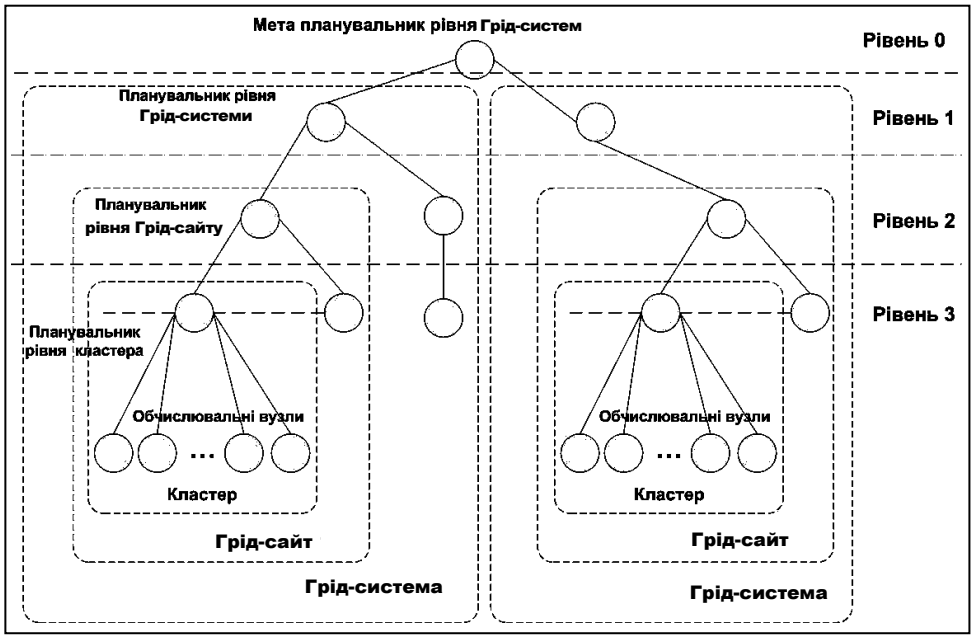

Рис. 1. Ієрархічна стратегія балансування навантаження в Грід-системі

Рівень 3: Балансування навантаження всередині кластера, коли завдання користувачів розподіляються між обчислювальними серверами, які входять у кластер, за допомогою локальної системи управління завданнями типу PBS, Condor тощо [15].

БАЛАНСУВАННЯ НАВАНТАЖЕННЯ МІЖ ГРІД-СИСТЕМАМИ

Для вибору ресурсів, моніторингу проходження завдань і доступу до даних з додатків застосовується *Resource Broker*, що відіграє роль зовнішнього *планувальника або мета-диспетчера робіт* і слугує інтерфейсом між Грід-службами і локальними планувальниками. Прикладами таких додатків є Nimrod/G [16], AppLeS Parameter Sweep Template (APST) [17], Gridbus Resource Broker (GRB) [18], GridWay [19].

При цьому для користувача забезпечується єдина точка входу до різних Грід-систем як до єдиного обчислювального ресурсу, функціональну сумісність яких засновано на використанні адаптерів і трансляторів. Для кожної Грід-структури в структурі мета-планувальника включено спеціальний адаптер, що змінює опис завдання користувача таким чином, щоб воно було зрозумілим іншій Грід-системі при наступній трансляції. У таблиці наведено характеристики досить простих алгоритмів, які використовуються в цих планувальниках ресурсів для забезпечення функціональної сумісності різних Грід-систем.

Таблиця. Характеристики алгоритмів планування ресурсів

Мета планувальника	**Інформаційна система**	**Застосований алгоритм планувальника**	**Особливості**
Nimrod/G	Інформаційний сервіс MDS Globus	В основу алгоритму покладено концепцію сервісу торгівлі ресурсами GRACE (Grid Architecture for Computational Economy — Грід-архітектура для «обчислювальної економіки»). Ключові параметри алгоритму: T — кінцевий термін виконання завдання, CR — умовна вартість ресурсу, CZ — умовний бюджет завдання. Алгоритми планування: вибір ресурсу найменшої вартості для виконання завдання в термін T (оптимізація вартості); вибір ресурсу для мінімізації терміну виконання завдання (оптимізація часу); оптимізація за критерієм час — вартість, яка подібна вартісній оптимізації, але за наявності множини ресурсів однакової вартості застосовується стратегія оптимізації часу; консервативна за часом стратегія, яка подібна оптимізації часу, але гарантує кожному завданню мінімум витрат на одне завдання	Для початку планування мають бути встановлено такі властивості завдання як «термін виконання» та «бюджет»
AppLeS Parameter Sweep Template (APST)	Network Weather Service (NWS), MDS, Ganglia	В якості метрик використовуються статичні характеристики (архітектура, обсяг пам'яті, дисковий обсяг), характеристики мережі (смуга пропускання, навантаження та затримка), пріоритет черги, надійність ресурсу, цінові характеристики ресурсу. Алгоритми планування: евристичні алгоритми Min-min, Max-min, Sufferage, и Xsufferage [20]; сторонні алгоритми планування виконання завдань [21]	Планування проводиться на короткий період часу з наступними перепланауваннями
Gridbus Resource Broker (GRB)	Grid Market Directory (GMD), Grid Index Information Service (GIIS)	Алгоритми планування проводять оптимізацію за критеріями мінімізації часу виконання завдання та ціни на основі параметрів наявних обчислювальних ресурсів та параметрів опису задачі (термін виконання, бюджет тощо)	Брокер не є Грід-сервісом, а встановлюється на кожній клієнтській машині

БАЛАНСУВАННЯ НАВАНТАЖЕННЯ ВСЕРЕДИНІ ГРІД-СИСТЕМИ

Час обробки завдання складається з часу: аналізу опису завдання, пошуку відповідного ресурсу, завантаження завдання на ресурс, очікування в черзі, виконання завдання на ресурсі та завантаження результатів користувачеві. Добре відомо, що задача оптимального розподілення робіт за ресурсами є NP-повною [36], тому існує лише наближені алгоритми балансування. Їх можна розділити на централізовані і децентралізовані, динамічні й статичні, періодичні й неперіодичні. Усі алгоритми балансування навантаження по-

будовано таким чином, щоб навантаження ресурсів було рівномірним, або використання ресурсів було максимальним при мінімізації загального часу виконання завдання. Прийнято розрізняти три типи таких алгоритмів, які здійснюють:

- *розподіл навантаження* у випадку розподілу навантажень на обчислювальний ресурс, який в даний час не використовується в обчисленнях, тобто вільний;
- *балансування навантаження*, коли завдання розподіляються між обчислювальними ресурсами таким чином, щоб забезпечити більш-менш рівномірне використання ресурсів або задовольнити певні критерії балансування (більше завдань на потужних кластерах);
- *вирівнювання навантаження*, коли не просто здійснюється рівномірний розподіл навантаження серед усіх ресурсів або вирівнювання навантаження, а планувальник прагне уникнути перенавантаження окремих ресурсів.

Вибір конкретного алгоритму балансування навантаження залежить від обраної моделі побудови брокера ресурсів, які можуть бути трьох таких типів.

Централізований брокер ресурсів — виконує вибір ресурсів за вимогами завдань користувачів для всієї Грід-системи, тобто повністю відповідає за розподіл усіх наявних ресурсів (рис. 2). Усю необхідну інформацію про динамічні характеристики ресурсів централізованого брокера він отримує від інформаційної системи. Прикладом реалізації такої моделі є проміжне забезпечення gLite [8], в якому WMS (Workfolw Management System — система управління потоком знань) відповідає за розподіл і управління завдань у межах наявних ресурсів [22–23].

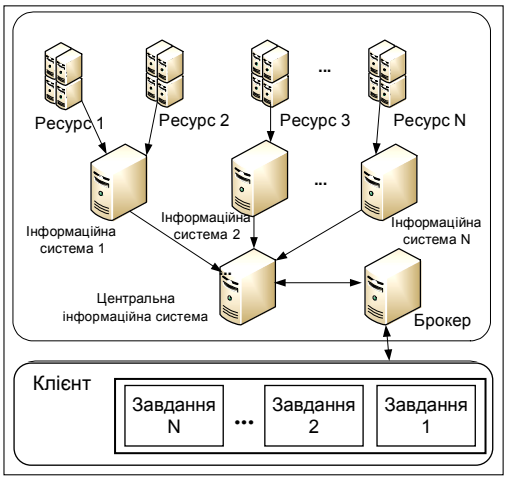

Рис. 2. Загальна архітектура Грід-системи з централізованим брокером

WMS також контролює процес виконання завдань та забезпечує отримання результатів виконання завдання користувачем. В gLite для балансування навантаження використовуються алгоритми «максимальної оцінки» та «нечіткої оцінки». Ці алгоритми проводять відбір ресурсів-кандидатів за допомогою оцінювання відповідності ресурсу вимогам завдання, яке надійшло в чергу.

Модель розподіленого брокера, в якому за результатами аналізу опису завдання на виконання кожен Грід-сайт формує свій перелік доступних обчислювальних ресурсів. Необхідну динамічну інформацію про стан Грід-системи такий брокер може отримувати від центральної інформаційної системи, а також шляхом запитів безпосередньо до доступних ресурсів (рис. 3), після чого починає пошук необхідного ресурсу для завдання користувача [24].

Прикладом реалізації такої моделі є проміжне забезпечення ARC [7, 38] проекту NorduGrid, у брокері якого реалізовано алгоритми вибору ресурсів, що забезпечують достатню масштабованість та надійність системи. У склад ARC включено алгоритми «Random» (проводить сортування кандидатів випадковим чином); «Benchmark» (проводить сортування згідно з оцінкою продуктивності ресурсу); «FastestQueueBroker» (сортує список кандидатів залежно від коефіцієнта наповненості черги ресурсу, наприклад, довжини черги, поділеної на загальну кількість процесорів); «Data» (сортує ресурси згідно з обсягом даних, що зберігаються в кеш обчислювального ресурсу).

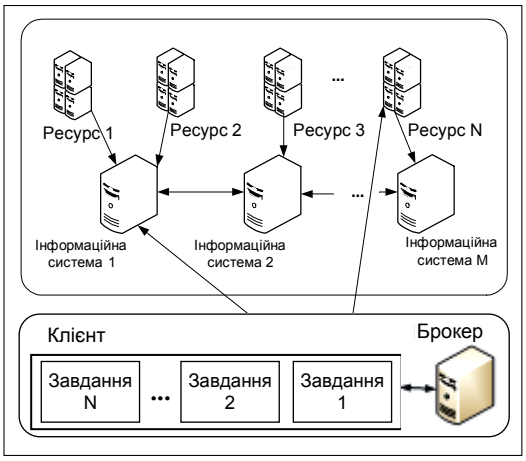

Рис. 3. Архітектура системи з розподіленим брокером ресурсів

Модель ієрархічного брокера поєднує в собі елементи централізованого та розподіленого підходів. На рис. 4 зображено узагальнену структуру ієрархічної моделі, що містить декілька екземплярів брокера ресурсів, які розподіляють свої функції за встановленими правилами. На відміну від одного великого ресурсного брокера існує декілька підходів для створення цієї ієрархічної структури, наприклад, віртуальна ієрархічна структура віртуальної організації, яку представлено в [25, 26], та різні ієрархічні структури використання Грід-систем на основі різного проміжного ПЗ [27].

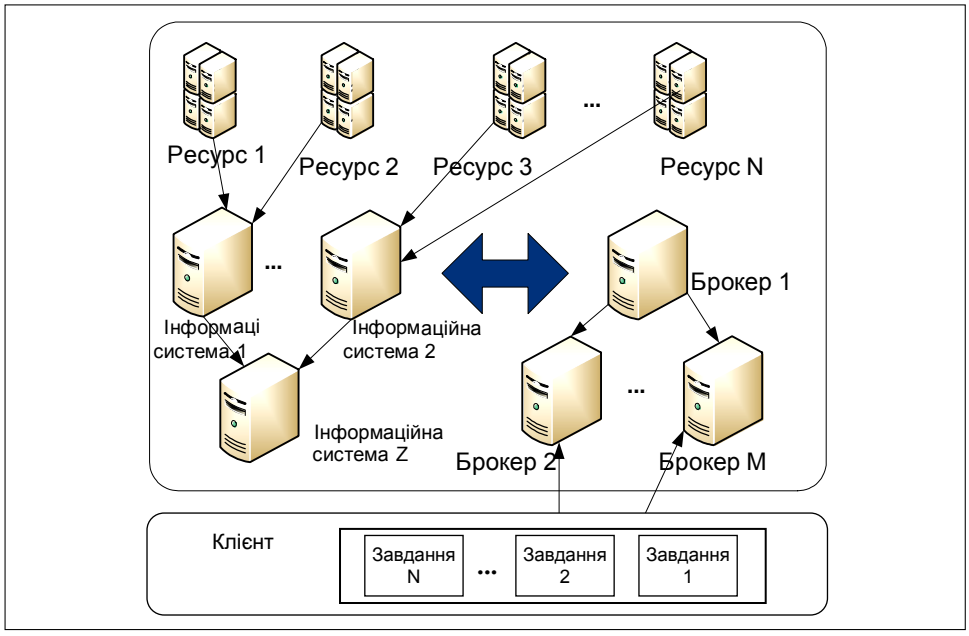

Рис. 4. Архітектура системи з ієрархічним брокером ресурсів

Одним із прикладів реалізації такої моделі є ієрархічна структура [27], для якої розроблено новий адаптер-планувальник, що дозволяє службі

управління Грід-ресурсами Globus GRAM відправляти завдання до GridWay [19], який потім використовує свій брокерський алгоритм для подання завдань до LRMS, або на Грід-системи з різним проміжним ПЗ, таким як AstorGrid та gLite.

Модель агентського брокера використовує спеціальні програмні компоненти (агенти), що реалізують алгоритм штучного інтелекту, автономні за своєю суттю, здатні до самовідновлення і прийняття рішень [28, 29, 30, 31]. Загальну структуру такої моделі показано на рис. 5. Динамічний характер агентів відрізняє цей підхід від традиційних, розглянених вище. Взаємодія між агентами у межах системи може бути децентралізованою, ієрархічною або представленою за схемою «головний–підлеглий».

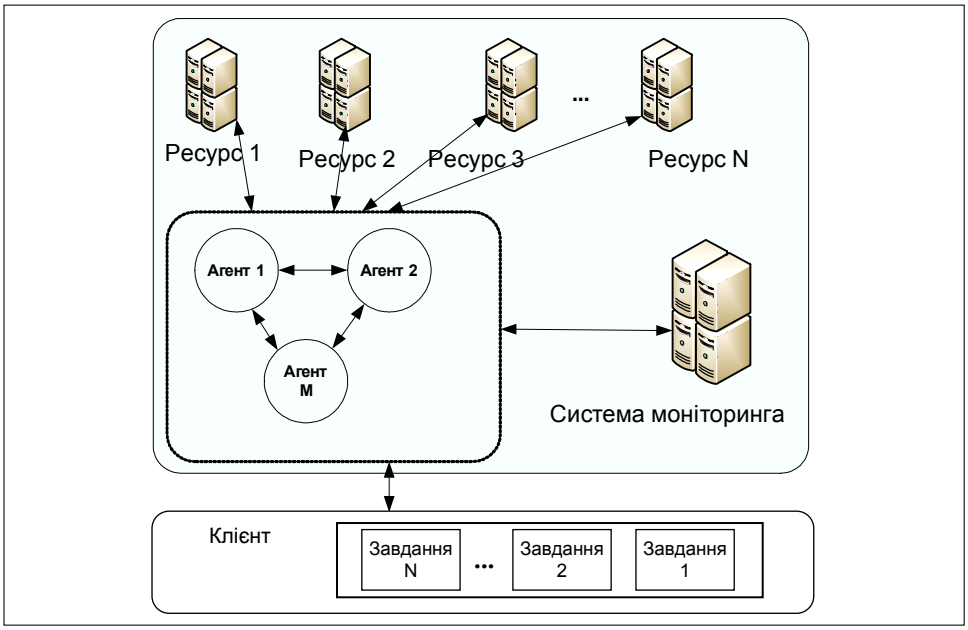

Рис. 5. Архітектура система з агентським брокером

Одним із прикладів реалізації агентської моделі є проміжне забезпечення AliEn [32], яке використовується для обробки даних в експерименті ALICE [33]. За виконання завдання на ресурсі відповідає Job Agent, який стартує на обчислювальному сервері кластера. Він звертається до центральної черги завдань; отримує JDL файл опису завдання, використовуючи сервіс PackMan, встановлює необхідні додатки; контролює виконання завдання та зберігання результатів виконання.

Іншим прикладом може бути Система Управління Ресурсами Агентів [26], в якої кожен агент відповідає за один або декілька ресурсів, що означає, що агенти в основному є постачальниками послуг. Інформація про ресурси, котрі надаються агентами, та зв'язок між ними забезпечується механізмом пошуку ресурсів.

БАЛАНСУВАННЯ НАВАНТАЖЕННЯ ВСЕРЕДИНІ КЛАСТЕРА

Завдання, призначені окремим обчислювальним ресурсам, виконуються на серверах кластера під наглядом локальної системи управління ресурсами, яку в різних джерелах називають по-різному: *JMS* (job management system —

система управління завданнями), *RMS* (resource management system або resource management and scheduling — система управління ресурсами або система управління та планування), *CMS* (cluster management software — система керування кластером) тощо. У ПЗ локальних систем можна виділити два компоненти: менеджер ресурсів і планувальник. Менеджер відповідає за розподіл процесорних вузлів (серверів), а планувальник визначає черговість виконання робіт (рис. 6). В основі багатьох локальних систем управління ресурсами лежить архітектура «клієнт-сервер». Найчастіше реалізується режим batch job (пакетної обробки).

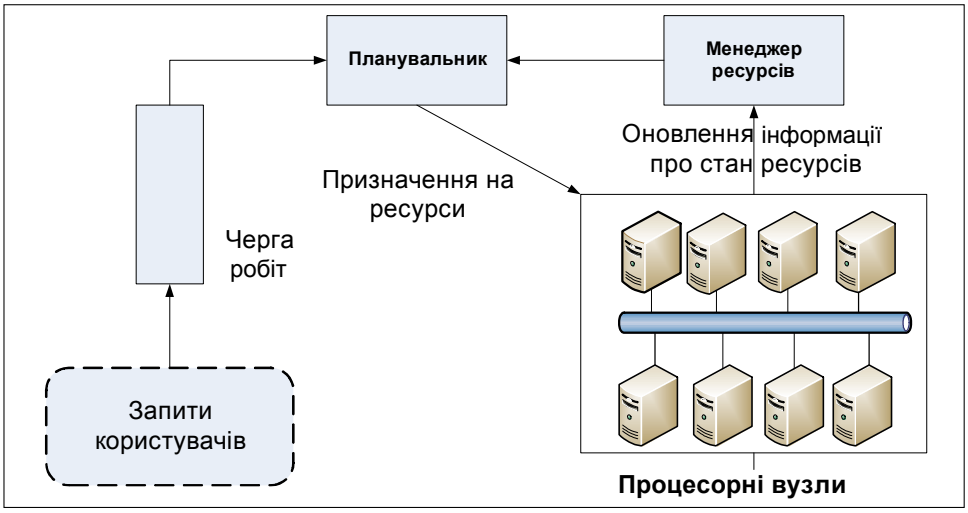

Рис. 6. Типова структура системи управління ресурсами

Під час управління ресурсами використовується *механізм черг* та *планування робіт*, або диспетчеризації. Два принципи організації черг завдань, які найчастіше використовуються, такі: FCFS (first come first serve — «першим прийшов — першим виконується») та LWF (least work first — «найменш трудомістка робота виконується першою»). У LWF трудомісткість може бути виражена через визначення необхідного числа процесорних вузлів та оцінку часу виконання роботи. Черги конфігуруються за допомогою спеціальних атрибутів.

Наприклад, у планувальнику Maui [34] кожна з робіт забезпечується атрибутами *geom* та *dux* [35]. Перший — описує необхідну конфігурацію ресурсів: запис $geom = 2,4$ означає, що для виконання роботи потрібно два вузли, в одному з яких задіюються два, а в іншому чотири процесори. Атрибут *dux* вказує на тривалість виконання завдання. Відповідно зі значеннями атрибутів алгоритм планування визначає черговість завдань і виділяє для них ресурси. Так, з черги може бути обрано завдання, для виконання якого достатньо наявних вільних процесорів.

Локальна система управління ресурсами, як правило, підтримує сукупність черг. Роботи можуть розподілятися за чергами в залежності від спеціальних вимог до ресурсів у запиті (необхідний доступ до високопродуктивних графічних станцій тощо). Ресурсний запит оформлюється у вигляді скрипту команди *qsub*. У цілому диспетчеризація в режимі пакетної обробки дозволяє ефективніше використовувати обчислювальні ресурси шляхом

перерозподілу початку виконання завдання та завантажувати трудомісткими, багатогодинними роботами ресурси, наприклад вночі.

АЛГОРИТМИ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ

Основною метою алгоритмів балансування навантаження є прискорення виконання завдань користувачів на ресурсах, робоче навантаження яких може бути непередбачуваним під час виконання завдання. Отже, важливо встановити метрику, яка визначає робоче навантаження ресурсу. Кожен алгоритм динамічного балансування навантаження має своєчасно оцінювати інформацію про стан обчислювального ресурсу [6], яка зводиться до відповідей на такі питання:

- «Як виміряти робоче навантаження ресурсу?»;
- «Які критерії є основними для визначення цього робочого навантаження?»;
- «Як уникнути негативного впливу динамічності ресурсу на робоче навантаження?»;
- «Як враховувати неоднорідність ресурсів для того, щоб миттєво отримати середній показник робочого навантаження системи?».

Балансування навантаження в розподілених середовищах має враховувати два аспекти: різні обсяги обчислень для різних типів завдань і неоднорідність ресурсів, зокрема, неоднакову продуктивність процесорів. Критерій ефективності використання процесорів можна визначити таким чином. Нехай *i*-й процесор реалізує свій обсяг обчислень w_i за час T_i. Час виконання програми дорівнює

$$T_f = \max\{T_i\},\ i = 1,\ldots,P,$$

де P — число процесорів. Час T_i включає в себе чистий час T_c, що витрачається на обчислення, а також деякий час на обмін даними з базовим процесором. Час T_c визначається обсягом обчислень w_i і продуктивністю процесора $p_i(t_i)$, що залежить від стану (ступеня завантаженості) системи в момент часу t_i:

$$T_{ci} = \frac{w_i}{p_i(t_i)}.$$

Ефективність використання процесорів формально визначається так:

$$RE = \sum_{i=1}^{p} \frac{T_{ci}}{T_f} \times \frac{p_i(t_i)}{\sum_{j=1}^{p} p_j(t_j)}.$$

Тоді, завдання балансування навантаження полягатиме в максимізації значення RE на відрізку часу T_f при заданому числі P процесорів.

Успіх алгоритму розподілу навантаження залежить від кількості параметрів, які входять у формули обчислення, а саме: часу їх вимірювання, часу обчислення робочого навантаження ресурсів та загального часу прийняття рішення за алгоритмом розподілу навантаження. Варто зазначити, що час

реалізації (виконання) алгоритму розподілу навантаження є найважливішим показником для користувача. Важливим є також урахування витрат на передачу завдання між вузлами Грід-інфраструктури, визначених операцією балансування навантаження, однак отримання та верифікація цієї інформації пов'язані з великими труднощами [37].

Алгоритми балансування навантаження можна поділити на дві категорії: статичні і динамічні [40]:

Статичні алгоритми — які розподіляють завдання на обчислювальні ресурси, виходячи з завантаження обчислювальних ресурсів у момент надходження завдання, і в якості критерію балансування вибирається рівномірне навантаження ресурсів (рис. 7). Для статичного планування необхідні методи визначення обсягів робіт w_i для кожного з ресурсів, а також методи оцінки продуктивності $p_i(t_i)$.

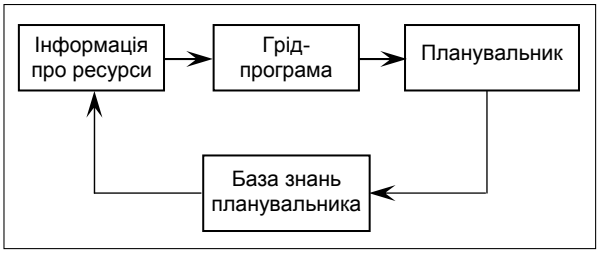

Рис. 7. Статичне балансування навантаження

Перевагою такого роду алгоритму є простота з точки зору програмної реалізації, оскільки немає необхідності постійно контролювати наявне завантаження обчислювальних ресурсів та накопичувати статистику завантаження [40]. До недоліків статичних алгоритмів слід віднести значну похибку в оцінках w_i та $p_i(t_i)$ при різних значеннях продуктивності процесорів обчислювальних ресурсів. Статичні алгоритми працюють добре тільки тоді, коли різниця в навантаженні обчислювальних ресурсів невелика. Корисні реалізації статичних методів балансування навантаження наведені в [16], серед яких слід визначити такі алгоритми:

- Round-Robin Algorithm (алгоритм кругового обслуговування) — завдання розподіляються за обчислювальними ресурсами в послідовному порядку з першого до останнього [39, 40];
- Randomized Algorithm (алгоритм випадкового вибору), коли обчислювальний ресурс для виконання завдання обирається за допомогою випадкового вибору [41];
- Simulated Annealing or Genetic Algorithms (генетичний алгоритм) забезпечує змішану процедуру розподілу завдань за обчислювальними ресурсами з використанням методів оптимізації [42].

До недоліків алгоритмів статичного балансування навантаження слід віднести труднощі з апріорною оцінкою часу виконання завдання.

Динамічні алгоритми вирішують задачу розподілення завдань на основі поточної інформації про завантаження всіх доступних обчислювальних ресурсів (рис. 8).

Як результат динамічні алгоритми виконують розподіл завдань коректніше, оскільки для призначення завдання на обчислювальний ресурс окрім статичної інформації про характеристики ресурсів, яка зберігається в інформаційній системі, використовується поточна інформація про реальне завант-

таження обчислювальних ресурсів, що надходить в інформаційну систему від постачальників у режимі реального часу (надходять усі зміни в значеннях параметрів оцінки стану ресурсів). У літературі [43] визначено три головних чинники, які зазвичай визначають стратегію та вибір конкретного алгоритму балансування завантаження. Ці чинники пов'язано з відповідями на такі основні питання:

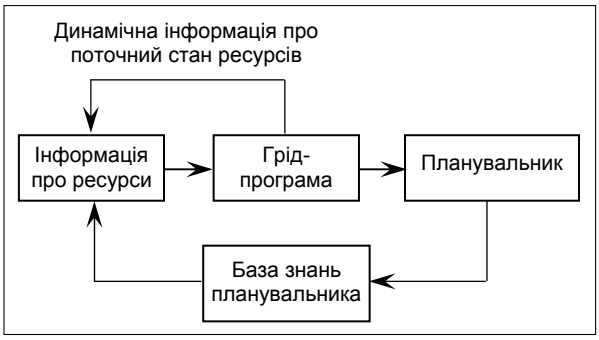

Рис.8. Динамічне балансування навантаження

- «Хто приймає рішення про розподіл завдань за ресурсами?»;
- «Яка інформація використовується для прийняття рішення про розподіл завдань за ресурсами?»;
- «Де приймається рішення в ієрархії структурних елементів проміжного ПЗ Грід-інфраструктури?».

Успішні реалізації динамічних методів балансування навантаження систематизовано в [3, 38], серед яких відзначимо такі:

- Sender-Initiated Strategy (балансування на вимогу клієнта) та Receiver-Initiated Strategy (балансування з серверного боку) — перевантажені ресурси намагаються перенести роботу до ненавантажених ресурсів (випадок балансування з серверного боку) рис. 9 чи, навпаки, ненавантажені ресурси шукають перенавантажені ресурси, з яких можна отримати роботу (випадок балансування на вимогу клієнта) (рис. 10) [45]. Методика балансування на вимогу клієнта ефективніша, ніж методика балансування з серверної сторони для ненавантажених Грід-систем, оскільки в цих умовах вірогідність знайти ненавантажений вузол вища, ніж знайти перенавантажений вузол [45, 46].

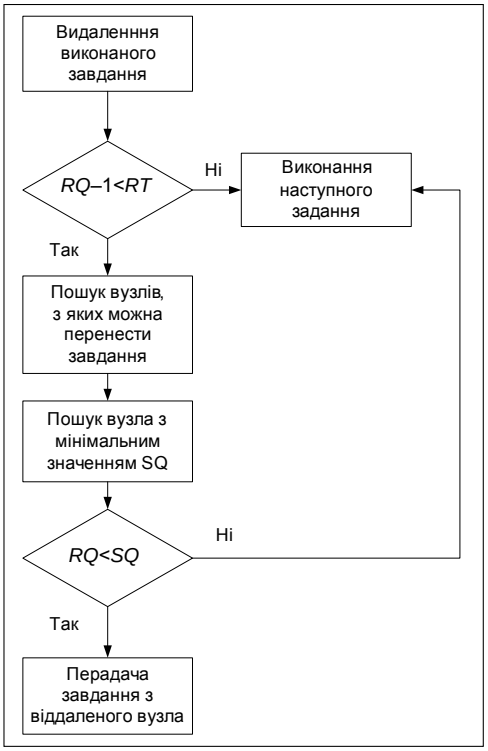

Рис. 9. Балансування з серверної сторони

- *Балансування на вимогу сервера* — обчислювальний ресурс проводить пошук іншого ресурсу, з якого можна перенести завдання у власну чергу і при цьому не перевершити максимальну розмірність черги T із урахуванням параметрів завдань. Якщо остання умова не виконується, то ресурс, що

ініціював пошук, не починає виконувати знайдене завдання і продовжує пошук, поки не будуть знайдено ресурси для завдання, що задовольняють наведеним умовам, або кількість спроб пошуку не досягне певного значення L, після якого пошук зупиняється. Цю стратегію із значенням $T = 1$ було досліджено в [47].

- *Резервування* — з одного боку, подібно до стратегії балансування на вимогу клієнта переадресовуються лише нові завдання, а ресурси намагаються зарезервувати для кожного з наявних завдань деяку частину процесорного часу, з іншого — подібно до стратегії балансування з серверного боку ресурс намагається замінити завдання, що виконуються, якщо на ньому в стані виконання залишаються менше ніж T завдань. Обчислювальний ресурс, з якого можна перенести завдання, обирається випадковим чином і перевіряється, чи буде довжина черги на ньому після трансферу деякого завдання нижче певного значення T. Якщо ця вимога не виконується і немає інших резервацій на ініціюючому ресурсу, то для обраного завдання резервується процесорний час і воно передається на ініціюючий ресурс (якщо за цей час не надійшли на виконання інші завдання).

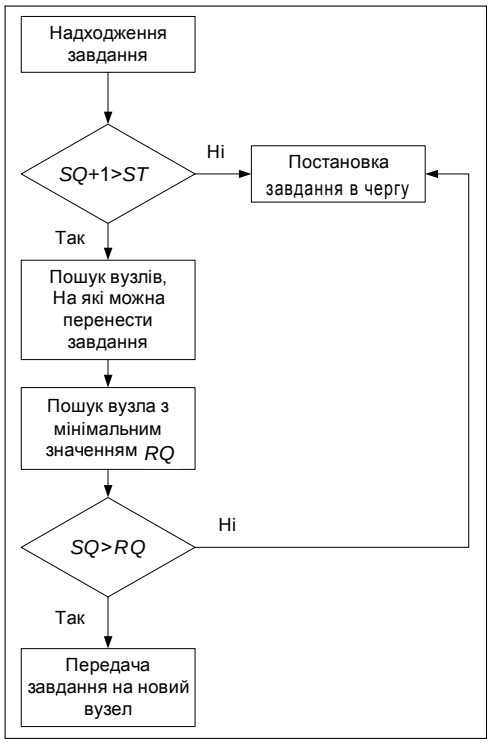

Рис. 10. Балансування на вимогу клієнта

Якщо умови розміру черги на віддаленому ресурсі не виконуються, то продовжується пошук іншого ресурсу, з якого можна перенести деяке завдання, поки не знайдеться відповідний ресурс або кількість спроб пошуку не досягне певного значення L. У цьому випадку ініціюючий ресурс має дочекатись, поки не буде завершено виконання іншого завдання, після чого процес розпочнеться з початку. Дослідження показують, що резервування поступається за продуктивністю двом попереднім підходам [40].

Якщо довжина черги є єдиним показником завантаження ресурсу, то може використовуватися трансферна політика пересилки завдань, яка базується на врахуванні співвідношення довжини вхідної черги (SQ) і її порогу ST після отримання нового завдання. Необхідна інформація про завантаження отримується опитуванням інших ресурсів клієнтом через broadcast-повідомлення про довжини їх черг. Отримавши необхідну інформацію, клієнт може обрати ресурс із найменшою довжиною черги (RQ), де значення RQ задовольняє нерівність $SQ > RQ$ (рис. 10).

- *Симетрично-ініційований алгоритм* — комбінація балансування з серверної та клієнтських сторін. Ресурс може використовувати алгоритми балансування з клієнтської сторони в тих випадках, коли довжина черги перевищує поріг ST, або алгоритми балансування з серверного боку, коли довжина черги знижується до значення, меншого за RT [48].

- *Евристичні алгоритми*, до яких відносяться такі алгоритми, як Min-min, Max-min, Sufferage, Xsufferage [20]. Алгоритм Min-min вибирає завдання з набору завдань, що має мінімальний час виконання на наявних ресурсах, і надсилає його на той ресурс, який за оцінкою забезпечує мінімальний час виконання. Алгоритм Max-min працює аналогічно, але першим опрацьовує завдання з максимальним оціночним часом, що може бути ефективно в наборі завдань, в якому домінують короткі завдання. В алгоритмах Sufferage та Xsufferage замість часу виконання використовується значення, яке розраховується, як різниця двох мінімальних оціночних [20].

ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ АЛГОРИТМІВ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ

Балансування є головною складовою існуючих підходів до динамічного планування, при чому самі підходи відрізняються за своєю точністю та кількістю інформації, необхідної для аналізу [45]. Алгоритм Джоу [46] проводить балансування навантаження, періодично вимагаючи від кожного ресурсу інформувати всі відомі йому ресурси про зміни в його навантаженні. Вілебек, Лем Маір та Рів [47] запропонували чотири політики планування, які динамічно балансують навантаження під час використання інформації про поточний стан інших ресурсів. Обидва підходи — централізований і децентралізований — вимагають досить частого обміну повідомленнями для збору інформації про поточний стан ресурсів і прийняття рішень. У повністю розподіленій системі з N обчислювальними ресурсами для прийняття рішення про планування ресурс має обмінятись $2(N-1)$ повідомленнями [50]. Гібридні алгоритми [45] об'єднують у собі переваги статичних і динамічних стратегій. У гібридних алгоритмах статичний алгоритм вважається «грубим» розподіленням завдань за ресурсами, а динамічний алгоритм — «тонким» регулюванням первинного розподілу. Під час використання тільки статичного алгоритму балансування навантаження може виникнути дисбаланс у Грід-системі. Коли це трапляється, динамічний алгоритм починає свою роботу і забезпечує збалансованість завдань у чергах по всій Грід-системі.

ВИСНОВКИ

Кожна з розглянутих вище моделей побудови архітектури брокера ресурсів має свої переваги та недоліки. Наприклад, широкі можливості функціонального використання централізованої моделі визначаються її масштабованістю та продуктивністю. Натомість, можливості розподіленого брокера звужені. Ієрархічний підхід може бути ефективним для розподілу завдань у Грід-середовищі, але потребує більше часу для визначення необхідних ресурсів. Використання автономних агентів у системі для отримання її поточного стану може ідеально дозволити керувати розподілом ресурсів у ефективний і надійний спосіб, але вимагає побудови чіткої системи зв'язків між ними.

Серед алгоритмів балансування навантаження ресурсів Грід-системи можна розрізнити статичні й динамічні залежно від параметрів, що враховуються під час прийняття рішення про призначення і виконання завдання. У статичному підході використовується середньостатистична інформація

про роботу системи, ігноруючи її поточний стан. Статичні алгоритми [49, 50] частіше застосовуються для балансування періодичних завдань із жорсткими вимогами щодо часу завершення.

Навпаки, динамічні алгоритми [51, 52] виконують балансування в реальному часі і динамічно визначають можливість поліпшення балансування навантаження з врахуванням вже призначених для оброблення розподілених задач [26]. Оскільки серед задач у Грід-середовищі періодичні є меншістю, то доцільніше використовувати динамічні алгоритми.

Під час обговорення згаданих вище брокерських структур і стратегій балансування було з'ясовано, що задача розподілу ресурсів пов'язана не лише з вибором механізму розподілу, але значною мірою визначається інформацією, доступною для брокера, і її актуальністю. Підтримка оновлення інформації про стан ресурсів може бути досить витратною і вартість її зростає разом із розміром системи. Чим ефективніше управління інформаційними потоками в Грід-системі, тим ефективніший процес розподілу ресурсів. Це положення використано авторами для побудови інноваційного брокера для нового покоління проміжного забезпечення NorduGrid, відомого як KnowARC [7], про який йтиметься в наступній статті.

ЛІТЕРАТУРА

1. *Yagoubi B., Slimani Y.* Task Load Balancing Strategy for Grid Computing // Journal of Computer Science . — 2007 — **3**, № 3. — P. 186–194.
2. *Heiss H.-U., Schmitz M.* Decentralized dynamic load balancing: The particles approach // Information Sciences. — 1995. — № 84. — P. 115–128.
3. *Martynov E., Zinovjev G., Svistunov S.* Academic segment of Ukrainian Grid infrastructure // System Research and Information Technologies. — 2009. — № 3. — P. 31–42.
4. *Загородний А., Зиновьев Г., Мартынов Е., Свистунов С.* Украинский академический Грид: Українсько-македонський наук. зб. — 4-ий вип. — Київ: Вид. Нац. б-ка України ім. В.І. Вернадського, 2009. — С. 140–150.
5. *Zagorodny A., Zgurovsky M., Zinovjev G., Petrenko A., Martynov E.* Integrating Ukraine into European Grid Infrastructure // Системні дослідження та інформаційні технології. — 2009. — № 2. — С. 35–49.
6. *Yagoubi B., Slimani Y.* Dynamic load balancing strategy for grid computing // Transactions on Engineering, Computing and Technology. — 2006. — № 13. — P. 260–265.
7. *Nordugrid* ARC. — http://www.nordugrid.org.
8. gLite. — http://glite.web.cern.ch/glite/.
9. UNICORE. — www.unicore.eu/.
10. *Nath R.* Efficient Load Balancing Algorithm in Grid Environment // Master Thesis, Thapar University, Patiala, May 2007. — 63 p.
11. *Петренко А.І.* Національна Grid — інфраструктура для забезпечення наукових досліджень і освіти. — http://netallted.cad.kiev.ua/downloads/Grid.pdf.
12. *Kenthapadi K., Manku G.* Decentralized algorithms using both local and random probes for P2P load balancing // Proceedings of SPAA, Las Vegas, Nevada, 2005. — P. 135–144.
13. *Buyya R., Murshed M*. GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing // Concurrency And Computation: Practice And Experience. — 2002. — **14**, № 13. — P. 1175–1220.

14. *Chapman C., Wilson P.* Condor services for the Global Grid: Interoperability between Condor and OGSA // Proceedings of the 2004 UK e-Science All Hands Meeting. — 2004. — № 2. — P. 870–877. — http://www.cs.wisc.edu/condor/doc/condor-ogsa-2004.pdf.
15. *Buyya R., Abramson D., Giddy J.* Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid // Fourth International Conference on High Performance Computing in Asia — Pacific Region, Beijing, China. — 2000. — P. 283–289.
16. *Casanova H.* et al. The AppLeS Parameter Sweep Template: User Level Middleware for the Grid. — http://www.sc2000.org/techpapr/papers/pap.pap169.pdf.
17. *Venugopal S., Nadiminti K., Gibbins H., Buyya R.* Designing a resource broker for heterogeneous grids // Software: Practice and Experience. — 2008. — **38**, Issue 8. — P. 793–825. — http://www.buyya.com/papers/ gridbus-broker-design-spe.pdf.
18. GridWay. — http://www.gridway.org/.
19. *Ming Wu, Xian-He Sun.* The GHS Grid Scheduling System:Implementation and Performance Comparison. — http://www.cs.iit.edu/~scs/psfiles/GHS-ngs-Sun.pdf.
20. *Berman F., Wolski R., Casanova H.* et al. Adaptive Computing on the Grid Using AppLeS. — http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.7606&rep=rep1&type=ps.
21. *Laure E., Hemmer F.* et al. Middleware for the next generation grid infrastructure. — http://cdsweb.cern.ch/record/865715/files/p826.pdf?version=1.
22. *Cecchi M., Capannini F.* et al. The glite workload management system // GPC '09: Proceedings of the 4-th International Conference on Advances in Grid and Pervasive Computing. — Berlin: Heidelberg, Springer-Verlag, 2009. — P. 256–268.
23. *Пономаренко В.С., Листровой С.В., Минухин С.В., Знахар С.В.* Методы и модели планирования ресурсов в GRID-системах. — Х.: ИНЖЭК, 2008. — 408 с.
24. *Foster I., Kesselman C., Tuecke S.* The anatomy of the grid: Enabling scalable virtual organizations // International Journal of High Performance Computing Applications. — 2001. — **15**, № 3. — P. 200–222.
25. *Ying Li, Feng Hong* et al. A framework for price-based resource allocation on the grid // Lecture Notes in Computer Science. — 2005. — **3320/2005**(1). — P. 341–344.
26. *Huedo E., Montero R., Llorente I.* A recursive architecture for hierarchical grid resource management // Future Generation Computer Systems. — 2009. — **25**(4). — P. 401–405.
27. *Cao J., Jarvis S.* et al. Arms: An agent-based resource management system for grid computing // Scientific Programming. — 2002. — **10**(2). — P. 135–148.
28. *Cao J., Spooner D.* et al. Grid load balancing using intelligent agents // Future Generation Computer Systems. — 2005. — **21**, № 1. — P. 135–149.
29. *Dugenie P., Cerri S., Duvert F., Jonquet C.* Agent-grid integration ontology // Lecture Notes in Computer Science. — 2006. — **4277/2006**(1). — P. 136–146.
30. *Ganzha M., Gawinecki M.* et al. Agents as resource brokers in grids forming agent teams // Future Generation Computer Systems. — 2008. — **4818/2008**(1). — P. 489–491.
31. *Buncic P.* et al. The AliEn system, status and perspectives // Computing in High Energy and Nuclear Physics, 24–28 March, La Jolla, California, Proceedings of CHEP. — 2003. — http://www.slac.stanford.edu/econf/C0303241/proc/papers/MOAT004.PDF.
32. *Betev L.* et al. The ALICE physics data challenge and the ALICE distributed analysis prototype. — http://indico.cern.ch/getFile.py/access?contribId=506&sessionId=9&resId=2&materialId=paper&confId=0.
33. *Maui* cluster. — http://www.clusterresources.com/products/maui-cluster-scheduler. php.
34. *Аветисян А.И., Гайсарян С.С., Грушин Д.А., Кузюрин Н.Н., Шокуров А.В.* Эвристики распределения задач для брокера ресурсов Grid // Тр. Ин-та системного программирования РАН. — 2004. — Т. 5. — С. 269–280.

35. *Веб-сайт* СпбГУ, Факультет Информационных технологий и программирования, кафедра Компьютерных технологий. — http://rain.ifmo.ru/cat/view.php/theory/algorithm-analysis/np-completeness-2004.
36. *Leinberger W., Karypis G., Kumar V., Biswas R.* Load balancing across near-homogeneous multi-resource servers // 9-th Heterogeneous Computing Workshop. — 2000. — P. 60–71.
37. *Javier Bustos-Jimenez*. Robin hood: An active objects load balancing mechanism for intranet. — http://www.dcc.uchile.cl/~jbustos/Pub/rh.pdf.
38. *Abdur Razzaque1 Md., Hong C.S.* Dynamic Load Balancing in Distributed System: An Efficient Approach. — http://networking.khu.ac.kr/publications/data/ Dynamic%20Load%20 Balancing%20in%20 Distributed%20System%20An% 20 Efficient%20Approach.pdf.
39. *Ramos J., Sang J.* Simulation of large scale networks III: an improved computational algorithm for round-robin service // Winter Simulation Conference. — New Orleans, Louisiana, USA. — 2003. — P. 721–728.
40. *Топорков В.В.* Модели распределенных вычислений. — М.: ФИЗМАТЛИТ, 2004. — 320 с.
41. *Leland R., Hendrickson B.* An empirical study of static load balancing algorithms // Proceedings of the Scalable High-Performance Computing Conference. — 1994. — P. 682–685.
42. *Subrata R., Zomaya A.Y., Landfeldt B.* Artificial life techniques for load balancing in computational Grids // Journal of Computer and System Sciences. — 2007. — **73**, № 8. — P. 1176–1190.
43. *Zaki M., Li W., Parthasarathy S.* Customized Dynamic Load Balancing for a Network of Workstations // Proc. of 5-th IEEE International Symposium on High Performance Distributed Computing, 6–9 August, Syracuse, NY, USA, 1996. — P. 282–291.
44. *Taibi T., Abid A., Azahan E.F.E.* A Comparison of Dynamic Load Balancing Algorithms // Jordan Journal of Applied Sciences. — 2007. — **9**, № 2. — P. 125–133.
45. *Derek L. Eager.* A Comparison of Receiver-Initiated and Sender-Initiated Adaptive Load Sharing. — http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.77.4986&rep=rep1 &type=pdf.
46. *Livny M., Melman M.* Load Balancing in Homogeneous Broadcast Distributed Systems // ACM Computer Network Performance Symposium, April. — 1982. — P. 47–55.
47. *Malik S.* Dynamic Load Balancing in a Network of Workstations. // Research Report. — 2000. — www.cs.toronto.edu/~smalik/downloads/paper_515.pdf.
48. *Caminero A., Rana O., Caminero B., Carrión C.* Autonomic Network-Aware Metascheduling for Grids: A Comprehensive Evaluation // Advances in Grid Computing, Febrary. — 2011. — P. 49–72. — http://www.intechopen.com/source/pdfs/13943/InTech-Autonomic_network_aware_metascheduling_for_grids_a_comprehensive_ evaluation.pdf.
49. *Lo V.M.* Heuristic Algorithms for Task Assignments in Distributed Systems // IEEE Transactions on Computers. — 1988. — **37**, № 11. — P. 1384–1397.
50. *Bokhari S.* Partitioning Problems in Parallel, Pipelined, and Distributed Computing // IEEE Transactions Computers. — 1988. — **37**, № 1. — P. 48–57.
51. *Cavanaugh C., Radhika A.* Dynamic Resource Management Algorithm for a Distributed Real-time System // Proceedings of the 19-th IEEE International Parallel and Distributed Processing Symposium, 4–8 April. — Denver, Cjljrado, USA. — 2005. — P. 131.
52. *Zeng Z., Veeravalli B.* Divisible load scheduling on arbitrary distributed networks via virtual routing approach // Proceedings of 10-th International Conference on Parallel and Distributed Systems, 7–9 July. — Newport Beach, CA, USA. — 2004. — P. 161.

Надійшла 05.03.2011