УДК 004.7

В. В. Литвин, А. С. Мельник[1]

[1]НУЛП, м. Львів, Україна, vasyl17.lytvyn@gmail.com

МЕТОД КООРДИНАЦІЇ ПЛАНУВАННЯ ГРІД-ОБЧИСЛЕНЬ

Розроблено метод планування в грід-системах з використанням координуючого органу. Розроблено метод планування, керований обмеженням часу та метод найскладнішої роботи, що дало змогу розробити алгоритм планування для зменшення загального часу виконання робіт в грід-системі. Здійснено апробацію розробленого методу шляхом побудови прикладної системи імітаційного моделювання грід-систем.

ГРІД-СИСТЕМА, КООРДИНАЦІЯ, ПЛАНУВАННЯ, ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ

Вступ

За останні роки стрімко зросли об'єми даних та об'єми обчислень. Доцільним є використання грід-систем як форми розподілених обчислень, в якій високопродуктивна обчислювальна система є сукупністю обчислювальних систем (обчислювальних вузлів та кластерів), що працюють разом для розв'язання однієї ресурсомісткої задачі [1, 2].

Грід – розподілена обчислювальна система, яка забезпечує гнучкий, безпечний, скоординований розподіл ресурсів. Грід може об'єднувати обчислювальні машини, розподілені як адміністративно, так і географічно. Паралельні та розподілені обчислення застосовуються для розв'язку фізичних, математичних, біологічних та будь-яких інших задач, що вимагають значних обчислювальних потужностей. Актуальність технології грід лише зростає з часом, що підтверджує динаміка появи та розвитку численних грід-проектів по всьому світу. В Україні цілий ряд наукових організацій застосовують паралельні обчислення та створили власні комп'ютерні кластери.

Для розподілу робіт по обчислювальних ресурсах грід виконують планування. Планування – визначення порядку виконання робіт та вибір ресурсів, на яких ці роботи будуть виконуватися. Задача планування є NP-повною і однією з основних в контексті грід-систем. Існуючі методи планування спрямовані на знаходження плану з мінімальним часом виконання окремих груп робіт. Ці методи не розглядають систему в цілому. Тому існують такі проблеми: мінімальний час виконання кожної роботи окремо не завжди дає мінімальний час виконання цих робіт загалом. Тому необхідний додатковий агент, який би координував план виконання робіт у вузлах грід-системи [3, 4].

1. Формальна постановка задачі планування

Кожен учасник грід-системи може завантажувати набори робіт, які система має виконати. Назвемо метароботою роботу, незалежну від інших метаробіт. Метаробота складається з взаємозалежних робіт меншого розміру. Ці роботи є неподільними і кожна неподільна робота повинна виконуватися на одному обчислювальному вузлі. Роботи залежні між собою – робота може вимагати результатів виконання іншої роботи або кількох інших робіт.

Для розподілу робіт по обчислювальних ресурсах виконують планування. Планування – визначення порядку виконання робіт та вибір ресурсів, на яких ці роботи будуть виконуватися.

Метароботу (загальну роботу), яку потрібно виконати у грід-системі, представимо у вигляду направленого ациклічного графа робіт $G(V,E)$, де V – множина вершин $v_i \in V$, які представляють окремі неподільні роботи – набори інструкцій, які мають виконуватися на одній машині; E – множина комунікаційних ребер, визначає порядок слідування робіт; напрямлене ребро $e_{ij} = (v_i, v_j) \in E$ означає, що роботу v_j можна виконувати тільки після завершення та отримання результатів роботи v_i; роботу v_i називають батьківською, v_j – дочірньою. Бінарне відношення $v_i \prec v_j$ означає, що роботу v_i потрібно виконати перед роботою v_j (при цьому v_i не обов'язково є безпосереднім батьком v_j). Вага $w(v_i)$ – специфікація роботи v_i (кількість обчислень), а вага $w(e_{ij})$ – специфікація комунікації ребра e_{ij} (кількість даних).

Функція $P(v)$ визначає множину безпосередніх батьківських робіт роботи v; функція $C(v)$ – множину безпосередніх дочірніх робіт. Роботу $v_{entry} = v \mid P(v) = \varnothing$ називають вхідною роботою (початковою), а $v_{exit} = v \mid C(v) = \varnothing$ – вихідною роботою (кінцевою). Якщо є більше однієї вхідної/вихідної роботи, то потрібно з'єднати їх єдиною вхідною/вихідною псевдо-роботою (роботою з нульовою кількістю обчислень та нульовими ребрами) [5].

Топологію гетерогенної грід-системи, представимо у вигляді ненаправленого графа $S(P,L)$, де вершина $p_i \in P$ представляє i-ту обчислювальну машину, а ребро $l_{ij} = (p_i, p_j) \in L$ – двонапрямлене комунікаційне з'єднання обчислювальних машин p_i та p_j. Вага $w(p_i)$ визначає специфікацію обчислювальної машини p_i (відносну швидкодію), а вага $w(l_{ij})$ – специфікацію комунікаційного з'єднання l_{ij} (пропускну здатність). Будь-яка обчислювальна машина системи може одночасно виконувати обчислення та передавати дані.

Задачею планувальника є розподіл роботи, представленої графом робіт G по ресурсах системи S

з врахуванням порядку слідування робіт. Розв'язком є план Ψ:

$$\Psi = G \to S = (\Phi, \mathrm{X}), \quad (1)$$

де Φ – розподіл робіт, множина значень ϕ_i: i-та робота має виконуватися на ϕ_i-й машині:

$$\Phi = \left\{ \varphi_i : \varphi_i \in \left[1, |P|\right] \middle| i = \overline{1, |V|} \right\}, \quad (2)$$

X – послідовність виконання робіт, множина унікальних значень χ_i: спочатку виконується робота χ_1, тоді χ_2 і так до останньої роботи:

$$\mathrm{X} = \left\{ \chi_i : \left(\chi_i < \chi_j \forall v_{\chi_i} \prec v_{\chi_j} \right) \wedge \left(\chi_i \neq \chi_j \forall i \neq j \right) \middle| i, j = \overline{1, |V|} \right\}. \quad (3)$$

Нехай функція $t(x, y)$ визначає: а) тривалість комунікації, якщо параметри – комунікаційне ребро e_{ij} та комунікаційне з'єднання l_{ij}; або б) тривалість виконання роботи, якщо параметри – робота v_i та машина p_i; при тому, якщо роботу v_i неможливо виконати на машині p_i, то результатом буде ∞. Аналогічно, функція $c(x, y)$ визначає вартість комунікації або вартість виконання роботи.

Визначимо час початку виконання роботи v_i:

$$t_{start}(v_i) = \max\left[t_{DR}(v_i), t_{PR}(v_i) \right], \quad (4)$$

де $t_{DR}(v_i)$ – час готовності даних роботи v_i, тобто час завершення передачі результатів виконання всіх батьківських робіт $v_j \in P(v_i)$ на машину p_{ϕ_i}; $t_{PR}(v_i)$ – час готовності машини p_{ϕ_i} до виконання роботи v_i, тобто час завершення виконання всіх попередніх робіт на цій машині.

Визначимо час завершення виконання роботи v_i – сума часу початку виконання та тривалості виконання роботи:

$$t_{finish}(v_i) = t_{start}(v_i) + t\left(v_i, p_{\varphi_i}\right). \quad (5)$$

Часом завершення виконання метароботи, представленої графом задач G, згідно з планом Ψ є час завершення виконання всіх робіт:

$$t(\Psi) = t_{finish}(v_{exit}). \quad (6)$$

Основними критеріями оптимізації при пошуку розв'язку (1) є мінімізація загального часу виконання метароботи в системі:

$$t(\Psi) \to \min. \quad (7)$$

Чим менша довжина плану, тим ефективніший алгоритм планування. Якщо розв'язком є ∞, то роботу, представлену графом задач G, неможливо виконати на системі S.

Деякі методи використовують також інші критерії. Наприклад, в комерційних системах потрібно враховувати вартість ресурсів. Для таких систем розглядають задачі часової або вартісної оптимізації:

1. Задача часової оптимізації – мінімізація часу виконання з врахуванням бюджетного обмеження:

$$\begin{cases} t(\Psi) \to \min, \\ c(\Psi) < B, \end{cases} \quad (8)$$

де $c(\Psi)$ – сумарна фінансова вартість ресурсів, використаних для виконання метароботи; B – максимальна вартість (бюджетне обмеження).

2. Задача вартісної оптимізації – мінімізація вартості виконання з врахуванням часового обмеження:

$$\begin{cases} c(\Psi) \to \min, \\ t(\Psi) < D, \end{cases} \quad (9)$$

де D – максимально допустимий час виконання метароботи (крайній термін).

Задачі, та також називають задачами відображення алгоритму на обчислювальну систему. Знаходження найкращого розв'язку цих задач є NP-повною задачею [4]. Тому використовують різноманітні евристичні алгоритми для пошуку наближених розв'язків.

Об'єктом наших досліджень є процеси планування в партнерських грід-системах, в яких ресурси надані безкоштовно. Ми розглядаємо задачу мінімізації загального часу виконання метароботи в системі. Існують різноманітні евристичні методи, які розв'язують цю задачу. Одними з найефективніших з них є методи плануванням списком HEFT та DLS.

В задачі розглядають мінімізацію часу виконання однієї метароботи. Однак в системі можуть паралельно виконуватися кілька метаробіт. Задача не враховує наявність інших метаробіт. Існуючий підхід виконує планування метаробіт в довільному порядку (в порядку їх надходження). Для кожної метароботи шукають план з мінімальним часом виконання. Але мінімальний час виконання кожної метароботи окремо не завжди дає мінімальний час виконання цих метаробіт загалом.

Ми пропонуємо розглянути задачу координації планування, яку подамо у вигляді:

$$\max\left[t\left(\Psi^1\right), t\left(\Psi^2\right), \cdots, t\left(\Psi^m\right) \right] \to \min, \quad (10)$$

де m – кількість метаробіт; Ψ^i – план для метароботи i.

Ще однією проблемою в партнерських грід-системах є недостатнє задоволення вимог обмеження часу виконання метаробіт (крайнього терміну). Координатор намагається мінімізувати час виконання окремої метароботи, використавши для цього найкращі ресурси. При цьому вимоги крайнього терміну інших метаробіт не враховуються, через що ці метароботи можуть бути не виконані вчасно, тобто, не вкластися в крайній термін.

Ще одна задача, яку ми розглянемо – задача координації планування з врахуванням часових обмежень:

$$\begin{cases} t\left(\Psi^1\right) < D^1, \\ t\left(\Psi^2\right) < D^2, \\ \vdots \\ t\left(\Psi^m\right) < D^m, \end{cases} \quad (11)$$

де D^i – максимально допустимий час виконання метароботи i (крайній термін) [6].

2. Координація планування

Для розв'язку задач (10) та (11) ми пропонуємо ввести в грід-систему координатора – планувальник, який визначає, в якому порядку система повинна виконувати планування метаробіт. Назвемо цей процес координацією планування.

На рис. 1 зображено архітектуру системи планування з координатором. Метароботи надходять в систему і стають в чергу метаробіт. Координатор сортує метароботи у черзі таким чином, щоб метаробота, яку потрібно виконати найшвидше, була першою в черзі. Планувальник бере першу роботу з черги і виконує для неї планування (одним з існуючих методів, наприклад HEFT, або DLS). Метароботу видаляють з черги після того, як для всіх робіт цієї метароботи були виділені ресурси.

Отож, у нашому підході процес планування виконується в два етапи:

1. Координатор визначає пріоритети метаробіт та сортує метароботи по спаданню пріоритетів.

2. Координатор виконує планування робіт. При цьому планувальник розглядає метароботи в порядку, визначеному на етапі 1.

В існуючих підходах координатор відсутній, тому метароботи розміщені в черзі в тому порядку, в якому вони надійшли в систему. Координатор дозволяє змінити порядок метаробіт на більш оптимальний. Метод визначення пріоритетів робіт координатором залежить від того, яку задачу потрібно розв'язати в системі – (10) чи (11).

Ми пропонуємо такі методи координації:

1. Метод, керований обмеженням часу (DD – Deadline Driven).

2. Метод найпростішої роботи (SF – Simple First).

3. Метод найскладнішої роботи (CF – Complex First).

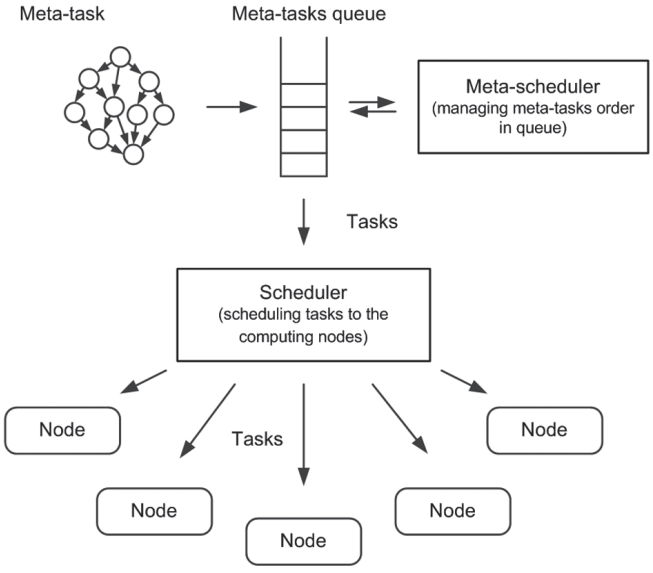


Рис. 1. Архітектура системи планування з координатором

Перший метод призначений для розв'язування задачі координації з врахуванням часових обмежень (11), а другий та третій – задачі планування (10).

3. Метод, керований обмеженням часу

В існуючих системах планування в грід планувальник опрацьовує метароботи в порядку їх надходження в систему. Основною задачею координатора є мінімізація часу виконання метароботи. В партнерських грід ресурси надані безкоштовно, тому планувальник використовує найкращі ресурси для поточної метароботи. При цьому планувальник не зважає на забезпечення якості обслуговування інших метаробіт. Основною вимогою користувачів щодо виконання метаробіт може бути обмеження часу виконання – крайній термін. Кожен користувач при додаванні своєї метароботи в грід задає також крайній термін виконання цієї роботи. Система планування грід повинна врахувати це обмеження і, якщо це можливо, виконати метароботу у визначений користувачем термін. Для розв'язування цієї задачі ми пропонуємо метод координації, керованим обмеженням часу (DD – Deadline Driven).

У методі, керованим обмеженням часу, ми пропонуємо:

1) Сортувати метароботи в черзі за зростанням крайнього терміну. Більший пріоритет має робота, у якої крайній термін менший.

2) Виконувати статичне планування робіт – планування з попередньою резервацією ресурсів відразу для цілої метароботи.

Критерієм координатора, за яким він сортує метароботи є крайній термін:

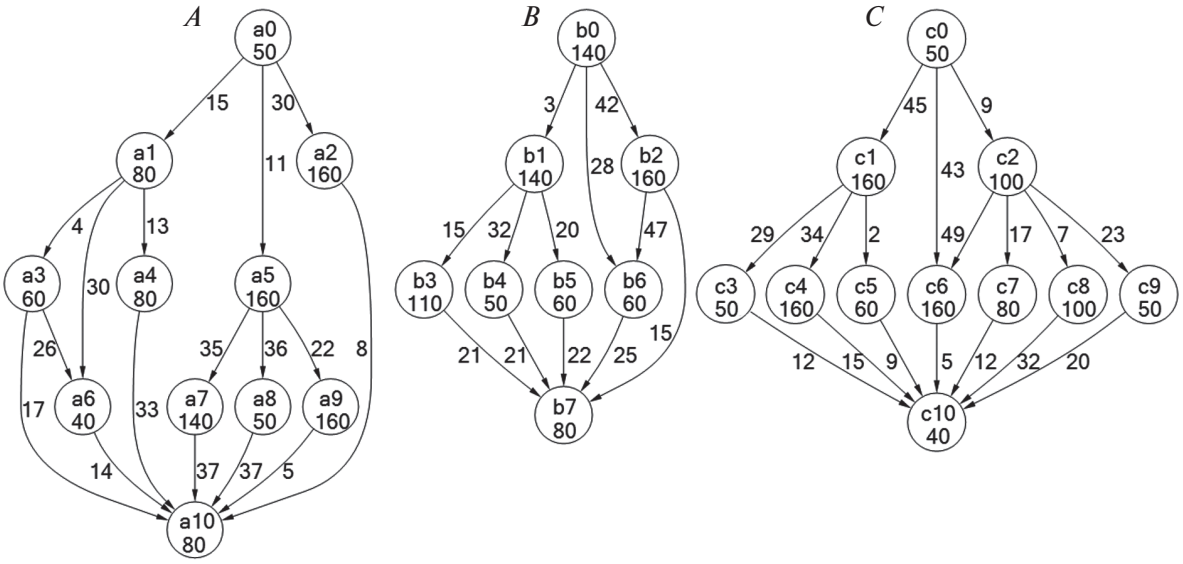
$$k_{DD}^{i} = D^{i}. \qquad (12)$$

Найпершою буде опрацьована метаробота, яку потрібно виконати найшвидше. Статичний планувальник розподілить всі роботи метароботи на ресурси з попередньою резервацією. Це дозволить гарантувати, що ресурси не будуть використані для інших робіт, і термін виконання роботи, визначений на етапі планування метароботи, не буде змінений. На цьому кроці ми можемо також сказати користувачу, коли його метаробота буде виконана, чи буде дотриманий крайній термін і яким буде понаднормовий час.

Приклад 1. Нехай в черзі є три метароботи із заданими обмеженнями часу виконання (рис. 2). Роботи надійшли в систему у послідовності $A; B; C$.

Нехай у системі є 5 обчислювальних вузлів. Специфікація вузлів:

$$w(p_0) = 4;$$
$$w(p_1) = 3;$$
$$w(p_2) = 5;$$
$$w(p_3) = 8;$$
$$w(p_4) = 5.$$



$D^A = 180$ $D^B = 70$ $D^C = 120$

Рис. 2. Метароботи з обмеженнями часу виконання

Таблиця 1

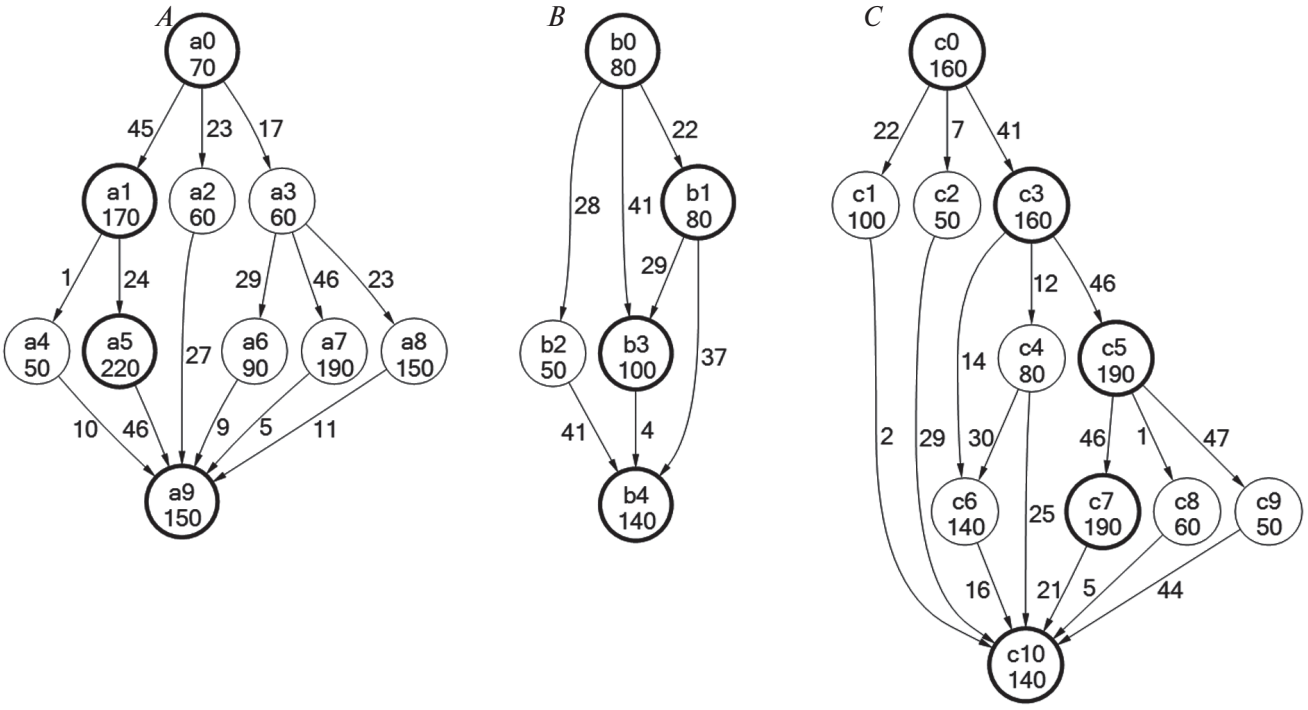
Результати планування без метапланувальника

	A	*B*	*C*
Хід планування	a0 --> p3; t_finish=6,25 a5 --> p3; t_finish=26,25 a1 --> p2; t_finish=25,25 a7 --> p3; t_finish=43,75 a2 --> p4; t_finish=44,25 a9 --> p2; t_finish=62,65 a3 --> p0; t_finish=41,05 a4 --> p3; t_finish=53,75 a8 --> p1; t_finish=50,12 a6 --> p0; t_finish=51,05 a10 --> p3; t_finish=**73,65**	b0 --> p4; t_finish=72,25 b2 --> p3; t_finish=100,65 b1 --> p4; t_finish=100,25 b3 --> p3; t_finish=117,00 b6 --> p2; t_finish=122,05 b5 --> p4; t_finish=112,25 b4 --> p0; t_finish=119,15 b7 --> p3; t_finish=**137,05**	c0 --> p4; t_finish=10,00 c1 --> p0; t_finish=91,05 c2 --> p2; t_finish=82,65 c4 --> p4; t_finish=144,25 c6 --> p1; t_finish=145,78 c8 --> p2; t_finish=102,65 c7 --> p3; t_finish=127,00 c9 --> p0; t_finish=103,55 c5 --> p2; t_finish=134,05 c3 --> p0; t_finish=131,65 c10 --> p3; t_finish=**152,25**
$t(\Psi^i)$	73,65	137,05	152,25
D^i	180	70	120
$t^i_{overtime}$	0 (0%)	67,05 (95,79%)	32,25 (26,88%)

Таблиця 2

Результати планування з DD-метапланувальником

Хід планування	b0 --> p3; t_finish=17,50 b2 --> p3; t_finish=37,50 b1 --> p2; t_finish=46,10 b3 --> p3; t_finish=62,85 b6 --> p3; t_finish=45,00 b5 --> p2; t_finish=58,10 b4 --> p4; t_finish=62,50 b7 --> p3; t_finish=**76,70**	c0 --> p2; t_finish=10,00 c1 --> p4; t_finish=51,00 c2 --> p0; t_finish=36,80 c4 --> p2; t_finish=90,10 c6 --> p0; t_finish=76,80 c8 --> p1; t_finish=71,53 c7 --> p4; t_finish=78,50 c9 --> p3; t_finish=82,95 c5 --> p3; t_finish=90,45 c3 --> p1; t_finish=88,20 c10 --> p3; t_finish=**98,10**	a0 --> p4; t_finish=10,00 a5 --> p4; t_finish=110,50 a1 --> p0; t_finish=96,80 a7 --> p3; t_finish=135,00 a2 --> p2; t_finish=122,10 a9 --> p4; t_finish=142,50 a3 --> p3; t_finish=105,60 a4 --> p3; t_finish=115,60 a8 --> p0; t_finish=130,20 a6 --> p1; t_finish=124,13 a10 --> p3; t_finish=**153,50**
$t(\Psi^i)$	76,70	98,10	153,50
D^i	70	120	180
$t^i_{overtime}$	6,70 (9,57%)	0 (0%)	0 (0%)



$w(CP^A) = 610$ $w(CP^B) = 4000$ $w(CP^C) = 840$

Рис. 3. Метароботи з визначеним критичним шляхом

Таблиця 3

Результати планування без метапланувальника

	A	*B*	*C*
Хід планування	a0 --> p3; t_finish=8,75 a1 --> p3; t_finish=30,00 a3 --> p2; t_finish=24,15 a5 --> p3; t_finish=57,50 a7 --> p2; t_finish=62,15 a8 --> p4; t_finish=58,75 a6 --> p0; t_finish=52,45 a2 --> p4; t_finish=25,35 a4 --> p1; t_finish=46,87 a9 --> p3; t_finish=**81,90**	b0 --> p0; t_finish=20,00 b1 --> p0; t_finish=72,45 b3 --> p3; t_finish=94,40 b2 --> p1; t_finish=63,53 b4 --> p3; t_finish=**111,90**	c0 --> p4; t_finish=90,75 c3 --> p4; t_finish=122,75 c5 --> p3; t_finish=155,70 c4 --> p4; t_finish=138,75 c7 --> p3; t_finish=179,45 c6 --> p4; t_finish=166,75 c1 --> p2; t_finish=115,15 c9 --> p2; t_finish=175,10 c2 --> p0; t_finish=104,65 c8 --> p0; t_finish=170,90 c10 --> p3; t_finish=**201,40**

Таблиця 4

Результати планування з CF-метапланувальником

	C	*A*	*B*
Хід планування	c0 --> p3; t_finish=20,00 c3 --> p3; t_finish=40,00 c5 --> p3; t_finish=63,75 c4 --> p2; t_finish=58,40 c7 --> p3; t_finish=87,50 c6 --> p2; t_finish=86,40 c1 --> p4; t_finish=44,40 c9 --> p4; t_finish=83,15 c2 --> p2; t_finish=31,40 c8 --> p0; t_finish=78,95 c10 --> p3; t_finish=**109,45**	a0 --> p2; t_finish=14,00 a1 --> p1; t_finish=79,67 a3 --> p0; t_finish=32,40 a5 --> p4; t_finish=128,47 a7 --> p2; t_finish=124,40 a8 --> p0; t_finish=116,45 a6 --> p0; t_finish=54,90 a2 --> p4; t_finish=56,40 a4 --> p1; t_finish=96,33 a9 --> p3; t_finish=**156,42**	b0 --> p4; t_finish=16,00 b1 --> p4; t_finish=72,40 b3 --> p3; t_finish=121,95 b2 --> p2; t_finish=41,40 b4 --> p2; t_finish=**152,40**
$t\left(\Psi^i\right)$	109,45	**156,42**	152,40

Всі вузли з'єднані між собою:

$$w\left(l_{ij}\right)=5;\quad i,j=\overline{0,4}.$$

Планування без координатора. Планувальник HEFT опрацьовує метароботи в тому порядку, в якому вони надійшли в систему: спочатку A, тоді B і C. Хід планування подано в табл. 1.

Перевищення обмеження часу виконання визначимо за формулою:

$$t^{i}_{overtime}=\begin{cases}0, & \text{якщо } t\left(\Psi^{i}\right)<D^{i};\\ t\left(\Psi^{i}\right)-D^{i}, & \text{в іншому випадку,}\end{cases} \quad (13)$$

де $t\left(\Psi^{i}\right)$ – час завершення виконання, D^{i} – Обмеження часу виконання.

Без координатора обмеження часу виконання, які користувачі задали для кожної мета роботи, ніяк не враховуються. Тому тільки першу метароботу було виконано вчасно, для другої та третої метаробіт крайній термін виконання було перевищено на 95,79% та 26,88% відповідно (табл. 1).

Планування з координатором, керованим обмеженням часу виконання. Координатор сортує метароботи згідно з критерієм (12), тобто за зростанням крайнього терміну:

$$G^{B}\left(k^{B}_{DD}=70\right),$$
$$G^{C}\left(k^{C}_{DD}=120\right),$$
$$G^{A}\left(k^{A}_{DD}=180\right).$$

Планувальник HEFT опрацьовує метароботи в порядку, визначеному координатором: спочатку B тоді C і в кінці A. Хід планування з використанням координатора, керованого обмеженням часу виконання, подано в табл. 2.

При плануванні з координатором тільки для другої метароботи було перевищено крайній термін виконання на 9,57% (табл. 4), всі інші роботи було виконано вчасно.

Слід зазначити, що загальний час виконання всіх метаробіт при використанні координатора, керованого обмеженням часу, більший, ніж без координатора. Це пояснюється тим, що для задоволення вимог обмеження часу метароботи були виконані в іншому порядку, який в результаті дав більший загальний час виконання. Не можна досягнути одночасно найкращих результатів за всіма показниками. Тому при конфігурації грід-системи та виборі політик планування потрібно визначити, що є важливішим – задоволення користувацьких вимог щодо крайнього терміну виконання робіт, чи мінімізація загального часу виконання робіт.

Ще одне зауваження, на яке слід звернути увагу при налаштуванні грід системи – навіть при використанні DD-планувальника система планування не гарантує, що всі роботи будуть виконані з дотриманням крайнього терміну. Причини цього можуть бути різні: збої живлення в деяких вузлах грід, проблеми з мережевим з'єднанням між вузлами, перевантаження грід великою кількістю робіт, нереалістична вимого щодо крайнього терміну. Для вирішення цих проблем потрібно аналізувати користувацькі вимоги щодо терміну виконання ще на етапі завантаження задачі в систему; перевірити поточний стан системи і виконати приблизний розрахунок, коли метаробота може бути виконана. Якщо користувач задав значення, яке є меншим, то відразу попереджувати його про неможливість виконання цієї метароботи в зазначений термін.

Одним із варіантів розрахунку приблизного часу виконання метароботи в системі є знаходження середньої тривалості виконання робіт, які лежать на критичному шляху:

$$t_{estimate}=\sum\nolimits_{v_i\in CP}\overline{w}\left(v_i\right), \quad (13)$$

де $\overline{w}\left(v_i\right)$ – середній час виконання роботи v_i в системі:

$$\overline{w}\left(v_i\right)=\frac{\sum\limits_{p_j\in P}t\left(v_i,p_j\right)}{|P|}. \quad (14)$$

4. Метод пріоритету найскладнішої роботи

Час виконання робіт в системі є критерієм мінімізації більшості існуючих методів планування. Однак ці методи виконують мінімізацію для однієї метароботи. Якщо в системі є декілька метаробіт, то, здебільшого, планувальник опрацьовує ці метароботи в тому порядку, в якому вони надійшли в систему, а це не завжди дає найкращий результат. Виникає задача про знаходження оптимального порядку планування метаробіт, так щоб мінімізувати загальний час виконання метаробіт.

Оскільки всі метароботи мають різну структуру графів робіт та різні складності робіт, то зміна порядку, в якому ці метароботи будуть опрацьовані, може суттєво вплинути на загальний час виконання. Експериментально було виявлено, що якщо в системі є дві метароботи, одна проста, яка складається з кількох робіт, а інша складніша, яка містить в собі більше робіт з більшою кількістю обчислень, то планування спочатку складнішої метароботи дає кращий загальний час виконання, ніж коли першою опрацювати простішу роботу. Це спостереження лягло в основу методу пріоритету найскладнішої роботи (CF – Complex First), який призначений для розв'язування задачі координації планування (10).

У методі пріоритету найскладнішої роботи ми пропонуємо:

1) Сортувати метароботи в черзі за спаданням довжини критичного шляху. Більший пріоритет має робота, у якої довший критичний шлях.

2) Виконувати планування робіт планувальником системи. Метароботи опрацьовувати в порядку, визначеному координатором.

Критерієм координатора, за яким він сортує метароботи, є обернене значення до довжина критичного шляху в графі робіт:

$$k_{CF} = \frac{1}{w(CP)}, \quad (15)$$

$$w(CP) = \sum_{v_i \in CP} w(v_i). \quad (16)$$

Якщо у графі робіт є декілька критичних шляхів, то розглядають довільний із них. Комунікаційні зв'язки між роботами при цьому не враховуються.

Найпершою буде опрацьована метаробота з найдовшим критичним шляхом. Для планування робіт може бути використаний як статичний, так і динамічний планувальник. Якщо в вимогах системи є пункт повідомляти користувача, коли його метаробота буде виконана, то потрібно використовувати статичний планувальник, який зарезервує ресурси для всіх робіт метароботи і тим самим дозволить визначити термін виконання метароботи.

Приклад. Нехай в черзі є три метароботи із заданими обмеженнями часу виконання (рис. 3). Роботи надійшли в систему у послідовності $A; B; C$. Система складається з 5 обчислювальних вузлів, всі вузли з'єднані між собою. Специфікація вузлів та з'єднання задані у прикладі 1.

Планування без координатора. Планувальник HEFT опрацьовує метароботи в тому порядку, в якому вони надійшли в систему: спочатку A, тоді B і C. Хід планування подано в табл. 3.

Загальний час виконання всіх метаробіт:

$$t_{finish}^{HEFT} = \max(81{,}90;\ \ 111{,}90;\ \ 201{,}40) = 201{,}40.$$

Планування методом пріоритету найскладнішої роботи. Координатор сортує метароботи згідно з критерієм (16), тобто за спаданням довжини критичного шляху:

$$C \quad \left(k_{CF} = \frac{1}{840}\right),$$

$$A \quad \left(k_{CF} = \frac{1}{610}\right),$$

$$B \quad \left(k_{CF} = \frac{1}{400}\right).$$

Планувальник HEFT опрацьовує метароботи в порядку, визначеному координатором: спочатку C, тоді A і в кінці B. Хід планування з використанням координатора на основі методу пріоритету найскладнішої роботи подано в табл. 4.

Загальний час виконання всіх метаробіт:

$$t_{finish}^{HEFT+CF} = \max(109{,}45;\ \ 156{,}42;\ \ 152{,}40) = 156{,}42.$$

В результаті планування за методом найскладнішої роботи ми отримали на 22% кращий результат, ніж за існуючим методом.

На рис. 4 подано графічне представлення результатів планування. При плануванні без координатора метароботи A і B бузи заплановані першими і зайняли всі основні ресурси. В результаті, метаробота C почала виконуватися тільки приблизно в час 60, тому що громіздка робота c_0 не змогла бути запланованою в жоден з проміжків простою вузлів (наприклад, на вузол p_4 перед роботою a_2, або на вузол p_1 перед роботою a_4). Пізній початок виконання метароботи C, найбільшої серед метаробіт, стало причиною пізнього завершення її виконання, а отже і загального результату.

Натомість при плануванні з координатором за методом пріоритету найскладнішої роботи першою була запланована метаробота C. Ця метаробота зайняла вузли p_2, p_3 та p_4 – найпотужніші обчислювальні вузли в системі, і єдина робота c_8 була призначена на вузол p_0. Наступною була запланована метаробота A, причому роботи a_0, a_2, a_3 та a_6 були вставлені в інтервали простою: робота a_0 на вузол p_2 перед роботою c_2; робота a_2 – на вузол p_4 між роботами c_1 та c_9; роботи a_3 та a_6 – на вузол p_0 перед c_8. Останньою була запланована метаробота B. Оскільки ця метаробота складається з відносно простих робіт, то всі роботи, крім останньої, були розміщені дуже ефективно – в проміжках простою між вже запланованими роботами. В результаті, цей метод планування дав на 22% менший загальний час виконання робіт, ніж попередній.

5. Перевірка розроблених методів

Для перевірки запропонованих методів було розроблено та використано засіб для моделювання грід-систем Sesame. Для генерації графів робіт використано засіб Task Graphs for Free (TGFF v3.0), який дозволяє генерувати графи з послідовно-паралельними групами робіт.

Параметри генерування графів робіт:

– кількість робіт в графі (task_cnt);

– середня ширина і довжина груп робіт (series_wid, series_len);

– кількість додаткових ребер в межах однієї групи робіт (series_local_xover);

– кількість додаткових ребер між окремими групами робіт (series_global_xover);

– відношення кількості комунікацій до кількості обчислень (CCR).

Ці параметри набували одне зі значень:

$$SET_{task\_cnt} = \{20; 40; 60; 80; 100\},$$

$$SET_{series\_wid\_len} = \left\{[w=2; l=3]; [w=3; l=2]; [w=5; l=2]; [w=10; l=1]; \left[w = \left\lceil \frac{v}{\sqrt{v}} \right\rceil; l = \left\lfloor \sqrt{v} \right\rfloor\right]\right\},$$

$$SET_{series\_local\_xover} = \{0; 1; 2; 5; 10\},$$

$$SET_{series\_global\_xover} = \{0; 1; 2; 5; 10\},$$

$$SET_{CCR} = \{0{,}1; 0{,}5; 0{,}1; 5; 10\}.$$

Комбінації цих параметрів дають 3125 різних типів графів.

Параметрами моделювання були кількість обчислювальних вузлів в системі та кількість метаробіт в черзі. Засіб випадковим чином вибирав задану кількість метаробіт зі згенерованої перед тим множини графів робіт. Тоді виконував моделювання для існуючого та розробленого методів. Результати моделювання подано на рис. 4.

Рис. 4. Відносна тривалість виконання робіт без координатора (HEFT) та за методом пріоритету найскладнішої роботи (HEFT+CF) для грід з 20 вузлами

Висновки

У статті запропоновано модель планування з метапланувальником в грід-системі як розвиток існуючої моделі планування з введенням метапланувальника для визначення порядку планування метаробіт, яка, на відміну від прийнятого підходу, дає змогу враховувати загальний стан системи для покращення ефективності планування.

Розроблено метод планування, керований обмеженням часу, що дало змогу розробити алгоритм планування з врахуванням вимог користувачів щодо терміну виконання робіт.

Розроблено метод найскладнішої роботи, що дало змогу розробити алгоритм планування для зменшення загального часу виконання робіт в грід-системі.

Одержала подальший розвиток задача планування – на її основі сформовані задачі метапланування та метапланування з врахуванням часових обмежень.

Розроблено програмне забезпечення для імітаційного моделювання грід-систем, яке ґрунтується на побудованих моделях, що дало можливість дослідити та оцінити ефективність розроблених методів та алгоритмів.

Список літератури: 1. *Foster I.* The Grid 2: Blueprint for a New Computing Infrastructure / I.Foster, C.Kesselman // Morgan Kaufmann, San Francisco, Calif, 2003. — 748 p. **2.** Грід — нова інформаційно-обчислювальна технологія для науки / А.Г. Загородній, Г.М. Зінов'єв, Є.С. Мартинов, С.Я. Свистунов, В.М. Шадура // Вісник НАН України. — 2005. — № 6. — С. 17-25. **3.** *Elmroth E.* Grid Resource Brokering Algorithms Enabling Advance Reservations and Resource Selection Based on Performance Predictions / E. Elmroth and J. Tordsson // Future Generation Computer Systems. The International Journal of Grid Computing: Theory, Methods and Applications. Elsevier, Vol 24, No. 6, pp. 585-593, 2008. **4.** *Baker M.* Grid and Grid technologies for wide-area distributed computing / Baker M., Buyya R., Laforenza D..- Software practice & Experience 2002, John Wiley & Sons Ltd. **5.** *Литвин В. В.* Алгоритми планування в обчислювальних грід-системах / В. В. Литвин, А. С. Мельник, О. Ю. Пшеничний, В. Я. Крайовський // Матеріали міжнародної науково-практичної конференції «Актуальні проблеми інфор¬маційних технологій, економіки та права», Чернівці, 23-24 лютого 2011.– Чернівці : ПВНЗ «Буковинський університет», 2011. – С. 61–62. **6.** *Мельник А.С.* Методи імітаційного моделювання грід-систем / А.С. Мельник // Матеріали Міжнародної конференції студентів і молодих науковців «Сучасні інформаційні технології 2011», Одеса, 2011. – С. 34-35.

Надійшла до редколегії 15.04.2013

УДК 004.7

Метод координации планирования грид-исчислений / В.В. Литвин, А.С. Мельник // Бионика интеллекта: науч.-техн. журнал. – 2013. – № 2 (81). – С. 93-100.

В статье рассматриваются методы повышения эффективности грид-исчислений благодаря использованию координатора планирования задач. Наводятся примеры применения разработаного метода.

Ил. 4. Табл. 4. Библиогр.: 6 назв.

UDK 004.7

The method of coordination grid calculations planning / V. Lytvyn, A. Meljnyk // Bionics of Intelligence: Sci. Mag. – 2013. – № 2 (81). – P. 93-100.

In article discusses methods to improve the efficiency of grid estimates due to the use of scheduling coordinator. Induced examples use elaboration of the method.

Fig. 4. Tab. 4. Ref.: 7 items.