УДК 683.322

РУСАНОВА О. В.

СПОСІБ ПЛАНУВАННЯ ОБЧИСЛЕНЬ ДЛЯ МУЛЬТИЯДЕРНИХ КЛАСТЕРІВ

У роботі запропонований універсальний спосіб планування обчислень *HMTS* для мультиядерних кластерних систем із будь-якою топологією. Даний підхід є комбінацією спискового та кластерного способів планування. Розроблена програмна модель для реалізації запропонованого та найбільш відомих способів планування обчислень для мультиядерних кластерів. Наведено результати досліджень, які підтверджують більш високу ефективність підходу *HMTS* у порівнянні з відомими способами.

The universal scheduling approach *HMTS* for multi-core cluster systems with any topologies is proposed in paper. This approach is the combination of list and cluster scheduling methods. Software for implementation of the proposed and most well-known scheduling methods in multi-core cluster systems was developed. The results of researches, which are presented in the paper, confirm the higher efficiency of the approach *HMTS* in comparison with known methods, were presented.

Ключові слова: мультиядерні кластери, планування обчислень, топологія системи, списковий, кластерний, генетичний способи планування

1. Вступ

Сьогодні прогрес суперкомп'ютерів безпосередньо пов'язаний із розвитком кластерних систем. Аналіз останньої 46-ї редакції списку найпотужніших комп'ютерів світу TOP-500 показує, що підвищення продуктивності кластерних систем здійснюється за рахунок використання багатоядерних вузлів та збільшення загальної кількості ядер у системах. Так, за останній рік середня кількість ядер у системах зросло на 12308 (або на 26,6%) з 46288 до 58596. Таке збільшення кількості ядер призводить до суттєвого ускладнення вирішення задачі ефективного використання ресурсів кластерних систем. Наприклад, ефективність найбільш потужної системи у світі *Tianhe-2*, яка складається з 3120000 ядер, дорівнює 0,62. Підвищення ефективності використання ресурсів пов'язано з удосконаленням методів планування обчислень.

У даній роботі пропонується ефективний кластерно-списковий спосіб статичного планування обчислень для підвищення реальної продуктивності мультиядерних кластерних систем.

2. Існуючі підходи до планування в кластерних системах

Існуючі способи статичного планування для кластерних систем можна розділити на чотири основні групи: генетичні, спискові, кластерні та з дублюванням [1]. Розглянемо найбільш відомі з них.

Гібридний генетичний алгоритм *HGAHS*. У роботі [2] запропоновано алгоритм планування, який є гибридом генетичного та спискового підходів. Суть цього алгоритму полягає в наступному. Спочатку для підзадач завдання визначаються пріоритети з використанням спискового алгоритму *HEFT* [3]. Потім відповідно до отриманих пріоритетів виконується первинне призначення підзадач на ресурси. Номери ресурсів (процесорів) генеруються випадковим чином. Після цього виконується аналіз отриманого призначення та визначаються підзадачі, вибрані для мутації. Цей процес повторюється до тих пір, поки задана ймовірність якості призначення не буде досягнута. Розглянутий алгоритм добре зарекомендував себе для гетерогенних кластерних систем, проте він не враховує можливу мультиядерну архітектуру системи. Крім того, використання алгоритму припускає, що обчислювальна система є повнозв'язною, що робить неможливим застосування алгоритму в системах з будь-якою іншою топологією.

Списковий алгоритм планування *SHTSA (HCPPEFT)* з використанням дублювання. У роботі [4] запропоновано алгоритм, що є комбінацією спискового підходу та дублювання обчислень, суть якого полягає в наступному. На першому етапі, відповідно до спискового підходу, визначаються пріоритети підзадач та формується черга для їх призначення на проце-

сори. Для визначення пріоритетів використовуються такі характеристики: критичні шляхи до початку графу задачі, критичні шляхи до кінця графу задачі, кількість батьківських підзадач та приналежність підзадач до критичного шляху графа задачі. На другому етапі виконується призначення підзадач на процесори з використанням процедури дублювання обчислень. Дублювання виконується для мінімізації часу виконання задачі. Даний алгоритм, так само, як і алгоритм *HGAHS,* підтримує лише повнозв'язну топологію кластерної системи і не враховує мультиядерність вузлів.

Алгоритм планування для мультиядерних систем *SAMCS*

У роботі [5] запропоновано алгоритм для мультиядерних систем, суть якого полягає в наступному. Даний алгоритм включає два етапи. На першому етапі групи підзадач призначаються по вузлах кластерної системи, а на другому – окремі підзадачі призначаються відповідно по ядрах вузлів. Для реалізації даного алгоритму спочатку формуються групи підзадач з метою мінімізації витрат на комунікацію (пересилки даних). Для цього об'єднуються в групи найбільш зв'язані підзадачі, а також використовується процедура дублювання. Основним недоліком даного алгоритму є те, що він, як і попередні, розрахований лише на повнозв'язні кластерні системи.

У даній роботі пропонується універсальний спосіб планування обчислень для мультиядерних кластерних систем із будь-якою топологією.

3. Постановка задачі планування для мультиядерних кластерів

Вхідними даними для статичного планування є моделі задачі та кластерної системи.

Модель задачі, зазвичай, може бути представлена у вигляді направленого ациклічного графу *DAG (Directed Acyclic Graph).* Вершинам такого графу відповідають підзадачі, а дугам – комунікаційні зв'язки між ними. Кожна вершина *DAG* характеризується номером та вагою, яка визначає її обчислювальну вартість. Дуги *DAG* також характеризуються вагою, що визначає комунікаційну вартість між відповідними вершинами. Комунікаційна вартість підзадач, призначених на одне й те саме ядро вузла системи, дорівнює нулю [6]. Підзадачі без вхідних дуг називаються вхідними, а підзадачі без вихідних дуг – кінцевими вершинами. Будь-яка підзадача вважається готовою до виконання, якщо всі її батьківські підзадачі виконані. Приклад графа задачі представлений на рис. 1.

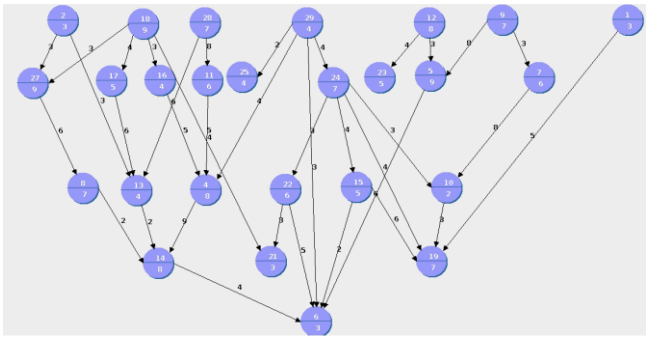

Рис. 1. Приклад графу задачі

У даній роботі кластерна система має ієрархічну структуру і складається з гомогенних (однорідних) вузлів. У свою чергу, кожен вузол системи є мультиядерним. Модель такої кластерної системи може бути представлена у вигляді ненаправленого графу системи. Вершинам такого графу відповідають вузли системи, а дугам – комунікаційні зв'язки між ними (топологія системи). Кожна вершина такого графу характеризується номером та кількістю ядер у вузлі системи. Крім того задається продуктивність усіх вузлів та кількість фізичних каналів для пересилки даних. Дуги графу системи характеризуються вагою, яка визначає їх пропускну спроможність щодо пересилки даних. Оскільки у даній роботі передбачається одна й та сама швидкість пересилки даних між сусідніми вузлами, тому і ваги дуг у графі системи мають одні й ті значення. Крім того передбачається, що всі процесори системи мають автономні контролери вводу-виводу, які дозволяють одночасно обчислювати підзадачі, а також виконувати обмін даних з іншими процесорами. Приклад графу системи представлений на рис.2. Модель цієї кластерної системи складається з 16 вузлів, кожен з яких має два ядра. У даному прикладі описана топологія гіперкуба 4-го порядку. Крім того, кожен вузол має продуктивність 1.0, пропускна спроможність усіх каналів також дорівнюють 1.0.

Результат планування може бути представлений за допомогою діаграми Ганта. Діаграма Ганта є розкладом роботи усіх обчислювальних вузлів системи і для кожного з них потактовий порядок виконання усіх завдань, призначених на цей вузол. Проте традиційна діаграма Ганта містить не усю інформацію про дані, що пере-

силаються, з одного вузла на інший. У роботі пропонується використати модифіковану діаграму Ганта [6], у якій розклад роботи кожного ядра вузлів системи включає не лише чергу виконуваних підзадач, але і порядок пересилки даних по усіх його каналах.

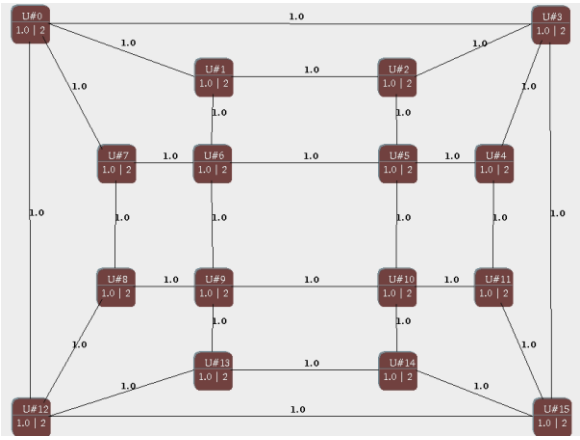

Рис. 2. Приклад графу системи

Загальна постановка задачі планування у даній роботі полягає в наступному. Нехай є кластерна система, що складається з k вузлів. Кожний j–й вузол включає p ядер. Топологія кластерної системи та її технічні характеристики задані за допомогою графа системи. Задана також задача користувача, що включає m підзадач, за допомогою графа задачі. Для кожної з m підзадач необхідно знайти таке z–е ядро кластерної системи, яке забезпечує мінімальний загальний час виконання заданої задачі (T_i). Математична модель цього завдання може бути записана таким чином. Знайти

$$\min_{i=1,R}\{T_i\}$$

$$T_i = \sum_{b=1}^{m}\sum_{j=1}^{k}\sum_{l=1}^{p} t_{bjl} * X_{bjl},$$

де $R -$ кількість варіантів вирішення задачі;
t_{bjl} –час виконання b–ї підзадачі в l–му ядрі j–го вузла кластерної системи;
$X_{bjl} = 1$, якщо b–а підзадача виконується в l–му ядрі j–го вузла кластерної системи;
$X_{bjl} = 0$, у противному випадку.

4. Запропонований спосіб планування обчислень для мультиядерних кластерів

У роботі запропоновано новий ефективний спосіб планування обчислень для гомогенних мультиядерних кластерних систем з довільною топологією на основі комбінації спискового та кластерного підходів – *HMTS (Homogeneous Multicore Task Scheduling).* Спосіб планування *HMTS,* як і всі спискові підходи має етап визначення пріоритетів підзадач графу задачі та формування черги на призначення. Також даний спосіб включає етап вибору ядра вузла кластерної системи для підзадач. Проте, крім цього, в *HMTS,* використовується ще один додатковий проміжний етап. На цьому етапі виконується групування (кластеризація) підзадач. У даному способі планування не передбачається виконання кластеризації усього графа задачі, як це виконується в алгоритмі *SAMCS.* У запропонованому підході в групи об'єднуються лише вхідні вершини (підзадачі) графа задачі, а в процесі призначення можливі випадки групування підзадач за умови наявності непризначених вхідних підзадач і присутності вільних ядер у вибраному вузлі у даний момент часу. Цей підхід дає підстави збільшити ефективність планування. Таким чином, *HMTS* складається з наступних трьох етапів, які виконуються послідовно:

1. Визначення пріоритетів підзадач.
2. Групування вхідних підзадач графа задачі.
3. Призначення підзадач по вузлах та ядрах кластерної системи та додаткове групування.

На першому етапі пріоритет кожної підзадачі в графі задачі визначається довжиною її критичного шляху до кінця графа задачі з урахуванням тільки ваг вершин. Такий спосіб обчислення пріоритетів називають *b-level* [1]. Значення підзадачі n_i визначається за формулою:

$$b(n_i) = w_{n_i} + \max_{n_j \in imed\_succ(n_i)} \{b(n_j)\},$$

де w_{n_i} – вага підздачі n_i;
imed_succ(n_i*)* – множина безпосередніх наступників (дочірніх підзадач) підзадачі n_i.
Значення *b-level* вихідних підзадач визначається їх власною вагою.

Етап групування підзадач (вершин) першого ярусу складається із наступних кроків:

1. Визначити множину вершин першого ярусу S_{1L}.

2. Відсортувати ці вершини у порядку спадання їх пріоритетів (*b-level*).

3. Визначити загальну кількість ядер у системі.

4. Якщо розмір множини вершин перевищує кількість ядер, зменшити розмір множини вершин до кількості ядер шляхом виключення вершин із найменшим пріоритетом.

5. Доки не порожня множина S_{1L}:

5.1. Створити нову групу вершин – множину G_i.

5.2. Виключити з S_{1L} першу вершину V_1 , включивши її в G_i.

5.3. Повторювати цикл *N-1* раз, де *N* – кількість ядер у вузлах:

5.3.1. Серед вершин із множини S_{1L} обрати таку вершину V_2, для якої коефіцієнт зв'язності пари $\{V_1, V_2\}$ буде максимальним. Коефіцієнт зв'язності *K* обчислюється за формулою:

$$K_{V_1,V_2} = \sum_{j=0}^{R} E_{i,j} * (L_{max} - L_i),$$

де $E_{i,j}$ - вартість (вага) *j*–ї спільної комунікації (дуги) між вершинами V_1 та V_2;

L_{max} - номер останнього ярусу графа задачі;

L_i - номер *i*–го ярусу, де починається спільна дуга.

5.3.2. Виключити з S_{1L} вершину V_2 та включити її в множину G_i.

Продемонструємо роботу етапу групування на прикладі графа задачі, зображенному на рис.1. Згідно алгоритму, визначимо множину вершин першого ярусу S_{1L} та впорядкуємо їх у порядку спадання пріоритетів. Значення *b-level* та пріоритети вершин першого ярусу зображені в табл.1.

Множина S_{1L} включає сім вершин і має вигляд: $S_{1L} = \{V_{18}, V_{28}, V_2, V_{29}, V_9, V_{12}, V_1\}$. Припустимо, що кластерна система складається з двох вузлів, кожен з яких має три ядра. Тоді загальна кількість ядер дорівнює шести, що перевищує розмір множини вершин S_{1L}. Необхідно з множини S_{1L} виключити вершину з найменшим пріоритетом, а саме V_1. Після цього множина S_{1L} має вигляд: $S_{1L} = \{V_{18}, V_{28}, V_2, V_{29}, V_9, V_{12}\}$.

Таблиця 1. Значення b-level та пріоритети вершин першого ярусу графа задачі

Вершина	*b-level*	Пріоритет
2	30	4
18	36	6
28	32	5
29	23	3
12	20	1
9	22	2
1	10	0

Виконаємо покроково операцію групування. Для цього створимо групу G_1, вершину V_{18} виключимо з множини S_{1L} і включимо в G_1. Тоді $G_1 = \{V_{18}\}$, $S_{1L} = \{V_{28}, V_2, V_{29}, V_9, V_{12}\}$. Тепер необхідно із множини S_{1L} знайти вершину, яка має максимальний коефіцієнт зв'язності з V_{18}. У табл.2 вказані коефіцієнти зв'язності вершини V_{18} з усіма вершинами з множини S_{1L}.

Таблиця 2. Коефіцієнти зв'язності вершин

	V_{28}	V_2	V_{29}	V_9	V_{12}
V_{18}	20	30	16	0	0

Із табл.2 видно, що максимальний коефіцієнт зв'язності з вершиною V_{18} має вершина V_2, тому вона має бути додана до множини G_1 і виключена з множини S_{1L}. У результаті $S_{1L} = \{V_{28}, V_{29}, V_9, V_{12}\}$, $G_1 = \{V_{18}, V_2\}$.

Для завершення формування групи G_1, необхідно повторити вищевказані кроки для пари V_{18}, V_2 та всіх вершин, що залишились у множині S_{1L}. Для визначення множини спільних дуг вершини V_x з групою вершин $\{V_y, V_z\}$ використовується співвідношення:

$$E = (E_Y \cup E_Z) \cap E_X ,$$

тобто перетин множини дуг графа з коренем у вершині V_X відбувається з об'єднанням множин дуг графів з коренями у вершинах V_y, V_z. Найбільший коефіцієнт зв'язності з парою вершин V_{18}, V_2 має вершина V_{28}. Таким чином, вершину V_{28} виключаємо з множини S_{1L} і включаємо її в G_1. У результаті $S_{1L} = \{V_{29}, V_9, V_{12}\}$ $G_1 = \{V_{18}, V_2, V_{28}\}$. Оскільки кількість вершин у множині S_{1L} співпадає із залишком ядер у кластерній системі, то група G_2 буде еквівалентною S_{1L}. Таким чином $G_1 = \{V_{18}, V_2, V_{28}\}$, $G_2 = \{V_{29}, V_9, V_{12}\}$. У результаті групування підзадач і призначення їх

на ядра одного й того самого вузла зменшуються витрати часу на взаємодію між вузлами, що дає можливість підвищити реальну продуктивність кластерної системи.

Етап призначення підзадач по вузлах та ядрах кластерної системи включає наступні кроки:

1. Для кожної групи, сформованої на другому етапі в порядку спадання пріоритету, виконати призначення на вузол системи. Вузол обрати із списку за пріоритетом. Пріоритети вузлів визначаються за їх зв'язністю у графі системи. Найбільший пріоритет мають вузли з максимальною зв'язністю.

2. Для кожної готової (доступної) підзадачі, у якої всі батьківські підзадачі вже призначені, у порядку спадання пріоритетів (*b-level*):

   2.1. Обрати вузол, який забезпечує мінімальний час старту підзадачі.

   2.2. Перевірити, чи не залишилось непризначенних підзадач з першого ярусу. Якщо такі підзадачі є, перевірити, кількість вільних ядер в даний момент часу у вибраному для призначення вузлі. Якщо в обраному вузлі більше одного вільного ядра, виконати групування обраної підзадачі з підзадачами першого ярусу за алгоритмом другого етапу.

3. Виконати призначення підзадачі або групи, в залежності від результату п.2.2.

Зазначимо, що при рівності пріоритетів, для призначення обирається підзадача з більшою зв'язністю, а при рівній зв'язності – з меншою вагою. Крім того, при виборі вузла для виконання враховуються усі вузли кластерної системи, незалежно від того зайняті вони, чи вільні в момент призначення.

Таким чином, описана процедура призначення дозволяє врахувати архітектурні топологічні особливості багатоядерної кластерної системи.

5. Порівняльний аналіз способу планування HMTS з існуючими підходами

Для підтвердження ефективності запропонованого підходу, розроблена програмна модель, за допомогою якої можна виконати порівняльний аналіз *HMTS* з широко відомими алгоритмами *HGAHS, SHTSA* та *SAMCS.*

При проведенні порівняльного аналізу використовувались наступні метрики:

- *Scheduling Length (SL)* або *makespan.* Визначається часом виконання задачі при плануванні обчислень.
- *Scheduling Length Ratio(SLR).* Головним індикатором потужності способу планування є *SL (makespan).* Оскільки при проведенні експериментів використовується велика кількість графів задач з різними властивостями, необхідно нормалізувати *SL* до меншої межі, якою є *SLR.* Значення *SLR* визначає у скільки разів *SL* більше *CPMIN* і визначається за формулою

$$SLR = \frac{makespan}{\sum_{n_i \in CP_{MIN}} \min_{p_j \in Q}\{w_{i,j}\}}$$

Знаменником є сума мінімальних обчислювальних вартостей підзадач *CPMIN*. Спосіб планування обчислень, який забезпечує найменше значення *SLR* графа, є кращим з точки зору часу виконання графа задачі.

- *Speedup (SU)* або прискорення. Значення *SU* визначається обчислюється шляхом ділення послідовного часу виконання графа задачі на час виконання графа задачі *(SL – makespan)* у паралельній системі:

$$Speedup = \frac{\min_{p_j \in Q}\{\sum_{n_i \in V} w_{i,j}\}}{makespan}$$

- *Efficiency (E)* або ефективність. Визначається співвідношенням значення прискорення до кількості ядер кластерної системию
- *Running time (RT)* або час роботи самого алгоритму планування.

Тестування алгоритмів планування виконується з використанням випадково згенерованих графів задач та заданих графів кластерних систем.

За допомогою розробленої програмної моделі у даній роботі проведені наступні дослідження. Згенеровано 2400 типів графів задач із наступними вхідними параметрами:

V = {20, 30, 40, 50, 60, 70, 80, 90, 100} – кількість підзадач у графі задачі;

a = {0.5, 1.0, 2.0} – форма графа;

CCR = {0.1, 0.25, 0.5, 0.75, 1.0} – співвідношення сумарної комунікаційної вартості до сумарної обчислювальної вартості графа задачі;

Wmin = {5, 10, 40, 75, 100} – мінімальна вага вершини графу;

Wmax = {200, 500, 1000} – максимальна вага вершини графу.

Окрім цього, для кожного з типів генерувалось 10 графів задач і таким чином для

порівняння способів планування усього згенеровано 24000 графів задач.

В якості графа системи використано топологію гіперкуба 4-го порядку (рис.2), що складається з 16 вузлів, кожен з яких має два ядра.

Результати проведення експериментів, щодо показника *SL,* наведені у табл.3

Попарно порівнюючи середні значення *SL* досліджувальних способів планування між собою, можна зазначити наступне:

- Запропонований спосіб планування *HMTS* у 95% варіантів графів задач забезпечив мінімальне значення *SL*, що є кращим результатом серед усіх досліджуваних способів планування.
- Спосіб планування *SHTSA* в 78% варіантів графів задач забезпечив максимальне значення *SL*, що є гіршим результатом серед усіх досліджуваних способів планування.
- За результатами порівняння значень *SL,* досліджувані способи планування можна розташувати таким чином: *{HMTS, SAMCS, HGAHS, SHTSA},* при цьому перший є кращим, а останній гіршим.

Середнє значення *SLR* на 24000 згенерованих графах завдань для *HMTS* = 2.09, для *SAMCS* = 2.36, для *HGAHS* = 2.48, для *SHTSA* = 2.68.

В усіх проведених тестах запропонований спосіб планування обчислень *HMTS* показав кращі результати, за винятком швидкості роботи, поступившись по цьому параметру підхорам *SAMCS* і *SHTSA*. По інших показниках досліджувані способи планування у порядку спадання ефективності можна розташувати наступним чином: *{HMTS, SAMCS, HGAHS, SHTSA}*. При цьому *SLR* запропонованого способу *HMTS* краще на 13%, ніж у *SAMCS*, на 19%, ніж у *HGAHS* і на 24%, ніж у *SHTSA*.

Таблиця 3. Результати попарного порівняння значень SL способів планування на множині випадково згенерованих графах завдань

		HMTS	***SAMCS***	***HGAHS***	***SHTSA***
HMTS	Краще Рівно Гірше	*	20657 197 3146	23776 114 110	22153 378 1469
SAMCS	Краще Рівно Гірше	3146 197 20657	*	21939 362 1699	19067 601 4332
HGAHS	Краще Рівно Гірше	110 114 23776	1699 362 21939	*	2152 340 21508
SHTSA	Краще Рівно Гірше	1469 378 22153	4332 601 19067	21508 340 2152	*

6. Висновки

У роботі запропоновано новий ефективний спосіб планування *HMTS* для мультиядерних кластерних систем із довільною топологією. Цей підхід є комбінацією спискового та кластерного способів планування. Розроблена програмна модель для виконання порівняльного аналізу запропонованого способу із найбільш відомими підходами, які можуть бути використані для планування обчислень мультиядерних кластерів. Приведені результати досліджень, що підтверджують більш високу ефективність способу *HMTS* у порівнянні з відомими підходами.

Запропонований підхід *HMTS* може бути використаний для підвищення реальної продуктивності як гомогенних мультиядерних кластерів, так і мультиядерних систем із масовим паралелізмом. Майбутні роботи у цьому напрямку пов’язані з адаптацією *HMTS* до гетерогенних кластерних систем.

Список посилань

1. Young Choon Lee, «Problem-Centric Scheduling for Heterogeneous Computing Systems», September 2007. – P.162.
2. Kamaljit Kaur. Heuristics Based Genetic Algorithm for Scheduling Static Tasks in Homogeneous Parallel System // Department of Computer Science & Engineering, Guru Nanak Dev University, Amritsar- 143001, Punjab, India, IJCSS, vol.4, pp.183-198, 2010.
3. H. Topcuoglu, S. Hariri and M. Wu, «Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing», IEEE Trans. Paralleland Distributed Systems, vol. 13, no. 3, pp. 260–274, March 2002.
4. Yanyan Dai. A Synthesized Heuristic Task Scheduling Algorithm // Institute of Information and Communication, Guilin University of Electronic Technology, Guilin 541004, China, 2014. – P. 67.
5. Xiaozhong Geng, Gaochao Xu, Xiaodong Fu, Yuan Zhang, «A Task Scheduling Algorithm For Multi-Core-Cluster Systems», Journal of Computers, vol.7,no11, pp.2797-2804, November 2012.
6. G. Loutsky, O. Rusanova, «Scheduling Problems on the Parallel and Distributed Systems - an Overwiew» – Poland, Rzeszow, tom 1, pp.101-105 (Engl), 2000.