УДК 681.51

С.В. Мінухін, О.О. Ганжа

Харківський національний економічний університет, Харків

ОПТИМІЗАЦІЯ ПЛАНУ ВИБІРКИ ЗАВДАНЬ З ЧЕРГИ КЛАСТЕРУ НА ОСНОВІ ПАРАЛЕЛЬНОГО ВИРІШЕННЯ БУЛЄВИХ НЕЛІНІЙНИХ РІВНЯНЬ З ОБМЕЖЕННЯМИ

Розглянуто метод групової обробки завдань, що надходять до черги кластеру, на основі моделі рангового підходу. Метод реалізовано на мові програмування MC#. Проведений аналіз результатів паралельної реалізації методу.

Ключові слова: *грід, кластер, диспетчер завдань, метод групової обробки завдань, ранговий підхід, FCFS, Backfill, MC#.*

Вступ

Обчислювальні системи кластерного типу набувають все більшого поширення. При розгортанні кластерних систем виникає ряд проблем. Помітне місце в цьому ряду займає проблема ефективного управління системою. Для ефективної реалізації вирішення завдань необхідно розробити систему, що забезпечуватиме можливість запуску завдань на кластері (систему доступу); систему, що забезпечуватиме планування виконання завдань на кластері (диспетчер завдань або планувальник), і систему моніторингу завантаженості вузлів кластера. У разі відсутності цих складових виникатимуть конфлікти в процесі роботи з обчислювальними потужностями. Таким чином, розроблення програмних систем та забезпечення, що вирішуватимуть поставленні завдання, є необхідною умовою успішного розвитку обчислювальних систем даного класу [1].

В даній роботі пропонується метод планування вибірки завдань з черги кластеру, за допомогою якого в диспетчері формується план розподілу завдань по вузлах. Метою розробленого методу є підвищення ефективності оброблення завдань, що надходять до кластеру.

Огляд існуючих способів вирішення задачі планування

Розглянемо типову структуру системи (рис. 1), що складається зі M типів ресурсів R_i, $i = \overline{1..M}$. До системи мають доступ N клієнтів O_j, $j = \overline{1..N}$, які формують завдання для їх виконання в системі.

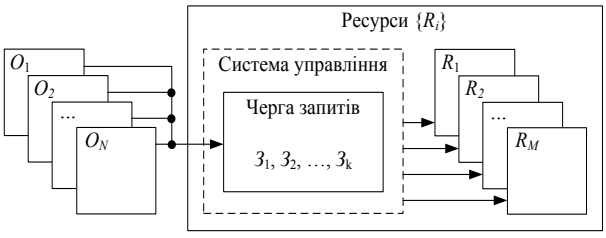

Рис. 1. Схема обробки завдань

Кожне завдання $З_k$ від будь-якого клієнта $О_j$ має свій пріоритет b_k, який залежить від рівня привілеїв клієнта. Момент часу, в який від клієнта надходить завдання – величина випадкова. При цьому на вході системи при високій інтенсивності надходження завдань утворюється черга.

За наявності обмежень на розмір черги в системі може виникнути ситуація, при якій черговий запит на виконання завдання, що поступив на вхід системи, не буде прийнятий до обслуговування, що приведе до його втрати або затримки в обслуговуванні. Тому актуальною є проблема розроблення системи з прийнятною (потрібною або директивною) швидкістю обслуговування завдань, що поступають із заданою інтенсивністю на їх виконання у кластері. визначимо додаткову вимогу до роботи системи управління кластером: вона має одночасний доступ до будь-якої кількості вільних ресурсів на даний момент часу і на одному ресурсі кластеру може виконуватися тільки одне завдання.

Існує велика кількість методів та їх модифікацій, що забезпечують розділ простору ресурсів кластеру. Найпростішим методом планування, є метод FCFS (First Come First Served) [2], згідно з яким завдання, що поступило у чергу раніше інших, має найвищий пріоритет і, отже, повинно бути виконуватися першим. Якщо ресурсів для його запуску виявляється недостатньо, очікується момент часу, коли накопичиться потрібний йому об'єм вільних ресурсів, після чого завдання буде запущено. Таким чином, за даною дисципліною обслуговування порядок завдань в черзі не може бути порушений: завдання запускаються одне за іншим згідно з їх пріоритетами. Метод гарантує запуск паралельного завдання проте неефективно використовує обчислювальні ресурси: в період їх накопичення для запуску більш пріоритетного завдання частина з них простоює. Завантаження ресурсів в цьому випадку складає приблизно 50 – 80%. З цієї причини метод FCFS не завжди є ефективним, а застосовуються його модифікації, здатні краще завантажувати ресурси.

Кращій спосіб рішення пропонує метод зворотного заповнення (Backfill), спочатку розроблений для масивно-паралельних систем типу IBM Sp2 [3], який достатньо широко застосовується на практиці, у тому числі і в кластерних системах. На відміну від FCFS він вимагає від користувачів оцінку часу виконання їх завдань, що дозволяє виділяти ресурси завданням не безпосередньо у момент звільнення, а завчасно (попереджуване планування). Для цього будується план розподілу ресурсів – розклад запусків завдань. При побудові розкладу ресурси виділяються завданням у порядку їх пріоритетів, причому завдання може отримати деякі ресурси тільки за умови, що вони вже не відведені більш пріоритетним завданням. За допомогою механізму попереднього резервування метод гарантує отримання ресурсів пріоритетними завданнями, та, разом з тим, допускає порушення порядку черги, що призводить до збільшення коефіцієнта загального завантаження ресурсів.

Важливо відзначити, що в методі Backfill переслідуються дві цілі, які конфліктують одна з одною: перша ціль – підвищення ефективності використання ресурсів шляхом заповнення «дірок» в розкладі дрібними завданнями, друга ціль – запобігання зависанню великих завдань з прогнозуванням часу або порядку їх запуску. Варіюючи кількістю резервувань, що створюються при одноразовому плануванні черги завдань, або комбінуючи метод Backfill з іншими відомими методами впорядковування завдань в черзі і оптимізаційними методами, можна отримати його різні варіанти, що збільшують вагу однієї з цих цілей. Недолік цього методу полягає в тому, що може бути затриманий запуск завдань, для яких резервування не робиться і в цілому ресурси використовуються неефективно.

У розглянутих методів планування існують різні модифікації, направлені на вирішення існуючих проблем щодо підвищення ефективності процесу планування. Проте й вони мають недоліки. Отже з метою вирішення цих проблем в даній роботі пропонується метод групової вибірки на основі рангового підходу [4, 5].

Метод групової вибірки завдань – це такий спосіб планування, при якому з черги обслуговується декілька завдань одночасно. Вибираються завдання, які потребують оброблення на різних ресурсах, та такі, для яких сума їх пріоритетів була максимальною і щоб вони задовольняли умовам обмежень, які накладаються на чергу. У разі наявності рівнозначних завдань вибираються ті, що надійшли до черги раніше. Для опису черги в розробленому методі було запропоновано використати модель на основі нелінійного булєвого рівняння, а для опису обмежень – систему нелінійних булєвих нерівностей [4, 5].

Математична постановка задачі

Зробимо такі припущення: $\vec{X}$ – множина всіх варіантів вибірки завдань з черги, $\vec{X}$ – один з варіантів вибірки завдань

$$\vec{X} = x_1, x_2, ..., x_p, ..., x_N \ , \quad p = \overline{1..N},$$

де N – кількість завдань в черзі; x_p – булєва змінна, яка дорівнює 1 або 0.

Таким чином, постановка завдання полягає в наступному: вибрати з черги як можна більшу кількість завдань, які звертаються до різних ресурсів кластера, та таких, щоб сума їх пріоритетів була б максимальною.

Позначимо змінною β_p пріоритет завдання $З_p$, тоді критерій оптимізації має такий вигляд:

$$F = \sum_{p=1}^{N} \beta_{pi} x_p x_i \to \max, \qquad (1)$$

або в окремому випадку, коли рівняння лінійне:

$$F = \sum_{p=1}^{N} \beta_p x_p \to \max. \qquad (2)$$

Зробимо такі позначення: A_{kg} – булєва змінна, яка дорівнює 1, якщо завдання $З_k$ використовує ресурс R_g, і 0 – якщо ні; B_g – кількість ресурсів типу R_g. Тоді, виходячи з умови, що у будь-який момент часу будь-який ресурс може бути використаний для виконання тільки одного завдання, отримуємо M обмежень:

$$\sum_{k=1}^{p} A_{kig} x_k x_i \le B_g, \quad g = \overline{1..M}, \qquad (3)$$

або в окремому випадку, якщо рівняння лінійне:

$$\sum_{k=1}^{p} A_{kg} x_k \le B_g, \quad g = \overline{1..M}. \qquad (4)$$

Отже, необхідно знайти таку вибірку $\vec{X}$ з множини завдань $\vec{X}$, для якої виконується (1) або (2), при виконанні системи обмежень (3) або (4). Це задача нелінійного (або в окремих випадках лінійного) програмування з булєвими змінними.

Метод вирішення задачі

Вирішення задачі відбувається поетапно і засновано на ідеях рангового підходу [5]. Кожен етап складається з визначення оптимальної вибірки $\vec{X}$, її обслуговування і зміни функціонала (2) або (3) та обмежень (4) або (5) з урахуванням змін в черзі після обслуговування вибірки. Деяка підмножина V множини можливих значень $\vec{X} = \vec{X}$, всі вектори якої задовольняють (4) або (5), утворює множину можливих рішень. Множину всіх можливих рішень можна представити у вигляді графа G_Δ який наведено на рис. 2.

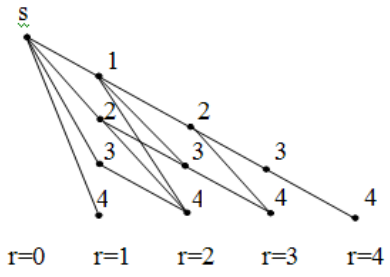

Рис. 2. Граф G_Δ

Усі можливі рішення можна розбити на групи векторів, які представляють собою шляхи графу G_Δ. Якщо позначити підмножину складників цих груп m^r, $r = \overline{1..n}$, то множину всіх можливих рішень можна записати як об'єднання підмножин m^r:

$$X = \bigcup_{j=1}^{n} m^r. \qquad (5)$$

Граф G_Δ будується таким чином: на нульовому ярусі (r = 0) розташовується вершина так звана фіктивна вершина s. На першому ярусі розміщуються всі вершини графа G_Δ, що мають зв'язок з вершиною s та з'єднують їх з s (при цьому утворюється підмножина шляхів рангу r = 1). На другому ярусі розміщуються всі вершини, що мають зв'язок з вершинами першого ярусу, без вершини з номером 1 і з'єднуються з вершинами першого ярусу (при цьому утворюються всі шляхи рангу r = 2) і т.д. до тих пір, поки на останньому ярусі не залишиться одна вершина n. Геометрично вершина p графу G_Δ рангу r – це множина векторів $\vec{X} = x_1, x_2, ..., x_p, ..., x_N$, у яких $x_p = 1$, а на позиціях від 1 до p знаходиться r одиниць. На рис. 2 наведено граф G_Δ для n = 4.

У графі G_Δ кожному ребру, що входить у вершину p, $p = \overline{1,n}$ відповідають дві ваги: вага β_{pi}, при складовій $x_p x_i$ у функціоналі (1), та вага A_{kig}, при складовій $x_k x_i$ у обмеженні (3), $k = \overline{1,n}$. Тоді шлях μ_{sj}^r у графі G_Δ з вершини s у вершину p характеризується двома довжинами: d_c μ_{sj}^r – довжиною за вагою функціонала та d_a μ_{sj}^r – довжиною за вагою обмежень. Множина шляхів $m_s^r(j)$ у графі G_Δ до вершин p, розташованим на ярусах $r = \overline{1,n}$ від вершини s, можна представити у вигляді:

$$m_s^r(j) = m_{sj}^{r=1} \cup m_{sj}^{r=2} \cup ... \cup m_{sj}^{r=n}, \quad j = \overline{1,n}, \qquad (6)$$

де m_{sj}^r – множина шляхів у графі G_Δ від вершини s до вершин p, розташованих на r-х ярусах графу G_Δ (ранг шляху $\mu_{sj}^r \in m_{sj}^r$ визначається кількістю ребер, що складають цей шлях). Слід мати на увазі, що множина шляхів $m_{sj}^{r=k}$ у графі G_Δ відповідає множині векторів $\vec{X}_1, \vec{X}_2, ..., \vec{X}_V$, що містять p одиниць. Отже, $\left|m_{sj}^r\right| = c_n^{r=k}$, тобто кожному шляхові в множини $m_{sj}^{r=k}$ відповідає деякий вектор: $x_1, x_2, ..., x_n$. З (6) виходить, що:

$$m_s^r(j) = c_n^{r=1} \cup c_n^{r=2} \cup ... \cup c_n^{r=n} = 2^n - 1. \qquad (7)$$

На основі наведеної математичної моделі на основі рангового підходу для побудови алгоритмів вирішення завдань нелінійного програмування з булєвими змінними використаємо принцип оптимізації за напрямком в дискретному просторі станів, що задається графом G_Δ.

Під оптимізацією за напрямком у графі G_Δ до деякої вершини p розуміють формування множин $m_{sp}^{r=r+1}$ наступного рангу, які виходять за рахунок виділення в m_{sj}^{r} тих шляхів, під'єднання (приєднання) до яких ребер (k, p) дозволить в множині $m_{sp}^{r=r+1}$ отримати шляхи, що задовольнятимуть правилам відсікань L_w на основі наступного рекурентного співвідношення:

$$\forall\ \mu_{sj}^{r} \in m_{sj}^{r}\ \left\lfloor \mu_{sp}^{r=r+1} = L_w\ \mu_{sj}^{r} \cup (j,p)\ \right\rfloor, \quad p = \overline{r+1,n}\ ,\ \ j = \overline{r,n}\ , \tag{8}$$

де $\mu_{sj}^{r} \cup (j,p)$ – шлях з вершини s графу G_Δ у вершину p, що проходить через проміжну вершину k, а також задовольняє правилам L_w, тобто отриманий за рахунок під'єднання до шляху μ_{sj}^{r} ребра (k, p), якщо таке з'єднання не суперечить правилам L_w. До правил відсікань відноситьсятакі умови:

1) довжина за вагою обмежень $d_a\ \mu_{sj}^{r}$ не відповідає умові обмеження (нерівності);

2) шлях μ_{sj}^{r} не є унікальним.

Узагальнена процедура вирішення задачі оптимізації вибірки завдань з черги

Процедура складається з наступних кроків:

1. З вершини s будується множина шляхів $m_{sj}^{r=1}$, $j = \overline{r,n}$. Виділяються шляхи $\mu_{sj}^{*r=1}$, що визначають локальні екстремуми на кожному ярусі графу G_Δ, тобто ті шляхи, які не суперечать правилам відсікань L_w.

2. Формується множина шляхів $m_{sp}^{r=r+1}$, $j = \overline{r+1,n}$ наступного рангу, на базі множини шляхів m_{sj}^{r} попереднього рангу відповідно до рекурентного співвідношення (8). В отриманих множинах $m_{sp}^{r=r+1}$ проводиться відсікання шляхів згідно з вибраним правилам відсікань L_w і виділяються шляхи $\mu_{sp}^{*r=r+1}$, що визначають локальні екстремуми на кожному ярусі графу G_Δ.

3. Перевіряємо чи є можливість сформувати множину шляхів $m_{sp}^{r=r+1}$ на наступному ранзі. Якщо немає можливості, то переходимо до кроку 4, а якщо є можливість, то перевіряємо r = (n – 1). У разі виконання рівняння переходимо до кроку 4, інакше збільшуємо r на 1 та виконуємо крок 2.

4. Виділяємо серед безлічі локальних екстремумів глобальний, тобто такий шлях, який має максимальну довжину за вагою функціоналу $d_c\ \mu_{sj}^{r}$ та мінімальну довжину за вагою обмежень $d_a\ \mu_{sj}^{r}$ і процедура закінчує роботу. Саме такий шлях є результатом роботи процедури.

Результати

Розроблений метод програмно реалізований на мові програмування MC# (Multiprocessor C#). Для проведення моделювання в програмі додані модулі генерації початкових даних та аналізу отриманих результатів.

В розробленій програмі скорочення часу роботи досягається відправкою деяких методів для виконання на процесори (робочі станції), які входять до кластеру. Такі методи називаються movable-методами.

Програма складається з трьох модулів: «Формування моделі черги», «Формування плану вибірки завдань», «Аналіз показників рішення задачі». Перераховані модулі формують модель функціонування диспетчера завдань кластеру. В модулі «Формування моделі черги» будуються нелінійні булєві рівняння та обмеження за заданими параметрами, які характеризують інтенсивність надходження завдань до черги.

Модуль «Формування плану вибірки завдань» є програмною реалізацією розглянутого методу. Далі наведено два фрагменти коду даного модуля (лістинги 1, 2):

Лістинг 1

```
public class MainClass {
private List<Equation> equationList;
…
private void MainMethod () {
MainClass mc = new MainClass();
PlanBuild pb = new PlanBuild();
foreach (Equation equation in equationList){
pb.Build( equation, mc.channel );
…
}
…
}
Equation Get() & Channel channel( Equation equation) { return equation; }
…
}
```

MainClass – головний клас програми. В типізованому списку equationList зберігається набір даних, сформованих при виконанні модулю «Формування моделі черги», які подаються на вхід методу Build класу PlanBuild. Метод Build є movable-методом. Особливістю movable-методів є те, що вони не повертають ніяких значень, тому для отримання результату використовують канали. Канал це спеціальний метод, що об'являється в зв'язці зі звичайним методом, який призначено для отримання значення із каналу.

Для отримання результату розрахунків використовуємо канал channel.

Лістинг 2 .

```
public class PlanBuild {
…
public movable Build ( Equation equation, Channel channel ){
…
}
…
}
```

Метод Build класу PlanBuild отримує в якості параметрів рівняння з обмеженнями та канал, який поверне результат розрахунків до головного класу програми. Отже, завдяки movable-методам та каналам було реалізовано паралельність виконання кроку 2 із узагальненої процедури вирішення задачі оптимізації вибірки завдань з черги.

В якості результату, необхідного для проведення аналізу ефективності методу, виступають показники часу вирішення задачі.

Для прикладу програмно була вирішена задача, початкові дані якої були сформовані за наступними параметрами:

1) діапазон змінних = 20 – діапазон змінних від x_1 до x_{20}, які можуть бути використані при формуванні рівнянь та обмежень;

2) кількість складників = 50 – кількість складників ($5*x_1x_3+x_2x_3+\ldots$) в рівнянні та обмеженнях;

3) ступінь нелінійності = 20 – максимально можлива кількість змінних в складнику;

4) діапазон коефіцієнтів при складниках: від 1 до 100 – діапазон цілих чисел від 1 до 100, які встановлюються при складниках рівняння та обмежень та є сумарною характеристикою завдань, що формують складник. В даному випадку, в рівнянні – це показник сумарного пріоритету завдань, а в обмеженнях – сумарна кількість процесорних елементів, яка необхідна для вирішення цієї групи завдань;

5) кількість обмежень 3 – кількість нелінійних булєвих нерівностей, що складають систему обмежень;

6) кількість реалізацій рівнянь 50 – кількість сформованих рівнянь без зміни параметрів;

7) кількість крапок 20 – кількість груп сформованих рівнянь за однаковими параметрами;

8) крок збільшення 10 – число на яке збільшиться вибраний параметр (інкремент). В даному прикладі для збільшення параметр обрано кількість складників.

Для прикладу наведемо фрагмент одного з множини рівнянь, яке сформовано за наведеними параметрами:

$$44*X_2X_3X_5X_6 + 22*X_{13}X_{18} + \ldots +$$
$$+13*X_{13}X_{17}X_{16}X_2X_3X_4X_{19}X_{11}X_9X_8 \rightarrow \max.$$

Порівняння результатів, отриманих при вирішенні задачі на чотирьох вузлах кластеру та у послідовному режимі, наведено в табл. 1.

Таблиця 1

Порівняння результатів

№ п/п	Кількість складників	Середній час розрахунку в послідовному режимі, с	Середній час розрахунку в паралельному режимі, с
1	50	0,05	0,04
2	60	0,07	0,05
3	70	0,1	0,06
4	80	0,13	0,09
5	90	0,16	0,1
6	100	0,2	0,12
7	110	0,23	0,15
8	120	0,27	0,17
9	130	0,31	0,2
10	140	0,37	0,21
11	150	0,41	0,24
12	160	0,46	0,28
13	170	0,52	0,3
14	180	0,58	0,32
15	190	0,62	0,34
16	200	0,69	0,37
17	210	0,73	0,39
18	220	0,8	0,4
19	230	0,85	0,43
20	240	0,91	0,48

Отже, в середньому, паралельний варіант вирішення задачі на 40% ефективніший за послідовний (рис. 3).

Рис. 3. Порівняння часу виконання алгоритму в послідовному та розподіленому режимах

Для аналізу складності методу в двох режимах було проведено апроксимацію (рис. 4).

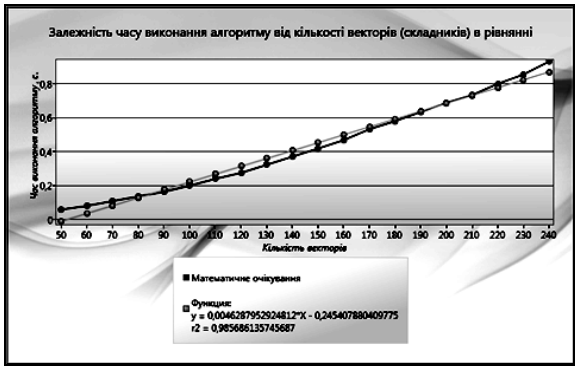

Рис. 4. Залежність часу виконання алгоритму від кількості векторів в рівнянні

З рис. 4 можна зробити висновок, що задача вибірки завдань з черги кластеру на основі рангового підходу є поліноміально-вирішуваною, та розроблений метод дозволяє управляти розподілом завдань в реальному часу.

Висновки

1. Запропоновано метод розв'язання задачі щодо оптимізації вибірки завдань кластеру на основі моделі системи нелінійних рівнянь з обмеженнями та сегментації черги на групові вибірки. Для вирішення задачі використано ранговий підхід.

2. Досліджуваний метод реалізовано засобами мови MC#.

3 Розроблений метод є ефективним, що підтверджують отримані дані. Паралельна реалізація ефективніша за послідовну, причому перевага паралельної реалізації зростає зі збільшенням складності задачі (рис. 3).

Надалі планується інтегрувати розроблену модель у функціонуючий диспетчер завдань кластеру і проаналізувати ефективність її роботи.

Список літератури

1. Кирьянов А.К. Введение в технологию Грид / А.К. Кирьянов, Ю.Ф. Рябов. – СПб: Гатчина, 2006. – 40 с.

2. U. Schwiegelshohn and R. Yahyapour. Analysis of First-Come-First-Serve Parallel Job Scheduling. In Proceedings of the 9th SIAM Symposium on Discrete Algorithms, 1998. – P. 629-638.

3. Коваленко В.Н. Использование алгоритма «Backfill» в грид / В.Н. Коваленко, Д.А. Семячкин // Тр. междунар. конф. «Распределенные вычисления и Грид-технологии в науке и образовании» (Дубна, 29 июня – 2 июля 2004 г.). – Дубна: ОИЯИ, 2004. – С. 139-144.

4. Листровой С.В. Метод решения задач целочисленного линейного программирования с булевыми переменными на основе рангового подхода / С.В. Листровой, Д.Ю. Голубничий, Е.С. Листровая // Электрон. Моделирование. – 1998. – Т. 20 „№ 6. – С. 14-32.

5. Методы и модели планирования ресурсов в GRID-системах: монография / В.С. Пономаренко, С.В. Листровой, С.В. Минухин, С.В. Знахур. – Х.: ИД «ИНЖЭК», 2008. – 408 с.

Надійшла до редколегії 1.10.2010

Рецензент: д-р техн. наук, проф. С.В. Лістровий, Харківська залізнична академія, Харків.

ОПТИМИЗАЦИЯ ПЛАНА ВЫБОРКИ ЗАДАНИЙ ИЗ ОЧЕРЕДИ КЛАСТЕРА НА ОСНОВЕ ПАРАЛЛЕЛЬНОГО РЕШЕНИЯ БУЛЕВЫХ НЕЛИНЕЙНЫХ УРАВНЕНИЙ С ОГРАНИЧЕНИЯМИ

С.В. Минухин, А.А. Ганжа

Рассмотрен метод групповой обработки заданий, которые поступают в очередь кластера, на основе модели рангового подхода. Метод реализован на языке программирования MC#. Проведен анализ результатов параллельной реализации метода.

Ключевые слова: *грид, кластер, диспетчер заданий, метод групповой обработки заданий, ранговый подход, FCFS, Backfill, MC#.*

OPTIMIZATION PLAN OF SELECTION TASKS FROM CLUSTER'S TURN ON BASIS PARALLEL DECISION OF BOOLE NONLINEAR EQUALIZATIONS WITH LIMITATIONS

S.V. Minukhin, A.A. Ganzha

The method batch-processing tasks which enter cluster's turn is considered, on the basis model of grade approach. A method is realized in programming of MC# language. The analysis result of parallel realization method is conducted.

Keywords: *grid, cluster, controller of tasks, method batch-processing tasks, grade approach, FCFS, Backfill, MC#.*