УДК 683.322

РУСАНОВА О. В.,
ЯРОХ Ю. А.

ПЛАНИРОВАНИЕ ВЫЧИСЛЕНИЙ В ГЕТЕРОГЕННЫХ КЛАСТЕРНЫХ СИСТЕМАХ

В статье предлагается универсальный подход HETS к планированию вычислений, как для однородных, так и неоднородных кластерных систем. Данный подход является комбинацией списочного метода и метода, использующего дублирование. Разработана программная модель для реализации предлагаемого и наиболее известных способов планирования вычислений в гетерогенных кластерных системах. Представлены результаты исследований, которые подтверждают более высокую эффективность подхода HETS по сравнению с известными способами.

The article proposes a universal approach for task scheduling, HETS, both for homogeneous and heterogeneous cluster systems. This approach is the combination of list and duplication scheduling methods. Software for implementation of the proposed and most well-known scheduling methods in heterogeneous computing cluster systems was developed. The results of researches, which are presented in the article, confirm the higher efficiency of the approach HETS in comparison with known methods, were presented.

1. Введение

Стремительное развитие кластерных систем последних лет привело к абсолютному их доминированию среди суперкомпьютеров мира. Повышение производительности кластерных систем непосредственно связано с увеличением общего числа ядер, используемых в них. Кроме того, увеличивается количество гетерогенных кластерных систем. Это приводит к усложнению решения задачи эффективного использования ресурсов в таких системах. Совершенствование методов планирования вычислений является одной из наиболее важных задач, решение которой ведет к росту реальной производительности и эффективности работы кластерных систем.

Существующие способы планирования вычислений обычно разделяют на четыре группы: списочные, кластерные, генетические и с использованием дублирования [1]. Списочные способы планирования являются на сегодняшний день наиболее предпочтительными для применения в кластерных системах, поскольку они, в большинстве случаев, позволяют обеспечить лучшие результаты планирования и имеют меньшую временную сложность по сравнению с другими способами [2]. В то же время известно, что использование способов планирования с дублированием в кластерных системах рекомендуется к применению для сильно связных задач [1].

Поэтому в работе предлагается комбинированный подход, в котором сочетаются преимущества двух способов – списочного и с использованием дублирования, что ведет к повышению эффективности планирования в гетерогенных кластерных системах.

2. Постановка задачи планирования

В соответствии с теорией расписаний задача планирования определена, если заданы модели пользовательской задачи и кластерной системы, результаты планирования, а также используемый критерий оптимизации.

Пользовательская задача, как правило, может быть представлена в виде направленного ациклического графа DAG *(Directed Acyclic Graph)*. DAG, $G = (V, E)$, состоит из множества V вершин (v) и множества E дуг (e). Обычно DAG называют графом задачи. Вершинам такого графа соответствуют различные части программы (подзадачи), а дугам – взаимосвязи между ними. Дуга $(i, j) \in E$ между подзадачами n_i и n_j представляет межвершинную коммуникацию. Результат подзадачи n_i должен быть передан подзадаче n_j до начала ее выполнения. Подзадача без предшественников называется входной, n_{entry}, а подзадача без приемников называется выходной n_{exit}. Среди предшественников n_i, тот, который последним производит коммуникацию, является наиболее влиятельной родительской вершиной (MIP – *Most Influential Parent*) и обозначается $MIP(n_i)$ [1]. Наиболее длинный путь графа задач является критическим путем (CP – *Critical Path*) [1]. Подзадача считается готовой к выполнению, если все подзадачи предшественники были выполнены.

Вес подзадачи, n_i, обозначаемой как w_i, представляет вычислительную стоимость подзадачи. Вычислительная стоимость подзадачи, n_i на процессоре p_j, – $w_{i,j}$. Вес дуги, обозначаемой как $c_{i,j}$, представляет коммуникационную стоимость между подзадачами n_i и n_j. Коммуникационная стоимость подзадач, назначенных на один и тот же процессор, равна нулю. Средняя вычислительная и коммуникационная стоимость подзадачи n_i обозначаются $\overline{w_i}$ и $c_{i,j}$ соответственно, где первый параметр – это средняя вычислительная стоимость всех процессоров системы, а второй – это средняя коммуникационная стоимость между подзадачей n_i и всеми подзадачами приемниками.

Моменты времени раннего начала *(Earliest Start Time)* и завершения *(Earliest Finish Time)* выполнения подзадачи n_i на процессоре p_j определяются следующим образом:

$$EST(n_i, p_j) = \begin{cases} 0 & n_i = n_{entry} \\ EFT(MIP(n_i), p_k),\ p_k \in P & j = k \\ EFT(MIP(n_i), p_k),\ p_k \in P + C_{MIP(n_i),i} & j \neq k \end{cases}$$

$$EFT(n_i, p_j) = EST(n_i, p_j) + w_{i,j}$$

Результат планирования может быть представлен с помощью модифицированной диаграммы Ганта [3], в которой расписание работы каждого процессора включает не только очередь исполняемых подзадач, но и порядок пересылаемых данных всех его каналов.

Задача планирования заключается в следующем. Пусть имеется кластерная система, состоящая из K узлов. Каждый i–й узел состоит из P_i процессоров. Задано приложение из m подзадач с помощью DAG графа. Известна также топология кластерной системы, описанная с помощью неориентированного графа. Требуется найти i–й узел кластерной системы, который обеспечивает оптимальное решение заданного приложения.

В данной работе решаются следующие задачи оптимизации планирования:

1. Необходимо найти такой i–й узел кластерной системы, который обеспечивает минимальное общее время выполнения заданного приложения (T_i). Математическая модель этой задачи может быть записана следующим образом. Найти

$$\min_{i=1,R} \{T_i\} \qquad (2.1)$$

$$T_i = \sum_{i=1}^{m} \sum_{l=1}^{P_i} t_{jli} * X_{jli} + S_i + \max\{Tr, Tf_i\} \qquad (2.2)$$

где t_{jli} – время выполнения j–й подзадачи в l-м процессоре i-го узла кластерной системы; S_i – время доставки входных данных и результатов приложения в (из) i-й (i-го) узел (узла) кластерной системы; Tr – время готовности приложения к выполнению в узлах системы; Tf_i – время освобождения i-го узла кластерной системы для выполнения заданного приложения в эксклюзивном режиме; $X_{jli} = 1$, если j–я подзадача выполняется l–м процессоре i-го узла, $X_{jli} = 0$ в противном случае.

2. Необходимо найти i–й узел кластерной системы с минимальной стоимостью, который обеспечивает выполнение заданного приложения за ограниченное время *(Tz)*. Математическая модель этой задачи может быть записана следующим образом. Найти

$$\min_{i=1,R} \{C_i\} \qquad (2.3)$$

при ограничении

$$T_i \leq Tz\ (i = 1, K) \qquad (2.4)$$

В соотношении (2.3) C_i – стоимость i-го узла при выполнении заданного приложения. В данном случае вначале находится подмножество узлов, где время выполнения приложения соответствует ограничению (2.4). Затем среди них определяется узел, в котором приложение выполняется с минимальной стоимостью (соотношение (2.3)).

3. Необходимо найти i-й узел кластерной системы с минимальной стоимостью, который обеспечивает минимальное общее время выполнения заданного приложения. Математическая модель этой задачи может быть записана следующим образом. Найти узел кластерной системы, удовлетворяющий соотношениям (2.1) и (2.2) при ограничении

$$C_i \leq C_{\min} \qquad (2.5)$$

В этом случае вначале, в соответствии с соотношениями (2.1) и (2.2), определяется подмножество узлов, где приложение выполняется за минимальное время. Затем среди них, в соответствии с (2.5), выбирается узел с минимальной стоимостью.

3. Существующие подходы планирования вычислений в кластерных системах

Генетические способы планирования.

Генетические способы планирования вычислений (GA [4], AIS [5], PSGA [6], Push-Pull [7]

и другие) являются наиболее широко известными подходами, использующими направленный случайный поиск [8].

С помощью генетических способов планирования можно получить, близкие к оптимальным [9]. Однако часто такие способы сводятся к полному перебору всех возможных вариантов. В связи с этим, время работы генетических алгоритмов несоизмеримо больше времени работы алгоритмов, основанных на любых других подходах.

Поэтому, использование генетических способов планирования для реальных кластерных систем проблематично [1].

Кластерные способы планирования.

Кластерные способы планирования обычно состоят из трех фаз. На первой, сильно связные задачи группируются в кластеры, которых может быть неограниченное количество, используя линейную или нелинейную кластерную эвристику. На второй фазе кластеры назначаются на процессоры, используя коммуникационную чувствительную или коммуникационную нечувствительную стратегию. На третьей фазе кластеры сливаются (объединяются) или декластеризуются в зависимости от количества доступных процессоров. Если объединенные два кластера обеспечивают уменьшение времени выполнения, то они сливаются.

Обычно кластерное планирование на первой фазе предполагает использование неограниченного количества процессоров, исходя из своих потребностей. Поэтому, когда нет нужного их количества, то происходит перестройка кластеров под доступное количество процессоров. Конечно, это влияет на качество и на эффективность планирования. Кроме этого, процесс кластеризации в общем виде имеет экспоненциальную временную сложность.

Кластерное планирование чаще всего применяется для полносвязных однородных систем, а также для систем, где нет ограничений на выделяемые ресурсы. Применение кластерных способов планирования вычислений для неполносвязных кластерных систем с неоднородными узлами не рекомендуется [1].

Способы планирования с дублированием.

Основная идея способов планирования вычислений с дублированием (HDBS [10], LDBS [11] , DBUS [12] и другие) состоит в том, чтобы производить назначение некоторых подзадач одновременно на несколько процессоров (рис.1). Таким образом, снижается негативное влияние пересылок данных при сильной связности графа задач. Способы планирования с дублированием отличаются стратегией выбора подзадач, которые стоит продублировать. Эти подходы показывают достаточно высокую эффективность на сильно связных графах задач. Однако его эффективность падает с уменьшением связности DAG.

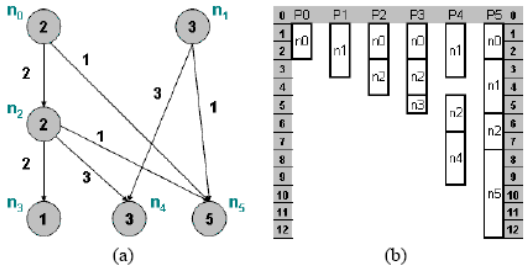

Рис. 1. Пример планирования с дублированием, где (a) граф задач, (b) диаграмма выполненного планирования с дублированием

Кроме того, способы планирования, основанные на дублировании, могут привести к необоснованной загрузке процессоров.

Поэтому использование способов планирования с дублированием в кластерных системах рекомендуется к применению на сильно связных графах задач [1].

Списочные способы планирования.

Списочные способы планирования (HEFT [13], LMT [14], MH [15], CPOP [13] и другие) являются на сегодняшний день доминантными среди эвристических способов планирования, поскольку они, в большинстве случаев, позволяют обеспечить лучшие результаты планирования и имеют меньшую временную сложность по сравнению с другими способами планирования.

Списочное планирование заключается в формировании очереди готовых подзадач с помощью одного из эвристических методов и затем оптимального назначения очереди вычислительных работ на процессоры. Все списочные способы отличаются подходами, применяемыми при выполнении указанных этапов.

На первом этапе для каждой подзадачи определяется приоритет. Основными способами определения приоритетов являются: *b-level* и *t-level* [1]. После этого подзадачи сортируются в порядке убывания их приоритетов. Способ определения приоритетов влияет на время выполнения задачи.

На втором этапе каждая готовая к выполнению подзадача назначается на процессор, исходя из политики выбора процессора.

В списочных способах планирования вычислений широко используется техника вставки, когда осуществляется назначение подзадачи на процессор между двумя уже погруженными подзадачами, если при этом имеется достаточный слот (промежуток) времени, а также техника дублирования подзадач (рис.2).

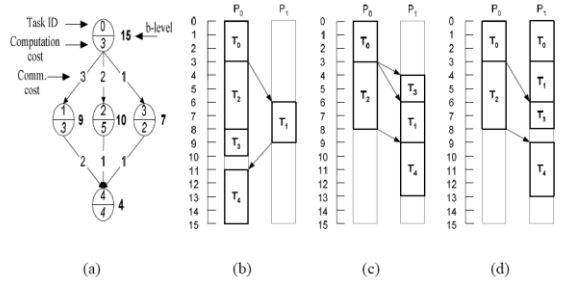

Рис.2. Примеры списочного планирования, где (a) граф задач, выполненное списочное планирование (диаграмма): (b) обычное, (c) с техникой вставки, (d) с дублированием

Для неоднородных кластерных систем более предпочтительно использовать способы списочного планирования вычислений, которые могут обеспечить субоптимальные результаты планирования при высокой скорости работы [1].

4. Способ планирования вычислений HETS

В работе предлагается новый эффективный способ планирования вычислений (HETS – *Heterogeneous Effective Task Scheduling*) в гетерогенных кластерных системах с произвольной топологией, на основе списочного подхода с использованием преобразования структуры DAG.

Целью предлагаемого способа планирования вычислений HETS является повышение реальной производительности кластерных систем с произвольной топологией. Данный подход является комбинацией списочного способа и способа, использующего дублирование, и позволяет минимизировать их недостатки [1].

Способ HETS, как и все списочные способы планирования, включает этап формирования очереди подзадач на основании их приоритетов, а также этап назначения готовых подзадач на процессоры. Однако HETS использует особую политику определения приоритетов: каждая подзадача графа имеет сложный приоритет, который состоит из статической и динамической составляющей. Статический приоритет каждой подзадачи вычисляется, так же как и в способе планирования HEFT, на основании значения *b-level* [1]. Динамический приоритет каждой подзадачи в начале погружения равен ее статическому приоритету, однако в процессе планирования он изменяется.

Способ планирования HETS явно не выполняет дублирование подзадач, однако, путем эквивалентного преобразования графа задач на втором этапе его работы, достигается эффект дублирования, при этом временная сложность такого подхода ниже.

Таким образом, HETS состоит из трех этапов, которые выполняются последовательно:

1. Начальное определение приоритетов подзадач графа (статическая приоритезация).

Статические приоритеты вершин определяются на основании критических путей до конца графа задачи:

$T_{кр.i} = b\text{-}level(n_i)$,

где $T_{кр.i}$ (CP_i) – критический путь *i*-й подзадачи до конца графа.

Значение *b-level* подзадачи определяется как сумма вычислительных и коммуникационных стоимостей самого длинного пути начиная от рассматриваемой вершины до результирующей в графе задачи. Значения *b-level* подзадачи n_i определяются как [1]:

$$b(n_i) = \overline{w}_i + \max_{n_j \in imed\_succ(n_i)} \left\{ \overline{c}_{i,j} + b(n_j) \right\},$$

где $imed\_succ(n_i)$ – множество непосредственных приемников подзадачи n_i, $\overline{w}_i$ – среднее значение веса *i*-й вершины (следствие гетерогенности кластерной системы), $c_{i,j}$ – среднее значение веса дуг от *i*-й к *j*-й вершине.

Таким образом: $Ps_i = T_{кр.i}$, где Ps_i – статический приоритет i-й вершины.

2. Преобразование графа задач.

В способе планирования HETS предусмотрены различные преобразования графа задачи. При этом выполняется «мягкое» дублирование, то есть подзадача дублируется только тогда, когда это целесообразно. Таким образом, сохраняется преимущество способов планирования с дублированием, заключающихся в эффективности их работы на сильносвязных графах задач.

Способ планирования вычислений HETS выполняет следующие преобразования графа задач [16]:

- Дублирование подзадач.

- Объединение предшествующей и последующей подзадач.
- Объединение двух параллельных ветвей графа задач.
- Объединение двух предшествующих подзадач.

Преобразование графа, путем дублирования задач, имеет смысл применять только тогда, когда подзадачи-приемники будут выполняться на тех же процессорах, что и подзадачи-источники. В этом случае веса пересылок будут аннулированы.

Эффективность данного подхода связана с соотношением потерь на дополнительные вычисления за счет дублирования подзадач и выигрыша за счет уменьшения времени пересылок. Благоприятным фактором является сохранение времени критического пути графа задач после применения процедуры дублирования.

Способ планирования вычислений HETS прдусматривает применение преобразование графа, путем дублирования, в случаях, когда:

- Подзадача является входной и имеет две или больше подзадач-наследников.
- Если подзадача имеет подзадачипредшественники и имеет две или больше подзадач-наследников, а также выполняется условие:

$$Sum\ (e_1 \dots e_n) > Sum\ (e_{01} * n \dots e_{0m} * n),$$

где $e_1 \dots e_n$ – веса дуг между подзадачей-кандидатом на дублирование и ее подзадачами-наследниками, $e_{01} \dots e_{0m}$ – веса дуг подзадачи-кандидата на дублирование и ее подзадачами-предшественниками, n – количество подзадач-наследников у подзадачи-кандидата на дублирование, m – количество подзадач предшественников у подзадачи-кандидата на дублирование.

На рис.3 продемонстрирован граф задач до применения преобразования (слева), путем дублирования подзадач, и граф задач после применения преобразования (справа), путем дублирования подзадач. Различные варианты объединения предусматривают слияние подзадач графа и инцидентных им соединений. В случае объединения выполняется «укрупнение» подзадач графа.

В способе планирования вычислений HETS применяется преобразование графа, путем объединение предшествующей и последующей подзадач в случаях, когда:

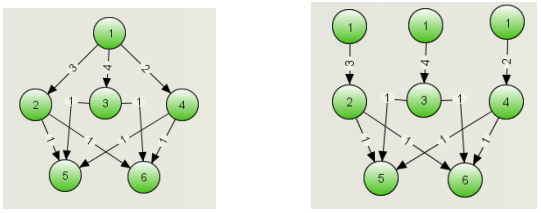

Рис. 3. Граф задач до и после применения преобразования, путем дублирования подзадач

- Было выполнено дублирование подзадач.
- Если подзадача имеет единственную предшествующую и единственную последующую вершину.

На рис. 4 продемонстрирован граф задач до применения преобразования, путем объединение предшествующей и последующей подзадач, и граф задач после применения такого преобразования. При этом веса подзадач 2, 3 и 4 увеличиваются на единицу.

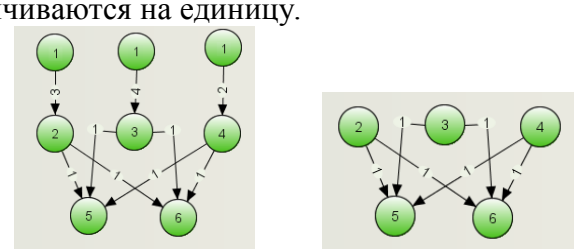

Рис. 4. Граф задач до и после применения преобразования, путем объединения предшествующей и последующей подзадач

В случае использования преобразования графа, путем объединение двух параллельных ветвей графа задач, производится слияние независимых подзадач графа. При этом веса сливаемых соединений и подзадач объединяются. В результате получается граф, в котором число соединений, а, следовательно, и количество пересылок уменьшается.

Алгоритм такой трансформации заключается в проверке целесообразности объединения соседних подзадач и в итеративном процессе их объединения. Резерв времени для каждой подзадачи определяется как разность между поздним и ранним сроками начала ее исполнения без изменения общего времени выполнения графа. Объединение проводится, если резерв времени одной ветви больше времени выполнения второй ветви, с которой она объединяется. Процесс продолжается итеративно до тех пор, пока уменьшается время выполнения задачи.

В случае объединения двух предшествующих подзадач производится слияние независимых подзадач и дуг графа, при этом их веса

объединяются. В результате уменьшается количество пересылок в графе.

Уменьшение количества пересылок приводит к сокращению числа конфликтов при пересылках в системе, однако при этом увеличивается объем самих пересылок (которые объединились в одну), а также объем вычислений (объединение вершин графа). Следовательно, эффективность такого преобразования зависит от производительности процессоров в системе, а также от числа и пропускной способности ее каналов.

3. Назначение задач процессорам и доопределение приоритетов задач (динамическая приоритезация).

До начала назначения на процессоры все динамические приоритеты подзадач равны их статическим приоритетам.

Перед выбором подзадачи для назначения осуществляется доопределение приоритетов, а точнее обновление их динамической составляющей:

- Для всех доступных (готовых) подзадач для всех процессоров определяется текущий запас времени по формуле:

$$FT_{i,j} = T_{кр.гр} - T_{кр.\,i} - T_{тек.j},$$

где $FT_{i,j}$ – запас времени i-й подзадачи на j-м процессоре и $T_{тек.j}$ – текущий такт j-го процессора.

- Для всех доступных (готовых) подзадач определяется средний запас времени по формуле:

$$FT_i = Sum\,(FT_{i,1} \ldots FT_{i,n}) / n,$$

где n – количество процессоров компьютерной системы.

- Если $FT_i < 0$, то $P_i = Ps_i + |FT|$, где P_i – комплексный приоритет i-ой вершины.

Подзадача с наибольшим комплексным приоритетом P выбирается для назначения. При равенстве комплексных приоритетов выбирается подзадача с большей связностью, а при равенстве связностей – с меньшим весом.

В HETS при выборе процессора для назначения учитываются все процессоры, даже если некоторые из них заняты в момент назначения.

Назначение подзадач на процессоры основано на выборе того из них, который обеспечивает минимальное время окончания выполнения выбранной подзадачи (EFT). Значение EFT подзадачи на конкретном процессоре включает время доставки для нее исходных данных с учетом топологии кластерной системы, а также время выполнения подзадачи на процессоре с учетом его производительности.

5. Методика сравнения способов планирования вичислений

Сравнение способов планирования вычислений основывается на следующих метриках:

- *Scheduling Length (SL). SL – makespan* определяется временем выполнения задач графа на кластерной системе.
- *Scheduling Length Radio (SLR).* Значение *SLR* графа задач фактически определяет, во сколько раз *SL* больше CP_{MIN} , и вычисляется по формуле:

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

где CP_{MIN}. – сумма минимальных вычислительных стоимостей подзадач. Способ планирования вычислений с минимальным значением *SLR* является лучшим, с точки зрения времени выполнения задачи.

- *Speedup (SU)* или ускорение. Значение *SU* определяется отношением времени выполнения задачи на однопроцессорной системе ко времения ее выполнения на параллельной системе:

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

- *Number of Occurrences of Better Quality of Schedules (NOBQS).* Данный показатель определяет число случаев лучшего качества планирования. Выполняется подсчет лучшего, равного и худшего качества планирования каждого из способов планирования в сравнении со всеми другими по отдельности и вместе. Более качественным является планирование, обеспечивающее меньшее время выполнения графа задач *(SL).*
- *Running time (RT). RT* определяет время работы самого алгоритма планирования. При статическом планировании этот показатель учитывается лишь при выборе алгоритмов планирования, с равными значениями *SLR*. Тот из них, который имеет меньшее *RT*, является более предпочтительным для практического применения.

6. Экспериментальные результаты проведенных исследований

Основываясь на рекомендациях использования и эффективности применения способов планирования вычислений в гетерогенных кла-

стерных системах [1], для исследования были отобраны три широко известных списочных способа планирования (Heterogeneous Earliest Finish Time – HEFT [13], Levelized Min Time – LMT [14], Critical Path On a Processor – CPOP [13]) и два наиболее эффективных способа планирования с дублированием (Heterogeneous Duplication-Based Scheduling – HDBS [10], Levelized Duplication-Based Scheduling – LDBS [11]).

Для сравнения вышеупомянутых способов планирования с предлагаемым подходом HETS разработан программный комплекс Owl.

Граф топологии гетерогенной кластерной системы для проведения сравнения способов планирования представлен на рис.5.

При проведении данных исследований оценивается качество планирования различных способов планирования на множестве случайно сгенерированных графов задач. Для этого используется генератор, описанный в работе[13], со следующими входными параметрами:

- v = {30, 40, 50, 60, 70, 80, 90, 100} – количество подзадач графа.
- a = {0.5, 1.0, 2.0} – форма графа.
- *out_degree* = {1, 2, 3, 4, 5} – выходная степень подзадачи.
- *CCR* = {0.1, 0.5, 1.0, 5.0, 10.0} – соотношение суммарной коммутационной

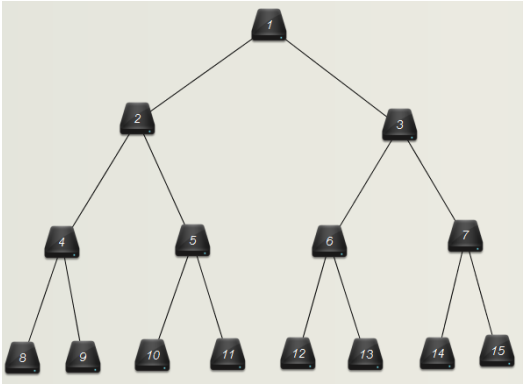

Рис. 5. Граф топологии кластерной системы для проведения исследований

стоимости к суммарной вычислительной стоимости графа задач.

- β = {0.25, 0.5, 0.75, 1.0} – диапазон отношений вычислительных стоимостей процессоров к самому производительному из них.

Все возможные комбинации этих входных параметров включают 2400 типов DAG. Кроме этого, для каждого из типов генерируется 10 графов для усреднения результатов планирования по каждому типу DAG. Таким образом, для сравнения способов планирования всего сгенерировано 24000 графов задач.

Результаты проведенных экспериментов представлены на рис.6-9 и в табл.1.

На основании результатов, приведенных на рис.6, можно отметить следующее:

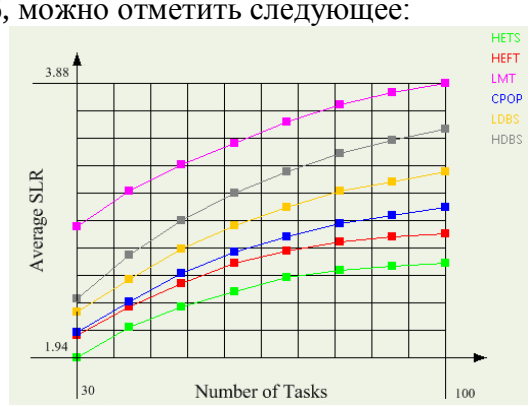

Рис.6. Зависимость среднего SLR от количества вершин графа задач

- По результатам проведенного эксперимента лучшим способом планирования, обеспечивающим минимальное среднее значение SLR из диапазона количества задач графа {30 - 100}, является HETS.
- По результатам сравнения средних значений *SLR*, исследуемые способы планирования можно расположить в порядке ухудшения характеристики *SLR* следующим образом: {HETS, HEFT, CPOP, LDBS, HDBS, LMT}.

Анализируя результаты, приведенные на рис.7, можно отметить следующее:

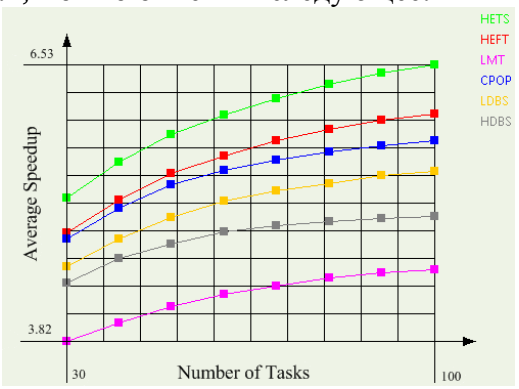

Рис. 7. Зависимость среднего SU от количества вершин графа задач

- По результатам проведенного теста лучшим способом планирования, обеспечивающим максимальное среднее значение *SU*, является HETS.
- По результатам сравнения значений среднего *SU*, исследуемые способы планирования можно расположить в порядке ухудшения характеристики *SU* следующим образом: {HETS, HEFT, CPOP, LDBS, HDBS, LMT}.

Анализируя зависимости среднего *SLR* от *CCR* (рис.8), можно отметить следующее:

- По результатам проведенного теста лучшим способом планирования является HETS, который обеспечивает минимальное

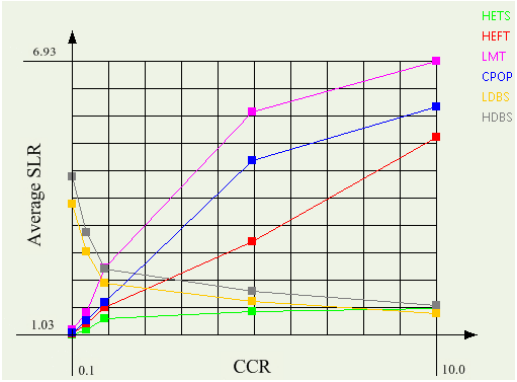

Рис. 8. Зависимость среднего SLR от CCR

среднее значение *SLR* практически на всем диапазона *CCR* {0.1 - 10.0} за исключением, когда *CCR* = {8.0, 9.0, 10.0}, где HETS уступает способу планирования LDBS.

Анализируя полученные зависимости среднего *RT* от количества вершин графа задач (рис.9), можно отметить следующее:

- По результатам проведенного теста лучшим способом планирования с минимальным средним значением *RT* является HEFT.
- По результатам сравнения средних значений *RT*, исследуемые способы планирования можно расположить в порядке ухудшения характеристики *RT* следующим образом: {HEFT, CPOP, HETS, LMT, LDBS, HDBS}.

Попарно сравнивая средние значения *SL* исследуемых способов планирования между собой (табл.1), можно отметить следующие:

- Способ планирования HETS для 91% графов задач обеспечил минимальное значение *SL*, что является лучшим результатом среди всех исследуемых способов планирования.

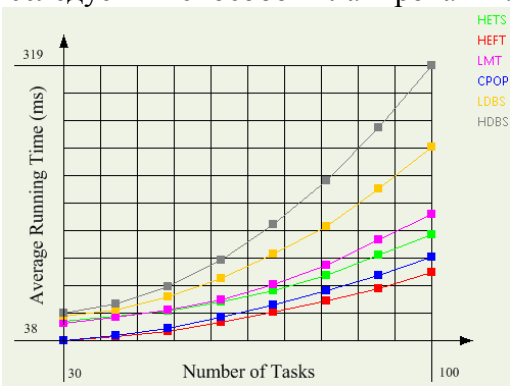

Рис. 9. Зависимость среднего RT от количества вершин графа задачи

- По результатам сравнения значений *SL*, исследуемые способы планирования можно расположить в порядке ухудшения характеристики *NOBQS* следующим образом: {HETS, HEFT, CPOP, LDBS, HDBS, LMT}.

Среднее значение *SLR* на 24000 сгенерированных графах задач для HETS = 2.39, для HEFT = 2.57, для LMT = 3.48, для CPOP = 2.68, для LDBS = 2.87, для HDBS = 3.09.

Таким образом, исследуемые способы планирования, в порядке убывания эффективности планирования по результатам экспериментов, можно расположить следующим образом: {HETS, HEFT, CPOP, LDBS, HDBS, LMT}. При этом среднее значение *SLR* у HETS лучше: на 8%, чем у HEFT; на 12%, чем у CPOP; на 20%, чем у LDBS; на 30%, чем у HDBS; на 45%, чем у LMT.

Табл.1. Результаты попарного сравнения значений SL способов планирования (NOBQS) на множестве случайно сгенерированных графов задач

		HETS	**HEFT**	**LMT**	**CPOP**	**HDBS**	**LDBS**	
HETS	Лучше		20657	23776	22153	21671	21092	91%
	Равно	*	197	114	378	684	575	2%
	Хуже		3146	110	1469	1645	2333	7%
HEFT	Лучше	3146		21939	19067	17167	13912	63%
	Равно	197	*	362	601	454	623	2%
	Хуже	20657		1699	4332	6379	9465	35%
LMT	Лучше	110	1699		2152	10061	6639	17%
	Равно	114	362	*	340	459	613	2%
	Хуже	23776	21939		21508	13480	16748	81%
CPOP	Лучше	1469	4332	21508		16043	12716	47%
	Равно	378	601	340	*	458	513	2%
	Хуже	22153	19067	2152		7499	10771	51%
HDBS	Лучше	1645	6379	13480	7499	*	8293	31%

	Равно	684	454	459	458		719	2%
	Хуже	21671	17167	10061	16043		14988	67%
LDBS	Лучше	2333	9465	16748	10771	14988		45%
	Равно	575	623	613	513	719	*	3%
	Хуже	21092	13912	6639	12716	8293		52%

7. Выводы

В статье предложен новый эффективный способ планирования вычислений HETS для гетерогенных кластерных систем с произвольной топологией. Данный подход является комбинацией списочного метода и метода, использующего дублирование. HETS позволяет объединить преимущества указанных методов и минимизировать их недостатки. Разработана программная модель для реализации предлагаемого и наиболее известных способов планирования вычислений для гетерогенных кластерных систем. Представлены результаты исследований, которые подтверждают более высокую эффективность подхода HETS по сравнению с известными способами.

Предложенный способ планирования вычислений HETS является универсальным и может быть использован для повышения реальной производительности, как масштабируемых систем с массовым параллелизмом, так и гетерогенных кластерных систем.

Список литературы

1. Young Choon Lee, «Problem-Centric Scheduling for Heterogeneous Computing Systems», September 2007.
2. Y. K. Kwok and I. Ahmad, «Benchmarking the Task Graph Scheduling Algorithms», Proc. First Merged Int'l Parallel Symposium/Symposium on Parallel and Distributed Processing (IPPS/SPDP '98), pp. 531–537, 1998.
3. G. Loutsky, O. Rusanova, «Scheduling Problems on the Parallel and Distributed Systems - an Overwiew» – Poland, Rzeszow, tom 1, pp.101-105 (Engl), 2000.
4. E. S. H. Hou, N. Ansari, and H. Ren, «A genetic algorithm for multiprocessor scheduling», IEEE Trans. Parallel and Distributed Systems, vol. 5, no. 2, pp. 113–120, 1994.
5. R. L. King, S. H. Russ, A. B. Lambert, and D. S. Reese, «An artificial immune system model for intelligent agents», Future Generation Computer Systems, vol. 17, no. 4, pp. 335–343, 2001.
6. M. K. Dhodhi, I. Ahmad, A. Yatama, «An integrated technique for task matching and scheduling onto distributed heterogeneous computing systems», J. Parallel and Distributed Computing, 62: 1338-1361, 2002.
7. S. C. Kim and S. Lee, «Push-pull: Guided search DAG scheduling for heterogeneous clusters», Proc. Intl. Conf. Parallel Processing (ICPP'05), 2005.
8. L. Wang, H. J. Siegel, V. P. Rowchoudhry and A. A. Maciejewski, «Task matching and scheduling in heterogeneous computing environments using a genetic algorithm-based approach», J. Parallel and Distributed Computing, 47: 8-22, 1997.
9. Cristina Boeres, Jos´e Viterbo Filho and Vinod E. F. Rebello, «A cluster-based strategy for scheduling task on heterogeneous processors», Proc. 16th Symp. on Computer Architecture and High Performance Computing (SBAC-PAD), 2004.
10. Y. K. Kwok, «Parallel program execution on a heterogeneous PC cluster using task duplication», Proc. 9th Heterogeneous Computing Workshop (HCW), pp. 364–374, May 2000.
11. A. Dogan and R. Ozguner «LDBS: A Duplication Based Scheduling Algorithm for Heterogeneous Computing Systems», Proc. Int'l Conf. Parallel Processing, pp. 352–359, August 2002.
12. D. Bozdag, U. Catalyurek and F. Ozguner, «A task duplication based bottom-up scheduling algorithm for heterogeneous environments», Proc. Int'l Parallel and Distributed Processing Symp., April 2005.
13. 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.
14. M. Iverson, F. Ozguner and G. Follen, «Parallelizing existing applications in distributed heterogeneous environments», Proc. Heterogeneous Computing Workshop, pp: 93-100, 1995.
15. H. EI-Rewini and T. G. Lewis, «Scheduling parallel program tasks onto arbitrary target machines», J. Parallel and Distributed Computing, 9: 138-153, 1990.
16. O.V.Rusanova, A.P.Shevelo, «List Scheduling Algorithm Modification for MPP Systems». Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техніка, Київ 2006 р.№45.-238 с.- С101-111.