АЛГОРИТМЫ ПОИСКА АЛЬТЕРНАТИВНЫХ НАБОРОВ СЛОТОВ В ЗАДАЧЕ ПЛАНИРОВАНИЯ ПАКЕТА ЗАДАНИЙ

В.В. Топорков, А.С. Топоркова, А.В. Бобченков, Д.М. Емельянов, А.С. Целищев

В статье рассматриваются алгоритмы поиска альтернативных наборов слотов в модели управления потоками независимых заданий в виртуальных организациях распределенных вычислительных сред с неотчуждаемыми ресурсами. Рассматриваемые алгоритмы позволяют учитывать ресурсные требования заданий и действующие в модели экономические механизмы.

1. Введение

Экономические модели выделения ресурсов и планирования являются весьма эффективными в распределенных вычислениях с неотчуждаемыми ресурсами, включая грид, мультиагентные системы и облачные вычисления [Garg et al., 2009; Bredin et al., 1999].

В [Ernemann et al., 2002] рассматриваются эвристические алгоритмы поиска наборов слотов на основе задаваемых пользователем функций полезности.

Среди различных подходов к организации вычислений в распределенных системах и средах можно выявить *две устойчивых тенденции*.

Одна из них основывается на использовании *брокеров ресурсов* [Buyya et al., 2002; Ernemann et al., 2002], которые, как правило, позволяют оптимизировать выполнение конкретного приложения [Toporkov, 2009].

Другая тенденция связана с образованием *виртуальных организаций* и ориентирована, прежде всего, на грид-системы. При образовании виртуальных организаций [Kurowski et al., 2003] осуществляется оптимизация планирования на уровне потоков заданий. В рамках предложенной модели планирования потоков заданий [Топорков и др., 2009], важным этапом является предварительный поиск альтернативных вариантов назначения выполнения каждого из заданий.

Новизна подхода, состоит в применении экономических механизмов как на этапе отбора альтернативных наборов слотов, так и на этапе планирования выполнения пакета заданий.

2. Схема планирования

В рамках рассматриваемой модели, в основу которой заложена иерархическая схема управления потоками заданий [Toporkov, 2009; Kurowski et al., 2003], фигурируют пользователи, запускающие задания, и собственники вычислительных ресурсов. Интересы пользователей и собственников зачастую противоречивы. Каждый из независимых пользователей заинтересован в наискорейшем запуске своих заданий с наименьшими издержками (например, платой за использование ресурса), а собственники, наоборот, стремятся получить наибольший доход от предоставления ресурсов.

Планирование выполнения совокупности (пакета) независимых заданий осуществляется циклично, на основе динамично обновляемых расписаний выполнения заданий в локальных процессорных узлах (рис. 1).

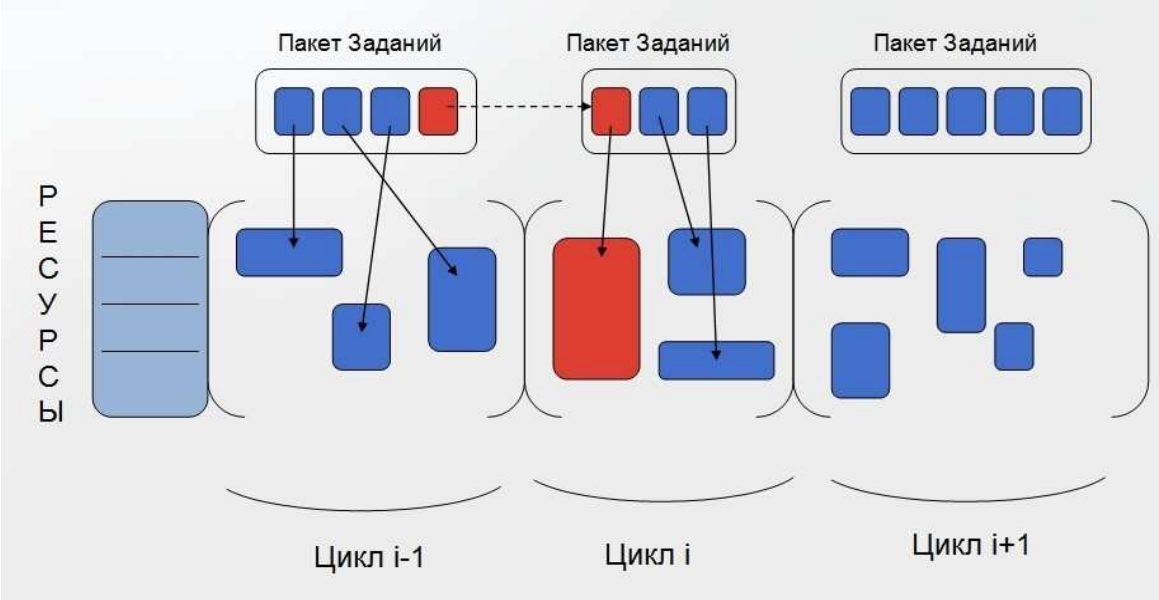


Рис. 1. Планирования потока заданий на основе циклов

В каждом цикле планирования запуска пакета заданий требуется решение двух задач. Во-первых, отбор подходящих (по ресурсу, времени, цене) слотов для каждого задания пакета [Ernemann et al., 2002] (в общем случае у каждого задания может оказаться несколько альтернативных наборов слотов для выполнения -

альтернатив). Во-вторых, необходим выбор комбинации альтернатив, являющейся эффективной или оптимальной с точки зрения прохождения всего пакета заданий в текущем цикле планирования.

В данной работе мы рассматриваем два типа критериев и две задачи однокритериальной оптимизации выполнения пакета заданий. Это *минимизация стоимости* выполнения пакета заданий при ограничении на суммарное время занятия слотов и *минимизация времени* прохождения пакета заданий при фиксированном бюджете виртуальной организации.

3. Алгоритмы отбора слотов

В начале каждого цикла динамично обновляются наборы доступных слотов на основе информации, поступающей от локальных менеджеров ресурсов. Каждый из слотов соответствует временному отрезку, который может быть использован для выполнения задачи, входящей в состав многопроцессорного задания, на том или ином типе ресурса. Требования задания к ресурсам оформляются в виде *ресурсного запроса*, содержащего тип, количество N и характеристики узлов (тактовую частоту процессора, емкость оперативной памяти, дисковое пространство, операционную систему и т.д.), а также оценку времени t их использования. Кроме того, пользователь указывает максимальную удельную стоимость C за использование отдельного узла, которую он готов заплатить. В рассматриваемой нами модели пользователь также указывает минимальную допустимую производительность P узлов, подходящих для выполнения его задания. Для запуска многопроцессорного задания необходимо согласованное выделение требуемого для его выполнения N слотов. В случае однородных узлов совокупность слотов для выполнения задания представляет собой прямоугольное *«окно»* размером Nxt, а для процессоров с разной производительностью это – «окно» с неровным правым краем, где – время выполнения составной части задания на наименее производительном процессоре (рис. 2).

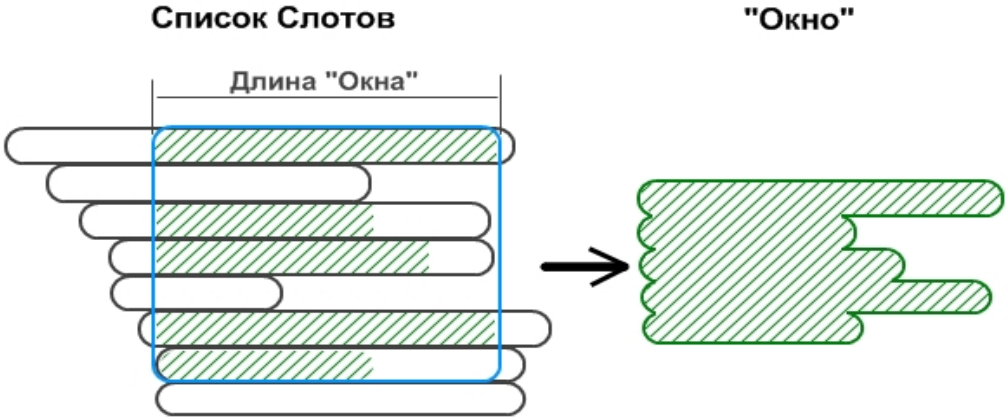


Рис. 2. Формирование окна с неровным правым краем

Нами рассматриваются два алгоритма поиска альтернативных наборов слотов.

Первый из них ALP, производит поиск окна на упорядоченном по неубыванию времени старта списке слотов системы. Его особенность состоит в том, что он учитывает ограничение C на максимальную удельную стоимость каждого отдельного слота. Приведем описание алгоритма ALP:

1. Доступные слоты системы упорядочиваются по неубыванию времени старта.
2. Из полученного списка выбирается следующий подходящий слот s. Слот считается подходящим, если удовлетворяет следующим требованиям:

a) производительность узла, на котором выделен слот P(s)>=P;
b) длины слота достаточно, чтобы выполнить часть многопроцессорного задания (с учетом актуальной производительности узла) L(s) >= tP(s)/P;
c) удельная стоимость использования слота не выше ограничения C(s)<=C.

Если условия a), b) и c) выполняются, то слот добавляется в список слотов окна.

3. Время старта всего окна полагается равным времени старта последнего добавленного слота. Далее происходит проверка всех текущих слотов списка окна. Если с учетом нового общего времени старта длины слота уже не хвататет, чтобы выполнить часть задания (пункт 2°b), этот слот удаляется из списка слотов окна и в дальнейшем не рассматривается.
4. Если число оставшихся слотов в списке окна меньше N, то переход на шаг 2°.
5. Конец алгоритма.

В случае, если до конца списка слотов системы мы не смогли найти подходящих слотов, то считается, что окно найти не удалось и задание переносится в следующий цикл планирования. Стоит отметить, что данный подход с ограничением на удельную стоимость отдельных слотов сужает область поиска и не позволяет использовать для составления окна более дорогие слоты. Отличие следующего алгоритма AMP состоит в том, что он оперирует с ограничением на максимальный бюджет выполнения задания. Максимальный бюджет формируется как S=CtN , где t - необходимое время резервирования, N - запрашиваемое количество слотов, а C -

указанное в ресурсном запросе ограничение на удельную стоимость отдельного слота. Таким образом, алгоритм AMP производит поиск окна, суммарная стоимость слотов которого не превышает максимальный бюджет S.

При описании алгоритма AMP мы воспользуемся дополнительными обозначениями: Ns – текущее количество слотов в списке окна; Mn – суммарная стоимость первых N слотов окна.

Рассмотрим алгоритм AMP для поиска отдельного окна:

1. Находится самое ранне окно из N слотов, используя алгоритм ALP без учета условия 2°c.
2. Слоты внути списка окна упорядочиваются по возрастанию их стоимости.

Далее вычисляется суммарная стоимость Mn первых N слотов окна. Если Mn<=S, то к шагу 4°, в этом случае результирующее окно формируетсся из первых N слотов, а остальные возвращаются в список слотов системы. Иначе к шагу 3°.

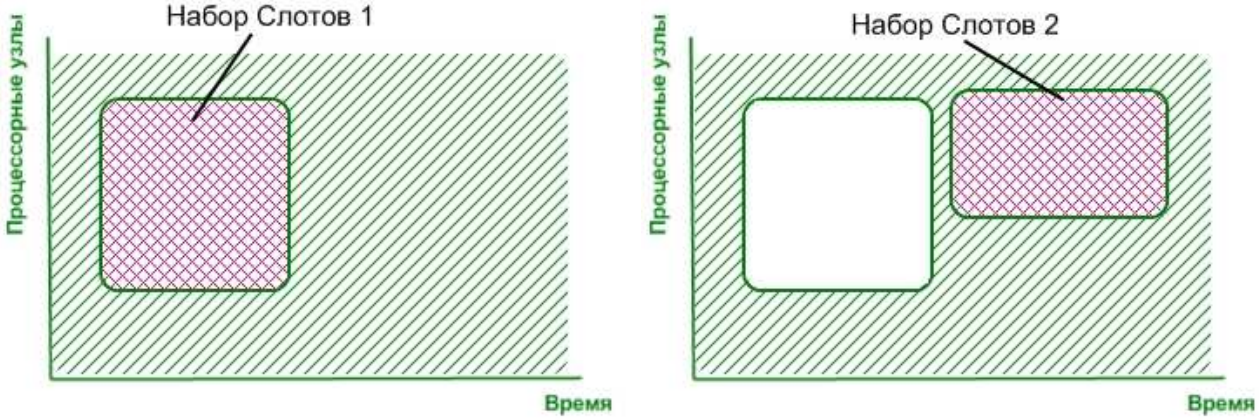
3. Добавляется следующий подходящий слот из списка с учетом ограничений 2°a и 2°b алгоритма ALP. Время старта формируемого окна полагается равным времени старта этого слота. Далее проверяется, все ли слоты успевают выполнить задание с учетом нового времени старта, а те, которые не успевают, удаляются (шаг 3° алгоритма ALP).
   Если Ns<N, то повтор шага 3°. Если Ns>=N, то к шагу 2°.
   Если список слотов системы закончен и Ns<N, то найти окно не удалось.
4. Конец алгоритма.

Следует отметить следующие три особенности предлагаемых алгоритмов. Во-первых, оба алгоритма учитывают производительность и разнородность вычислительных узлов. Это позволяет формировать окна с неровным правым краем. Во-вторых, оба алгоритма учитывают экономические ограничения пользователя:

ограничение на максимальную допустимую удельную стоимость использования ресурсов. В-третьих, оба алгоритма обладают линейной сложностью O(m) относительно общего количества слотов системы m, алгоритмы двигаются по списку слотов в направлении неубывания времени старта, без возвращения назад и пересмотра предыдущих шагов.

4. Схема поиска независимых альтернативных наборов слотов для пакета заданий

Для поиска множества альтернатив выполнения пакета заданий необходим алгоритм, который при поиске каждой следующей альтернативы учитывал бы ранее отобранные альтернативы, чтобы не возникало конфликтов и пересечений во времени. Для этого каждую найденную альтернативу – «окно», надо «вырезать» из списка доступных слотов, то есть на каждом шаге модифицировать список слотов (Рис. 3).



a b

Рис. 3. а, б. Схема поиска независимых альтернатив

Если для задания i было успешно найдено подходящее окно, то поиск альтернатив выполнения для последующих заданий должен происходить на модифицированном списке доступных слотов.

В результате работы описанных выше алгоритмов поиска альтернатив, формируется набор слотов, удовлетворяющий заданию и ресурсному запросу. Но для получения искомого окна каждый слот набора преобразуется так, чтобы его время старта было равно времени старта окна window.start, а длительность была равна оценке длительности выполнения задания на том типе ресурса, на котором выделен слот. Набор таких отформатированных слотов как раз и формирует искомое окно.

Если, к примеру, среди подходящих слотов окна оказался слот k, то в окне этот слот следует преобразовать (назовем этот преобразованный слот k’), присвоить время старта k’.start = window.start, а время окончания k’.end = k’.start + Ti (где Ti – оценка времени выполнения задания на ресурсе, на котором выделен слот k). Сформированный слот k’ и следует вычесть из исходного списка доступных слотов, то есть из соответствующего слота k, из которого он и был получен.

В общем случае при вычитании из упорядоченного списка слотов следует удалить слот k и добавить

два новых слота k1 и k2, у которых

k1.start = k.start,

k1.end = k’.start,

k2.start = k’.end

и k2.end = k.end (Рис. 4).

Слоты k1 и k2 необходимо добавить в список слотов, учитывая то, что список отсортирован по неубыванию времени старта. Слот k1 будет иметь ту же позицию в списке, что и слот k, так как они имеют одинаковое время старта. Если слоты k1 и k2 оказались нулевой длины, их в список добавлять не надо.

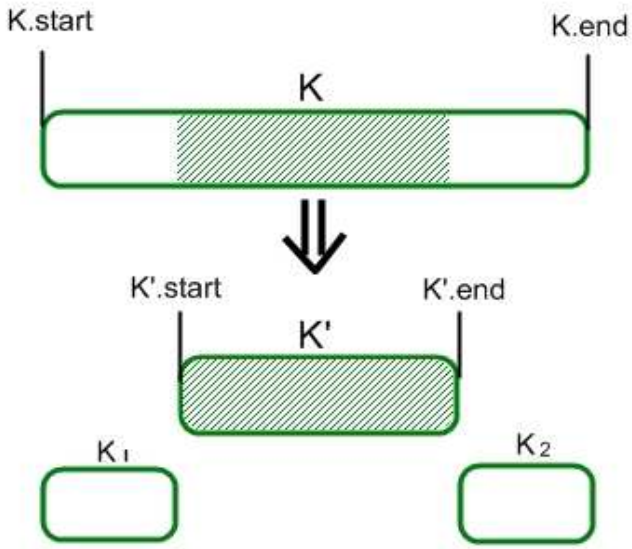


Рис. 4 Модификация списка слотов

5. Пример работы алгоритма

Для простоты и удобства демонстрации рассмотрим задачу поиска альтернатив выполнения для пакета заданий на однородном наборе ресурсов: искомые окна будут иметь прямоугольный вид. При учете ограничения на максимальную стоимость окна мы будем оперировать с величиной его суммарной удельной стоимости (получаемой суммированием удельных стоимостей всех слотов, из которых оно состоит).

Рассмотрим следующее начальное состояние распределенной вычислительной среды (Рис. 5). В представленной модели имеется 6 процессорных узлов cpu1 – cpu6, каждый из которых имеет собственную удельную стоимость (стоимость использования его единицы времени), которая указана в столбце справа от имени процессора. Текущее модельное время системы t = 0. Кроме того в системе уже спланировано выполнение семи задач p1-p7.

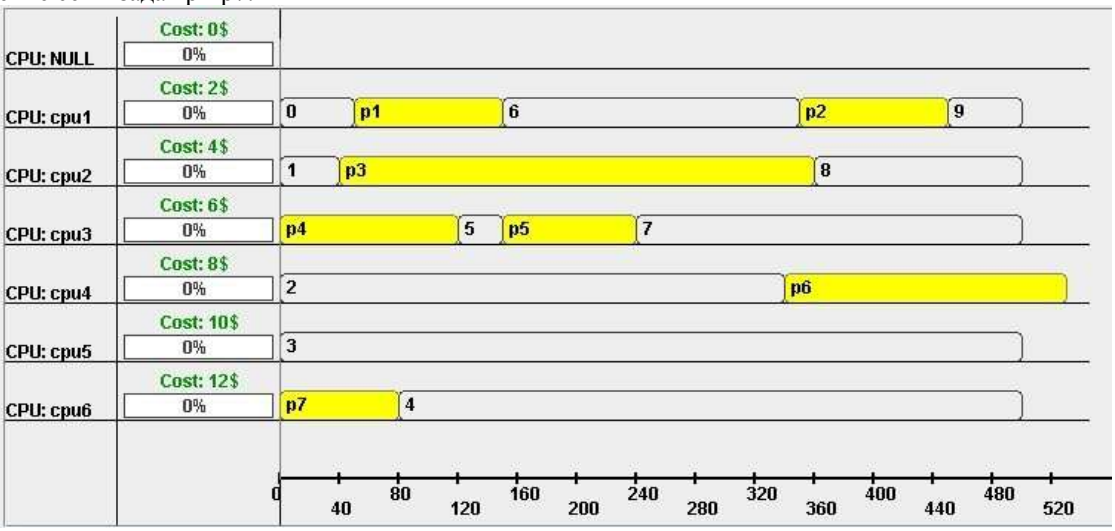


Рис. 5. Начальное состояние среды с упорядоченными слотами

Слоты на Рис. 5 упорядочены по неубыванию времени старта. Порядковый номер каждого слота указан на диаграмме на теле слота (0-9). Слоты изображены прозрачными (белыми) прямоугольниками.

Для наглядности рассмотрим ситуацию, когда в цикле планирования всего три задания со следующими

параметрами.

Задание 1:

Число необходимых процессоров: 2

Время выполнения: 80

Максимальная суммарная удельная стоимость выполнения: 10

Задание 2:

Число необходимых процессоров: 3

Время выполнения: 30

Максимальная суммарная удельная стоимость выполнения: 30

Задание 3:

Число необходимых процессоров: 2

Время выполнения: 50

Максимальная суммарная удельная стоимость выполнения: 6

Далее, согласно предложенной схеме поиска альтернатив, формируется список доступных слотов и с помощью алгоритма AMP находится самая ранняя подходящая альтернатива (первое подходящее окно) для первого задания. Затем окно, найденное для первого задания, вычитается из списка слотов и на оставшихся слотах находится самая ранняя подходящая альтернатива для второго задания. Далее выполняется аналогичная операция для третьего задания. На рис. 6 представлен результат моделирования этого этапа.

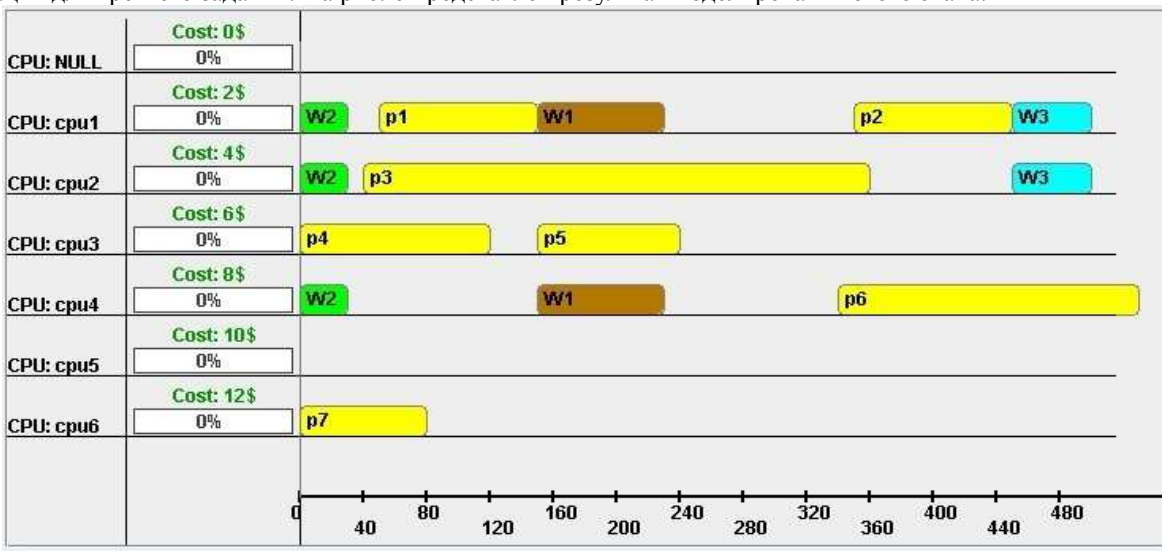


Рис. 6. Результат работы алгоритма поиска слотов

На диаграмме представлены все три найденные альтернативы. Самое раннее подходящее окно для первого задания отмечено именем W1 и выделено на двух процессорных узлах cpu1 и cpu4.

Суммарная удельная стоимость этого окна – 10. Заметим, что возможные окна с более ранним временем старта не подходят по суммарной удельной стоимости: не устраивают пользователя. Самое раннее подходящее окно для второго задания (с учетом альтернативы W1 для первого задания) представлено на диаграмме именем W2. Оно состоит из трех слотов на процессорах cpu1, cpu2 и cpu4 с общей удельной стоимостью 14. Самая ранняя подходящая альтернатива для третьего задания представлена на диаграмме окном W3.

Далее с учетом ранее найденных находятся следующие альтернативные наборы. Сначала для первого задания, а затем для второго и третьего. Результат поиска альтернатив для всех трех заданий представлен на рис. 7.

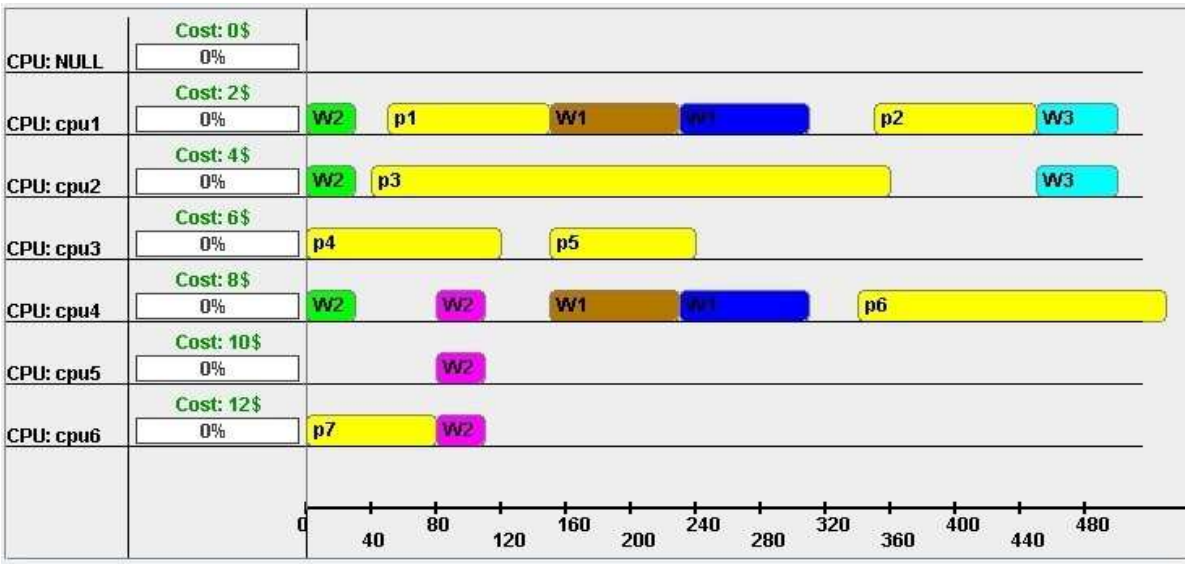


Рис. 7. Второе семейство альтернативных наборов слотов

Наконец рис. 8 демонстрирует весь набор найденных в соответствии с алгоритмом альтернатив для всех трех заданий.

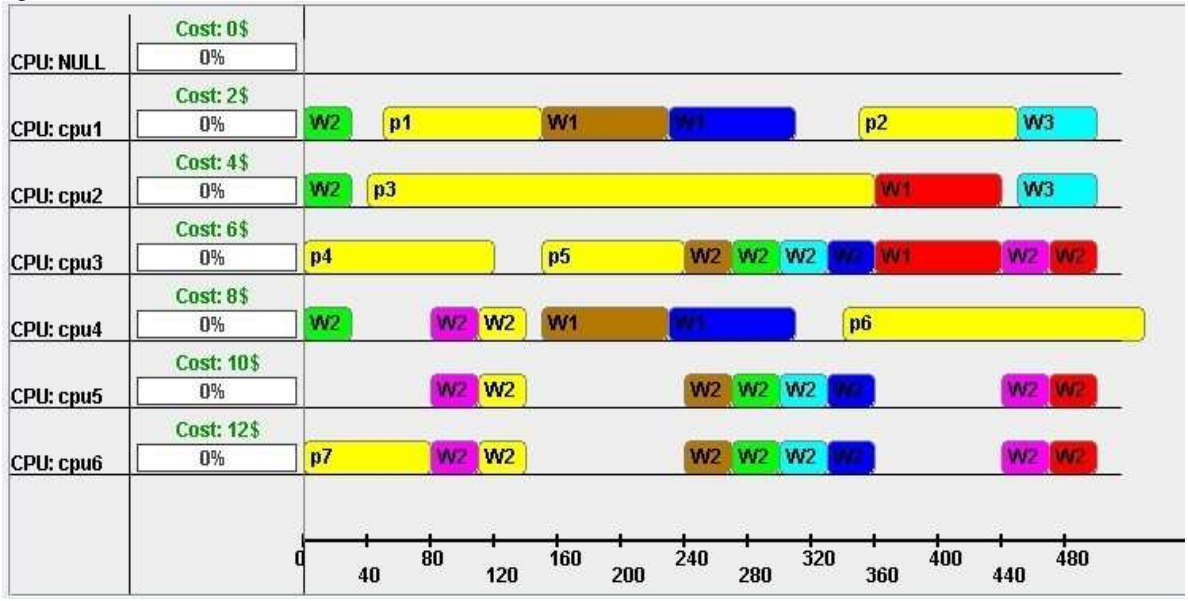


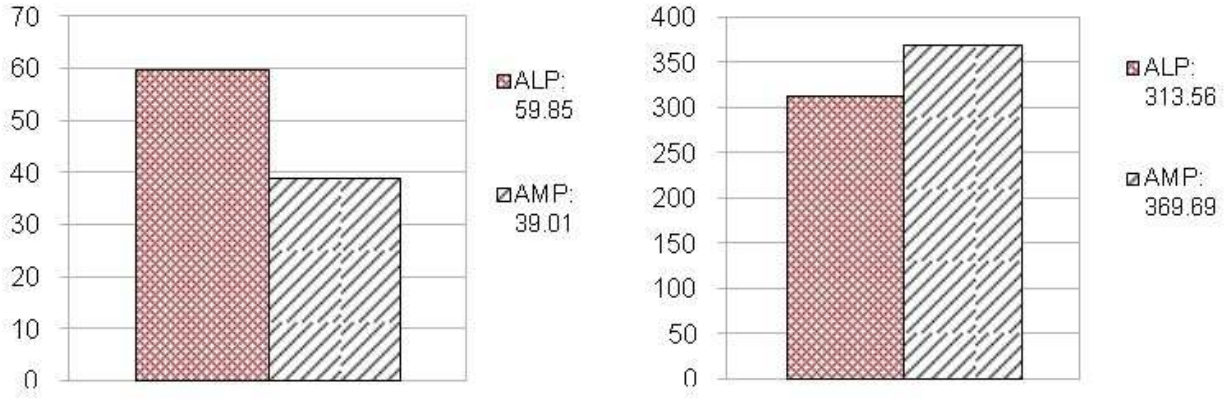
Рис. 8. Результат поиска альтернативных наборов слотов

6. Результаты моделирования циклов планирования

Эксперимент состоит в сравнении результатов планирования пакета заданий на одинаковом наборе ресурсов, но с использованием различных алгоритмов поиска альтернативных наборов слотов ALP и AMP. На каждом цикле планирования происходит генерация исходных данных: списка доступных слотов системы и пакета пользовательских заданий. Рассмотрим задачу поиска альтернативынх наборов слотов при *минимизации суммарного времени выполнения пакета заданий* min T(s) с ограничением на общий бюджет B*.

В общей сложности было промоделировано 25000 циклов планирования. Для подсчета результатов использовались только те 8589 из них, в которых все задачи пакета имели хотя бы одну альтернативу выполнения. При этом среднее количество слотов в отдельном эксперименте составило 135.2. Это число совпадает со средним количеством слотов по всем 25000 экспериментам, что говорит об отсутствии решающего влияния общего количества слотов на результаты поиска альтернатив. Общее количество найденных альтернатив алгоритмом ALP равно 258079 или в среднем 7.39 на задание. Алгоритм AMP при этом нашел 1160029 альтернативы или 34.28 на задание.

По результатам моделирования алгоритм AMP превзошел ALP по целевому критерию минимизации суммарного времени T(s) выполнения пакета заданий на 35%. Среднее время выполнения одного задания при использовании ALP составиоло 59.85, а при использовании AMP - 39.01 (рис. 9 (а)).



(а) (б)

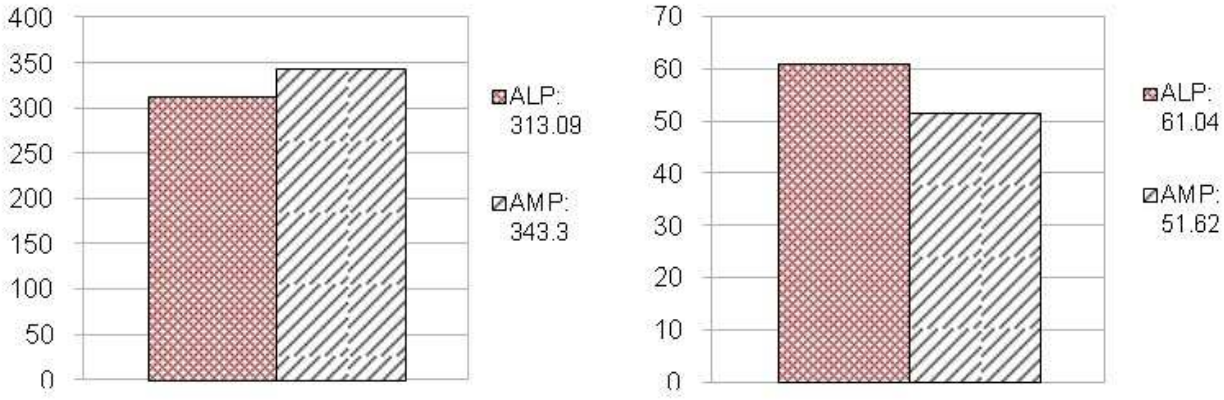
Рис. 9 Минимизация времени выполнения пакета заданий: среднее время выполнения задания (а); средняя стоимость выполнения задания (б).

При этом средняя стоимость выполнения задания с использованием ALP составила 313.56, а для AMP на 15% больше - 369.69 (рис. 9 (б)).

При решении задачи *минимизации суммарной стоимости выполнения пакета заданий***: min C(s) с ограничением на суммарное время выполнения T*, были собраны результаты 8571 отдельных циклов планирования, в каждом из которых все задания имели хотя бы по одной альтернативе выполнения.

Среднее количество слотов в отдельном эксперименте составило 135.11. И снова это число совпадает со средним количеством слотов по всем 25000 экспериментам. Общее количество найденных альтернатив алгоритмом ALP равно 253855 или в среднем 7.28 на задание. Алгоритм AMP при этом нашел 1154116 альтернативы или 34.23 на задание.

Средняя стоимость выполнения задания в схеме с использованием алгоритма ALP составила 313.09, а при использовании AMP - 343.3, что демонстрирует преимущество алгоритма ALP в 9% (рис. 10 (а)). Более высокая стоимость объясняется тем, что алгоритм AMP стремится использовать весь доступный бюджет задания и в среднем находит более дорогие по сравнению с алгоритмом ALP альтернативы. Среднее же время выполнния задания для ALP составило 61.04, а с использованием AMP - 51.62, что на 15% меньше (рис. 10 (б)).



(а) (б)

Рис. 10. Минимизация стоимости выполнения пакета заданий: средняя стоимость выполнения задания (а); среднее время выполнения задания (б).

Стоит отдельно отметить, что в ходе проведения эксперимента, алгоритм AMP позволял находить в среднем в 4-5 раз больше альтернатив выполнения для каждого задания, чем ALP, на одном и том же наборе ресурсов. Отчасти этим и объясняется его преимущество: на этапе планирования большее количество альтернатив позволяет выбрать более эффективную их комбинацию.

Заключение

Предложены и рассмотрены алгоритмы поиска альтернативных наборов слотов в контексте модели управления потоками заданий на основе иерархической структуры виртуальной организации распределенной вычислительной среды. Алгоритмы учитывают разнородность среды и действующие экономические механизмы. С помощью проведенного моделирования, были рассмотрены преимущества и недостатки использования этих алгоритмов на этапе поиска подходящих наборов слотов.

Работа выполнена при содействии Совета по грантам Президента РФ для поддержки ведущих научных школ (грант НШ-7239.2010.9), РФФИ (проект № 09-01-00095), Минобрнауки в рамках аналитической ведомственной целевой программы «Развитие научного потенциала высшей школы» (проекты № 2.1.2/6718; 2.1.2/13283) и федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009-2013 годы (государственные контракты № П2227; № 16.740.11.0038; № 16.740.11.0516).

ЛИТЕРАТУРА:

1. [Garg et al., 2009] Garg S.K., Buyya R., Siegel H.J. Scheduling parallel applications on utility Grids: time and cost trade-off management, Proceedings of the 32nd Australasian computer science conference (ACSC 2009), Wellington, New Zealand.
2. [Bredin et al., 1999] Bredin J., Kotz D., Rus D. Economic markets as a means of open mobile-agent systems, Proceedings of the workshop “Mobile agents in the context of competition and cooperation (mac3)”, 1999.
3. [Buyya et al., 2002] Buyya R., Abramson D., Giddy J. Economic models for resource management and scheduling in grid computing, J. of concurrency and computation: practice and experience, vol. 14, no. 5, 2002.
4. [Ernemann et al., 2002] Ernemann C., Hamscher V., Yahyapour R. Economic scheduling in grid computing. In: Proceedings of the 8th job scheduling strategies for parallel processing, D.G. Feitelson, L. Rudolph, U. Schwiegelshohn (eds.), Springer, Heidelberg, LNCS, vol. 2537, 2002.
5. [Toporkov, 2009] Toporkov V. Application-level and job-flow scheduling: an approach for achieving quality of service in distributed computing, Proceedings of the 10th international conference on parallel computing technologies, Springer, Heidelberg, LNCS, vol. 5698, 2009.
6. [Kurowski, 2003] Kurowski K., Nabrzyski J., Oleksiak A. et al. Multicriteria Aspects of Grid Resource Management // In: J. Nabrzyski, J.M. Schopf, and J. Weglarz (eds.), Grid resource management. State of the art and future trends. - Boston: Kluwer Academic Publishers, 2003.
7. [Топорков и др., 2009] Топорков В.В, Топоркова А.С., Целищев А.С., Бобченков А.В., Емельянов Д.М. Масштабируемые модели планирования и управления потоками заданий в распределенных вычислениях // Науч-ный сервис в сети Интернет: масштабируемость, параллельность, эффективность: Труды Всероссийской суперкомпьютерной конференции. - М.: Изд-во МГУ, 2009.