СПРАВЕДЛИВОЕ РАЗДЕЛЕНИЕ РЕСУРСОВ В ЗАДАЧЕ ПЛАНИРОВАНИЯ ПАКЕТА ЗАДАНИЙ В РАСПРЕДЕЛЕНЫХ СРЕДАХ

В.В. Топорков, Д.М. Емельянов

Предлагается модель планирования потока заданий и справедливого разделения ресурсов в распределенных вычислительных средах, позволяющая учитывать требования пользователей виртуальной организации к эффективности и качеству выполнения своих заданий. С помощью предложенных алгоритмов отбора слотов возможно находить эффективные по заданному критерию альтернативы выполнения для каждого задания пакета.

Введение

Для планирования и эффективного выбора ресурсов в распределенных вычислениях с неотчуждаемыми ресурсами, включая грид и облачные вычисления, широко и эффективно используются экономические модели [1; 2]. Среди различных подходов с применением экономических моделей к организации вычислений в распределенных средах можно выделить *две устойчивых тенденции*.

Одна из них для планирования отдельных заданий использует *брокеры ресурсов* [3; 4], которые, как правило, оптимизируют выполнение конкретного приложения [5].

Другая тенденция связана с образованием *виртуальных организаций* и ориентирована, прежде всего, на грид-системы. При образовании виртуальных организаций [6] осуществляется оптимизация планирования на уровне потоков заданий. Потоки заданий разбиваются на пакеты или на отдельные очереди, план выполнения которых составляется планировщиками локального уровня.

Авторами предлагается концепция планирования, позволяющая повысить эффективность выполнения отдельных заданий при оптимизации выполнения потока заданий в виртуальных организациях.

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

Схема планирования

В настоящей работе предлагается алгоритм, применяемый в рамках модели, в основу которой заложена иерархическая схема управления потоками заданий [1; 2]. В отличие от известных, рассматриваемая модель предполагает группирование заданий в пакеты в соответствии с их свойствами и потребностями в ресурсах, а также цикличное планирование системы заданий на основе динамично обновляемых критериев и ограничений. Доступные ресурсы представляются списком слотов – временных интервалов доступности вычислительных узлов на определенных условиях оплаты. В каждом цикле планирования локальные расписания обновляются, и осуществляется решение двух задач. Во-первых, отбираются подходящие (по ресурсу, цене *Ci*, времени *Ti*) наборы слотов – альтернативы выполнения для каждого задания пакета.

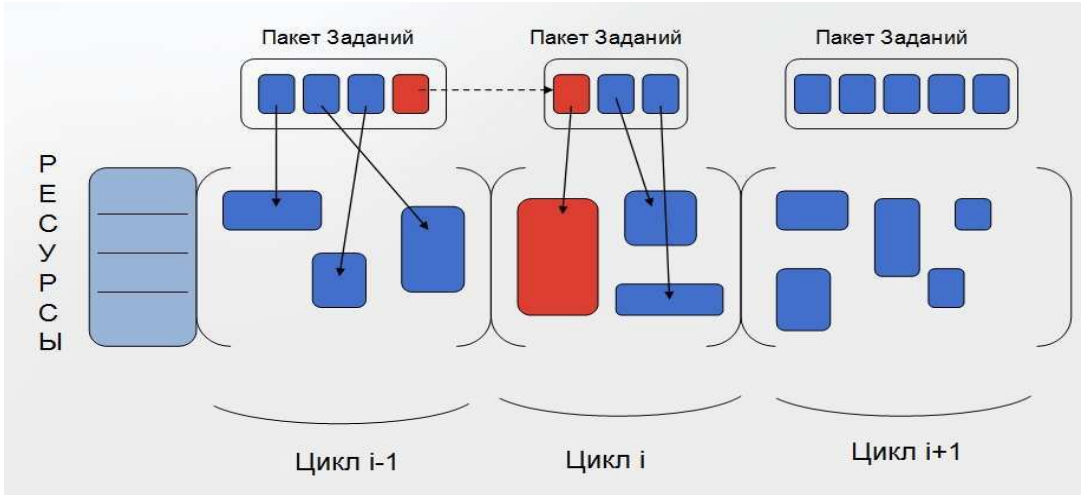

Рис. 1. Планирования потока заданий на основе циклов

Во-вторых, выбирается комбинация альтернатив, эффективная или оптимальная с точки зрения прохождения всего пакета заданий в текущем цикле планирования. Линейные по сложности алгоритмы отбора слотов предложены в [7]. В [4] описана общая схема выбора лучшей комбинации альтернатив.

Поиск подходящих наборов слотов

Для выполнения задания, согласно ресурсному запросу, требуется найти «окно» размером *n*: то есть то есть множество из n слотов, имеющих одинаковое время старта. Время, на которое выделяются ресурсы, определяется пользовательской оценкой времени выполнения. Каждая *альтернатива* (набор слотов) для выполнения задания характеризуется временем *Ta* и стоимостью *Ca*.

В различных системах для диспетчеризации заданий в грид используются различные подходы.

В *планировщике Maui (Moab)*[8], который реализует алгоритм Бэкфиллинга, поиск окон для выполнения заданий происходит без учета аддитивных требований (например, на суммарную стоимость найденного окна, или на минимальный суммарный объем памяти).

Система NWIRE [4] подразделяется на несколько доменов, состоящих из множества локальных ресурсов. Каждый из доменов находится под управлением собственного метаменеджера, выступающего в роли локального планировщика, представителя домена и брокера ресурсов. Метаменеджеры могут взаимодействовать и обмениваться данными посредством общей сети.

Задания от пользователей могут поступать на вход любого из локальных метаменеджеров и, кроме ресурсного запроса, содержат критериальные функции вида *UF = (-StartTime)* (оптимизация по данному приведенному критерию должна позволить выбрать план выполнения задания с минимальным временем старта). Далее метаменеджер генерирует вариант выполнения задания на базе ресурсов собственного домена, а также посылает данные задания связанным метаменеджерам. Связанные метаменеджеры также генерируют альтернативы выполнения, которые поступают первому метаменеджеру. Из них выбирается наиболее подходящий по заданному критерию и другим показателям, в том числе экономическим возможностям пользователя.

Следует отметить, что на этапе поиска альтернатив в домене, *не учитывается ограничение на общую стоимость*. Выборка множества предложений из найденного окна происходит по эвристике путем *перебора подходящих слотов*, а поиск прекращается при достижении заданного необходимого количества альтернатив. Оптимизация происходит на этапе выбора лучшей из предложенных альтернатив.

Нами предлагается общий вид алгоритма поиска окна, оптимального по заданному критерию с учетом заданных ограничений.

Общий вид поиска альтернативного набора слотов

Положим, согласно ресурсному запросу, требуется найти «окно» из *n* слотов, имеющих одинаковое время старта. При этом длина каждого из *n* слотов определяется производительностью того ресурса, на котором он выделен. В результате мы получим «окно» с неровным правым краем (рис. 2). В ресурсном запросе указывается требуемое время выделения ресурсов, необходимые характеристики узлов, а также ограничение на максимальную стоимость выделения всего «окна». Задается *критерий*, по которому будет выбираться наиболее подходящее для конкретного задания «окно», например, стоимость или время выполнения или потребление электроэнергии.

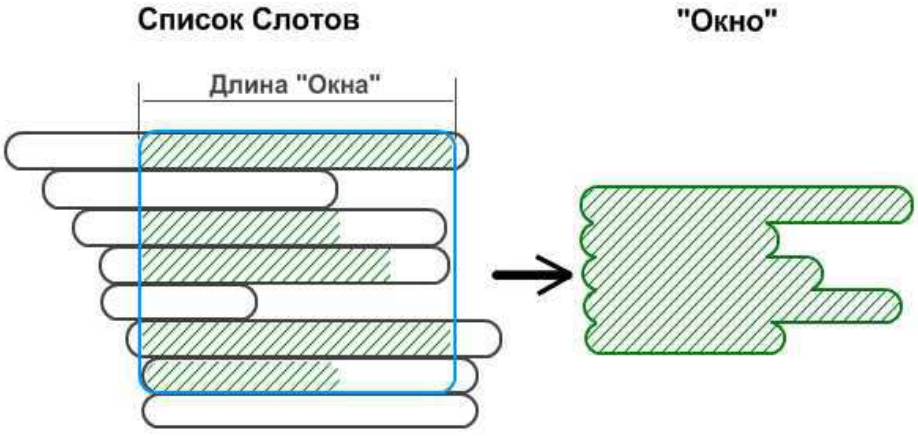

Рис.2. «Окно» с неровным правым краем на множестве слотов

Поиск «окна» осуществляется на списке всех доступных слотов, упорядоченных по неубыванию времени старта. Это условие необходимо для поочередного просмотра всего списка слотов и построения алгоритмов линейной сложности [7] .

Схема поиска «окна», удовлетворяющего требованиям и эффективного по заданному критерию, может быть представлена следующим образом.

1. Из упорядоченного списка слотов выбирается очередной подходящий слот и добавляется в список слотов «окна».
2. Время старта «окна» полагается равным времени старта последнего из добавленных слотов.
3. Происходит проверка, хватает ли длины слотов, находящихся в «окне» для выполнения части задания, с учетом нового, более позднего времени старта. Слоты, время действия которых истекает, удаляются из «окна» и исключаются из дальнейшего рассмотрения.
4. Если количество слотов $m > n$, то далее требуется выбрать n слотов, эффективных по заданному критерию [1], но в тоже время удовлетворяющих ограничению на суммарную стоимость. Предположим было выбрано «окно» W со значением целевого критерия crW. Задача выбора эффективного «окна», состоящего из n слотов в случае, если $m > n$ будет описана ниже.
5. Значение целевого критерия crW найденного «окна» W сравнивается со значением cr' – текущим лучшим значением целевого критерия для всех найденных ранее «окон». В случае, если $cr' > crW$ (при решении задачи минимизации), окно W объявляется новым кандидатом, а crW становится новым лучшим значением критерия: $cr' = crW$.
6. Алгоритм заканчивает работу после рассмотрения последнего слота. Результатом является «окно»-кандидат с наилучшим значением целевого критерия, оно объявляется эффективным по заданному критерию.

Рассмотрим задачу выбора окна из n слотов с общей стоимостью не более n на списке из m > n слотов (для случая m = n выбор тривиален). Текущее расширенное «окно» состоит из m слотов. Стоимость использования каждого из слотов с учетом их необходимой длины: c1, c2,…,cm. Предположим, что слоты имеют характеристику zi, суммарное значение которой, согласно целевому критерию, необходимо минимизировать в итоговом «окне».

Тогда задачу можно сформулировать следующим образом:

$$a_1 z_1 + a_2 z_2 + \ldots + a_m z_m \to \min$$

при ограничениях:

$$a_1 c_1 + a_2 c_2 + \ldots + a_m c_m \le S, \quad a_1 + a_2 + \ldots + a_m = n,$$

$$a_r \in \{0,1\}, r = 1,\ldots, m .$$

Коэффициэнты ai будут определять выбранное «окно».

Описанная схема поиска подходящей альтернативы выполнения обладает линейной сложностью $O(k)$ относительно общего количества слотов системы k: алгоритм двигается по списку слотов в направлении неубывания времени старта, без возвращения назад и пересмотра предыдущих шагов. Стоит отметить, что решение представленной задачи оптимизации является достаточно трудоемким, а необходимость выбора окна на каждом шаге представленного алгоритма, свидетельствует о его еще большей сложности. При большом количестве слотов время его работы может оказаться неадекватным. Однако, можно выделить некоторые часто встречающиеся задачи оптимизации, сложность которых возможно уменьшить. К ним относятся задачи *минимизации стоимости (MinCost) и минимизации общего времени выполнения задания (minRuntime), поиск окна с минимальным временем старта (minStart, AMP[]), поиск окна с минимальным временем завершения (minFinish).* Экспериментальные данные подтверждают, что время работы этих алгоритмов превышают время выполнения алгоритма AMP [7] не более, чем на порядок.

Результаты моделирования

Использование алгоритмов поиска альтернативных наборов слотов эффективных по заданному критерию на этапе поиска множества альтернатив выполнения не может гарантировать повышения эффективности итогового распределения для каждого задания. В общем случае задание может иметь несколько альтернатив выполнения, некоторые из которых были получены в условиях ограниченных доступных ресурсов и не являются эффективными. В то же время алгоритм выбора лучшей комбинации альтернатив для повышения эффективности выполнения всего пакета заданий выбирает альтернативы вне зависимости от их эффективности по локальному пользовательскому критерию. Однако, можно показать, что в среднем, для каждого задания будут выбираться более эффективные варианты выполнения, что обеспечит в среднем.

Авторами предлагается схема планирования пакета независимых заданий с учетом требований пользователей к эффективности выполнения отдельных заданий и справедливого разделения ресурсов. Справедливое разделение ресурсов в рамках виртуальной организации осуществляется алгоритмом выбора

лучшей комбинации альтернатив (AS, BS). Примером задачи оптимизации в таком случае может служить минимизация общей стоимости выполнения пакета заданий при заданном ограничении на общее время выполнения. Повышение эффективности выполнения отдельных заданий достигается с помощью использования описанных алгоритмов поиска «окон», эффективных по заданному пользовательскому критерию, на этапе поиска альтернативных наборов слотов. Каждый пользователь виртуальной организации имеет собственные приоритеты и требования к выполнению своего задания, поэтому в общем случае все задания пакета могут иметь различные критерии поиска альтернатив выполнения (рис. 3).

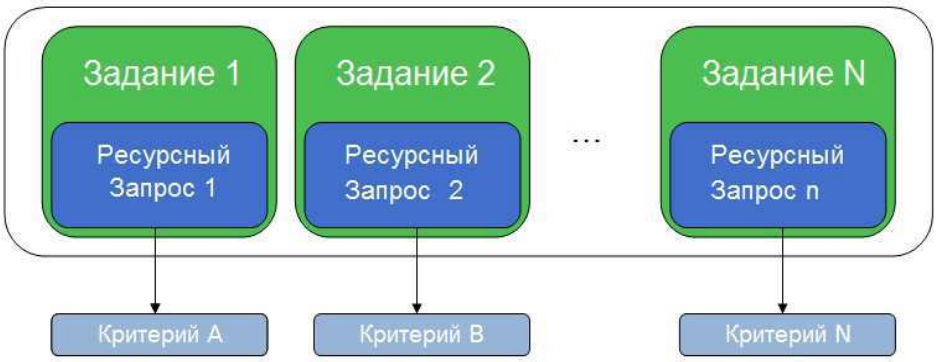

Рис. 3. Пакет заданий с заданными критериями поиска альтернативных наборов слотов

Для демонстрации идеи проведен эксперимент, моделирующий более 4000 независимых циклов планирования. На модельном цикле планирования происходила генерация пакета из 15 заданий со случайными критериеми оптимизации на этапе поиска альтернатив. Были использованы следующие типы критериев: минимизации стоимости (MinCost), минимизации времени завершения (MinFinish), минимизации времени выполнения (MinTime) и минимизации времени старта (AMP) [10, 11]. При этом для всего пакета заданий решалась задача минимизации стоимости с ограничением на общее время выполнения. В таблице 1 представлены средние результаты планирования заданий, сгруппированные по использовавшемуся пользовательскому критерию.

Таблица 1.

Характеристика	**AMP**	**Min Cost**	**Min Finish**	**Min Runtime**
Среднее количество заданий	5,4	5,1	5,2	5,1
Среднее время выполнения	66,7	70,4	52	44,2
Среднее процессорное время	188,7	193,8	169	152,1
Средняя стоимость	733	599,4	727,7	728,8
Среднее время старта	191,9	259,3	187,3	202,9
Среднеей время з	258,6	329,7	239,3	247,1

В таблице 2 представлены результаты планирования всего пакета заданий согласно описанной схеме с оптимизацией выполнения отдельных заданий (Extreme) и планирования без оптимизации, то есть использованием алгоритма AMP для всех заданий пакета (Normal).

Таблица 2.

Характеристика	**Extreme**	**Normal**
Количество Экспериментов	4204	4204
Количество заданий в пакете	15	15
Среднее количество альтернатив на задание	6,9	6,7
Среднее время выполнения	57,8	62,8
Среднее процессорное время	175	178,5
Средняя стоимость	697,8	703,2
Среднее время старта	222	231,5
Среднее время завершения	279,9	294,3

Из анализа данных в таблицах 1 и 2 следует, что использование критериев на этапе поиска альтернатив позволяет заметно повысить эффективность выполнения отдельных задания по заданному пользовательскому критерию (преимущество над алгоритмом AMP составляет от 14% до 30%). При этом отклонения в результатах планирования пакета заданий в целом незначительны (до 8%) и даже эффективнее по заданному критерию минимизации стоимости. Эти результаты позволяют говорить о возможности повышения эффективности выполнения отдельных задания (согласно пользовательским критериям) с помощью предложенных алгоритмов поиска окон при практически неизменной эффективности выполнения всего пакета заданий (согласно политикам, принятым в виртуальной организации). Кроме того, из результатов следует, что администраторы виртуальной организации могут сами дополнительно влиять на эффективность прохождения пакета заданий, принудительно назначая критерии поиска альтернативных наборов слотов для повышения общей эффективности (для этого, например, можно использовать одинаковый критерий поиска альтернатив для всех заданий пакета).

Заключение

Предложена модель планирования потока заданий и справедливого разделения ресурсов в распределенных вычислительных средах, позволяющая учитывать требования пользователей виртуальной организации к эффективности и качеству выполнения своих заданий. Представлена и обоснована общая схема алгоритма поиска альтернативных наборов слотов, эффективных по заданному критерию. Рассматриваемая модель, с помощью комбинирования оптимизационных критериев позволяет повысить эффективность выполнения отдельных пользовательских заданий при соблюдении политик, принятых в виртуальной организации.

Работа выполнена при частичном содействии Совета по грантам Президента Российской Федерации для поддержки ведущих научных школ (шифр НШ-316.2012.9), Российского фонда фундаментальных исследований (проект № 12-07-00042), Министерства образования и науки Российской Федерации в рамках федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009-2013 годы (государственные контракты № 16.740.11.0038 и 16.740.11.0516).

ЛИТЕРАТУРА:

1. 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 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 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 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 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 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. Toporkov V, Yemelyanov D, Toporkova A, Bobchenkov A (2011) Resource co-allocation algorithms for job batch scheduling in dependable distributed computing. Dependable Computer Systems. Springer-Verlag, AICS. V. 97. Berlin, Heidelberg: 243-256.
8. Интернет-ресурс www.clusterresources.org (разделы по использованию алгоритма Backfill в системе планирования потоков заданий MAUI, справка для администраторов систем планирования Moab и MAUI по настройкам алгоритма Backfill), 2012.