МОДИФИКАЦИЯ МЕТОДА КРИТИЧЕСКИХ РАБОТ С УЧЁТОМ ДИНАМИЧЕСКИ ИЗМЕНЯЕМОГО СОСТАВА РЕСУРСОВ В РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЯХ

А.С. Целищев, В.В. Топорков

Задача оптимального планирования вычислений в распределенных системах на сегодняшний день не имеет универсального подхода к своему решению. Известные на сегодняшний день алгоритмы и их комбинации, используемые в конечных планировщиках, в большинстве случаев не позволяют получать планы, оптимальные с точки зрения поставленных условий, так как частично или полностью не учитывают разнообразные, сопряженные с вычислениями факторы. Предлагаемый авторами комплексный подход к решению задачи планирования основывается на экономических моделях поддержки политики предоставления и потребления ресурсов в виртуальных организациях, а также анализе свойств заданий, подлежащих планированию (ресурсных запросов, структуры, статистики загрузки процессоров и потребности в данных), причем собственно планирование и коаллокация заданий реализуются с помощью обновляемых стратегий (списков возможных времен запусков и назначений задач) в соответствии с динамикой загрузки, освобождения, резервирования неотчуждаемых ресурсов и локальными политиками обработки заданий в доменах узлов. В данной статье описаны основные моменты модернизации разработанного и используемого авторами метода критических работ при применении его в планировании распределённых вычислений с учётом динамики загрузки вычислительных узлов, а также анализ качества обслуживания конечных пользователей.

Для подтверждения эффективности предлагаемого подхода авторами была создана среда имитационного моделирования описанных процессов, которая позволила доработать предлагаемые алгоритмы, дополнив их вспомогательными механизмами и эвристиками ([1]), а также собрать и проанализировать статистику исходов для разных стратегий планирования ([2]). На основе полученных результатов предлагаемая концепция [3] была расширена до масштабов планирования на метауровне. Как отмечено в [4], при моделировании на этапе распределения заданий локальными менеджерами ресурсов назначение задач происходило с использованием упрощенной модели (FCFS), что не позволяет говорить о рассмотрении максимально эффективного использования ресурсов. Также стоит отметить, что применение более эффективных методов локального планирования позволило бы представить более объективную картину работы глобального подхода.

Проблема интеграции продвинутых алгоритмов локального планирования заключалась в специфике лежащего в основе подхода метода критических работ, который рассматривает упрощённую ситуацию наличия неограниченного числа ресурсов заданного *типа*, не беря в расчёт вероятную занятость конкретного *экземпляра* другими задачами. В реальной ситуации планировщик должен учитывать информацию об ограниченности ресурсов, а выделяемый набор вычислительных узлов не всегда будет доступен в течение всего времени, отведённого на выполнение задания. Исходя из этого, авторами было принято решение модифицировать среду моделирования и дополнить методы планирования дополнительными функциями, которые позволят *динамически* выделять ресурсы и проводить планирование с использованием уже имеющихся наработок. Для решения этой задачи для начала необходимо хранить и динамически обновлять информацию о вычислительных узлах и интервалах их занятости и учитывать эти данные при любом назначении и переназначении задач в рамках планирования и разрешения коллизий. Только лишь внеся модификации в *реализованные* алгоритмы, можно сохранить близкие к оптимальным расписания, которые учитывают структуру заданий и дополнительные критерии. Это обусловлено тем, что в системе с частично занятым набором ресурсов построенное оригинальным, немодифицированным методом расписание неизбежно подвергнется временным сдвигам и уже не будет соответствовать полученным ранее модулем планировщика значениям критериальных функций. Для визуальной оценки допустимости построенных планов в среду имитационного моделирования был введен новый вид временных диаграмм, отображающих занятость каждого конечного узла с течением времени.

Диаграмма, представленная на рис. 1, показывает выполнение двух заданий-графов (каждое из которых состоит из подзадач-вершин и информационных связей-дуг) на абстрактном фиксированном наборе локальных ресурсов CPU0-CPU8, причём интервалы выполнения заданий не пересекаются (первый граф выполняется во временном интервале (0;110), второй – (120;270), ось абсцисс), что обеспечивает *полностью* свободный набор ресурсов при поступлении нового задания-графа. Каждая строка в диаграмме показывает интервалы занятости соответствующего процессора соответствующими задачами в течение заданного интервала. Отметим, что вид диаграммы близкого к оптимальному расписания для каждого задания обусловлен структурой этого задания и информационными связями между составляющими его задачами (см. левую часть рис. 1), а само построенное расписание обеспечивает близкое к оптимальному значение выбранной критериальной функции (в данном случае — минимальной стоимости выполнения всего задания).

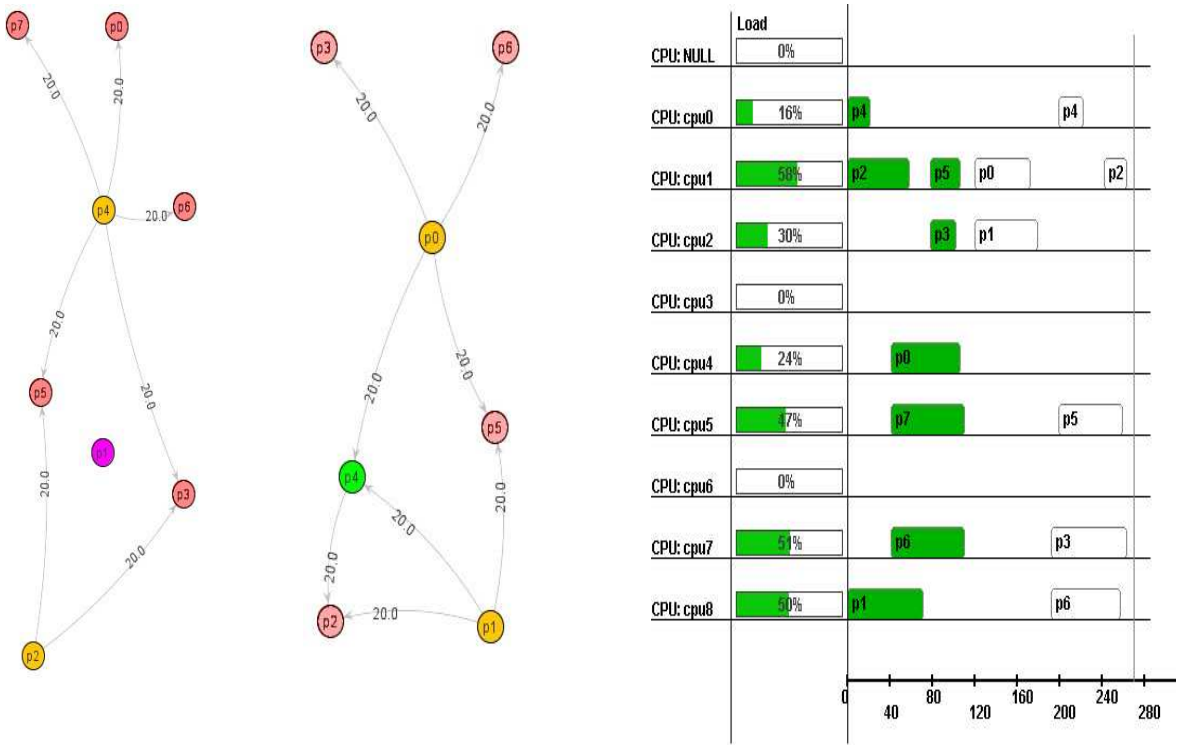

Рис. 1 Диаграмма двух непересекающихся заданий

Очевидно, что в этом случае при поступлении в систему нового задания полученный для него близкий к оптимальному план будет успешно выполнен на полностью свободном наборе ресурсов, так как подобная ситуация совпадает с условиями метода критических работ. Если, однако, второе задание поступит *раньше* окончания выполнения первого, то в предложенной модели подзадачи *нового* задания будут выполняться лишь после освобождения ресурса подзадачами *предыдущего* задания, даже несмотря на то, что занятость ресурса далека от ста процентов. Так, например, из рис. 1: CPU2 занят лишь в интервале (80,100), и при поступлении нового задания в момент времени 0 локальный планировщик распределит задачу на этот ресурс только в момент времени 100, что вызовет нежелательный простой узла, обеспечит более позднее время высвобождения ресурса и ухудшит общие показатели планирования. Логично рассуждать, что поступление нового задания-графа в реальной системе может произойти в любой момент, более того, не исключена и ситуация одновременной подачи нескольких заданий, поэтому предлагается ввести следующие изменения в работу системы:

- Вместе с данными об информационных связях и наборе ресурсов подавать модулю планировщика времена освобождения каждого ресурса (в общем случае массив слотов – интервалов простоя ресурса или массив интервалов занятости).
- При рассмотрении планировщиком ситуации назначения задачи на определённый ресурс динамически формировать список доступных ресурсов.
- При разрешении коллизий за ресурсы динамически формировать список доступных ресурсов.
- С течением времени на каждом шаге моделирования обновлять информацию о текущем состоянии ресурсов.

Более подробного пояснения требует пп. 2, так как назначение и просчёт критериальных функций проводится в нетривиальных динамических циклах. При рассмотрении ситуации, в которой часть ресурсов занята под ранее распланированные задачи, модель среды метапланировщика становится динамической. Лежащий в основе работы алгоритм условной оптимизации изначально был предназначен для статической модели, не учитывающей начальное состояние системы. При этом количество единиц каждого ресурса принималось бесконечным. Внося же изменения в механизм вычисления критериальной функции на этапе *планирования*, можно перейти от рассмотрения методом критических работ не типов, а конкретных экземпляров ресурсов, получая, таким образом, близкое к оптимальному расписание для реального набора вычислительных узлов.

Для реализации описанной идеи было необходимо ввести временной курсор для вычисления начального момента планируемой задачи. Так как заполнение таблиц дерева поиска значениями критериальной функции производится на этапе "обратной прогонки" ([6]) в направлении от последней задачи критической работы к первой, то разумно в качестве временного курсора использовать запас времени на текущую задачу и следующие за ней (запас является собственным параметром каждой таблицы в дереве поиска). Запас времени в

данном случае рассматривается как отрицательное смещение относительно момента окончания критической работы, который нетрудно вычислить: либо он соответствует всему масштабу планирования для главной критической работы, либо определяется временем уже распределенной последней задачи в критической работе. При расчете начального момента планируемой задачи необходимо принимать во внимание все времена передачи, так как из-за особенностей алгоритма они не учитываются при поиске и перед планированием вычитаются из запаса на критическую работу.

Далее, при расчете критериальной функции необходимо найти оптимальный с точки зрения критерия ресурс, например, самый «медленный», исполняющий задачу не медленнее заданного времени T. При этом вводится дополнительный параметр $t_{нач}$, определяющий планируемый начальный момент задачи и производятся следующие проверки:

- Ресурс не занят в момент времени $t_{нач}$
- Ресурс не занят в момент времени $t_{нач} + T$
- Ресурс не занят между моментами времени $t_{нач}$ и $t_{нач} + T$

Данные об интервалах занятости ресурса, как уже было сказано, идут непосредственно на вход планировщика. В том случае, когда для варианта распределения задачи не находится подходящего ресурса по условиям критериальной функции, её значение устанавливается ошибочным, и, таким образом, оно не участвует в поиске условного оптимума.

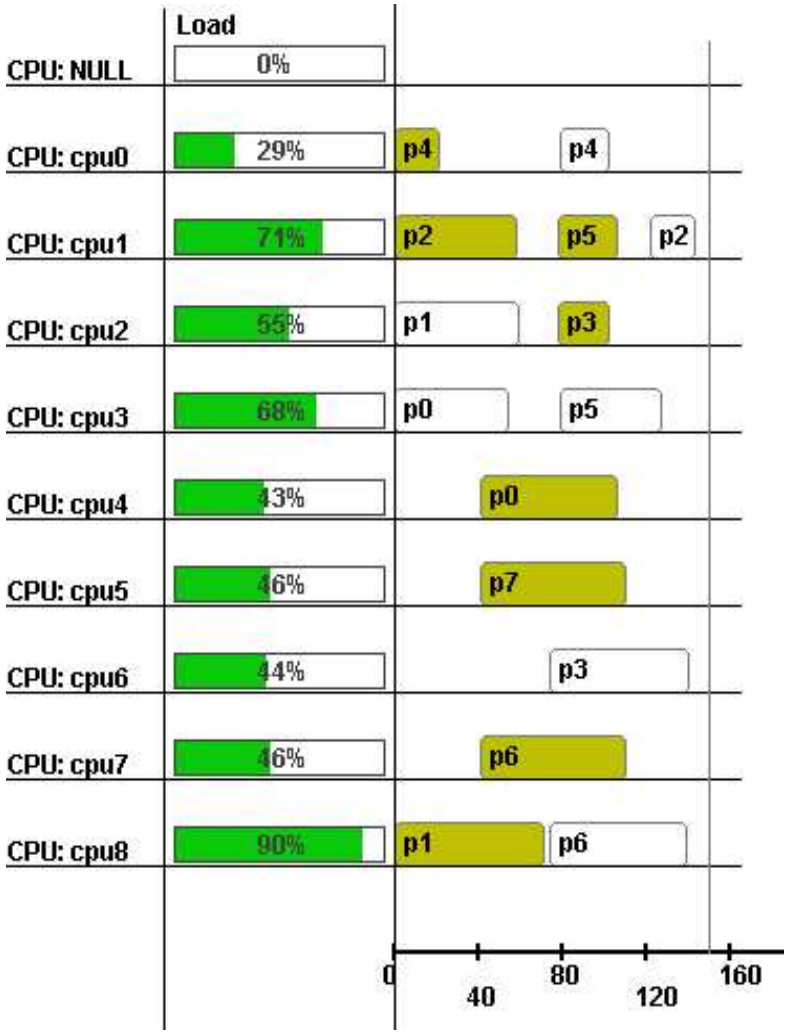

Рис. 2. Диаграмма двух пересекающихся заданий

Последняя версия среды моделирования метапланировщика, разрабатываемого авторами, поддерживает возможность получения и обработки заданий-графов в *любой* момент времени и осуществляет планирование с учётом информационных связей и доступности ресурсов по описанной методике. Возвращаясь к рис. 1, можно рассмотреть ситуацию, когда второе задание подаётся сразу после распределения первого (рис. 2). При применении модифицированного метода критических работ стало возможно проводить составление расписаний с учётом занятости ресурсов и, таким образом, уплотнять общее время выполнения потока заданий и увеличивать общее время занятости ресурсов.

Важно понимать, что косвенно реализованный здесь механизм backfilling'a (из рис. 2 видно, что задачи второго задания заполняют интервалы простоя ресурсов) не стоит путать с его классическим пониманием как

алгоритма планирования, в котором он используется в существующих планировщиках. Так, например, планировщик Maui [5] использует backfilling в локальных масштабах, но поддерживает описание составного задания только в виде простой *очереди* однородных подзадач с возможным назначением приоритетов (то есть не рассматривает более сложные по *структуре* задания) и не осуществляет планирование с учётом возможных требований минимизации простоя ресурсов или оптимального их использования. В предлагаемом подходе получающийся по факту backfilling является результатом учёта структуры заданий, разнотипности их подзадач, особенностей политики выделения ресурсов и хранения данных в распределенной неоднородной среде с динамично изменяющимся составом вычислительных узлов, на которых выполняются эти задачи, а также других необходимых введенных критериев. Ни один из существующих планировщиков на сегодняшний день не обеспечивает соблюдения подобного набора факторов при планировании вычислений в реальном времени.

В настоящий момент проводится заключительный этап моделирования работы с потоками случайно генерируемых сценариев распределённых вычислений. Имитационная модель позволяет формировать поток заданий от пользователей. Каждое из заданий представляется соответствующим параметризованным графом. Степень информационной связности вершин-задач может варьироваться. Число базовых процессорных узлов принимается кратным максимальной ширине яруса графа. Оценки длительности выполнения задач, объемов вычислений, времени доступа к данным – целые случайные числа, равномерно распределенные в заданном диапазоне. Конфликты между задачами, конкурирующими за один и тот же узел, разрешаются за счет незадействованных базовых узлов так, что обеспечивается минимум функции штрафа – например, суммы отношений объемов вычислений к оценкам длительности выполнения задач.

Программная модель позволяет выбирать описанные параметры случайных сценариев, критерии составления расписаний, функции штрафа, а так же *частоту* подачи заданий. В качестве результатов моделирования на выходе отображаются временные диаграммы загрузки ресурсов и распределения критических работ сценария, а также подробная информация о встреченных коллизиях и ходе их разрешения. После успешных результатов моделирования разработанных алгоритмов и концепций было принято решение переработать функциональную часть планировщика и подготовить ядро для программно-аппаратной интеграции решения в реальных условиях. Для этого было проведено разделение модулей, полный рефакторинг и профайлинг кода проекта, что привело к увеличению производительности планировщика более чем в пять раз. Разработанная система моделирования позволила оценить качество обслуживания конечных пользователей, а также создать механизмы априорной оценки вероятности успеха выполнения задания на заданном наборе ресурсов за заданное время. Так, при проведении исследований по формированию стратегий ([2],[4]) для анализа принимались во внимание лишь случаи успешно составленных, близких к оптимальным расписаний с разрешёнными коллизиями, тогда как статистика причин невозможности составления расписаний не рассматривалась. В то же самое время по результатам моделирования успешно составлялось лишь порядка трети планов. Новые модули мониторинга позволили проанализировать статистику исходов работы планировщика, загрузки ресурсов, а также сформировать эмпирические требования к ресурсам со стороны поступающего задания. Далее эти результаты рассмотрены подробнее.

Временной диапазон, в рамках которого можно составить расписание для сложноструктурированной задачи, можно определить как $T = [0;[T_{K1min} ; \Sigma\ t_j]]$, где T_{K1min} – минимальное время выполнения самой длинной (по сумме весов вершин и рёбер графа) критической работы на определённом наборе ресурсов, а $\Sigma\ t_j$ - сумма времён выполнения всех подзадач. Иными словами, при отсутствии параллелизма время выполнения всего сценария равно сумме времён выполнения каждой из подзадач (верхняя граница), а при максимально возможном параллелизме время выполнения сценария ограничено длиной максимальной критической работы (нижняя граница), то есть самым длинным последовательным участком графа. Дополнительным подтверждением процента успешно спланированных сценариев из [2] послужила новая статистика, отражающая тот факт, что при назначении диапазона планирования одного сценария, равному сумме весов вершин (соответствующих временам выполнения подзадач на определённых ресурсах) и рёбер самого длинного последовательного участка графа, успешно и оптимально распределяется около 35% сценариев, 29% сценариев не имеют оптимального плана (но для них существует план, близкий к оптимальному), а для остальных 35% корректных планов не существует. Проблема выбора интервала планирования вызвана тем, что для случайно генерируемых сценариев сложно дать оценку необходимого времени планирования ввиду абстрактности самого сценария, тогда как в реальной ситуации это прерогатива пользователя, инициирующего задание. Таким образом, предлагаемый подход позволит сразу отклонить запросы пользователей с заведомо некорректными требованиями к интервалу выполнения сценария (в том случае, когда предлагаемый интервал будет короче $[0; T_{K1min}]$), а также дать вероятностную оценку успеха распределения ещё до его начала. На рис. 3 представлена статистическая оценка исхода планирования для интервала, равного $T = [0;[n*T_{K1min}]]$ для n=1, 1.33, 1.66 (10000 прогонов для каждого n)

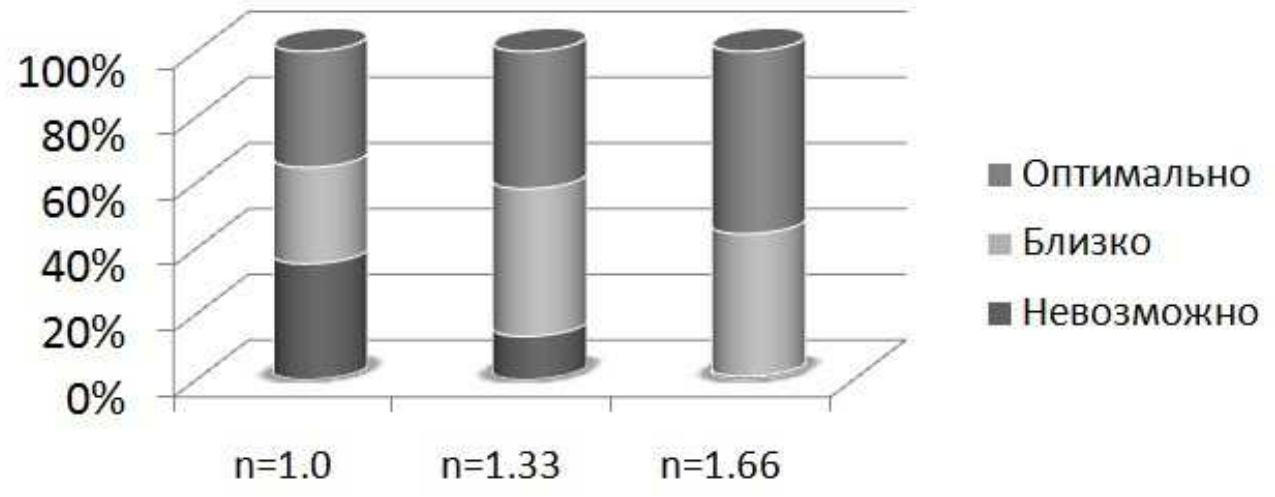

Рис. 3. Статистика распределения для разных интервалов

Таким образом, эксперимент показал, что при длине интервала в 1.66 раз больше, чем длина максимальной критической работы графа, вероятность неудачи при планировании менее 0.01. При этом в 55% случаев планировщик выдаёт план, близкий к оптимальному, обеспечивающий минимальный простой ресурсов и минимальную стоимость их использования (возможно использование и других критериев).

При успешном составлении расписания подзадачи каждого сценария конкурируют за один и тот же ресурс в среднем 2.25 раза (1800 коллизий на 800 сценариев). На рис. 4 приведена статистика исходов комплексного алгоритма разрешения коллизий при числе доступных процессоров, равному максимальной ширине ярусов исходного графа (то есть числу вершин максимальной критической работы), равному двухкратному и трёхкратному числу (по 10000 прогонов).

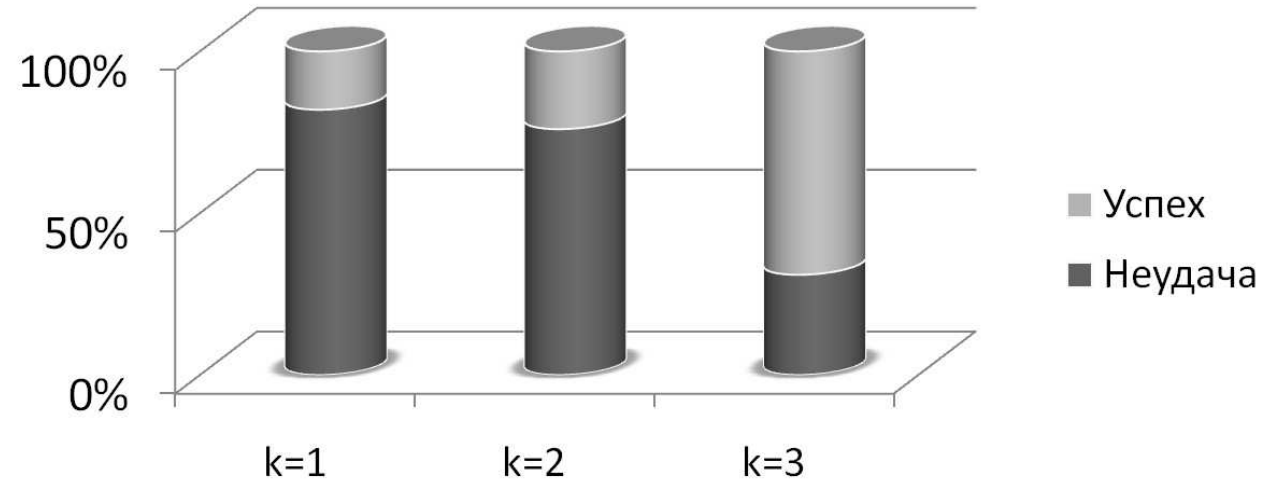

Рис. 4. Статистика распределений с различным количеством ресурсов

Как видно из диаграмм, при наличии свободных процессоров в количестве, втрое превышающем максимальную ширину яруса графа, коллизии не могут быть разрешены лишь в 18% случаев. Дальнейшие исследования результатов показали, что в большинстве случаев отсутствие в графе изолированных вершин (то есть подзадач без информационных зависимостей) позволяет сократить число коллизий почти вдвое. Очевидный вывод - вычисления, соответствующие изолированным вершинам, можно передавать на системы без параллелизма, так как они являются наиболее частыми участниками коллизий (под такую вершину планировщик выделяет всё время планирования и, назначая её на самый медленный процессорный узел, вынуждает её участвовать в коллизиях). Очевидно, что не для каждого сценария возможно составить близкое к оптимальному расписание и обеспечить его выполнение, однако предлагаемые подходы позволяют вплотную подойти к решению этой задачи, а также дать априорные оценки вероятности успешного распределения. На текущий момент примерно в половине рассмотренных случаев (47%) планировщик предлагает близкий к оптимальному план с разрешёнными возможными коллизиями. В настоящий момент идёт оптимизация работы

планировщика с неоптимальными планами и стратегиями выбора планов, близких к оптимальным, а также подготовка к моделированию и тестированию практической части проекта в реальных условиях.

Таким образом, полученные нами результаты позволяют считать, что описанная разработка вплотную подходит к решению задачи создания планировщика, который позволит составлять близкие к оптимальным планы для связанных по данным сложно структурированных сценариев распределённой обработки данных с учётом динамического состава ресурсов.

На сегодняшний день, когда важность эффективного использования распределённых ресурсов необычайно высока, очевидно, что успешное завершение разработки авторов позволит создать мощный инструмент, позволяющий избежать значительных экономических потерь – как временных, так и финансовых - в самых различных областях деятельности.

Работа выполнена при финансовой поддержке РФФИ (грант 09-01-00095) и аналитической ведомственной целевой программы «Развитие научного потенциала высшей школы» (проект 2.1.2/6718).

ЛИТЕРАТУРА:

1. В.В Топорков, А.С. Целищев, А.В. Бобченков, П.В. Рычкова “Метапланировщик – генератор стратегий распределённых вычислений“ // Труды XV международной научно-технической конференции "Информационные средства и технологии". В 3 томах. Т. 1. - М.: Издательский дом МЭИ, 2007.
2. В.В Топорков, А.С. Топоркова, А.С. Целищев “Стратегии коаллокаций для решения больших задач в распределенных средах“ // Научный сервис в сети Интернет: решение больших задач: Труды Всероссийской научной конференции (22-27 сентября 2008 г., г. Новороссийск). - М.: Изд-во МГУ, 2008. С. 3-6.
3. V.V. Toporkov, “Multicriteria Scheduling Strategies in Scalable Computing Systems”, Proc. of the 9th Int. Conf. on Parallel Computing Technologies (PaCT) 2007, LNCS, Vol. 4671. Springer-Verlag Berlin Heidelberg, 2007, pp. 313-317.
4. V.V. Toporkov, A.S. Tselishchev, “Safety Strategies of Scheduling and Resource Co-allocation in Distributed Computing“ // Proc. of the 3rd Int. conf. on Dependability of Computer Systems DepCoS-RELCOMEX 2008. 2008. IEEE CS. P. 152-159.
5. http://www.clusterresources.com
6. В.В. Топорков “Модели распределённых вычислений“ // М.: ФИЗМАТЛИТ, 2004. 320с.