ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ №1(32), 2017, с. 63–81

УДК 004.75

О. В. Сухорослов, А. М. Назаренко

Сравнительная оценка методов планирования приложений в распределенных вычислительных средах

Аннотация. Работа посвящена анализу эффективности известных алгоритмов планирования пакетов задач и композитных приложений в распределенных вычислительных средах (РВС). Сравнение алгоритмов производится на основе результатов имитационных экспериментов для различных примеров приложений и конфигураций РВС. Имитационное моделирование позволяет избежать проведения длительных натурных экспериментов и обеспечить воспроизводимость результатов. Помимо полученных результатов описывается используемая имитационная модель (симулятор) РВС на базе платформы SimGrid.

Ключевые слова и фразы: распределенные вычисления, алгоритмы планирования, пакет задач, композитное приложение, workflow, имитационное моделирование.

Введение

Распределенные вычислительные среды (РВС) активно используются для выполнения различного рода приложений, состоящих из набора отдельных вычислительных задач. Выделяют два важных класса данных приложений. В случае если все задачи являются независимыми, говорят о *пакетах задач* (bag of tasks), типичным примером которых являются многовариантные расчеты. В случае наличия зависимостей между задачами, ограничивающих очередность их запуска, говорят о *композитных приложениях* (КП) или workflow.

При планировании выполнения подобных приложений в РВС, заключающемся в назначении отдельных задач ресурсам среды, необходимо удовлетворить определенным требованиям. Примерами подобных требований являются минимизация времени выполнения приложения, соблюдение срока выполнения расчета или минимизация времени выполнения при ограниченном бюджете. Подобного рода

Работа выполнена при поддержке РФФИ (проекты 15-29-07068, 15-29-07043).

задачи планирования, даже в самых простых постановках, представляющих практический интерес, являются NP-полными [1]. Это делает точное решение задачи — выбор строго оптимального плана выполнения – практически недостижимым.

С конца 1990-х годов и по настоящее время было предложено множество приближенных методов планирования приложений в РВС, основанных на различного рода (мета)эвристиках и ориентированных на различные классы приложений и сред [2, 3]. При этом недостаточно хорошо изучен вопрос об области применимости и сравнительной эффективности данных алгоритмов в различных ситуациях, в зависимости от отдельных характеристик как приложений, так и среды. Кроме того, плохо изучены вопросы работы алгоритмов в условиях получения неточных оценок требуемых характеристик задач и среды, что часто имеет место при реальном использовании.

В работе делается попытка сравнительного анализа эффективности известных алгоритмов планирования указанных классов приложений в РВС. Сравнение производится на основе результатов имитационных экспериментов для различных примеров приложений и конфигураций среды. Имитационное моделирование позволяет избежать проведения длительных натурных экспериментов и обеспечить воспроизводимость результатов. В главе 1 дается краткое описание исследуемых алгоритмов. В главе 2 описывается используемая имитационная модель (симулятор) РВС и приложений на базе платформы SimGrid. В главе 3 приводятся результаты нескольких серий имитационных экспериментов. В заключении приводятся основные результаты и планы дальнейших исследований.

1. Планирование приложений в распределенной среде

В данной главе приведено краткое описание исследуемых далее алгоритмов планирования приложений в РВС.

1.1. Алгоритмы планирования пакета задач

Random: Простейший статический алгоритм, использующий случайное распределение задач по ресурсам.

RRobin (Round-Robin)**:** Простейший статический алгоритм, использующий циклическое распределение задач по ресурсам.

OLB (Opportunistic Load Balancing)**:** Динамический алгоритм, распределяющий задачи по ресурсам по мере освобождения последних [4]. В общем случае порядок назначения задач может быть произвольным. В данной работе для обеспечения детерминированности результатов задачи назначаются в порядке их следования в пакете. Преимуществами данного алгоритма являются его простота, адаптивность, отсутствие необходимости оценок времен выполнения задач. Вследствие этого он часто используется в системах распределенных вычислений. Под этим именем известен также статический вариант данного алгоритма, оперирующий оценками времен выполнения задач (в данной работе не рассматривается).

Min-Min: Статический алгоритм, относящийся к эвристикам пакетного планирования [5, 6]. Данные эвристики оперируют так называемой ECT-матрицей, содержащей оценки времени завершения каждой задачи (estimated completion time, ECT) на каждом из ресурсов. На каждом шаге алгоритма для каждой задачи определяется наилучший ресурс, обеспечивающий минимальное время её завершения. После этого в алгоритме Min-Min выбирается задача с минимальным временем выполнения на наилучшем ресурсе и назначается на данный ресурс. В конце шага назначенная задача (строка ECT-матрицы) удаляется, а значения матрицы пересчитываются с учетом выполненного назначения. Данные шаги повторяются до тех пор, пока не будут назначены все задачи.

Max-Min: Статический алгоритм, относящийся к эвристикам пакетного планирования [5, 6]. Данный алгоритм отличается от алгоритма Min-Min только критерием выбора задачи, назначаемой в рамках каждого шага. В алгоритме Max-Min выбирается задача с *максимальным* временем выполнения на наилучшем ресурсе и назначается на данный ресурс. Таким образом, данная эвристика старается как можно раньше планировать задачи с наибольшим временем выполнения.

Sufferage: Статический алгоритм, относящийся к эвристикам пакетного планирования [6]. Данный алгоритм отличается от алгоритма Min-Min только критерием выбора задачи, назначаемой в рамках каждого шага. В алгоритме Sufferage выбирается задача с максимальной разницей времен завершения на двух наилучших ресурсах. Таким образом, данная эвристика отдает предпочтение задачам, время завершения которых может наиболее сильно пострадать в случае альтернативного выбора.

1.2. Алгоритмы планирования композитных приложений

Алгоритмы планирования пакета задач применимы для планирования композитных приложений (КП) с минимальными изменениями.

Random, Round-Robin: Чтобы расписание оказалось выполнимым, необходимо учитывать зависимости между задачами — распределение ресурсов проводится при прямом обходе графа задач.

OLB (Opportunistic Load Balancing)**:** Алгоритм применим к композитным приложениям без изменений. Кроме того, существует множество алгоритмов планирования, разработанных специально для композитных приложений. Приведенные далее алгоритмы являются статическими методами так называемого приоритетного планирования и учитывают полную структуру композитного приложения. Статический подход позволяет повысить эффективность планирования, но вместе с тем делает данные методы чувствительными к точности оценки времени выполнения задачи на ресурсе (EET) и времени передачи данных на ресурс (ECOMT). Если приложение включает в себя длинные последовательные секции, то ошибка оценки будет накапливаться.

HEFT (Heterogeneous Earliest Finish Time)**:** Один из наиболее цитируемых в литературе алгоритмов планирования [7]. Задачи ранжируются в порядке убывания критерия

$$\mathrm{rank}(T_a) = \overline{\mathsf{EET}}(T_a) + \max_{T_b \in \{\text{доч. задачи } T_a\}} \left(\overline{\mathsf{ECOMT}}(c_{ab}) + \mathrm{rank}(T_b)\right),$$

после чего каждая задача назначается на ресурс с минимальным значением ECT. Отметим важную особенность функции ранжирования — родительская задача всегда имеет более высокий приоритет, чем все ее дочерние задачи, что позволяет вычислять ECOMT.

HCPT (Heterogeneous Critical Parent Trees)**:** Данный алгоритм реализует альтернативный подход к ранжированию задач [8]. Для каждой задачи вычисляется два критерия. Первый критерий — усредненное минимальное время старта задачи

$$\mathsf{AEST}(T_a) = \max_{T_b \in \{\text{род. задачи } T_a\}} \left(\overline{\mathsf{ECOMT}}(c_{ab}) + \overline{\mathsf{EET}}(T_b) + \mathsf{AEST}(T_b)\right).$$

Данный критерий вычисляется при прямом обходе графа задач. AEST входной задачи полагается равным нулю. Второй критерий — усредненное максимальное время старта задачи

$$\mathsf{ALST}(T_a) = \min_{T_b \in \{\text{доч. задачи } T_a\}} \left(\mathsf{ALST}(T_b) - \overline{\mathsf{ECOMT}}(c_{ab})\right) - \overline{\mathsf{EET}}(T_a).$$

Этот критерий вычисляется при обратном проходе графа задач, причем ALST для выходной задачи полагается равным уже вычисленному значению AEST. Задачи планируются на наилучший ресурс в порядке убывания ALST. В отличие от критерия, используемого в алгоритме HEFT, критерий ALST не гарантирует ранжирование с учетом зависимостей, поэтому применяется дополнительный шаг, восстанавливающий порядок задач.

LA (Lookahead)**:** Алгоритм является развитием алгоритма HEFT [9]. Ранжирование проходит полностью аналогично оригинальному алгоритму, однако при планировании наилучший ресурс определяется не ECT отдельной задачи, а общим временем выполнения приложения, если завершить составление расписания алгоритмом HEFT. Таким образом, конкретная задача может быть выполнена на менее производительном ресурсе, если это сокращает общее время выполнения. Алгоритм составляет качественные расписания, однако имеет высокую вычислительную сложность.

PEFT (Predict Earliest Finish Time)**:** Алгоритм также является развитием алгоритмов HEFT и LA [10]. Для каждой пары задача-ресурс вычисляется критерий OCT (Optimistic Cost Table):

$$\mathsf{OCT}(T_a, R_i) = \max_{T_b \in \text{доч. задачи} T_a} \min_{R_j} (\mathsf{OCT}(T_b, R_j) + \mathsf{EET}(T_b, R_j) + \overline{\mathsf{ECOMT}}(c_{ab}, R_i, R_j)).$$

Идея данного критерия — оценить время от начала выполнения задачи до завершения выполнения приложения без учета доступности ресурсов. Задачи планируются в порядке убывания OCT, при этом критерием для выбора наилучшего ресурса является сумма ECT и OCT. Как и LAgj, алгоритм PEFT можно назвать менее «жадной» версией HEFT, однако в данном случае вычислительная сложность алгоритма остается без изменений.

Наконец, в работе рассматривается динамический алгоритм MCT, занимающий промежуточное положение между статическим и динамическим подходом.

MCT (Minimum Completion Time)**:** Алгоритм рассматривает задачи по мере выполнения их зависимостей, что позволяет отнести его к динамическим. Однако при планировании алгоритм рассматривает не только свободные узлы, но и уже занятые. При этом для каждого узла поддерживается оценка момента времени освобожения ресурса $\mathsf{EST}(R_i)$. Задачи планируются на ресурс, обеспечивающий минимальное значение ECT с учетом времени старта задачи. Такой подход

позволяет сочетать сильные стороны динамического и статического подходов: вычисление ECT для задачи неявно учитывает (локальную) структуру композитного приложения, а влияние неточностей в оценках EET и ECOMT снижается за счет одновременного планирования ограниченного числа задач.

2. Имитационная модель

Для исследования алгоритмов планирования в данной работе используется имитационное моделирование. Данный подход, подразумевающий компьютерное моделирование процесса выполнения приложения в распределенной среде, активно применяется в данного рода исследованиях. В сравнении с натурными экспериментами имитационное моделирование позволяет значительно сократить время проведения экспериментов и обеспечить воспроизводимость результатов при невысоких требованиях к используемым аппаратным ресурсам. Однако при этом важно обеспечить точность (минимальное отклонение результатов от натурных экспериментов) и масштабируемость (возможность проведения крупномасштабных экспериментов) используемой модели.

Используемая в данной работе имитационная модель реализована на базе платформы SimGrid [11], предназначенной для реализации симуляторов РВС. Выбор данной платформы обусловлен её зрелостью, высоким уровнем верификации используемых моделей (в частности, сетевых ресурсов) и активной поддержкой разработчиками. Важным обстоятельством также является универсальность данной платформы, что позволяет моделировать с помощью SimGrid не только грид-системы, но и облачные инфраструктуры, пиринговые системы и MPI-приложения.

Все возможности платформы SimGrid доступны только при использовании языков C/C++ (также авторы поддерживают язык Java, однако в нем представлены не все функции платформы). Проведение сравнительной оценки методов планирования требует реализации множества программных компонент — генераторов модельных РВС и наборов задач, непосредственно алгоритмов планирования и средств анализа результатов, которые сами по себе не являются вычислительно сложными и, таким образом, не требуют использования низкоуровнего языка. Поэтому для данной работы была разработана библиотека pysimgrid, позволяющая использовать платформу SimGrid в языке Python. Использование Python позволило существенно ускорить разработку тестовой платформы, как за счет простоты самого языка

Python, так и за счет использования сторонних библиотек. Библиотека pysimgrid опубликована с открытым исходным кодом[1].

В рамках реализованной имитационной модели РВС представлена в виде совокупности вычислительных узлов и сетевых каналов между ними. Вычислительные узлы характеризуются производительностью, выражаемой в FLOPS. В текущей реализации предполагается, что каждый узел имеет одно процессорное ядро, ресурсы которого равномерно распределяются между выполняющимися на узле задачами. Сетевые каналы характеризуются пропускной способностью и латентностью. При моделировании передачи данных по каналу реализуется разделение пропускной способности канала между использующими его потоками.

Платформа SimGrid поддерживает моделирование различных сетевых топологий, включая иерархии и комбинации автономных систем с различными стратегиями маршрутизации. На данном этапе исследований рассматривается простая топология, в которой каждый узел подключен к общей сети через собственный сетевой канал, и маршрут между двумя узлами состоит из двух соответствующих каналов. Общая сеть в рамках данной топологии может соответствовать как коммутатору, так и глобальной сети. При этом задержки и ограничения по пропускной способности на уровне общей сети не моделируются.

В рамках реализованной имитационной модели приложения моделируются как направленные ациклические графы, вершинами которых являются задачи, а ребра задают зависимости по данным между задачами. Каждая задача характеризуется своим размером (объемом вычислений) в flops, а ребро — объемом передаваемых между задачами данных. Объем входных данных задачи равен сумме объемов входящих ребер. Также вводятся специальные вершины-задачи root и end, имеющие нулевой объем вычислений. Задача root передает через соответствующие ребра входные данные для начальных задач приложения (не зависят от других задач). Задача end принимает через соответствующие ребра выходные данные для терминальных задач приложения (не передают данные другим задачам). В случе пакета задач все задачи являются как начальными, так и терминальными, то есть связаны с root и end.

Для выполнения специальных задач root и end в рамках модели РВС создается специальный узел, называемый *мастером*, который не участвует в выполнении задач. Данный узел соответствует машине, на

[1] https://github.com/alexmnazarenko/pysimgrid

Таблица 1. Среднее время выполнения пакета задач (первая серия экспериментов)

# узлов	Random	RRobin	OLB	Min-Min	Max-Min	Sufferage
Степень неоднородности узлов = 2						
10	1.419	1.367	1.0	0.999	0.999	0.999
100	2.107	1.378	1.0	0.973	0.973	0.973
1000	4.456	1.0	1.0	1.0	1.0	1.0
Степень неоднородности узлов = 5						
10	2.211	2.175	1.0	0.994	0.994	0.994
100	3.347	2.450	1.0	0.893	0.893	0.893
1000	3.728	1.0	1.0	0.503	0.503	0.503
Степень неоднородности узлов = 10						
10	3.252	3.224	1.0	0.989	0.989	0.989
100	4.828	3.805	1.0	0.788	0.788	0.788
1000	3.329	1.0	1.0	0.275	0.275	0.275

которой размещаются входные данные и куда требуется загрузить выходные данные приложения. На практике данный узел обычно также осуществляет запуск и управляет ходом выполнения приложения.

3. Имитационные эксперименты

В данном разделе приведены результаты нескольких серий имитационных экспериментов с использованием описанных ранее алгоритмов и имитационной модели.

3.1. Алгоритмы планирования пакета задач

Во всех экспериментах генерировались синтетические пакеты из 1000 задач. Запуски пакетов осуществлялись на синтетических системах от 10 до 2000 узлов.

В первых двух сериях экспериментов не учитывались накладные расходы на передачу данных по сети, аналогично большинству работ в данной области. Для этого использовались пакеты с нулевыми объемами входных и выходных данных задач, а используемые системы имели бесконечную пропускную способность и нулевую латентность.

В первой серии экспериментов размер задач был зафиксирован, а варьировались число узлов (от 10 до 1000) и степень их неоднородности (отношение максимальной и минимальной производительностей,

Таблица 2. Среднее время выполнения пакета задач (вторая серия экспериментов)

# узлов	Random	RRobin	OLB	Min-Min	Max-Min	Sufferage
Степень неоднородности задач = 2						
10	1.151	1.024	1.0	0.998	0.995	1.0
100	1.774	1.072	1.0	0.964	0.935	1.0
Степень неоднородности задач = 10						
10	1.164	1.068	1.0	1.0	0.992	0.999
100	1.808	1.243	1.0	0.978	0.907	1.0
Степень неоднородности задач = 100						
10	1.172	1.076	1.0	1.001	0.992	1.0
100	1.828	1.302	1.0	0.984	0.901	0.998

от 2 до 10). Для каждой комбинации варьируемых параметров генерировалось по 100 систем. Результаты экспериментов для каждой комбинации параметров, усредненные по 100 запускам, приведены в таблице 1. Здесь и далее для каждого алгоритма приведено среднее значение времени выполнения пакета, нормализованного относительно времени алгоритма OLB. Эвристики пакетного планирования показывают лучшие результаты, чем алгоритм OLB. При этом наблюдаемый отрыв увеличивается с ростом степени неоднородности узлов и их числа. Отметим также, что в случае однородного пакета задач все эвристики пакетного планирования показывают одинаковые результаты.

Во второй серии экспериментов использовались гомогенные системы, а варьировались степень неоднородности задач в пакете (отношение максимального и минимального размеров задач, от 2 до 100) и число узлов (от 10 до 100). Для каждого значения степени неоднородности задач генерировалось по 100 пакетов. Результаты экспериментов для каждой комбинации параметров, усредненные по 100 запускам, приведены в таблице 2. В случае неоднородного пакета задач результаты различных эвристик пакетного планирования отличаются. Лучшие результаты во всех экспериментах показывает эвристика Max-Min. При этом отрыв от алгоритма OLB меньше, чем в предыдущей серии, и слабее зависит от степени неоднородности задач.

В последующих сериях экспериментов вводится учет накладных расходов на передачу данных по сети. В данных экспериментах задаются ненулевые объемы входных и выходных данных задач. Отношение

Таблица 3. Среднее время выполнения пакета задач (третья серия экспериментов)

Гранулярность	Random	RRobin	OLB	Min-Min	Max-Min	Sufferage
Степень неоднородности узлов = 2						
1	1.499	1.128	1.0	0.991	0.991	0.991
10	2.100	1.374	1.0	0.973	0.973	0.973
100	2.111	1.380	1.0	0.973	0.973	0.973
Степень неоднородности узлов = 5						
1	1.416	1.175	1.0	0.972	0.972	0.972
10	3.248	2.410	1.0	0.900	0.900	0.900
100	3.293	2.450	1.0	0.894	0.894	0.894
Степень неоднородности узлов = 10						
1	1.374	1.200	1.0	0.951	0.951	0.951
10	4.580	3.629	1.0	0.801	0.801	0.801
100	4.764	3.787	1.0	0.785	0.785	0.785

размера задачи в GFLOPS к объему входных данных в MB будем называть *гранулярностью* задачи. Также в данных экспериментах задаются фиксированные значения пропускной способности (100 МБ/с) и латентности (100 мкс) сетевых каналов, близкие к характеристикам локальной сети на базе Gigabit Ethernet.

В третьей серии экспериментов зафиксированы размер задач и число узлов (100), а варьировались гранулярность задач (от 1 до 100) и степень неоднородности узлов (от 2 до 10). Для каждой степени неоднородности узлов генерировалось по 100 систем. Результаты экспериментов для каждой комбинации параметров, усредненные по 100 запускам, приведены в таблице 3. В случае крупнозернистых задач (гранулярность 100) эффект от передачи данных по сети минимальный, и результаты очень близки к результатам серии 1 для 100 узлов. Однако с уменьшением гранулярности задач эффективность работы эвристик значительно снижается, приближаясь к OLB, поскольку используемые в них оценки времени передачи данных по сети не учитывают разделение пропускной способности канала между несколькими потоками.

В четвертой серии экспериментов использовалась гомогенная система из 100 узлов, а варьировались степень неоднородности (от 2 до 100) и гранулярность (от 1 до 100) задач. Для каждой комбинации

Таблица 4. Среднее время выполнения пакета задач (четвертая серия экспериментов)

Гранулярность	Random	RRobin	OLB	Min-Min	Max-Min	Sufferage
Степень неоднородности задач = 2						
1	1.562	1.067	1.0	1.200	1.405	1.017
10	1.763	1.071	1.0	0.965	0.971	1.000
100	1.776	1.069	1.0	0.964	0.936	1.000
Степень неоднородности задач = 10						
1	1.598	1.186	1.0	1.002	1.357	1.022
10	1.795	1.239	1.0	0.972	0.940	0.997
100	1.798	1.242	1.0	0.978	0.909	0.999
Степень неоднородности задач = 100						
1	1.613	1.217	1.0	0.971	1.346	1.057
10	1.841	1.299	1.0	0.979	0.937	0.998
100	1.831	1.281	1.0	0.983	0.901	0.993

параметров генерировалось по 100 пакетов. Результаты экспериментов, усредненные по 100 запускам, приведены в таблице 4. Аналогично предыдущей серии, в случае крупнозернистых задач результаты очень близки к результатам серии 2 для 100 узлов. С уменьшением гранулярности задач эффективность работы эвристик аналогично заметно падает, причем в данном случае показывая значения даже хуже алгоритма OLB. Отметим, что наибольший проигрыш показывает эвристика Max-Min, лучше все работавшая в аналогичной серии 2 без учета эффектов сети.

3.2. Алгоритмы планирования композитных приложений

В первой серии экспериментов использованы следующие тестовые классы композитных приложений (КП), соответствующие реальным научным расчетам [12]:

GENOME — КП для параллельной обработки нуклеотидных последовательностей (USC Epigenome Center);

LIGO — КП для обработки данных с детекторов в эксперименте по поиску гравитационных волн (LIGO);

MONTAGE — КП для объединения множественных снимков звездного неба в единое изображение (NASA/IPAC).

Таблица 5. Среднее время выполнения композитного приложения (первая серия экспериментов)

# узлов	OLB	MCT	Random	RRobin	HCPT	HEFT	LA	PEFT
GENOME								
5	1.0000	0.9715	2.8848	2.7558	0.9298	0.9057	0.8757	0.9106
10	1.0000	0.9147	3.2970	2.6183	0.8285	0.7980	0.7622	0.8168
20	1.0000	0.7893	3.2248	2.4883	0.7185	0.6407	0.6097	0.7294
LIGO								
5	1.0000	0.9792	2.5143	1.9667	1.0427	0.9522	0.9289	1.0086
10	1.0000	0.9232	2.9062	2.4588	1.0289	0.8547	0.8181	0.9763
20	1.0000	0.7769	2.8594	2.0179	0.8514	0.6341	0.6093	0.8311
MONTAGE								
5	1.0000	0.9771	2.0134	1.6907	1.0114	0.9759	0.9667	0.9801
10	1.0000	0.9635	2.5474	1.8999	0.9920	0.9621	0.9519	0.9744
20	1.0000	0.9114	2.7759	1.8974	0.9111	0.9140	0.8957	0.9252

Зафиксированы число задач в каждом КП (100) и степень неоднородности РВС (4), а варьировались класс КП и число узлов (от 5 до 20). Пропускная способность моделируемой сети 100 МБ/с, латентность — 100 мкс. Для каждой комбинации варьируемых параметров генерировалось по 100 систем.

Результаты первой серии экспериментов приведены в таблице 5. Во всех экспериментах лидирует алгоритм LA, за ним следуют остальные статические методы (HEFT, HCPT и PEFT). Алгоритм MCT ожидаемо показывает промежуточный результат между простым динамическим планированием и статическими методами. Наконец, алгоритмы Random и RRobin строят наиболее неэффективные расписания, так как не учитывают зависимости между задачами.

Для второй и третьей серии экспериментов использованы случайные КП, полученные по методу послойной генерации, специально разработанному для сравнения алгоритмов планирования [**13**]. Подход позволяет варьировать следующие параметры:

- число задач;
- среднее число вершин на слое графа (ширина);
- плотность связей между слоями графа;
- максимальное расстояние между слоями, для которых может быть добавлена связь.

Таблица 6. Среднее время выполнения композитного приложения (вторая серия экспериментов)

# узлов	OLB	MCT	Random	RRobin	HCPT	HEFT	LA	PEFT
Ширина слоя = 0.2								
5	1.0000	0.9637	2.6144	2.3130	0.9970	0.8932	0.8895	0.9261
10	1.0000	0.8655	2.4461	2.1967	0.8176	0.7481	0.7544	0.7942
20	1.0000	0.9473	2.6735	2.0184	0.8964	0.8749	0.8907	0.8961
Ширина слоя = 0.4								
5	1.0000	1.0061	2.8485	2.6580	0.9421	0.8527	0.8592	0.9490
10	1.0000	0.9388	2.9678	2.5944	0.8291	0.7290	0.7625	0.8623
20	1.0000	0.9830	2.8739	2.0620	0.8777	0.8639	0.9031	0.9369
Ширина слоя = 0.6								
5	1.0000	1.0283	2.6434	2.3073	0.8846	0.8512	0.8527	0.8672
10	1.0000	0.9336	2.6970	1.9665	0.7331	0.7082	0.8022	0.7604
20	1.0000	0.9694	2.7335	1.9157	0.8080	0.7984	0.9397	0.8319

Полученные в итоге композитные приложения обладают более сложной и несимметричной структурой, чем КП, использованные в первом серии экспериментов.

Вторая серия экспериментов построена аналогично первой. Сгенерировано три КП с различной шириной (20, 40, 60 задач на слой). Зафиксированы число задач в каждом КП (100) и степень неоднородности РВС (4). Варьируется ширина КП и число узлов (от 5 до 20). Пропускная способность моделируемой сети 100 МБ/с, латентность — 100 мкс. Для каждой комбинации варьируемых параметров генерировалось по 100 систем.

Результаты второй серии экспериментов приведены в таблице 6. Ширина слоя неявно определяет число одновременно выполняемых задач. Это позволяет пронаблюдать эффект от использования имитационной модели с учетом топологии и загрузки сети — по мере роста ширины слоя, растет ошибка оценки времени передачи данных (ECOMT). В частности, за счет этой ошибки значительно ухудшаются результаты наиболее гибкого и вычислительно сложного алгоритма LA. Для подтверждения этого вывода в таблице 7 приведены оценочные времена выполнения, достигнутые статическими алгоритмами во время построения расписания. Видно, что без учета эффекта конкуренции потоков данных в сети, алгоритм LA по прежнему является лучшим.

Таблица 7. Среднее оценочное время выполнения композитного приложения (вторая серия экспериментов)

# узлов	HCPT	HEFT	LA	PEFT
Ширина слоя = 0.2				
5	1.1242	1.0000	0.9900	1.0439
10	1.0946	1.0000	0.9984	1.0634
20	1.0250	1.0000	0.9996	1.0230
Ширина слоя = 0.4				
5	1.1164	1.0000	0.9798	1.1094
10	1.1263	1.0000	0.9917	1.1594
20	1.0141	1.0000	0.9981	1.0746
Ширина слоя = 0.6				
5	1.0721	1.0000	0.9858	1.0381
10	1.0385	1.0000	0.9849	1.0608
20	1.0096	1.0000	0.9974	1.0291

В третьей серии экспериментов использовался набор из 2700 случайных КП с широким разбросом характеристик, а варьировались число узлов (от 5 до 20) и пропускная способность сети (от 10 до 100 МБ/с). Отметим, что с увеличением пропускной способности также увеличивается временная гранулярность задач (соотношение между временами вычислений и передачи данных). Для каждой комбинации варьируемых параметров генерировалась одна система.

Результаты третьей серии экспериментов приведены в таблице 8. Результаты схожи со второй серией экспериментов, но, благодаря использованию систем с различной пропускной способностью сети, эффект неточной оценки ECOMT задач выражен более явно. Например, при пропускной способности сети 10 МБ/с, алгоритм LA проигрывает алгоритму HEFT 10-15%, в то время как алгоритм MCT приближается к результатам статических методов.

4. Заключение

В работе рассмотрен вопрос сравнительного анализа эффективности алгоритмов планирования приложений распределенных вычислительных средах. Реализован прототип программного инструментария для тестирования алгоритмов планирования на основе имитационной модели PBC. На базе данного инструментария реализовано несколько известных алгоритмов планирования пакетов задач и композитных

Таблица 8. Среднее время выполнения композитного приложения (третья серия экспериментов)

# узлов	OLB	MCT	Random	RRobin	HCPT	HEFT	LA	PEFT
Пропускная способность сети = 10 МБ/с								
5	1.0000	0.9085	2.0572	1.9281	0.8393	0.8465	0.8963	0.8509
10	1.0000	0.9259	2.3853	2.0350	0.8169	0.8007	0.9100	0.8317
20	1.0000	0.9198	2.2603	1.8399	0.8422	0.8355	0.9848	0.8602
Пропускная способность сети = 20 МБ/с								
5	1.0000	0.9564	2.3212	2.1245	0.8605	0.8136	0.8292	0.8550
10	1.0000	0.9559	2.4592	2.0599	0.8151	0.7665	0.8199	0.8262
20	1.0000	0.9618	2.5567	2.0459	0.8357	0.8036	0.8889	0.8489
Пропускная способность сети = 100 МБ/с								
5	1.0000	0.9535	2.3597	2.1476	0.9054	0.8308	0.8338	0.8950
10	1.0000	0.9561	2.4201	2.0004	0.8789	0.8306	0.8490	0.8834
20	1.0000	0.9716	2.7589	2.1751	0.8777	0.8592	0.8827	0.8975

приложений, проведены имитационные эксперименты с варьированием параметров среды и приложений. Выявлен ряд особенностей поведения алгоритмов, в частности деградация производительности в условиях конкуренции потоков в сети при использовании простых оценок времени передачи данных.

Дальнейшие исследования будут посвящены развитию имеющегося инструментария, реализации на его базе других существующих алгоритмов, проведению дополнительных экспериментов, в том числе в реальных РВС, и реализации набора тестов (benchmark) для сравнительного анализа алгоритмов планирования.

Список литературы

[1] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. R. Kan. “Optimization and approximation in deterministic sequencing and scheduling: a survey”, *Annals of Discrete Mathematics*, **5** (1979), pp. 287–326. ↑ 64

[2] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen et al. “A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems”, *Journal of Parallel and Distributed computing*, **61**:6 (2001), pp. 810–837. ↑ 64

[3] J. Yu, R. Buyya, K. Ramamohanarao. “Workflow scheduling algorithms for grid computing”, *Metaheuristics for scheduling in distributed computing environments*, Studies in Computational Intelligence, vol. **146**, Springer, Berlin–Heidelberg, 2008, pp. 173–214. ↑ 64

[4] R. Armstrong, D. Hensgen, T. Kidd. “The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions”, *Proceedings of the Seventh Heterogeneous Computing Workshop*, HCW’98 (Orlando, Florida, USA, 30 March, 1998), IEEE, 1998, pp. 79–87. ↑ 65

[5] R. F. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J. D. Lima et al. “Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet”, *Proceedings of the Seventh Heterogeneous Computing Workshop*, HCW’98 (Orlando, Florida, USA, 30 March, 1998), IEEE, 1998, pp. 184–199. ↑ 65

[6] M. Maheswaran, S. Ali, H. J. Siegal, D. Hensgen, R. F. Freund. “Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems”, *Proceedings of the Eighth Heterogeneous Computing Workshop*, HCW’99 (San Juan, Puerto Rico, 12 April, 1999), IEEE, 1999, pp. 30–44. ↑ 65

[7] H. Topcuoglu, S. Hariri, M. Wu. “Performance-effective and low-complexity task scheduling for heterogeneous computing”, *IEEE Transactions on Parallel and Distributed Systems*, **13**:3 (2002), pp. 260–274. ↑ 66

[8] T. Hagras, J. Janecek. “A simple scheduling heuristic for heterogeneous computing environments”, *Proceedings of the Second International Symposium on Parallel and Distributed Computing*, ISPDC’03 (Ljubljana, Slovenia, 13–14 October, 2003), 2003, pp. 104–110. ↑ 66

[9] L. F. Bittencourt, R. Sakellariou, E. R. M. Madeira. “DAG Scheduling Using a Lookahead Variant of the Heterogeneous Earliest Finish Time Algorithm”, *Proceedings of the 18th Euromicro International Conference on Parallel, Distributed and Network-Based Computing*, PDP’10 (Pisa, Italy, 17–19 February, 2010), 2010, pp. 27–34. ↑ 67

[10] H. Arabnejad, J. G. Barbosa. “List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table”, *IEEE Transactions on Parallel and Distributed Systems*, **25**:3 (2014), pp. 682-694. ↑ 67

[11] H. Casanova, A. Giersch, A. Legrand, M. Quinson, F. Suter. “Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms”, *Journal of Parallel and Distributed Computing*, **74**:10 (2014), pp. 2899–2917. ↑ 68

[12] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. H. Su, K. Vahi. “Characterization of scientific workflows”, *Proceedings of the Third Workshop on Workflows in Support of Large-Scale Science*, WORKS’08

(Austin Convention Center, Austin, TX, USA, 17 November, 2008), 2008, pp. 1–10. ↑ 73

[13] T. Tobita, H. Kasahara. “A standard task graph set for fair evaluation of multiprocessor scheduling algorithms”, *Journal of Scheduling*, **5**:5 (2002), pp. 379–394. ↑ 74

Рекомендовал к публикации *Программный комитет*
Пятого национального суперкомпьютерного форума ***НСКФ-2016***

Пример ссылки на эту публикацию:

О. В. Сухорослов, А. М. Назаренко. «Сравнительная оценка методов планирования приложений в распределенных вычислительных средах», *Программные системы: теория и приложения*, 2017, **8**:1(32), с. 63–81.
URL: http://psta.psiras.ru/read/psta2017_1_63-81.pdf

Об авторах:

Олег Викторович Сухорослов

Кандидат технических наук, старший научный сотрудник лаборатории распределенных вычислительных систем Института проблем передачи информации РАН. Области научных интересов: распределенные вычисления, грид-технологии, сервис-ориентированные и облачные вычисления.

e-mail: sukhoroslov@iitp.ru

Алексей Михайлович Назаренко

Старший программист в компании DATADVANCE, младший научный сотрудник лаборатории интеллектуального анализа данных и предсказательного моделирования Института проблем передачи информации РАН. Области научных интересов: композитные приложения, распределенные вычисления, облачные вычисления.

e-mail: nazar@phystech.edu

Oleg Sukhoroslov, Alexey Nazarenko. *Comparative study of scheduling algorithms for distributed computing environments.*

Abstract. The paper analyzes the efficiency of known algorithms for scheduling of bag-of-tasks and workflow applications in distributed computing environments (DCE). The comparison of algorithms is performed on the base of simulation experiments for various application cases and DCE configurations. Simulation allows to avoid the time consuming real-world experiments and to ensure reproducible results. Besides the experimental results, the used simulation model (simulator) of DCE based on the SimGrid framework is described. (*in Russian*).

Key words and phrases: distributed computing, scheduling algorithms, bag of tasks, workflow, simulation.

References

[1] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. R. Kan. "Optimization and approximation in deterministic sequencing and scheduling: a survey", *Annals of Discrete Mathematics*, **5** (1979), pp. 287–326.

[2] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen et al. "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems", *Journal of Parallel and Distributed computing*, **61**:6 (2001), pp. 810–837.

[3] J. Yu, R. Buyya, K. Ramamohanarao. "Workflow scheduling algorithms for grid computing", *Metaheuristics for scheduling in distributed computing environments*, Studies in Computational Intelligence, vol. **146**, Springer, Berlin–Heidelberg, 2008, pp. 173–214.

[4] R. Armstrong, D. Hensgen, T. Kidd. "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions", *Proceedings of the Seventh Heterogeneous Computing Workshop*, HCW'98 (Orlando, Florida, USA, 30 March, 1998), IEEE, 1998, pp. 79–87.

[5] R. F. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J. D. Lima et al. "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet", *Proceedings of the Seventh Heterogeneous Computing Workshop*, HCW'98 (Orlando, Florida, USA, 30 March, 1998), IEEE, 1998, pp. 184–199.

[6] M. Maheswaran, S. Ali, H. J. Siegal, D. Hensgen, R. F. Freund. "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems", *Proceedings of the Eighth Heterogeneous Computing Workshop*, HCW'99 (San Juan, Puerto Rico, 12 April, 1999), IEEE, 1999, pp. 30–44.

[7] H. Topcuoglu, S. Hariri, M. Wu. "Performance-effective and low-complexity task scheduling for heterogeneous computing", *IEEE Transactions on Parallel and Distributed Systems*, **13**:3 (2002), pp. 260–274.

[8] T. Hagras, J. Janecek. “A simple scheduling heuristic for heterogeneous computing environments”, *Proceedings of the Second International Symposium on Parallel and Distributed Computing*, ISPDC’03 (Ljubljana, Slovenia, 13–14 October, 2003), 2003, pp. 104–110.

[9] L. F. Bittencourt, R. Sakellariou, E. R. M. Madeira. “DAG Scheduling Using a Lookahead Variant of the Heterogeneous Earliest Finish Time Algorithm”, *Proceedings of the 18th Euromicro International Conference on Parallel, Distributed and Network-Based Computing*, PDP’10 (Pisa, Italy, 17–19 February, 2010), 2010, pp. 27–34.

[10] H. Arabnejad, J. G. Barbosa. “List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table”, *IEEE Transactions on Parallel and Distributed Systems*, **25**:3 (2014), pp. 682-694.

[11] H. Casanova, A. Giersch, A. Legrand, M. Quinson, F. Suter. “Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms”, *Journal of Parallel and Distributed Computing*, **74**:10 (2014), pp. 2899–2917.

[12] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. H. Su, K. Vahi. “Characterization of scientific workflows”, *Proceedings of the Third Workshop on Workflows in Support of Large-Scale Science*, WORKS’08 (Austin Convention Center, Austin, TX, USA, 17 November, 2008), 2008, pp. 1–10.

[13] T. Tobita, H. Kasahara. “A standard task graph set for fair evaluation of multiprocessor scheduling algorithms”, *Journal of Scheduling*, **5**:5 (2002), pp. 379–394.

Sample citation of this publication:

Oleg Sukhoroslov, Alexey Nazarenko. “Comparative study of scheduling algorithms for distributed computing environments”, *Program systems: Theory and applications*, 2017, **8**:1(32), pp. 63–81. (*In Russian*).

URL: http://psta.psiras.ru/read/psta2017_1_63-81.pdf