Планирование с наилучшим качеством для выполнения композитных приложений в гетерогенной вычислительной среде

А. А. Прохоров[a], А. М. Назаренко[b]

Институт проблем передачи информации РАН им. А.А. Харкевича,
125051, г. Москва, Б. Каретный пер., 19

E-mail: [a] prokher@gmail.com, [b] nazar@phystech.edu

Вычисления на основе композитных приложений (КП) заняли важное место в самых различных областях — от исследовательских задач до прикладных инженерных рассчетов. Независимые задачи в КП могут быть выполнены параллельно, что определяет одно из ключевых преимуществ данного подхода — возможность использовать распределенную вычислительную среду вне зависимости от поддержки распределенных вычислений каждой из задач в КП. Эффективное использование распределенной, часто гетерогенной, среды является сложной задачей и требует планирования выполнения — алгоритма принятия решений, какая задача и в какой момент должна быть выполнена на определенном ресурсе. В литературе рассматриваются различные постановки задачи планирования, однако самым развитым направлением является планирование с наилучшим качеством — минимизация общего времени выполнения КП без дополнительных критериев. Данная работа посвящена систематизации и обзору существующих результатов. Рассмотрены основные классы алгоритмов планирования с наилучшим качеством и выделены их основные преимущества и недостатки.

Ключевые слова: распределенные вычисления, планирование, композитное приложение, поток работ.

Работа выполнена при финансовой поддержке РФФИ в рамках научного проекта № 15-29-07043. Условия для выполнения работ по проекту предоставлены ООО "ДАТАДВАНС".

1. Введение

На сегодняшний день доступно огромное разнообразие вычислительных пакетов и алгоритмов для самого широкого круга задач. Для некоторых, обычно коммерчески востребованных, задач существуют интегрированные решения. Однако в большинстве случаев приходится применять целый ряд разнородных инструментов, связав их в единую вычислительную схему — так называемое композитное приложение (КП).

Композитные приложения могут составляться из сотен и даже тысяч отдельных задач, многие из которых могут быть выполнены параллельно. Для этого можно использовать распределенные вычислительные среды (РВС) самой различной структуры, так как необходимая частота обмена данными в композитном приложении гораздо ниже, чем в классическом параллельном приложении.

Использование распределенной среды ставит задачу эффективной организации вычислений. Рассматриваются различные постановки, учитывающие такие критерии как равномерность загрузки узлов или соблюдение некоторого временного бюджета, однако наиболее изученной является задача планирования с наилучшим качеством (best-effort scheduling), когда единственной целью планирования является минимизация общего времени выполнения.

Планирование выполнения КП в РВС имеет ряд особенностей. В первую очередь, это гетерогенность самой вычислительной среды - ресурсы обладают различной производительностью для различных задач. Кроме того, обычно невозможен перенос задачи между ресурсами без полного ее перезапуска, поэтому возможно только невытесняющее планирование. Таким образом, задача относится к одному из самых сложных теоретических классов и является NP-полной [Graham, Lawler, …, 1979]. Это делает точное решение задачи практически недостижимым. Поэтому для планирования используются различные приближенные методы.

2. Модель выполнения

Для строгой постановки задачи планирования необходимо ввести две модели - модель вычислительной среды и модель композитного приложения. Кроме того, неявно предполагается наличие управляющей системы или брокера выполнения, то есть программного комплекса, непосредственно запускающего задачи и контролирующего их выполнение (подробнее об архитектуре и функциональности таких систем см. [Yu, Rajkumar, 2005]). Однако с точки зрения задачи планирования достаточно предположить, что накладные расходы в управляющей системе пренебрежимо малы.

Вычислительная среда состоит из ресурсов R_i $(1 \le i \le M)$ с различной производительностью. Между каждой парой ресурсов есть сетевое соединение с заданными пропускной способностью и латентностью. При этом реальной топологией сети и эффектами взаимодействия потоков данных обычно пренебрегают. Так как в данной работе рассматриваются только методы с наилучшим качеством, дополнительные свойства ресурсов, такие как стоимость использования, не рассматриваются.

Композитные приложения удобно описывать на языке потоков работ (workflow). Поток работ обычно представляется в виде направленного ациклического графа. Вершины графа $T_a(1 \le a \le N)$ отражают выполнение отдельной задачи в КП, а ребра C_{ab} задают зависимость или связь между задачами. Если присутствует связь C_{ab}, задачу T_a называют (непосредственным) родителем задачи T_b, а задачу T_b - дочерней задачей T_a. Дочерние задачи не могут быть начаты, пока не завершились все их родители.

Для всех рассматриваемых далее методов планирования необходима возможность оценить время выполнения задач на любом ресурсе. Обозначим эту оценку как $EET(T\_a, R\_i)$ (EET = Estimated Execution Time) и будем считать ее известной.

Для каждой связи C_{ab} определен объем данных c_{ab}, передаваемых от родительской задачи к дочерней. Аналогично времени выполнения EET, обозначим время передачи данных от задачи T_a к T_b как $ECOMT(c_{ab}, R_i, R_j)$ (Estimated Communication Time). Есть различные подходы к оценке $ECOMT$, но чаще всего используют простую линейную модель:

$$ECOMT(c_{ab}, R_i, R_j) = t_{con} + \frac{c_{ab}}{L_{ij}},$$

где t_{con} - латентность соединения, L_{ij} - пропускная способность сети между ресурсами R_i и R_j [Topcuoglu, Hariri, Wu Min-You, 2002].

В рамках введенной модели мы можем описать время завершения каждой задачи и всего приложения в целом. Пусть задача T_a выполняется на ресурсе R_i, а ее родительские задачи T_b - на ресурсах R_{j_b}. Тогда:

$$ECT(T_a, R_i) = EET(T_a, R_i) + \max_{b \in \text{родительские задачи}} \left(ECOMT(c_{ab}, R_i, R_{j_b}) + ECT(T_b, R_{j_b}) \right).$$

Оценка времени завершения последней задачи в композитном приложении $ECT_{max} = \max_a ECT(T_a)$ определяет время выполнения всего композитного приложения.

3. Алгоритмы планирования

3.1. Планирование индивидуальных задач

Самый простой подход к планированию — выполнять каждую задачу на наилучшем для нее ресурсе, не учитывая структуру приложения.

Такой подход реализуется в эвристике MCT (Minimum Completion Time) [Maheswaran, Ali, …, 1999]. На каждом шаге алгоритм запускается для одной готовой к выполнению задачи T и планирует выполнение на ресурсе, минимизирующем $ECT(T_a, R_i)$.

Несмотря на то, что задачи планируются независимо, алгоритм MCT может неявно учитывать структуру композитного приложения за счет вычисления оценки ECT, включающей в себя время передачи данных. Если планировщик обладает актуальной информацией о состоянии вычислительной сети, этого достаточно чтобы эффективно планировать приложения с несложной структурой, избегая использования перегруженных ресурсов или ресурсов в плохой сетевой доступности.

3.2. Пакетное планирование

В пакетном планировании на каждом шаге алгоритма рассматриваются все готовые к выполнению задачи [Mandal, Kennedy, …, 2005]. Для каждой задачи оценивается некоторый критерий, который определяет приоритет данной задачи в рамках планируемого пакета. После этого задача с наивысшим приоритетом планируется на ресурс с минимальной оценкой $ECT(T_a, R_i)$, а приоритеты перерасчитываются. Это повторяется пока не будет распланирован весь пакет.

Авторами эвристики было предложено 3 критерия: MinMin — первыми планируются задачи с минимальным оценочным временем выполнения на наилучшем ресурсе, MaxMin — первыми планируются задачи с максимальным оценочным временем выполнения на наилучшем ресурсе, Sufferage — первыми планируются задачи с максимальной разницей оценок времени выполнения между двумя наилучшими ресурсами.

Методы пакетного планирования сочетают гибкость динамического планирования с приоритезацией задач для повышения качества плана, что делает их эффективным решением при планировании в условиях неполной информации о структуре КП.

3.3. Приоритетное планирование

В отличие от уже рассмотренных алгоритмов, методы приоритетного планирования (в разных источниках – priority, level или list scheduling) учитывают полную структуру приложения. Приоритетное планирование основано на идее, что лучшие ресурсы должны быть выделены задачам, которые оказывают наибольшее влияние на общее время выполнения КП.

В алгоритмах приоритетного планирования можно явно выделить два этапа – упорядочивание задач и размещение их на ресурсы.

Для примера рассмотрим, вероятно, самый известный алгоритм этого типа – HEFT (Heterogeneous Earliest Finish Time [Topcuoglu, Hariri, Wu Min-You, 2002]).

На первом шаге производится ранжирование задач по следующему признаку:

$$rank\,(T_a) = \overline{EET}(T_a) + \max_{T_b \in \text{ дочерние задачи } T_a} (\overline{ECOMT}(c_{ab}) + rank(T_b)),$$

где $\overline{EET}(T_a)$ – средняя оценка времени выполнения задачи T_a по всем ресурсам, $\overline{ECOMT}(c_{ab})$ – среднее время передачи объема данных c_{ab} между всевозможными парами ресурсов.

Отметим важную особенность функции ранжирования — родительская задача всегда имеет более высокий приоритет, чем все ее дочерние задачи. Благодаря этому на момент планирования каждой задачи известно, где будут выполнены все ее родительские задачи.

На втором шаге все задачи распределяются на ресурсы, минимизирующие ECT, в порядке убывания ранга.

Приоритетное планирование можно назвать самым развитым подходом, представленным наибольшим количеством публикаций и алгоритмов. Разработка алгоритма HEFT установила новый стандарт в этой области, после чего были предолжены как новые методы упорядочивания задач, так и альтернативные алгоритмы размещения задач по вычислительным ресурсам [Arabnejad, Barbosa, 2014].

3.4. Кластерное планирование

Кластерное планирование – класс эвристических методов планирования, основанных на разбиении задач в КП на группы, выполняемые на одном ресурсе. Так как время передачи данных в пределах ресурса обычно полагают нулевым, эффективная группировка задач позволяет сократить время ожидания данных во время выполнения, и, таким образом, общее время выполнения КП.

Особенностью многих методов кластерного планирования является формирование заранее неизвестного числа кластеров, которое может быть больше числа доступных ресурсов. В таком случае, формирование финального плана выполнения требует использования дополнительного алгоритма, например, модифицированной версии приоритетного планирования, работающей с кластерами вместо отдельных задач [Kianzad, Bhattacharyya, 2006]. Кроме того, дополнительный шаг планирования позволяет адаптировать метод для использования гетерогенной вычислительной среды (кластеризация обычно проводится без учета гетерогенности). Однако существуют и алгоритмы кластерного планирования, изначально разработанные для использования в ограниченной гетерогенной среде [Cirou, Jeannot, 2001].

3.5. Планирование с дублированием задач

Планирование с дублированием задач допускает многократное выполнение некоторых задач КП на различных ресурсах, для сокращения времени передачи данных между ресурсами.

Однако, дублирование задач создает дополнительную нагрузку на ресурсы, что ограничивает применимость таких методов в загруженных вычислительных средах.

Алгоритмы с дублированием задач, как правило, построены на одном из уже рассмотренных эвристических подходов — приоритетном или кластерном. Однако именно дублирование задач принято выделять как основную особенность алгоритма.

Сравнение с алгоритмами приоритетного планирования показывает преимущество методов с дублированием задач при условии достаточно высокой доли обмена данными в общем времени выполнения, например, в работе [Hagras, Janecek, 2004] авторы демонстрируют уменьшение времени выполнения по сравнению с алгоритмом HEFT для ~90% тестовых КП.

4. Заключение

В работе рассмотрена классификация методов планирования с наилучшим качеством, структура и основные особенности каждого из классов. Кроме того, введены необходимые модели, позволяющие сформулировать задачу планирования.

Важным направлением дальнейших исследований является проведение вычислительных экспериментов для определения практической эффективности алгоритмов. Особый интерес представляет сравнение между собой алгоритмов различных классов, так как в большинстве работ новые методы сравнивают с аналогами.

Список литературы

Arabnejad H., Barbosa J.G. List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table // IEEE Transactions on Parallel and Distributed Systems. — 2014. — Vol. 25, No. 3. — P. 682–694.

Cirou B., Jeannot E. Triplet: A clustering scheduling algorithm for heterogeneous systems // International Conference on Parallel Processing Workshops. Procedings. — 2001. — P. 231–236.

Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey // Annals of Discrete Mathematics. — 1979. — Vol. 5. — P. 287–326.

Hagras T., Janecek J. A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems // 18th International Parallel and Distributed Processing Symposium. Proceedings. — 2004. — 107 p.

Kianzad V., Bhattacharyya S.S. Efficient techniques for clustering and scheduling onto embedded multiprocessors // IEEE Transactions on Parallel and Distributed Systems. — 2006. — Vol. 17, No. 7. — P. 667–680.

Maheswaran M., Ali S., Siegal H.J., Hensgen D., Freund R.F. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems // Heterogeneous Computing Workshop. Proceedings. — 1999. — P. 30–44.

Mandal A., Kennedy K., Koelbel C., Marin G., Mellor-Crummey J., Liu B., Johnsson L. Scheduling strategies for mapping application workflows onto the grid // HPDC-14. Proceedings. — 2005. — P. 125–134.

Topcuoglu H., Hariri S., Wu Min-You Performance-effective and low-complexity task scheduling for heterogeneous computing // IEEE Transactions on Parallel and Distributed Systems. — 2002. — Vol. 13, No. 3. — P. 260–274.

Yu J., Rajkumar B. A Taxonomy of Scientific Workflow Systems for Grid Computing // SIGMOD Rec. — 2005. — Vol. 34, No. 3. — P. 44–49.

Best-effort workflow scheduling in a heterogeneous environment

A. A. Prokhorov[a], A. M. Nazarenko[b]

Institute for information transmission problems,
19, Bolshoy Karetnyy per., Moscow, 127051, Russia

E-mail: [a] prokher@gmail.com, [b] nazar@phystech.edu

Workflow-based computations play a significant role in multiple domains ranging from scientific research to applied engineering. Independent tasks in a workflow can be executed in parallel, which allows using distributed computing environments even if none of the tasks themselves can be scaled to multiple processors. Efficient usage of complex heterogeneous environments demands a scheduling procedure — algorithm deciding what task should be executed on a processor at a given time. There are different scheduling problem statements, but the most explored is so-called best-effort scheduling problem — minimizing total execution time without any additional criteria. This paper is an overview of the existing results in the area. We consider major classes of best-effort scheduling algorithms and highlight their pros and cons.

Keywords: distributed computing, scheduling, workflow.

The study is supported by the Russian Foundation for Basic Research (RFBR), research project No. 15-29-07043. Research facilities are provided by DATADVANCE LLC.