Распределение заданий в Desktop grid: обзор подходов к проблеме

И. А. Чернов[a]

ИПМИ КарНЦ РАН,
Россия, 185910, г. Петрозаводск, ул. Пушкинская, д. 11

E-mail: [a] chernov@krc.karelia.ru

Вычислительные сети из настольных компьютеров, связанных через Интернет или локальные сети (Desktop grid) являются эффективным и недорогим средством для решения широкого класса вычислительных задач. Необходимым требованием к задаче является возможность разбиения ее на большое число несложных слабо связанных или независимых расчетных заданий. Для эффективной работы нужен алгоритм распределения заданий по вычислительным узлам — диспетчеризации. Возможно множество критериев оптимизации, а сама задача NP-сложна во многих случаях. Для определенности имеем в виду грид-системы на базе BOINC. Это открытое программное обеспечение позволяет развернуть грид-систему с клиент-серверной архитектурой, в которой сервер создает задания и принимает ответы, а узлы обращаются к нему за заданиями и возвращают ответы. На узлах устанавливается клиент, использующий ресурсы во время их простоя. Можно выделить два класса грид-систем: проекты добровольных вычислений (volunteer computing) и гриды в пределах организации (Enterprise Desktop Grid). При добровольных вычислениях стоит вопрос противодействия злонамеренным действиям, актуальна задача привлечения добровольцев, требуется система вознаграждения за участие в проекте или за получение желаемого результата. В другом случае важна эффективность, можно точнее предсказать доступность узлов, известны их количество, производительность, объем свободной памяти, установленное программное обеспечение и операционная система. Такие системы существенно менее изучены. В этой работе мы дадим краткий обзор проблем, решаемых посредством соответствующего распределения заданий, а также подходов к оценке качества такого распределения, применительно к EDG. Задача распределения заданий NP-сложна, поэтому предлагаются различные эвристики или применяются генетические алгоритмы. Возможные критерии качества: makespan (время между постановкой задания и получением ответа), throughput (число заданий в единицу времени), полное время работы, максимальная нагрузка, однородная нагрузка, надежность (риск ложного ответа), средние затраты на расчет, нарушение сроков. Алгоритмы тестируют на симуляторах и эмуляторах.

Ключевые слова: диспетчеризация заданий, гриды, BOINC

Работа выполнена при поддержке грантов РФФИ 16-07-00622 и 15-29-07974

Введение

Вычислительные сети из настольных компьютеров, связанных через Интернет или локальные сети (Desktop grid) являются эффективным и недорогим средством для высокоэффективного решения широкого класса задач. Необходимым требованием к задаче является возможность разбиения ее на большое число простых слабо связанных или независимых расчетных заданий. Примерами таких задач являются: идентификация параметров модели гидридного фазового перехода методом слепого поиска [Чернов, 2013] и виртуальный скрининг [Ивашко, 2015]; много проектов добровольных вычислений описано на сайте boinc.berkelev.edu. Для эффективной работы нужен алгоритм распределения заданий по вычислительным узлам — диспетчеризации. Возможно множество критериев оптимизации, а сама задача NP-сложна во многих случаях.

Для определенности имеем в виду грид-системы на базе BOINC (boinc.berkelev.edu). Это открытое программное обеспечение позволяет развернуть грид-систему с клиент- серверной архитектурой, в которой сервер создает задания и принимает ответы, а узлы обращаются к нему за заданиями и возвращают ответы. На узлах устанавливается клиент, использующий ресурсы во время их простоя.

Можно выделить два класса грид-систем: проекты добровольных вычислений (volunteer computing) и гриды в пределах организации (Enterprise desktop grid [Ivashko, 2015]). При добровольных вычислениях стоит вопрос противодействия злонамеренным действиям, актуальна задача привлечения добровольцев, требуется система вознаграждения за участие в проекте или за получение желаемого результата. В грид организации важна эффективность, можно точнее предсказать доступность узлов, известны их количество, производительность, объем свободной памяти, установленное программное обеспечение и операционную систему. Такие системы существенно менее изучены.

В этой работе мы дадим краткий обзор проблем, решаемых посредством соответствующего распределения заданий, а также подходов к оценке качества такого распределения, применительно к EDG. Так, не рассматриваем методы антисаботажа, которым посвящено много работ.

Имеются следующие обзорные работы по методам диспетчеризации: [Xhafa, 2010; Choi, 2006; Estrada, 2012]. Задача распределения заданий NP-полна, поэтому предлагаются различные эвристики. Возможные критерии качества: makespan (время между постановкой задания и получением ответа), throughput (число заданий в единицу времени), полное время работы, максимальная нагрузка, однородная нагрузка, надежность (риск ложного ответа), средние затраты на расчет, нарушение сроков. Алгоритмы тестируют на симуляторах и эмуляторах: [Taufer, 2007; Beaumont, 2011; Estrada, 2009; Anderson, 2011]. Статьи [Qu, 2010] и [Wang, 2011] оптимизируют одновременно makespan и надежность при помощи генетического алгоритма. В работе [Estrada, 2008] организована эволюция на множестве правил распределения заданий, описанных в заданной грамматике. Естественный отбор приводит к оптимальному набору правил диспетчеризации.

Пакетирование заданий

Возможно объединение расчетных заданий в сложные задания (пакеты) с разными целями: во-первых, если затраты времени на коммуникацию сравнимы со временем расчета, пакетирование резко увеличивает производительность. Такая ситуация описана в нашей работе [Чернов, 2013]; во-вторых, большое число быстро решаемых заданий может создавать нагрузку на сервер вплоть до эффекта DDoS-атаки. В статье [Mazalov, 2014] описана игровая модель взаимодействия узлов и сервера, снижающая нагрузку за счет выбора размера пакета заданий. Наконец, в пакеты заданий можно внедрять тестовые задания или реплики заданий [Domingues, 2007].

Надежность и доступность

Надежность — это устойчивость к сбоям, ошибкам, поломкам. Различается надежность в смысле вероятности получить ответ или — правильный ответ. Пример источника ошибок — сходимость алгоритма типа спуска к локальному минимуму. Иногда ошибку может исправить репликация. Она же повышает вероятность получения хоть какого-нибудь ответа в срок. Контрольные точки — сохранение состояния алгоритма — другой подход повышения надежности. Диспетчеризация заданий сама может повышать надежность: либо распределяя задания во времени так, чтобы снизить последствия сбоя, либо оптимально раздавая их разным узлам. Техники heartbeat (проверки пульса) позволяют определить, работает ли узел. Эти техники пригождаются и для проблемы доступности. Работа [Sathva, 2010] содержит обзор-классификацию различных методов защиты от сбоев при вычислениях в грид-системах, все методы описаны более или менее подробно, указаны преимущества и недостатки в сравнении друг с другом.

Доступность — это наличие узла в сети тогда, когда он нужен. В ряде работ это называют надежностью, что создает путаницу. В некоторых работах различают состояния «включен, но занят» и «выключен». Применяются статистические методы для выявления шаблонов: от простых, выявляющих зависимости между временем суток и доступностью, до весьма изощренных. Предсказание доступности узла по статистике — похожая, но иная технология. Применяются репутационные методы — хорошая репутация означает частую доступность или предсказуемое поведение — и кластеризация узлов по признаку доступности. Статьи [Nouman, 2014; Rubab, 2014] дают обзор по этой теме.

Зависимые задания

Вопросы надежности и доступности особенно важны, если задания зависимы. Зависимости описываются орграфом без циклов (деревом), который заранее известен [Lee, 2010; Gao, 2007]. Возникают и новые задачи: например, выявление критических для производительности заданий, которые важно решить в срок. Статья [Cordasco, 2012] рассматривает новую метрику «площадь», названную так по аналогии с суммами Римана: это число заданий, которые можно решать в каждый момент времени.

Список литературы

Choi S. J., A taxonomy of DG systems focusing on scheduling // Technical Report: KU-CSE- 2006-1120-02. — 2006 .— 17 p.

Ivashko E. Enterprise Desktop Grids // BOINC-based High Performance Computing: Fundamental Research and Development. Proceedings of the Second International Conference BOINC:FAST. — 2015. — P. 16-21.

Xhafa, F., Ajith A. Computational Models and Heuristic Methods for Grid Scheduling Problems // Future Generation Computer Systems. — 2010. — № 26(4). Pp. 608-21. Estrada T., Taufer M. Challenges in Designing Scheduling Policies in Volunteer Computing // Desktop Grid Computing. — 2012. — P. 167-190.

Taufer M., Kerstens A., Estrada T., Flores D.A., Teller P.J. SimBA: A Discrete Event Simulator for Performance Prediction of Volunteer Computing Projects // Principles of Advanced and Distributed Simulation. — 2007. — P. 189-197.

Beaumont O., Bobelin L., Casanova H. et al Towards Scalable, Accurate, and Usable Simulations of Distributed Applications and Systems // Technical report RR-7761. — 2011. — 36 p.

Estrada T., Taufer M., Reed K., Anderson D.P. EmBOINC: An emulator for performance analysis of BOINC projects // Parallel and Distributed Processing. — 2009. — P. 1-8.

Anderson D.P. Emulating volunteer computing scheduling policies // Parallel and Distributed Processing Workshops and Phd Forum. — 2011. — P. 1839-1846.

Chernov I.A., Ivashko E.E., Nikitina N.N., Gabis I.E. Chislennaya identificatsiya modeli degidrirovaniya v grid-sisteme yf baze BOINC [Numerical identification of dehydriding model in a BOINC-based grid] // Kompyuternyye issledovaniya i modelirovaniye. — 2013. — Vol. 5, № 1. — S. 37-45 (in Russian).

Mazalov V.V., Nikitina N.N., Ivashko E.E. Hierarchical two-level game model for tasks scheduling in a desktop grid // Ultra Modern Telecommunications and Control Systems and Workshops. — 2014. — P. 541-545.

Ivashko E.E., Nikitina N.N., Moeller S. Vysokoproizvoditelnyi virtualnyi skrining v Enterprise Desktop Grid na baze BOINC [High-performance virtual screening on enterprise BOINC-based desktop grid] // Vestnik Yuzhno-Uralskogo gosudarstvennono universiteta. — 2015. — Vol. 4, № 1. — S. 57-63 (in Russian).

Domingues P., Sousa B., Moura Silva L. Sabotage-tolerance and trust management in desktop grid computing // Future Generation Computer Systems. — 2007. — Vol. 23, №7. — P. 904-912.

Sathva S.S., Babu S.K. Survey of fault tolerant techniques for grid // Computer Science Review. - 2010. - Vol. 4. - P. 101-120.

Nouman D.M., Shamsi J.A. Volunteer computing: requirements, challenges, and solutions // Journal of Network and Computer Applications. — 2013. — Vol. 39. — P. 369-380.

Lee Y.C., Zomava A.Y., Siegel H.J. Robust task scheduling for volunteer computing systems // The Journal of Supercomputing. — 2010. — Vol. 53. — P. 163-181.

Gao L., Malewicz G. Toward maximizing the quality of results of dependent tasks computed unreliably // Theory of Computing Systems. - 2007. - Vol. 41. - P. 731-752.

Cordasco G., De Chiara R., Rosenberg A.L. On scheduling DAGs for volatile computing platforms: Area-maximizing schedules // Journal of Parallel and Distributed Computing. — 2012. - Vol. 72. - P. 1347-1360.

Estrada T., Fuentes O., Taufer M. A distributed evolutionary method to design scheduling policies for volunteer computing // ACM SIGMETRICS Performance Evaluation Review. — 2008. - Vol. 36. - P. 40-49.

Qu B., Lei Y., Zhao Y. A new genetic algorithm based scheduling for volunteer computing // Computer and Communication Technologies in Agriculture Engineering. — 2010. — P. 228231.

Wang X., Yeo C.S., Buvva R., Su J. Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm // Future Generation Computer Systems. - 2011. - Vol. 27. - P. 1124-1134.

Rubab et al. A review on resource availability prediction methods in volunteer grid computing. // IEEE International Conference on Control System, Computing and Engineering. — 2014. — P. 478-483.

Scheduling in Desktop grid: review of approaches

I. A. Chernov[a]

IAMR KRC RAS,
11 Pushkinskaya street, Petrozavodsk, Russia

E-mail: [a] chernov@krc.karelia.ru

Computational networks build of desktop computers connected by LAN of internet (Desktop grids) are an efficient and cheap tool for solving multiple problems that demand high-performance computing. The main restriction is possibility to decompose the problem into many simple loosely dependent or independent tasks. For efficient work a scheduling algorithm is necessary. A few criteria are possible for optimization and comparing methods. The problem itself is often NP- hard. To be specific, we keep in mind the BOINC-based desktop grids. BOINC is an open-source software for constructing client-server grids: the server constructs tasks, sends them to computing nodes, and receives the results, while the nodes request tasks and return the answers. A client software is installed on the nodes; it uses idle resources of the client. Desktop grids can be divided in two classes: volunteer computing projects and enterprise desktop grids. In volunteer computing the organizer needs to fight malefactors, attract volunteers, a reward system for participating in the project or for obtaining the desired result is necessary. In an enterprise desktop grid efficiency is most important; availability of nodes can be predicted more precisely, their total number is known, as well as their performance, amount of memory, installed software and operation system. Such grids are mush less studied. In this article we give a review of challenges of scheduling in desktop grids and methods of that focusing on enterprise desktop grids. The scheduling problem is NP-hard, so usually different heuristics are proposed or genetic algorithms are applied. Popular metrics are: makespan (time between preparing a task and receiving its answer), throughput (number of tasks solved per a time unit), total time needed to solve all tasks, maximal load, homogeneity of the load, reliability (risk of errors), average cost, deadline violation. Algorithms are tested on simulators and emulators.

Keywords: scheduling, desktop grid, BOINC

This work was supported by RFBR grants 16-07-00622 and 15-29-07974