УДК 004.94:519.876.5

Динамическое управление вычислениями в распределенных системах

С.А. Лупин, А.В. Пачин, О.А. Кострова, Д.А. Федяшин

Национальный исследовательский университет «МИЭТ»

lupin@miee.ru

***Аннотация* — Рассматриваются вопросы повышения эффективности распределенных вычислительных сред с помощью метода, использующего динамическое переопределение параметров выполняемых узлами сети подзадач. Реализация метода основана на многократном запуске оптимизационных приложений. Представлены результаты аналитической оценки эффективности метода, а также результаты тестирования программной реализации, подтверждающие его практическую эффективность.**

***Ключевые слова* — распределенные вычисления, GRID системы, математическое моделирование.**

I. Введение

Распределенные вычисления получают все большее распространение в современном мире. Вычислительные сети, состоящие из тысяч узлов, формируются для осуществления вычислительно сложных расчетов. Одним из методов организации распределенной вычислительной среды являются GRID системы. Несомненным достоинством таких систем является их высокая масштабируемость и маневренность, низкая стоимость как на этапе создания, так и на стадии эксплуатации. Под маневренностью в данном случае понимается возможность оперативной трансформации вычислительного пространства.

Однако при всех преимуществах у GRID систем есть и существенный недостаток – невозможность управления ходом вычислений после запуска приложения. Это приводит к тому, что даже в том случае, когда узлы сети будут выполнять неэффективную работу, мы не сможем остановить расчеты. Решением проблемы может стать корректировка хода вычислительного процесса путем дробления основного задания на более мелкие с вычислительной точки зрения подзадачи с последующим анализом промежуточных результатов.

II. Распределенные вычисления

Отличительной особенностью, определяющей основные преимущества распределённых вычислений, является то, что в качестве отдельных узлов вычислительной системы могут быть использованы обычные персональные компьютеры и рабочие станции. Это позволяет получить вычислительные мощности, сопоставимые с ресурсами суперкомпьютера, но с гораздо меньшими затратами.

Высокопроизводительные вычислительные системы становятся обычным инструментом исследователя и инженера. Однако любая организация, становясь обладателем вычислительного кластера, не использует его постоянно (в режиме «365/24/7»). Более того, достаточно часто значительная часть дорогостоящих вычислительных ресурсов простаивает. В какой-то мере эти проблемы свойственны и распределенным вычислениям, использующим добровольно предоставляемые ресурсы.

GRID позволяет совместно использовать вычислительные ресурсы, которые принадлежат различным организациям и частным лицам. В GRID могут объединяться разнородные вычислительные ресурсы: персональные компьютеры, рабочие станции, кластеры и даже суперкомпьютеры. Для GRID систем также остро стоит вопрос эффективного использования объединенных в единую среду ресурсов.

GRID – это форма распределенных вычислений, при котором множество географически распределенных, слабо связанных гетерогенных вычислительных ресурсов соединены через сеть для решения ресурсоемких вычислительных задач или обработки больших массивов данных. Идея GRID-систем аналогична идее гибридной электрической сети, где каждый клиент может подключаться к ней и либо получить электроэнергию, либо отдавать в сеть ее излишки. Таким образом, предоставляя для общего использования относительно небольшие локальные ресурсы, участник добровольных вычислений может получить для своих расчетов вычислительную мощность всех узлов сети.

Сетевая архитектура GRID-системы может объединять несколько административных доменов, принадлежащих разным организациям, университетам или локальным сетям обычных пользователей. Обычно GRID не имеет централизованного контроля над ресурсами, которые объединены через локальную сеть, глобальную сеть (WAN) или через сеть Интернет. По этой причине GRID-системы обладают большой масштабируемостью и гибкостью. Географический охват может изменяться от небольшого количества локально соединенных компьютеров в одной

организации или даже одного компьютерного класса в университете до таких глобальных проектов, как SETI@home (проект по поиску внеземного разума) с более чем миллионом участниками.

Одной из наиболее распространенных платформ для реализаций распределенных вычислений является платформа BOINC [1, 2]. Основной задачей программного обеспечения BOINC является управление работой тысяч узлов единого GRID пространства. Модульный подход дает серверу BOINC гибкость, необходимую для обеспечения требуемого уровня безопасности, отказоустойчивости и производительности при больших нагрузках.

Система BOINC состоит из программы-клиента, общей для всех BOINC-проектов, составного сервера, который физически может состоять из нескольких отдельных компьютеров, и прикладного программного обеспечения. Для выполнения распределенных вычислений используется архитектура клиент-сервер. В основу архитектуры BOINC заложена идея конечного автомата – сервер состоит из набора отдельных подсистем, каждая из которых отвечает за свою вполне конкретную задачу, к примеру, выполнение вычислений, передачу файлов и т.д. Каждая из подсистем проверяет состояние подзадачи, производит какие-то действия и изменяет состояние подзадачи – так они работают в бесконечном цикле [3].

В ходе тестирования распределенной системы на платформе BOINC были выявлены некоторые особенности ее работы: сложность определения уровня загруженности узлов, относительно долгое время «подхвата» задачи конечными вычислительными узлами, сильная зависимость параметров системы от качества сети. Можно считать, что платформа BOINC лучше подходит для управления масштабными проектами уровня LHC@home (расчеты для исследований большого адронного коллайдера [4]), а не для небольших проектов.

Для устранения отмеченных недостатков BOINC систем при работе с небольшими проектами может быть использован метод динамического управления вычислениями [5], который позволит повысить эффективность использования узлов в GRID системах.

В основе предлагаемого подхода лежит многократный запуск распределенных приложений с переопределением заданий при каждом очередном запуске. Такой подход может быть использован, например, при решении задач поиска экстремума функций с помощью алгоритма перебора вариантов.

На каждой стадии работы приложения зона поиска равномерно распределяется между всеми GRID-узлами, и по завершении выбирается подобласть, в которой получен наилучший результат. На следующей стадии поиск оптимума локализуется только в этой области, а шаг изменения параметров функции уменьшается. Процесс продолжается до тех пор, пока не будет достигнута требуемая точность решения или не будет исчерпан лимит времени. В следующем разделе мы поясним работу метода на примере.

III. Оценка эффективности динамического управления вычислениями

В качестве примера будем рассматривать поиск экстремума многопараметрической функции в ограниченной области. Для простоты будем искать решение в n-мерном кубе.

Пусть задана многопараметрическая функция $F = f(x_1,....x_n)$ и необходимо найди ее экстремум.

Оценим эффективность метода динамического управления работой распределенного приложения при следующих параметрах:

n – количество параметров функции;

$0 < x_i < x_{\max}$ – диапазон изменения параметров;

Δx – шаг изменения значений параметров;

t_{exe} – время вычисления значения функции в одной точке n-мерного пространства;

$t_{restart}$ – время повторного запуска приложения;

M – количество узлов в распределенной среде;

N – число запусков приложения;

q – кратность изменения шага вычислений.

Количество точек n-мерного пространства P (*Points*), которые должны быть проанализированы для нахождения экстремума при $N = 1$, можно определить как:

$$P = P_1 = \left(\frac{x_{\max}}{\Delta x} \right)^n . \quad (1)$$

Соотношение (1) справедливо для случая n-мерного куба с равным значением Δx для всех параметров.

При этом время нахождения экстремума функции $F = f(x_1,....x_n)$ составит:

$$T(N, M) = T(1,1) = P \cdot t_{exe} . \quad (2)$$

Тогда для вычислительного пространства, содержащего M узлов, время решения задачи будет определяться как:

$$T(1, M) = T(1,1) / M = (P \cdot t_{exe}) / M . \quad (3)$$

Соотношение (3) справедливо для гомогенной среды и равномерного распределения нагрузки в ней.

Оценим время работы приложения при использовании метода многократного запуска. Число анализируемых точек пространства для $N = 2$ составит:

$$P = P_1 + P_2 = \left(\frac{x_{\max}}{q \cdot \Delta x}\right)^n + \frac{1}{M}\left(\frac{x_{\max}}{\Delta x}\right)^n =$$

$$= \left(\frac{1}{q^n} + \frac{1}{M}\right) \cdot \left(\frac{x_{\max}}{\Delta x}\right)^n \quad . \qquad (4)$$

Поскольку сложность задачи нахождения экстремума функции определяется числом анализируемых точек, для оценки получаемого ускорения S_P найдем отношение значений, вычисленных по формулам (2) и (4):

$$S_P = \frac{1}{\left(\frac{1}{q^n} + \frac{1}{M}\right)} = \left(\frac{M \cdot q^n}{M + q^n}\right) . \qquad (5)$$

При соотношении значений параметров этого соотношения, используемых на практике ($M > 10, q \le 10, n \ge 3$), можно приближенно оценить получаемое ускорение:

$$S_P \approx M \quad . \qquad (6)$$

Для нахождения времени решения используем (2). С учетом времени повторного запуска приложения получим:

$$T = T_1 + T_2 + t_{restart} . \qquad (7).$$

С учетом соотношения (3) общее ускорение S при двукратном использовании метода повторного запуска приложений по сравнению с вычислениями на одном узле составит:

$$S = S_P \cdot S_M \approx M \cdot M = M^2 \quad . \qquad (8).$$

Выражение (8) не учитывает время повторного запуска приложения и может быть использовано для задач, требующих значительных ресурсов вычислительной среды.

Распространяя рассуждения, аналогичные (1) – (8), для случаев $N > 2$ приходим к выводу, что:

$$S(N) = S_P \cdot S_M \approx M^N \quad . \qquad (9).$$

Для аналитической оценки эффективности метода многократного запуска приложений в среде MATLAB построена математическая модель, позволяющая находить оптимальные параметры вычислительного процесса для различных параметров среды и исследуемой функции.

IV. Программная реализация алгоритма и результаты тестов

В качестве тестовой использована функция вида:

$$F(z) = -\left(\frac{\sin(x) \cdot \sin(y)}{x \cdot y} + 2\sin(x)\right) . \qquad (10)$$

График анализируемой функции и точка экстремума представлены на рис. 1.

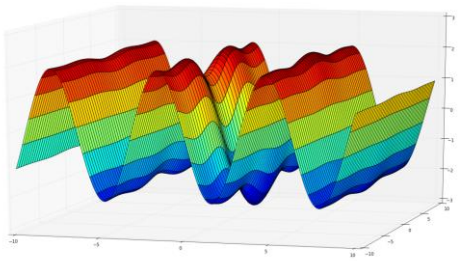

Рис. 1. График функции

Результаты вычислительных экспериментов, представленные в таблице 1, подтверждают возможность применения метода многократного запуска приложения для решения оптимизационных задач.

При запуске задачи на одном узле время решения составило более 11 часов. Отметим, что для нахождения экстремума функции мы использовали наиболее универсальный алгоритм поиска, инвариантный к исследуемой функции. Конечно, градиентные методы могут решать подобные задачи быстрее, но у них есть ограничения – функция должна быть дифференцируема во всем диапазоне значений аргументов.

Метод поиска экстремума, основанный на анализе множества точек в n-мерном кубе, хорошо масштабируется. Это подтверждают результаты его реализации на 5 узлах распределенной вычислительной системы. Время решения задачи составило немногим более 2 часов или в 5 раз меньше, чем на одном узле. Отметим, что время работы узлов не одинаково, хотя каждый из них анализирует равное количество точек. Это можно объяснить тем, что время вычисления тригонометрических функций зависит от аргумента. В более простых функциях этот эффект не будет проявляться.

Рассмотрим теперь результаты двукратного запуска приложения. На первом этапе для локализации пространства поиска используется увеличенный в 10 раз шаг изменения параметров функции. Это позволяет в 100 раз уменьшить время выполнения приложения и найти перспективную область. В таблице она выделена цветом. Отметим, что пространство поиска ограничивается только по одной переменной. При большом числе узлов GRID системы возможно ограничить область и по нескольким параметрам. На втором этапе работы алгоритма используется заданный шаг, но уже для ограниченного пространства. Поскольку время работы приложения пропорционально числу анализируемых точек (2), ускорение вычислений составляет величину порядка 25, что согласуется с результатом, который дает выражение (8).

Таблица 1

Результаты работы распределенного приложения (q = 10)

N	Δx	M_i	$x_{\min}, x_{\max}$	T	$\min F = f(x)$	x	y	S
1	0.01	1	-500..500	41608	-2.6751951	1.3799	$-4.5474*10^{-10}$	-
1	0.01	1	-500..-300	7540	-2.0007	-300.02	-4.49	5.2
		2	-300..-100	8010	-2.002	-105.24	-4.48	
		3	-100..100	7501	-2.675	1.380	$-4.547*10^{-10}$	
		4	100..300	7553	-2.008	102.1	$-4.547*10^{-10}$	
		5	300..500	7599	-2.003	303.16	$-4.547*10^{-10}$	
2	0.1	1	-500..-300	77.67	-2.00068	-306.03	-4.49	25.1
		2	-300..-100	79.89	-2.00174	-117.8	-4.49	
		3	-100..100	74.61	-2.67479	1.3999	$-1.13686*10^{-10}$	
		4	100..300	77.95	-2.00879	102.1	$-1.13686*10^{-10}$	
		5	300..500	76.65	-2.003	328.3	$-1.13686*10^{-10}$	
	0.01	1	-100..-60	1508.9	-2.003	-61.26	4.49	
		2	-60..-20	1576.5	-2.009	-23.56	4.49	
		3	-20..20	1501.7	-2.6751951	1.38	$5.115907*10^{-11}$	
		4	20..60	1499.2	-2.048	20.42	$5.115907*10^{-11}$	
		5	60..100	1525.4	-2.015	64.4	$5.115907*10^{-11}$	

Необходимо отметить, что повторный запуск приложения осуществлялся вручную, на что требуется порядка 10 минут. Конечно, это не дает возможности точно оценить время $t_{restart}$., но не сказывается на справедливости вывода о практической эффективности метода динамического планирования нагрузки узлов GRID систем. Выделение перспективной области и формирование задания для повторного запуска выполнялось автоматически.

Разработанная модель позволяет решать задачу нахождения оптимальных параметров запуска распределенного приложения. Например, можно определить размер вычислительного пространства, необходимого для реализации приложения в заданное время. Для этого необходимо осуществить пробный запуск и оценить время выполнения расчета значения функции в одной точке t_{exe} .

V. Заключение

При исследовании эффективности метода многократного запуска приложения размер вычислительного пространства был сознательно ограничен для того, чтобы сосредоточить внимание на вопросах динамического управления вычислительной сложностью подзадач, а не на масштабируемости. Последняя не требует доказательств, поскольку опирается на основополагающие свойства GRID вычислений.

В экспериментах на каждом узле работало однопоточное приложение, однако GRID системы могут включать и узлы, содержащие ускорители Intel Xeon Phi. Это позволяет использовать гибридную модель программирования, но трудно администрируется в BOINC. В ходе дальнейших исследований планируется проведение экспериментов по реализации различных этапов расчетов на разных узлах распределенной системы. Это позволит нивелировать разнородность узлов и повысить эффективность их использования.

Кроме оптимизационных, метод многократного запуска приложения может быть использован и для решения других классов задач. В качестве следующего шага исследований будет проведена оценка эффективности предлагаемого подхода для задачи планирования топологии беспроводных сетей.

Поддержка

Статья подготовлена в рамках исследовательского проекта РФФИ № 16-07-01055\165 "Адаптация ресурсоемких алгоритмов к распределенной вычислительной среде".

Литература

[1] D. Anderson, "BOINC: A System for Public-Resource Computing and Storage", Proceedings of the 5th IEEE/ACM International GRID Workshop

[2] D. P. Anderson, "Volunteer computing: the ultimate cloud, " Crossroads, vol. 16, no. 3, pp. 7-10, Mar. 2010. [Online], Available: http://doi.acm.org/10.1145/1734160.1734164
[3] D. Ferreira, F. Araujo, and P. Domingues, "libboincexec: A generic virtualization approach for the boinc middleware, " in Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum, ser. IPDPSW '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1903-1908
[4] McIntosh E., Schmidt F., de Dinechin. F. Massive tracking on heterogeneous platforms // 9th International Computational Accelerator Physics Conference (ICAP), October 2006
[5] Sergey Lupin, Aleksandr Pachin, Olga Kostrova, Dmitriy Fedyashin. Using the Applications' Reentering for Improvement the GRID Computing Efficiency // Proceedings of the NW Russia Young Researchers in Electrical and Electronic Engineering Conference, St.Petersburg, Russia, 2016.

Dynamic management of computations in distributed systems

S. Lupin, A. Pachin, O. Kostrova, D. Fedyashin

National Research University of Electronic Technology, Zelenograd, Moscow, Russia

lupin@miee.ru

ABSTRACT

Keywords – distributed computations, GRID, mathematical modeling.

In this article, we solve the problem of increasing the efficiency of the distributed computing environment by using a method that dynamically redefines parameters of the subtasks, which are running on the network nodes. The implementation of this method is based on multiple run of the optimization applications. The results of theoretical estimation and program implementation show practical effectiveness of this method.

There is a task of using effectively the resources of GRID system, which exploits one single integrated environment. In order to eliminate some problems of BOINC systems for small projects, dynamic computing control method, which increases the efficiency of the nodes in the GRID systems, can be used.

The idea of the proposed method is to launch distributed applications repeatedly when redefining the tasks each time. This approach can be used for solving function optimization problems when the variants enumeration algorithm is applied.

The algorithm can be described as follows: at first, we define the searching area and distribute it uniformly among the GRID nodes, and then we choose sub-area, which contains the best result. Next, we decrease the step size of function parameters and use the aforementioned procedure to continue optimum searching. The process continues until the required accuracy of solution or time limit is reached.

In order to evaluate the effectiveness of the proposed method, we have built a MATLAB mathematical model that helps us to find the optimal parameters. The results of the computation experiments show the possibility of using this method for solving optimization problems.

To find the extremum of the function, we use the most universal searching algorithm. Despite the fact that the application execution time equals 11 hours on one node, we use this algorithm because of the following advantages. It is invariant to the test function, therefore it can be used when the function is not differentiable on the entire domain what is mandatory for the gradient methods that can solve such problems more quickly.

Experimental results show the advantages of using the proposed algorithm. In order to localize the search area at first stage we use the initial step size multiplied by 10.

This allows us to reduce the execution time by 100 times and to select a perspective search area. Note that we limit this area by using only one variable; it is possible to limit the area by using several ones when a large number of nodes are used. On the last stage we apply the algorithm to a previously limited area. Since the runtime is proportional to the number of analyzed points, the distinguished acceleration is equal to 25, which is consistent with the theoretical evaluation.

REFERENCES

[1] D. Anderson, "BOINC: A System for Public-Resource Computing and Storage", Proceedings of the 5th IEEE/ACM International GRID Workshop
[2] D. P. Anderson, "Volunteer computing: the ultimate cloud, " Crossroads, vol. 16, no. 3, pp. 7-10, Mar. 2010. [Online], Available: http://doi.acm.org/10.1145/1734160.1734164
[3] D. Ferreira, F. Araujo, and P. Domingues, "libboincexec: A generic virtualization approach for the boinc middleware, " in Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum, ser. IPDPSW '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1903-1908
[4] McIntosh E., Schmidt F., de Dinechin. F. Massive tracking on heterogeneous platforms // 9th International Computational Accelerator Physics Conference (ICAP), October 2006
[5] Sergey Lupin, Aleksandr Pachin, Olga Kostrova, Dmitriy Fedyashin. Using the Applications' Reentering for Improvement the GRID Computing Efficiency // Proceedings of the NW Russia Young Researchers in Electrical and Electronic Engineering Conference, St.Petersburg, Russia, 2016.