Оценка эффективности методов балансировки нагрузки в распределенных вычислительных системах

Мин Тху Кхаинг, С.А. Лупин, Аунг Тху

***Аннотация*— В настоящее время распределенные вычисления являются одним из популярных подходов, который широко применяется при проведении научно-исследовательских работ, связанных с использованием ресурсоемких приложений. Доступные распределенные вычислительные системы опираются на современные GRID-технологии. Основной проблемой при их использование, особенно для гетерогенных систем, является обеспечение равномерной загрузки всех узлов, участвующих в совместном решении задачи. Это достигается за счет различных методов балансировки нагрузки или оптимального распределения задач между узлами вычислительной системы. В статье представлены результаты сравнения трёх методов распределения нагрузки. Определено, что наиболее эффективен метод, основанный на учёте реальной производительности узлов. Полученные результаты могут быть полезны широкому кругу научных работников.**

***Ключевые слова*— распределенная вычислительная система; грид-технологии; параллельные вычисления; балансировка нагрузки.**

I. Введение

Современные компьютерные технологии в самых разных областях базируются на ресурсоёмких алгоритмах, что делает возможным их реализацию только на высокопроизводительных вычислительных системах (ВВС или High Performance Computing - HPC). Доминирующей платформой при создании ВВС сегодня являются многоядерные процессоры и ускорители. На их основе строятся как узлы суперкомпьютеров, так и мощные рабочие станции. Конечно, именно такие системы в состоянии обеспечить максимально достижимую на сегодняшний день производительность, но требуют очень высоких финансовых затрат на их реализацию. На наш взгляд, это и является основной причиной того, что не угасает интерес и к гораздо менее затратным, даже условно бесплатным, решениям, основанным на распределенных вычислениях. За подобными технологиями создания вычислительных сред закрепилось название GRID-системы. В этой статье речь пойдет именно о них.

В архитектуре распределенных систем определяют три уровня:

Статья получена 28 сентября 2021.Статья подготовлена в рамках гранта РФФИ № 19-01-00666 "Современные высокопроизводительные методы оптимизационного моделирования".

Мин Тху Кхаинг - аспирант Национального исследовательского университета «МИЭТ»; (email: minthukhaing55@gmail.com)
Аунг Тху – аспирант Национального исследовательского университета «МИЭТ»; (email: aungaungthu61050@gmail.com)
С.А. Лупин, профессор, Национальный исследовательский университет «МИЭТ», (e-mail: lupin@miee.ru).

- аппаратный уровень (компьютеры, коммуникации, периферийные устройства);
- программный уровень (операционная среда, приложения);
- пользовательский уровень (порядок взаимодействия с интегрированной средой).

В основе работы распределенных систем лежит принцип параллельного исполнения приложения на всех узлах вычислительной сети. При этом необходимо учитывать, что узлы сети обладают различной производительностью, т.е. GRID-система в общем случае является неоднородной или гетерогенной. Кроме того, различные участки сети имеют разную топологию и коммуникационную среду, разные версии операционных систем. В крупных проектах типа Seti@home (проект добровольных вычислений на платформе BOINC) общее число участников составляло более 1,4 млн., а узлы сети были расположены по всему миру [1,2]. Этот и подобные ему проекты поддерживаются большими группами ученых, но не менее популярны и проекты, реализуемые в рамках отдельной лаборатории и использующие для вычислений доступные персональные компьютеры и рабочие станции сотрудников института.

Параллельные вычисления начинаются с разбиения больших задач на отдельные подзадачи (часто с помощью параметризации), которые могут выполняться одновременно множеством процессоров в узлах сети. Во время работы процессоры могут взаимодействовать через общую память или через сеть. Результат параллельных вычислений получается посредством объединения локальных результатов каждого узла в рамках общего алгоритма [3]. Параллельные вычисления сегодня являются основным методом повышения производительности вычислительных систем и минимизации времени выполнения программы.

В распределенных вычислениях компьютеры в одной сети совместно используют один или несколько ресурсов. В идеальной вычислительной системе каждый ресурс является общим, что превращает компьютерную сеть в высокопроизводительную вычислительную среду.

У параллельных и распределенных вычислений есть много схожих черт, но есть также некоторые различия, которые очень важны с точки зрения вычислений, затрат и времени. Как правило, при параллельных вычислениях в узлах имеется несколько обрабатывающих элементов (CPU, GPU, ускорители), которые могут взаимодействовать между собой через общую память, а сами узлы взаимодействуют через

специализированную сеть. В распределенных вычислениях узлы не имеют общей памяти, поэтому используют для коммуникации обычную сеть. С учетом географической распределенности узлов сети и существенной гетерогенности ее отдельных сегментов, при распределенных вычислениях обмены между узлами минимизируют. В идеале, как в проекте Seti@home, каждый узел только дважды взаимодействует с сервером – при получении задании и при отправке ответа.

Параллельные вычисления требуют дорогостоящего оборудования для координации действий процессоров в ходе вычислений, а распределенные вычисления используют доступные отдельные узлы и обычную сеть, поэтому стоимость их создания низкая. В некоторых приложениях параллельные и распределенные вычисления могут использовать одинаковые алгоритмы, но в общем случае, они существенно отличаются.

Поскольку суперкомпьютеры очень дороги и не всегда доступны небольшим группам исследователей, появилась новая альтернативная концепция, называемая параллельными распределенными вычислениями. Для достижения максимальной эффективности такой системы рабочая нагрузка должна быть распределена между всеми узлами сети. Распределение нагрузки на обрабатывающие узлы называют еще задачей балансировки нагрузки. Для многопроцессорных вычислительных систем с распределенной памятью, к которым относятся и GRID-системы, проблема балансировки нагрузки является актуальной [4-7].

Балансировка нагрузки влияет на множество параметров распределенной вычислительной системы:

- повышает утилизацию каждого узла и, следовательно, общую производительность системы;
- сокращает время ожидания заданий в очереди;
- позволяет использовать и узлы сети с невысокой производительностью;
- максимизирует использование ресурсов среды;
- снижает время отклика вычислительной системы;
- повышает пропускная способность GRID-системы;
- повышает надежность вычислений;
- позволяет получить прирост производительности вычислительной системы без материальных затрат;
- позволяет прогнозировать рост производительности при расширении системы.

Благодаря очевидным преимуществам балансировка нагрузки является областью интенсивных исследований. В последние годы было разработано много различных балансировщиков, но ни один из них не является инвариантным по отношению к архитектуре вычислительной системы и решаемой задаче. Выбор правильного балансировщика нагрузки зависит от параметров приложения, таких как качество балансировки, шаблоны генерации нагрузки, а также от аппаратных параметров, таких как накладные расходы на связь.

В этой статье представлены результаты исследования простого метода балансировки нагрузки узлов в существенно гетерогенной системе, основанного на учете их пиковой и реальной производительности.

II. Распределенная вычислительная среда

Исследование эффективности балансировки нагрузки узлов проводилось в вычислительной среде, которая была организована в студенческом общежитии МИЭТ с использованием GRID-технологий [8]. Созданная распределенная вычислительная система объединяет разнородные персональные компьютеры, расположенные в разных корпусах общежития. Это достаточно традиционный вариант реализации GRID-вычислений - использование обычных персональных компьютеров или рабочих станций. Такой вариант распределённой ВВС требует учёта особенностей администрирования локальной сети и архитектуры процессоров в узлах сети.

На рисунке 1 показана логическая схема распределенной системы. Для настройки и управления вычислительной средой необходимо использовать некоторый инструментарий, обеспечивающий удаленный запуск приложения на каждом узле и сбор результатов.

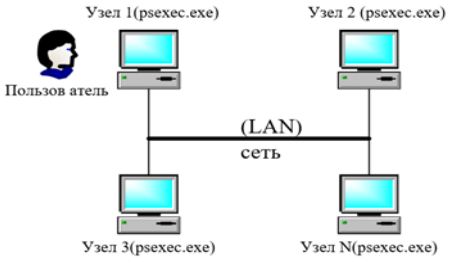

Рис. 1. Схема распределенных вычислений

Все узлы обладают различными характеристиками и соединены с помощью локальной сети. Скорость передачи данных между узлами сети составляет около 50 Мбит/с. Все рабочие узлы имеют внутренний IP адрес, в то время как локальный узел имеет как внутренний, так и внешний IP-адреса. Сеть состоит из гетерогенного оборудования, но на всех узлах установлены различные версии операционной системы Windows. Для удаленного запуска команд и процессов необходимо установить утилиту *PsExec*, чтобы службы сервера и рабочей станции функционировали на удаленном и локальном компьютерах, а на удаленном компьютере должен быть доступен стандартный общий ресурс Admin$. Утилита только работает под ОС Windows.

III. Тестовое приложение

В качестве тестового приложения мы используем задачу численного интегрирования по методу трапеций. Такой выбор объясняется, с одной стороны, возможностью её параллельной реализации в распределенной среде в виде невзаимодействующих процессов, а с другой стороны, существованием простого механизма распределения нагрузки между узлами – параметризации.

В проводимых исследованиях не предъявлялись специальные требования к интегрируемой функции. В частности, мы не использовали методы, основанные на изменении шага интегрирования в зависимости от градиента функции, поскольку это делает невозможным применять статическую балансировку нагрузки. Ниже показана выбранная для исследований функция, содержащая основные математические операции:

$$Y = \int_a^b \frac{1}{X^2+1} dx.$$

При вычислениях используется следующая формула:

$$\int_a^b f(x)dx \approx \frac{h}{2} \cdot \sum_{i=1}^{N} \big(f(i \cdot h) + f(i \cdot h + h)\big); (1)$$

$$h = (b - a)/N.$$

Отметим, что в качестве параметров, определяющих вычислительную нагрузку узла, можно использовать два равноправных метода:

- узлу передается число шагов и интервал, шаг интегрирования определяется самим узлом;
- узлу передается шаг интегрирования и интервал, число шагов определяется самим узлом.

С учетом гетерогенности вычислительной среды, теоретически эти методы могут давать разную точность, но этот вопрос в работе не исследовался.

IV. Организация удаленного запуска приложений на узлах сети

Запуск тестового приложения осуществлялся удаленно с одного из узлов (ведущего), с помощью утилиты PsExec [9]. Для использования утилиты необходимо скопировать ее в папку с исполняемыми файлами и запустить из любой оболочки командной строки: Cmd или PowerShell. Принцип работы программы состоит в следующем: в ресурсах исполняемого файла PsExec.exe помещаются еще один исполняемый файл – PSEXESVC, который является службой Windows. Перед выполнением команды PsExec распаковывает этот ресурс в скрытую административную папку удалённого компьютера Admin$ в файл C:\Windows\system32\psexesvc.exe.

Распределенное приложение для вычислительной среды написано на языке C++. Для его запуска на удаленном компьютере используется следующая команда:

START /B CMD /C CALL psexec \\ip-address -u login -p pass -s -c -f myprog.exe parameters > localpc_output.txt,

где:

START /B CMD /C CALL - запуск командной строки на узле;

psexec - вызов утилиты удаленного управления узлом;

\\ **ip-address** – ip-адрес вычислительного узла в сети;

-u login -p pass – логин и пароль администратора на локальном узле;

s – запуск удаленного процесса с правами системной учетной записи;

c – копирование файла указанной программы (myprog.exe) на вызывающий узел для выполнения;

f – файл с указанной программой будет скопирован, даже если он уже существует на этом узле;

myprog.exe – приложение, которое будет запущено на узле под операционной системой Windows;

parameters – параметры приложения;

localpc_output.txt – путь к директории, куда помещаются полученные на узлах результаты.

В узлах распределенной вычислительной сети расположены компьютеры, имеющие многоядерные процессоры, поэтому для их эффективной загрузки требуется многопоточное приложение. Оно было разработано с использованием технологии OpenMP (Open Multi-Processing), которая является одним из самых популярных методов программирования многоядерных и многопроцессорных систем с общей памятью. OpenMP поддерживает языки C, C++ и Fortran [10]. При использовании OpenMP все потоки совместно используют память и данные. Приложение **myprog.exe** – это многопоточная программа, позволяющая с помощью передаваемых ей параметров изменять число потоков.

Проводимые эксперименты предполагают многократный запуск приложения. Для снижения трудоемкости этого процесса в командной строке ведущего узла можно вызывать bat-файл, где вышеописанная команда вызывается для множества узлов с различными параметрами.

Содержание bat-файла для M узлов сети:

START /B CMD /C CALL psexec \\ip1-address -u login -p pass -s -c -f myprog.exe parameters > ip1_output.txt.

START /B CMD /C CALL psexec \\ip2-address -u login -p pass -s -c -f myprog.exe parameters > ip2_output.txt.

.....

START /B CMD /C CALL psexec \\ipM-address -u login -p pass -s -c -f myprog.exe parameters > ipM_output.txt.

V. Характеристики узлов сети

Для оценки эффективности различных методов балансировки нагрузки были проведены серии экспериментов в распределенной среде, включающей 10 разных узлов, характеристики которых представлены в таблице 1.

Таблица 1. Узлы вычислительной сети

Узлы	CPU	Число ядер	Производительность, P (GFlops)	
			$Pcore$	$Pprc$
1	i5-4460	4	25,6	102,4
2	i7-8700	6	51,2	307,2
3, 4, 7, 8	i3-3220	2	26,4	52,8
5	i3-8100	4	57,6	230,4
6	i5-3470	4	25,6	102,4
9	i5-2500K	4	19,8	79,2
10	i5-9400F	6	46,4	278,4

В общем случае производительность вычислительной системы характеризуется различными показателями. В зависимости от целевой функции это могут быть:

- время отклика;
- пропускная способность;
- утилизация вычислительных ресурсов;
- время передачи данных.

Проводимые исследования направлены на повышение производительности гетерогенной РВС за счёт оптимизации нагрузки её узлов, что повышает утилизацию вычислительных ресурсов.

В качестве оценки вычислительной мощности узла выступает значение пиковой производительности его процессора. В таблице 1 показана производительность одного ядра и процессора узлов сети.

VI. Методы балансировки нагрузки и экспериментальные результаты

Для балансировки нагрузки узлов вычислительной системы используются статические и динамические методы. Статическая балансировка в вычислительной системе происходит до запуска программы, при этом структура системы не меняется во время работы программы. В отличие от статических методов, где система определяет только начальное распределение задач между узлами системы, нагрузка узлов в динамической системе может изменяться во время работы приложения [11]. При таком типе балансировки можно учитывать изменения в нагрузке узлов вычислительной системы, происходящие в процессе работы приложения. Динамическая балансировка использует и перенос некоторых вычислений на менее загруженные узлы, что может повысить эффективность GRID-системы в целом. Следует отметить, что применение динамической балансировки при добровольных вычислениях можно оценить как трудно реализуемое.

При решении задачи балансировки нагрузки узлов необходимо учитывать однородность узлов вычислительной сети. Для гомогенной сети можно использовать равномерное распределение нагрузки. Гетерогенные сети, особенно с высоким уровнем неоднородности узлов, требуют учета их производительности.

В настоящей работе проведён сравнительный анализ трёх методов статической балансировки нагрузки. Первый основан на равномерном распределении, не учитывающем производительность узлов. Результаты работы приложения показаны в таблице 2.

Таблица 2. Вычисления при равномерной нагрузке

Узел	Время вычислений (сек)	
	$Tcore(i)$	$Tprc(i)$
1	66,736	18,756
2	15,364	2,6954
3	68,889	42,3228
4	69,081	43,361
5	18,389	4,46763
6	66,607	18,1821
7	68,832	42,596
8	70,8	43,789
9	103,212	26,0994
10	16,753	3,16215
GRID	103,212	43,789

При равномерном распределении имеется значительный дисбаланс – почти семикратная разница времени работы узлов (однопоточное приложение, $Tcore(i)$), что объясняется существенной гетерогенностью узлов GRID. Отметим, что при запуске на каждом узле максимально возможного числа потоков ($Tprc(i)$), дисбаланс значительно возрастает и достигает 16-ти кратной разницы.

Второй метод базируется на распределении, учитывающем пиковую производительность вычислительных узлов. В таблице 3 представлены её теоретические значения для всех ядер и процессоров, а также коэффициенты относительной мощности, необходимые для распределения нагрузки.

Таблица 3. Учет разнородности узлов (теория)

Узел (i)	$Pcore(i)$ (Gflops)	$K1(i)$	$Pprc(i)$ (GFlops)	$K2(i)$
1	25,6	0,4444	102,4	0,3333
2	51,2	0,8889	307,2	1
3	26,4	0,4583	52,8	0,1719
4	26,4	0,4583	52,8	0,1719
5	57,6	1	230,4	0,75
6	25,6	0,4444	102,4	0,3333
7	26,4	0,4583	52,8	0,1719
8	26,4	0,4583	52,8	0,1719
9	19,8	0,34375	79,2	0,2578
10	46,4	0,8055	278,4	0,9063
Max	57,6		307,2	

При этом использованы следующие соотношения:

$$K1(i) = Pcore(i)/max\ Pcore \quad (2)$$

$$K2(i) = Pprc(i)/max\ Pprc. \quad (3)$$

А доля нагрузки или интервала интегрирования в задаче (1), передаваемая соответствующему узлу, будет определяться как:

- однопоточное приложение

$$Load(i) = K1(i)/\sum_1^{10} K1(i), \quad (4)$$

- многопоточное приложение

$$Load(i) = K2(i)/\sum_1^{10} K2(i). \quad (5)$$

Результаты работы приложения показаны в таблице 4.

Таблица 4. Вычисления при балансировке нагрузки (учет пиковой производительности узлов)

Узел	Время вычислений (сек)	
	$Tcore(i)$	$Tprc(i)$
1	52,524	13,9907
2	24,742	8,90836
3	56,512	18,941
4	55,394	19,325
5	31,831	7,90924
6	51,366	13,2491
7	55,174	20,102
8	54,933	19,521
9	58,681	17,553

10	24,058	6,58393
GRID	58,681	19,325

Результаты по отношению к равномерному распределению заметно улучшились: дисбаланс при однопоточном запуске не превосходит двух, а при многопоточном – 2,4. Время решения задачи сократилось практически в два раза. Однако и этот результат нельзя ещё считать удовлетворительным, если под целью балансировки понимать выравнивание времени работы всех узлов.

Третий метод распределения нагрузки базируется на учёте реальной производительности вычислительных узлов. В качестве оценки реальной производительности узлов использованы временные параметры, полученные при равномерном распределении (табл. 2).

Это вполне правомерно, поскольку в этом случае нагрузка у каждого узла была одинаковой.

В таблице 5 показаны коэффициенты относительной мощности узлов, рассчитанные по формулам (6) и (7).

Таблица 5. Учет разнородности узлов (практика)

Узел (i)	$Tcore(i)$ (сек)	$K3(i)$	$Tprc(i)$ (сек)	$K4(i)$
1	66,736	0,2302	18,756	0,1437
2	15,364	1	2,6954	1
3	68,889	0,2230	42,3228	0,0637
4	69,081	0,2224	43,361	0,0622
5	18,389	0,8355	4,46763	0,6033
6	66,607	0,2307	18,1821	0,1482
7	68,832	0,2232	42,596	0,0633
8	70,8	0,2170	43,789	0,0616
9	103,212	0,1489	26,0994	0,1033
10	16,753	0,9171	3,16215	0,8524
Min	15,364		2,6954	

При этом использованы следующие соотношения:

$$K3(i) = Tcore(i)/min\,Tcore \quad (6)$$

$$K4(i) = Tprc(i)/min\,Tprc. \quad (7)$$

Доля нагрузки или интервала интегрирования в задаче (1), передаваемая соответствующему узлу, будет определяться как:

- однопоточное приложение

$$Load(i) = K1(i)/\sum_1^{10} K1(i), \quad (8)$$

- многопоточное приложение

$$Load(i) = K2(i)/\sum_1^{10} K2(i). \quad (9)$$

Результаты работы приложения показаны в таблице 6.

Следует отметить, что дисбаланс нагрузки D при таком подходе практически устранен. Он не превышает 10% для однопоточного приложения и 16% для многопоточного.

В проводимых исследованиях мы не изучали причины отличия теоретической и практической производительности узлов. Из самых общих соображений можно предположить, что разница объясняется использованием одного транслятора для разных архитектур процессоров. Для одних узлов код приложения был оптимизирован, а для других нет. Другой возможной причиной может быть загруженность части ресурсов процессора приоритетными системными вызовами. Влияние этого фактора на результаты исследований устранялось за счёт трёхкратного запуска приложений.

Таблица 6. Вычисления при балансировке нагрузки (учет реальной производительности узлов)

Узел	Время вычислений (сек)	
	$Tcore(i)$	$Tprc(i)$
1	36,447	8,51662
2	36,636	8,82616
3	34,562	8,6859
4	34,566	9,02611
5	36,612	8,78523
6	37,283	8,75975
7	35,71	9,50272
8	34,554	8,8103
9	36,311	8,9256
10	37,055	8,14688
GRID	37,283	9,50272

Для удобства сравнения результаты оценки эффективности всех исследованных методов балансировки собраны в таблице 7.

Таблица 7. Сравнение эффективности методов балансировки

Метод балансировки	Время вычислений			
	1 поток		36 потоков	
	T_1 (сек)	D_1 (%)	T_2 (сек)	D_2 (%)
Равномерное распределение	103,2	85,1	43,8	93,8
Учёт пиковой производительности	58,7	59,0	19,3	65,9
Учёт реальной производительности	37,3	7,3	9,50	14,3

Неравномерность нагрузки узлов оценивается по следующей формуле:

$$D = 100 \cdot (T_{max} - T_{min})/T_{max}, \quad (10)$$

$$0 \le D < 100.$$

Распределение нагрузки, основанное на реальной производительности узлов распределенной вычислительной системы, обеспечивает максимальную эффективность вычислений и минимальный дисбаланс. При необходимости для устранения оставшейся неравномерности можно выполнить более точную настройку за несколько итераций.

VII. Заключение

Вычислительная платформа, на которой проводились эксперименты, организована с использованием технологии GRID. Она состоит из разнородных компьютеров, расположенных в здании общежития. Полученные результаты подтверждают эффективность

метода балансировки нагрузки, основанного на учете реальной производительности узлов. Максимум производительности GRID-системы может быть получен при запуске на узлах многопоточных приложений.

Проведение GRID-вычислений с помощью программного обеспечения, удовлетворяющего принципам free license или open license - Microsoft Visual Studio, OpenMP и PsExec, позволяют считать, что полученные результаты могут быть интересны широкому кругу пользователей.

Дальнейшие исследования будут направлены на оценку применимости гетерогенных GRID-вычислений для решения ресурсоемких задач дискретной оптимизации, например, описанных в [12, 13], а также оценке эффективности балансировки в более крупных системах, состоящих из ресурсов нескольких компьютерных классов.

Поддержка

Статья подготовлена в рамках гранта РФФИ 19-01-00666А "Современные высокопроизводительные методы оптимизационного моделирования"

Библиография

[1] Sheng Y., Gui L., Wei Z., Duan J., and Liu Y. Layered Models for General Parallel Computation Based on Heterogeneous System. 2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies. doi:10.1109/pdcat.2012.85.

[2] Md. Firoj Ali. Distributed Computing: An Overview, Int. J. Advanced Networking and Applications Volume: 07 Issue: 01 Pages: 2630-2635 (2015) ISSN: 0975-0290.

[3] Бабичев С. Л. Распределенные системы: учебное пособие для вузов / С. Л. Бабичев, К. А. Коньков. — Москва : Издательство Юрайт, 2019. — 507 с. — (Высшее образование). — ISBN 978-5-534-11380-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/445188 (дата обращения: 25.09.2021).

[4] A. Chhabra, G. Singh, S.S. Waraich, B. Sidhu and G. Kumar. Qualitative Parametric Comparison of Load Balancing Algorithms in parallel and Distributed Computing Environment, Word Academy of Science, Engineering and Technology, 2006, 39-42.

[5] D.Z. Gu, L. Yang and L.R. Welch. A Predictive, Decentralized Load Balancing Approach, in: Proceedings of the 19th IEEE International Paralleland Distributed Processing Symposium, Denver, Colorado, April 2005, 04-08.

[6] Tabuk. Load Balancing in Distributed Computer Systems, International Journal of Computer Science and Information Security ,Vol. 8, No. 4, 2010.

[7] Md. Firoj Ali1, Rafiqul Zaman Khan. The study on load balancing strategies in distributed computing system, International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.2, April 2012.

[8] Лупин С.А., А.В. Пачин, О.А. Кострова, Д.А. Федяшин. Динамическое управление вычислениями в распределенных системах. // Проблемы разработки перспективных микро- и наноэлектронных систем - 2016. Сборник трудов / под общ. ред. Академика РАН А.Л.Стемпковского. - М.: ИППМ РАН, 2016. Часть II - 185 с., С. 185-189

[9] JJ'S WEBSITE, PsExec Simple Tutorial, [Online]. Доступ: http://jjasser.com/2014/04/psexec-simple-tutorial/.

[10] OpenMP technology (Open Multi-Processing) [Online]. Доступ: https://ru.wikipedia.org/wiki/OpenMP

[11] S.S. Manvi and M.N. Birje. A Review on Wireless Grid Computing, International Journal of Computer and Electrical Engineering, Vol. 2, No. 3, June, 2010.

[12] Sergey Lupin, Aleksandr Pachin, Olga Kostrova, Dmitriy Fedyashin. Using the Applications' Reentering for Improvement the GRID Computing Efficiency / Proceedings of 2016 IEEE NW Russia Young Researches in Electrical and Electronic Engineering Conference (EIConRusNW) St. Petersburg, Russia, February 2 - 4, 2016, pp. 256 - 258, DOI: 10.1109/EIConRusNW.2016.7448168

[13] Лупин С.А., Ай Мин Тайк, Федяшин Д.А. Оптимизация топологии беспроводных сетей - решение для направленных антенн // International Journal of Open Information Technologies. №7, 2018, с. 32-38

Evaluating the effectiveness of load balancing methods in distributed computing systems

Min Thu Khaing, S. Lupin, Aung Thu

Abstract*— Nowadays, distributed computing is one of the popular approaches that is widely used in research and development related to the use of resource-intensive applications. The available distributed computing systems are based on modern GRID technologies. The main problem when using them, especially for heterogeneous systems, is to ensure uniform load of all nodes participating in the joint solution of the problem. This is achieved through various methods of load balancing or optimal distribution of tasks between the nodes of the computing system. The article presents the results of comparing three methods of load balancing. It has been determined that the most effective method is based on taking into account the real performance of the nodes. The results obtained can be useful to a wide range of scientific workers.

Keywords*— distributed computing system; grid technologies; parallel computing; load balancing.

REFERENCES

[1] Sheng Y., Gui L., Wei Z., Duan J., and Liu Y. Layered Models for General Parallel Computation Based on Heterogeneous System. 2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies. doi:10.1109/pdcat.2012.85.

[2] Md. Firoj Ali. Distributed Computing: An Overview, Int. J. Advanced Networking and Applications Volume: 07 Issue: 01 Pages: 2630-2635 (2015) ISSN: 0975-0290.

[3] Babichev S. L. Raspredelennye sistemy: uchebnoe posobie dlja vuzov / S. L. Babichev, K. A. Kon'kov. — Moskva : Izdatel'stvo Jurajt, 2019. — 507 s. — (Vysshee obrazovanie). — ISBN 978-5-534-11380-8. — Tekst : jelektronnyj // Obrazovatel'naja platforma Jurajt [sajt]. — URL: https://urait.ru/bcode/445188 (data obrashhenija: 25.09.2021).

[4] A. Chhabra, G. Singh, S.S. Waraich, B. Sidhu and G. Kumar. Qualitative Parametric Comparison of Load Balancing Algorithms in parallel and Distributed Computing Environment, Word Academy of Science, Engineering and Technology, 2006, 39-42.

[5] D.Z. Gu, L. Yang and L.R. Welch. A Predictive, Decentralized Load Balancing Approach, in: Proceedings of the 19th IEEE International Paralleland Distributed Processing Symposium, Denver, Colorado, April 2005, 04-08.

[6] Tabuk. Load Balancing in Distributed Computer Systems, International Journal of Computer Science and Information Security ,Vol. 8, No. 4, 2010.

[7] Md. Firoj Ali1, Rafiqul Zaman Khan. The study on load balancing strategies in distributed computing system, International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.2, April 2012.

[8] Lupin S.A., A.V. Pachin, O.A. Kostrova, D.A. Fedjashin. Dinamicheskoe upravlenie vychislenijami v raspredelennyh sistemah. // Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem - 2016. Sbornik trudov / pod obshh. red. Akademika RAN A.L.Stempkovskogo. - M.: IPPM RAN, 2016. Chast' II - 185 s., S. 185-189

[9] JJ'S WEBSITE, PsExec Simple Tutorial, [Online]. Dostup: http://jjasser.com/2014/04/psexec-simple-tutorial/.

[10] OpenMP technology (Open Multi-Processing) [Online]. Dostup: https://ru.wikipedia.org/wiki/OpenMP

[11] S.S. Manvi and M.N. Birje. A Review on Wireless Grid Computing, International Journal of Computer and Electrical Engineering, Vol. 2, No. 3, June, 2010.

[12] Sergey Lupin, Aleksandr Pachin, Olga Kostrova, Dmitriy Fedyashin. Using the Applications' Reentering for Improvement the GRID Computing Efficiency / Proceedings of 2016 IEEE NW Russia Young Researches in Electrical and Electronic Engineering Conference (EIConRusNW) St. Petersburg, Russia, February 2 - 4, 2016, pp. 256 - 258, DOI: 10.1109/EIConRusNW.2016.7448168

[13] Lupin S.A., Aj Min Tajk, Fedjashin D.A. Optimizacija topologii besprovodnyh setej - reshenie dlja napravlennyh antenn // International Journal of Open Information Technologies. #7, 2018, s. 32-38