Система моделирования Grid: реализация и возможности применения

Грушин Д.А., Поспелов А.И.

Аннотация. В статье описывается разработанная в ИСП РАН система моделирования распределенных вычислительных сред Grid. С помощью этой системы проведен анализ реальной вычислительной среды Sharcnet. На основе анализа были выявлены возможные способы существенного увеличения эффективности работы среды.

1. Введение

В последнее время к вычислительным кластерам проявляется повышенный интерес со стороны науки, образования и промышленности. С доступностью кластерных технологий связан рост числа установок, которые строятся, устанавливаются и которые специалисты пытаются применять для решения своих производственных задач. Кластерные вычислительные системы становятся повседневным инструментом исследователя и инженера. Однако, любая организация, становясь обладателем вычислительного кластера, не использует его постоянно, в режиме “24/7”, более того, очень часто такой дорогостоящий вычислительный ресурс простаивает.

В связи с увеличением количества кластеров набирает все большую популярность концепция Grid [12, 16]. Grid позволяет совместно использовать вычислительные ресурсы, которые принадлежат различным организациям и которые могут быть расположены в различных административных областях. В Grid могут объединяться разнородные вычислительные ресурсы: персональные компьютеры, рабочие станции, кластеры и супер-компьютеры.

Одним из наиболее распространенных в настоящее время программных средств для реализации Grid является пакет Globus Toolkit [7]. Пакет Globus Toolkit разрабатывается, поддерживается и продвигается международным альянсом разработчиков из университетов США и Великобритании, а также научных лабораторий и вычислительных центров. Globus Toolkit является свободно распространяемым с открытым исходным кодом программным пакетом и предлагает базовые средства для создания Grid инфраструктуры: средства обеспечения безопасности в распределенной среде, средства надежной передачи больших объемов данных, средства запуска и получения

результатов выполнения задач на удаленных вычислительных ресурсах. На базе пакета Globus Toolkit создаются промышленные версии реализаций Grid инфраструктуры, например, такие как Univa [19] и Platform Globus Toolkit [17].

Однако, несмотря на то, что уже сейчас предлагаются, ставшие “де-факто” стандартными, средства создания Grid-инфраструктур, существует ряд важных научных задач, в том числе и теоретических, без решения которых полномасштабное использования возможностей Grid технологий в промышленности невозможно. Одной из актуальных задач в настоящее время является эффективное управление вычислительными ресурсами в распределенной среде. С ростом числа ресурсных центров входящих в распределенную инфраструктуру, отсутствие хорошего планировщика, обеспечивающего управление потоком задач, не только значительно снижает эффективность использования всей Grid-инфраструктуры, но может сделать бессмысленным ее создание. При этом, следует отметить, что для таких распределенных систем характерным является динамичное развитие, что делает невозможным решение задачи эффективного управления “в статике” – один раз и навсегда.

С другой стороны, оптимизация алгоритмов управления распределенной средой на непосредственно уже существующей Grid-инфраструктуре затруднено и связано со значительными издержками и простоями ресурсных центров, а часто в силу масштабности распределенной среды вообще не возможно. В связи с этим, актуальной задачей является создание системы моделирования Grid-инфраструктуры, которая позволит адекватно оценивать ее поведение при изменяющихся условиях и, на основе этого, оптимизировать стратегии управления потоками задач.

Система моделирования может быть использована для оценки эффективности распределенной вычислительной среды в различных ситуациях, например:

- при изменении нагрузки: количества поступающих задач, их размерности, приоритета, периода поступления и т.д.;
- при отключении части вычислительных ресурсов или добавлении новых ресурсов;
- при увеличении количества передаваемых данных;
- при выходе из строя части коммуникационных каналов

При этом оценка эффективности управления может проводиться по следующим наиболее популярным критериям [4]:

- минимизация среднего времени ожидания задачи в очереди;
- минимизация максимального времени выполнения группы задач (makespan);
- максимизация пропускной способности – числа завершенных задач в единицу времени;

- минимизация простоев процессоров
- и т.д.

В настоящее время существует несколько проектов по разработке систем моделирования Grid. Среди них наиболее известны: Bricks [14], MicroGrid [11], OptorSim [13], SimGrid [10] и GridSim [1]. Данные системы обладают как достоинствами, так и недостатками. Среди недостатков можно отметить узкую специализацию систем, отсутствие публично доступных версий, а также ограниченность моделируемых архитектур Grid систем. Особенности реализации некоторых из них накладывают ограничения на количество одновременно существующих элементов в Grid системе и требуют от пользователя знания специальных языков программирования, что значительно снижает эффективность работы с такими системами.

2. Система моделирования Grid

С 2007 года в ИСП РАН разрабатывается система моделирования Grid. При разработке мы старались избежать недостатков присущих существующим системам, а также реализовать некоторые новые интересные идеи.

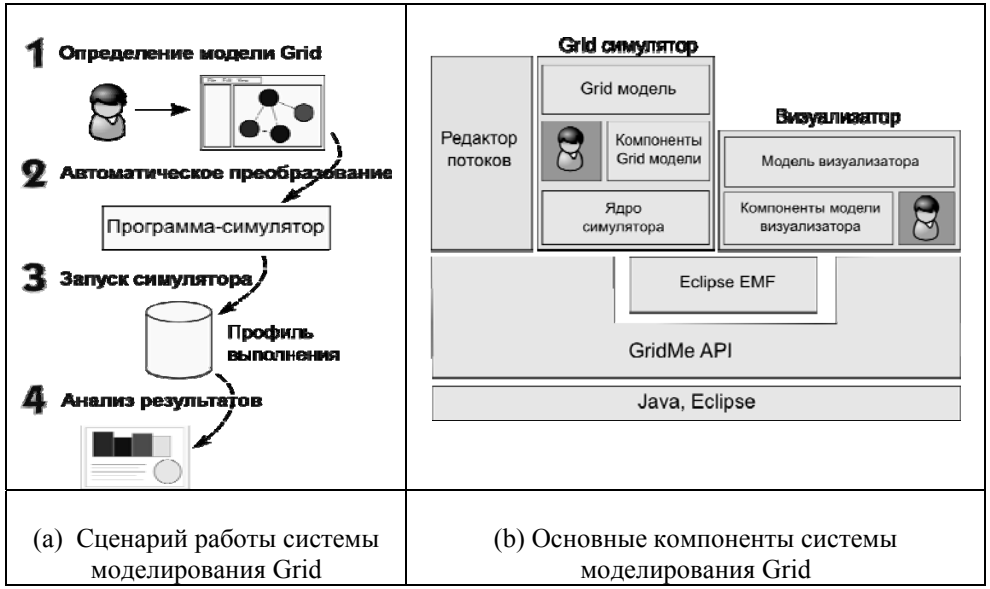
В частности, система проектировалась так, чтобы сделать работу пользователя максимально удобной и быстрой. В отличие от перечисленных выше систем в разработанной системе не нужно вручную писать программу моделирования. Пользователь работает в специальном редакторе, задавая топологию Grid системы и свойства отдельных элементов. При этом, автоматически проверяются различные виды ошибок: значения параметров, выходящие за область допустимых значений, несовместимость различных элементов, соединенных между собой, и т.п.

Сценарий работы с системой изображен на рисунке 1(а). Пользователь задает описание моделируемой среды, и указывает различные параметры. Система автоматически генерирует код программы моделирования и компилирует его. Программа-симулятор запускается и создает в результате своей работы профиль выполнения. Полученный профиль анализируется и представляется пользователю в виде HTML документа.

Система моделирования реализована на основе платформы Eclipse [2], с использованием только языка Java. Это дает возможность интеграции с другими Eclipse приложениями, например, средой разработки Java, системами контроля версий, и т.п. и позволяет использовать систему моделирования под различными операционными системами – Linux, Windows, Solaris и др.

Система расширяема и рассчитана на гибкое использование. Система позволяет моделировать различные Grid архитектуры: одно и двух-уровневые системы с одним или несколькими брокерами, добавлять хранилища данных, определять топологию сетевых соединений и т.д. Система включает в себя множество реализованных компонент, таких как брокер, кластер, поток задач

и т.д. Кроме того, пользователи могут расширять систему, реализовывая свои собственные компоненты.



(a) Сценарий работы системы моделирования Grid

(b) Основные компоненты системы моделирования Grid

Рис. 1. Система моделирования Grid.

Поведение отдельных элементов моделируется с помощью конечных автоматов, что позволяет работать с моделями больших систем – порядка тысяч процессоров и более миллиона задач.

Система предоставляет возможность для быстрого описания алгоритмов распределения задач с помощью набора правил. При моделировании распределения задач в Grid очень часто требуется проверить несколько алгоритмов, незначительно отличающихся друг от друга, например, сортировкой входного потока задач, способом выбора очередной задачи или ресурса и т.п. Описание алгоритма с помощью набора правил в такой ситуации позволяет гораздо быстрее проверить работу алгоритма, чем в случае реализации его в виде, например, Java класса, с последующей отладкой и тестированием.

В системе поддерживается возможность проведения серии экспериментов, состоящей из последовательных запусков выполняемой модели с изменением некоторых параметров при каждом следующем запуске. Например, может изменяться поток задач, конфигурация кластеров, сетевых соединений и т.п. Это позволяет в рамках одного эксперимента посмотреть динамику изменения эффективности системы и определить узкие места.

В системе реализован удобный механизм обработки результатов моделирования. Результат выполнения модели хранится в отдельном профиле и может обрабатываться независимо. Пользователь может использовать свой шаблон для выбора и визуализации только необходимой в данный момент информации. Это позволяет нескольким исследователям провести моделирование один раз, а затем независимо анализировать полученную информацию.

Система также включает в себя редактор и анализатор записей потоков задач (workload) [15, 8]. Запись потока представляет собой текстовый файл, каждая строка которого содержит характеристики отдельной задачи: время порождения, время запуска, общее время выполнения, количество занимаемых процессоров и т.д. Анализатор позволяет отобразить различные характеристики потока – количество задач, соотношение однопроцессорных и параллельных задач, график порождения задач во времени и т.п. С помощью редактора можно изменять поток – копировать и перемещать части потока, соединять несколько потоков в один, изменять характеристики группы задач и т.п. Также, редактор позволяет создавать синтетический поток по заданным параметрам.

Основные компоненты системы изображены на Рис. 1(b). Это – редактор и анализатор потоков, симулятор Grid системы, визуализатор.

2.1. Модель представления Grid системы

Модель для представления Grid инфраструктуры (мета-модель Grid) в нашей системе изображена на Рис. 2.

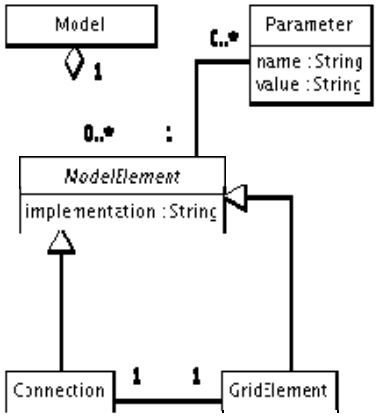


Рис. 2. Мета-модель Grid.

Модель Grid состоит из множества элементов (`ModelElement`), связанных между собой. Доступно два вида элементов – соединение (`Connection`) и Grid-элемент (`GridElement`). Grid-элементом может быть кластер, брокер, пользователь, хранилище данных и т.п. Соединения предназначены для передачи данных между Grid-элементами. У каждого элемента есть строковый атрибут реализация (implementation). Значение данного атрибута указывает на имя Java-класса, определяющего поведение элемента. Каждый элемент может быть параметризован. Параметр представляет собой пару строк (имя, значение) и может иметь дочерние параметры. Дочерние параметры используются, например, при задании свойств алгоритмов распределения задач. Предположим, для элемента мы выбираем реализацию “кластер” и для кластера определяем значение параметра “schedulerClass” как “BackfillLocal”. В данном случае реализация “BackfillLocal” может также требовать задания значений параметров. В этом случае параметр “schedulerClass” будет иметь дочерние параметры.

Таким образом, модель Grid-системы определяется в несколько этапов (Рис. 3). Сначала мы создаем Grid элементы, из которых будет состоять Grid система. Затем задаем топологию сети путем создания связей между элементами и сетевыми соединениями. И на последнем этапе выбираем реализацию каждого элемента и определяем значения параметров для конкретной реализации.

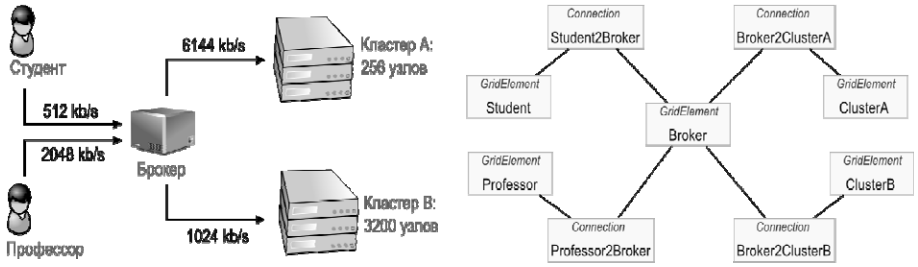
Рис. 3. Этапы определение модели Grid-системы.

В системе представлены основные элементы, необходимые для создания моделей Grid. Это – кластер, брокер, поток задач, сетевое соединение. Базовый набор реализованных алгоритмов распределения для кластера и брокера включают:

BackfillLocal	реализация алгоритма Backfill (алгоритм обратного заполнения) для кластера. Задание с меньшим приоритетом может быть запущено вне очереди, но только в том случае, если оно не будет мешать запуску более приоритетных заданий
BestFitLocal	реализация алгоритма “наилучший подходящий” для кластера. Для данного текущего количества свободных узлов подбирается задача, наиболее близкая по ширине
FirstFitLocal	реализация алгоритма “первый подходящий” для кластера. Размещается задача из начала очереди. Если узлов для запуска задачи не достаточно, то размещения не происходит
RandomFitGlobal	“случайный подходящий” для брокера. Для текущей задачи случайным образом выбирается кластер из множества подходящих

и другие.

В качестве примера рассмотрим, каким образом в системе моделирования будет определяться архитектура, изображенная на рисунке 4(a).



(a) Система (b) Модель

Рис. 4. Пример определения Grid системы.

Пользователи, брокер и кластеры представляются с помощью Grid-элементов, с именами Student, Professor, Broker, ClusterA, ClusterB соответственно – рисунок 4(b). Для задания соединений используются элементы `Connection` с именами Student2Broker, Professor2Broker, Broker2ClusterA, и Broker2ClusterB. В таблице 1 перечислены параметры и имена классов, реализующих поведение элементов.

Имя	Реализация	Параметр	Значение
ClusterA	SimpleCluster		
		nodes	256
		speedup	1
		schedulerClass	”BackfillLocal“
ClusterB	SimpleCluster		
		nodes	3200
		speedup	1
		schedulerClass	”BackfillLocal“
Student	WorkloadTaskFlow		
		wfile	”student.sfw.zip“
		startDelay	0
Professor	WorkloadTaskFlow		
		wfile	”professor.sfw.zip“
		startDelay	0
Broker	SimpleBroker		
		schedulerClass	”RandomFitGlobal“
Student2Broker	DelayedConstantConnection		
		count	512000
		period	1
Professor2Broker	DelayedConstantConnection		
		count	2048000
		period	1
Broker2ClusterA	DelayedConstantConnection		
		count	6144000
		period	1
Broker2ClusterB	DelayedConstantConnection		
		count	1024000
		period	1

Таб. 1: Параметры элементов

Для кластеров мы используем реализацию `SimpleCluster`. В качестве параметров необходимо указать количество узлов – параметр `nodes` и коэффициент ускорения – параметр `speedup`. Ускорение определяет, насколько быстрее, по сравнению с некоторым эталонным кластером, задачи будут выполняться на данном кластере. В нашем примере мы предполагаем, что задачи выполняются с одинаковой скоростью на обоих кластерах. Параметр `schedulerClass` определяет алгоритм распределения задач

локальным планировщиком. В примере мы задаем алгоритм `BackfillLocal`, представляющий собой реализацию алгоритма Backfill [6].

Для пользователей ”студент“ и ”профессор“ мы используем реализацию `WorkloadTaskFlow`. Данная реализация позволяет порождать задачи в моделируемой системе на основе собранных статистических данных использования реально существующей среды Grid. В качестве параметров задается имя файла в формате ”workload“ – параметр `wfile` и время до начала порождения первой задачи – параметр `startDelay`. Период позволяет активизировать различные потоки задач в различное время.

Брокер задается реализацией `SimpleBroker`. Единственным параметром является алгоритм распределения задач глобальным планировщиком – параметр `schedulerClass`. Мы указываем значение `RandomFitGlobal`. Это реализация алгоритма ”случайный из подходящих“ – для очередной задачи брокер выбирает множество кластеров, которые могут выполнить данную задачу, и затем случайным образом выбирает один кластер из данного множества.

Для сетевых соединений Student2Broker, Professor2Broker, Broker2ClusterA и Broker2ClusterB мы используем реализацию `DelayedConstantConnection`. Это простая реализация сетевого соединения позволяет передавать заданное количество данных с некоторой задержкой. В нашем примере это 512, 2048, 6144 и 1024 kb/sec соответственно.

После того, как описание модели завершено, трансляция данного описания, и последующая компиляция исходного кода происходит автоматически. На выходе получается выполняемая программа-симулятор, которую можно запустить и получить результат в виде профиля выполнения.

Для визуализации результатов система предоставляет готовые шаблоны, отображающие:

- загруженность системы – общую и с разбивкой по отдельным кластерам
- время ожидания задач в очереди – среднее и пиковое с разбивкой по классам задач и по отдельным кластерам
- пропускной способности – как по количеству задач, так и используя интегральную оценку

3. Эксперименты

Цель экспериментов заключалась в следующем. С использованием реализованного прототипа среды смоделировать поведение реально существующей Grid системы при различных условиях. В качестве распределенной Grid системы была выбрана сеть Sharcnet [18].

Распределенная вычислительная система Sharcnet (Shared Hierarchical Academic Research Computing Network) – это консорциум из 16 колледжей и университетов в юго-западной части провинции Канады Онтарио, вычислительные ресурсы которых объединены высокоскоростной оптической сетью (Рис. 5).

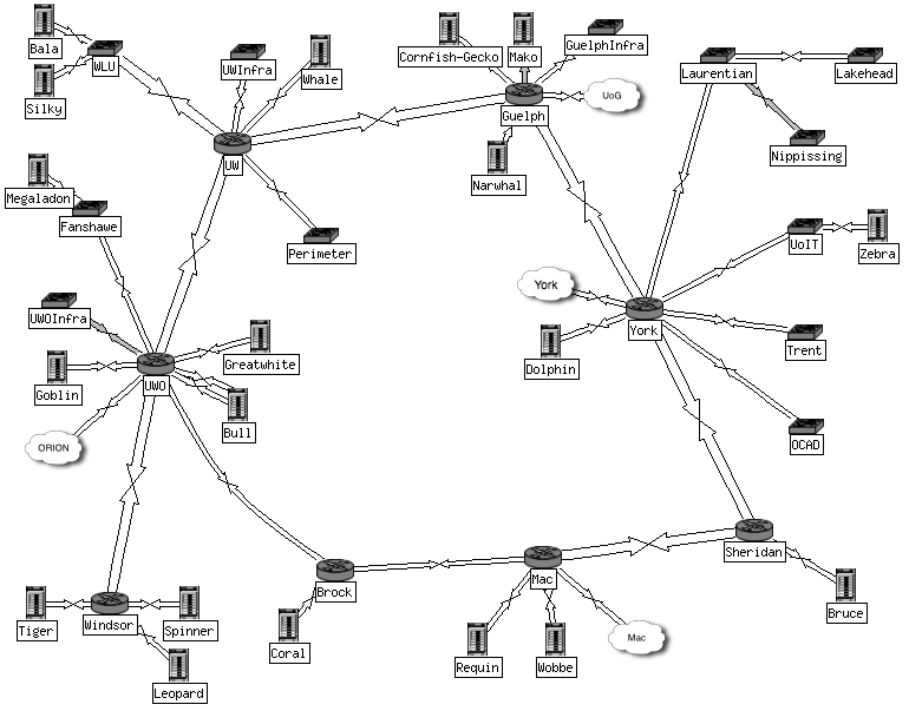


Рис. 5. Схема распределенной вычислительной системы Sharcnet.

Характеристики вычислительных ресурсов сети Sharcnet представлены в таблице 2.

Имя	Процессоры	Узлы
bruce	128	32 x 4 x Opteron
narwhal	1068	267 x 4 x Opteron dual core
tiger	128	32 x 4 x Opteron
bull	384	96 x 4 x Opteron
megaladon	128	32 x 4 x Opteron
dolphin	128	32 x 4 x Opteron
requin	1536	768 x 2 x Opteron

whale	3072	768 x 4 x Opteron
zebra	128	32 x 4 x Opteron
bala	128	32 x 4 x Opteron

Таб. 2. Характеристики вычислительных ресурсов сети Sharcnet.

В качестве входных данных была использована запись реального потока задач (“workload“ поток), выполнявшихся на кластерах с декабря 2005 по январь 2007 года [15]. Характеристики потока представлены на Рис. 6.

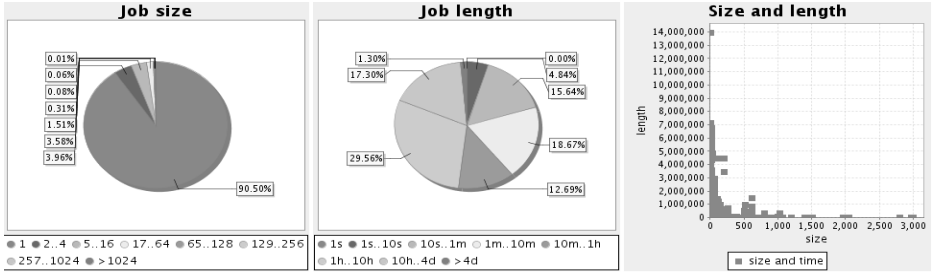


Рис. 6. Характеристики потока задач в сети Sharcnet: распределение ширины задач (количество запрашиваемых процессоров), длины задач, соотношение ширины и длины. Время показано в секундах.

Данный поток состоит по большей части из однопроцессорных задач. Однако, параллельных задач достаточно много – примерно 10%. Длина задач распределена более равномерно – большая часть, примерно 30%, имеет длину от 1 до 10 часов и чуть более половины всех задач по длительности составляют меньше одного часа, включая все большие параллельные задачи.

На Рис. 7 представлен график суммарного числа запрашиваемых процессоров в единицу времени. По оси абсцисс отложено время в секундах, по оси ординат – суммарное число запрашиваемых процессоров. Синяя горизонтальная линия показывает общее число процессоров в системе.

Мы видим, что поток не равномерный, на протяжении всего интервала присутствуют всплески запрашиваемого количества процессоров. Также, на графике можно заметить периодичность изменения нагрузки – временные интервалы 5000000, 1000000, 15000000 и т.д. Данные особенности присущи практически всем реальным потокам [3]. Примерно в середине временного интервала происходит перегрузка системы – запрашиваемая ширина становится больше доступной. Это говорит о том, что при любом распределении задач, в системе будут присутствовать очереди. Для данного потока, с 10% параллельных задач, будет происходить неполное заполнение кластеров, что еще больше увеличит размер очередей.

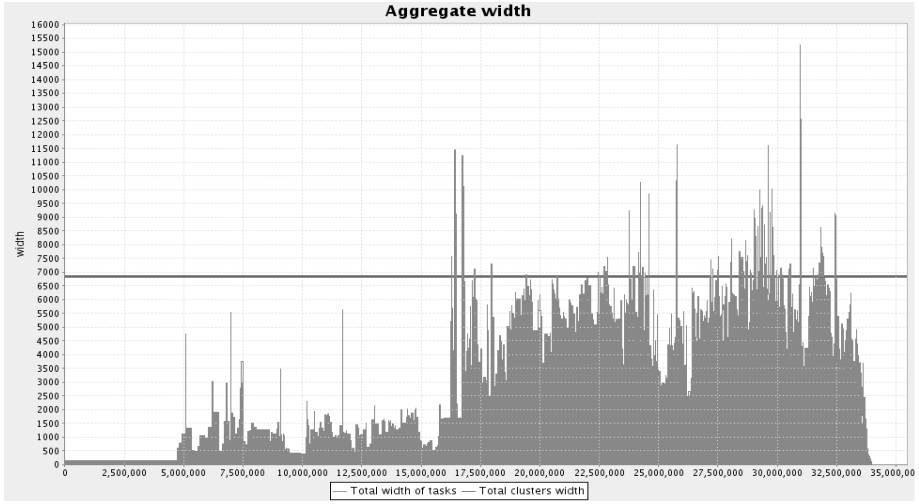
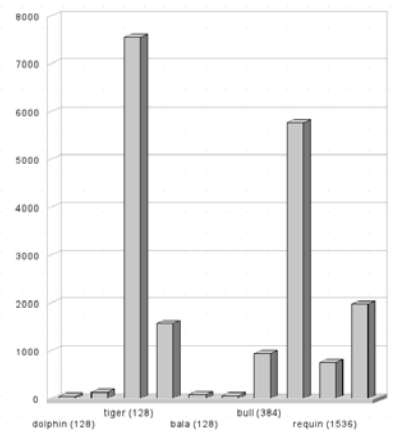
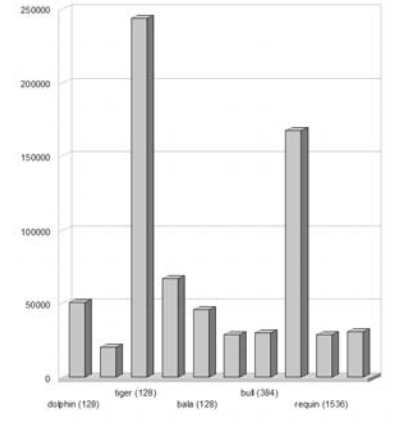


Рис. 7. Суммарное число запрашиваемых процессоров в единицу времени.

Особенность данного потока состоит также в том, что пользователи направляли задачи на кластеры непосредственно – для распределения не использовался брокер. При анализе потока оказалось, что в системе присутствует существенный дисбаланс нагрузки (Рис. 8).



Средняя длина очереди на каждом кластере



Среднее время ожидания на каждом кластере

Рис. 8. Дисбаланс в системе Sharcnet

Средние длины очередей в некоторых случаях отличаются почти в 100 раз. Это приводит к тому, что среднее время ожидания в очереди для различных кластеров отличается в несколько десятков раз.

Для проведения экспериментов, на основе данного потока мы создали несколько синтетических потоков. Необходимость использования синтетических потоков обусловлена различными факторами:

- запись оригинального потока может содержать различные несистематические особенности и всплески, связанные с конкретными событиями, что делает его неподходящим для анализа и прогнозирования поведения данной системы, но эти особенности могут быть сглажены при создании синтетических потоков;
- оригинальные потоки часто имеют большой размер и поэтому не очень удобны для выявления локальных свойств алгоритмов;
- синтетические потоки могут позволить создавать новые ситуации, которых не было в исходных данных.

Для создания синтетических потоков могут быть использованы различные подходы. Полностью синтетические потоки иногда бывают удобны для отладки, однако для анализа поведения моделируемых систем более подходящим является использование синтетических потоков, основывающихся на записях оригинальных потоков [5].

Для создания таких синтетических потоков мы использовали следующий поход. Для того чтобы определить поток задач

$$\{P_j\}_{j=1}^{M}$$

необходимо определить следующие параметры:

- R_j - промежуток времени между поступлением j и $j+1$ задачами ($j = 1, 2, \dots M-1$);
- H_j - запрашиваемое время исполнения($j = 1, 2, \dots M$);
- W_j - запрашиваемое число процессоров($j = 1, 2, \dots M$).

На основе оригинального потока задач для этих параметров оцениваются кумулятивные функции распределения и первые моменты. Далее, для каждого из параметров подбирается функция распределения в виде свертки нескольких распространенных функций распределения. Подбор осуществляется с помощью минимизации отклонения моментов и графиков функций по параметрам распределений в свертке и коэффициентам свертки.

Полученные таким образом распределения были использованы для генерирования нескольких синтетических потоков.

Мы применили разработанную систему моделирования и сравнили эффективность распределения задач в сети Sharcnet в оригинальном случае (без брокера) с распределением, получаемым с помощью брокера. Без брокера задачи поступали на кластеры в оригинальной последовательности, указанной

в файле загрузки. В обоих случаях на каждом кластере использовалась реализация алгоритма Backfill.

Алгоритм обратного заполнения Backfill работает по следующему принципу: размещая наиболее приоритетное задание, определяется момент времени, когда освободится достаточное количество ресурсов, занятых уже выполняющимися заданиями, затем производится резервирование этих ресурсов. Задание с меньшим приоритетом может быть запущено вне очереди, но только в том случае, если оно не будет мешать запуску более приоритетных заданий [6].

На брокере использовались различные алгоритмы распределения. В каждом случае вначале брокер выбирает множество кластеров, которые могут выполнить данную задачу – $W \geq W_j$, где W – число узлов кластера, а W_j – число узлов, запрашиваемых задачей, а затем выбирает один кластер исходя из заданного критерия:

N/W Выбирается кластер с наименьшим числом задач в очереди. Для данного кластера отношение N/W имеет минимальное значение, где N – число задач, стоящих в очереди, W – число процессоров кластера.

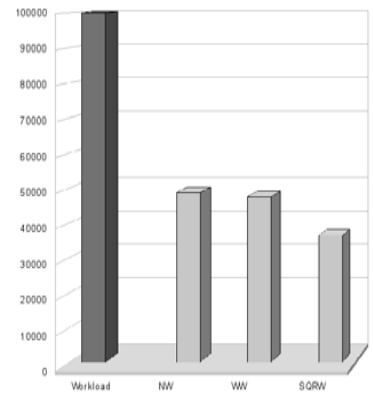
W/W Выбирается кластер с минимальной общей шириной задач в очереди. Для данного кластера отношение $(\sum_{j=1}^{N} W_j + \bar{W})/W$ имеет минимальное значение, где N – число задач, стоящих в очереди, W_j – ширина задачи, $\bar{W}$ – ширина отправляемой задачи, W – число процессоров кластера.

Sqr/W Выбирается кластер с минимальной общей площадью задач в очереди. Для данного кластера отношение $(\sum_{j=1}^{N} S_j + \bar{S})/W$ имеет минимальное значение, где N – число задач, стоящих в очереди, S_j – площадь задачи, $\bar{S}$ – площадь отправляемой задачи, W – число процессоров кластера.

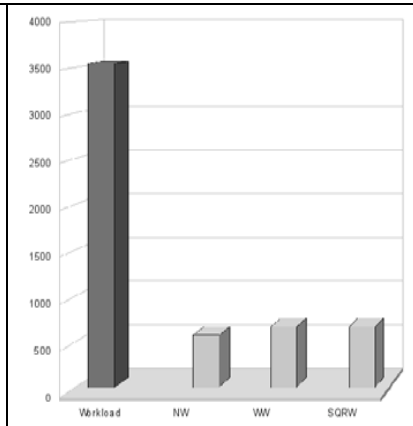
Всего было проведено 7 экспериментов:

1. задачи распределялись на кластеры согласно файлу загрузки;
2. задачи направлялись на брокер, который затем распределял их на кластеры. На брокере использовалась эвристика N/W;
3. на брокере использовалась эвристика W/W;
4. на брокере использовалась эвристика Sqr/W;
5. на брокер направлялись только однопроцессорные задачи. Параллельные задачи направлялись на кластеры согласно файлу загрузки. На брокере использовалась эвристика N/W;
6. однопроцессорные задачи, на брокере использовалась эвристика W/W;
7. однопроцессорные задачи, на брокере использовалась эвристика Sqr/W

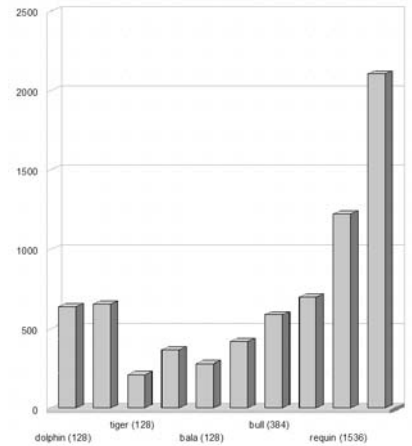
Мы сравнивали среднее время ожидания задач в очереди, а также характеристики самих очередей – длину и площадь, среднюю по всем кластерам и отдельно для каждого кластера.



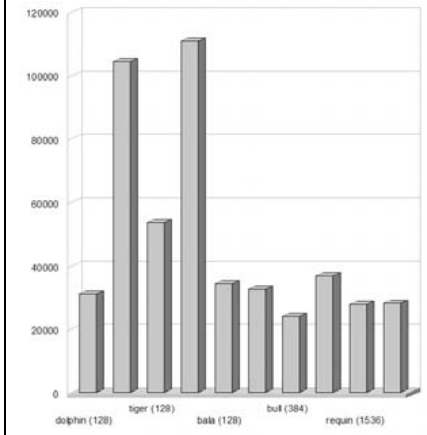
(a) Среднее время ожидания в очереди



(b) Средняя длина очереди



(c) Средняя длина очереди на каждом кластере. Распределение однопроцессорных задач, алгоритм Sqr/W



(d) Среднее время ожидания на каждом кластере. Распределение однопроцессорных задач, алгоритм Sqr/W

Рис. 9. Результаты экспериментов.

На рисунке 9 представлены результаты экспериментов при распределении только однопроцессорных задач. На рисунке 9(a) показано среднее время ожидания в очереди, определяемое как

$$T_{avg} = \frac{\sum_{j=1}^{N} T_{start} - T_{submit}}{N}$$

где . N – общее число запущенных задач, . T_{start} – время запуска задачи, . T_{submit} – время постановки задачи в очередь кластера. На рисунке 9(b) показана средняя длина очереди. Первый столбец на обеих диаграммах соответствует распределению задач без брокера. На рисунках 9(c) и 9(d) показаны средняя длина очереди и среднее время ожидания на каждом кластере в секундах.

Результаты показывают, что для данной вычислительной системы распределение потока однопроцессорных заданий через брокер дает значительный эффект: снижается среднее время ожидания заданий в очереди, а также происходит более равномерная загрузка вычислительных ресурсов.

При распределении всех задач (не только однопроцессорных) через брокер среднее время ожидания становится примерно на 5-7% меньше, чем в приведенных результатах. Однако, мы приводим результаты для распределения только однопроцессорных задач, поскольку, для проведения данных экспериментов нам была доступна информация только в виде записи потока задач. Из записи потока невозможно определить почему задача отправляется пользователем на тот или иной вычислительный ресурс. Причиной может быть архитектура системы, наличие специального программного обеспечения, личные пристрастия и т.п. Мы сделали предположение, что однопроцессорные задачи менее привязаны к конкретному кластеру, так как не зависят от среды передачи данных, которая установлена на кластере. Для системы Sharcnet такое предположение наиболее логично, так как среда передачи данных не одинакова для всех кластеров – таблица 3.

Имя	Процессоры	Архитектура
bruce	128	Myrinet 2g (gm)
narwhal	1068	Myrinet 2g (gm)
tiger	128	Myrinet 2g (gm)
bull	384	Quadrics Elan4
megaladon	128	Myrinet 2g (gm)
dolphin	128	Myrinet 2g (gm)
requin	1536	Quadrics Elan4
whale	3072	Gigabit Ethernet
zebra	128	Myrinet 2g (gm)
bala	128	Myrinet 2g (gm)

Таблица 3. Среда передачи данных кластеров сети Sharcnet.

Похожие результаты были получены в работе голландских исследователей при объединении двух Grid систем – Grid5000 и DAS2 [9]. В их работе отмечается наличие дисбаланса в обеих системах, и предлагается метод для его устранения используя глобальный планировщик. Результаты также показывают существенное уменьшение времени ожидания, примерно на 60%, и более равномерную загруженность кластеров.

В ходе проведенных нами экспериментов было замечено, что результаты распределения сильно зависят от входного потока задач. Очень трудно найти алгоритм распределения, который давал бы одинаково хорошие результаты на всех возможных потоках. Однако, зная характеристики вычислительной системы и характеристики предполагаемого потока задач, мы можем провести моделирование и определить какой алгоритм распределения показывает наиболее хороший результат. В некоторых случаях простая эвристика может давать лучшие результаты по сравнению с более сложной.

В связи с этим, актуальной представляется задача разработки такого алгоритма управления вычислительными ресурсами, при котором брокер анализирует поступающий к нему поток задач и, в зависимости от характеристик потока, выбирает эвристику, дающую оптимальное, согласно заданным критериям, распределение. Выбранная эвристика используется брокером для распределения задач по кластерам до тех пор, пока не произойдет ”переключение“ на другую эвристику.

4. Заключение

В статье представлена среда моделирования, разработанная в ИСП РАН, позволяющая оценивать поведение распределенных вычислительных систем при изменяющихся условиях и, на основе этого, оптимизировать стратегии управления потоками задач. Также представлены результаты использования реализованного прототипа данной среды на моделировании реально существующей вычислительной системы Sharcnet.

В будущем нам хотелось бы развивать данную систему как инструмент для оценки эффективности управления вычислительными ресурсами в Grid. Пользователями такой системы могут быть администраторы и исследователи, разрабатывающие новые алгоритмы управления ресурсами.

В ближайшее время предполагается провести эксперименты с задачами, требующими передачи больших объемов данных. Также, мы планируем расширить функциональность генератора синтетических потоков.

Система является свободно распространяемым с открытым исходным кодом программным пакетом и доступна по адресу http://gridme.googlecode.com.

Литература

[1] Buyya R., Murshed M. Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing // Concurrency and computation: practice and experience. "— 2002. "— Vol. 14. "— Pp. 1175–1220.

[2] Eclipse - an open development platform www.eclipse.org.

[3] Feitelson D. G. Locality of sampling and diversity in parallel system workloads // ICS ’07: Proceedings of the 21st annual international conference on Supercomputing. "— ACM, 2007. "— Pp. 53–63.

[4] Feitelson D. G., Rudolph L. Metrics and benchmarking for parallel job scheduling // Lecture Notes in Computer Science. "— 1998. "— Vol. 1459. "— Pp. 1+.

[5] Feitelson D. G. Workload modeling for computer systems performance evaluation book draft. "— since 2005.

[6] Feitelson D. G., Weil A. M. Utilization and predictability in scheduling the IBM SP2 with backfilling // 12th Intl. Parallel Processing Symp. "— 1998. "— Pp. 542–546.

[7] Globus alliance. "— www.globus.org.

[8] The grid workloads archive. "— http://gwa.ewi.tudelft.nl/pmwiki/.

[9] Inter-operating grids through delegated matchmaking / A. Iosup, D. H. Epema, T. Tannenbaum et al. // Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC07). "— Reno, NV: 2007. "— November.

[10] Legrand A., Marchal L., Casanova H. Scheduling distributed applications: The simgrid simulation framework // Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid 2003 (CCGrid2003). "— 2003. "— Pp. 138–145.

[11] The microgrid: Using emulation to predict application performance in diverse grid network environments / H. Xia, H. Dail, H. Casanova, A. Chien // In Proceedings of the Workshop on Challenges of Large Applications in Distributed Environments (CLADE’04). IEEE Press. "— 2004.

[12] Tuecke S., Czajkowski K., Foster I. et al. Open grid services infrastructure (ogsi) version 1.0. "— 2003. "— June.

[13] Optorsim - a grid simulator for studying dynamic data replication strategies / W. Bell, D. Cameron, L. Capozza et al. // International Journal of High Performance Computing Applications. "— 2003. "— Vol. 17, no. 4. "— Pp. 403–416.

[14] Overview of a performance evaluation system for global computing scheduling algorithms / A. Takefusa, S. Matsuoka, K. Aida et al. // Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing (HPDC’99). "— 1999. "— Pp. 97–104.

[15] Parallel workloads archive. "— http://www.cs.huji.ac.il/labs/parallel/workload/.

[16] Foster I., Kesselman C., Nick J., Tuecke S. The physiology of the grid an open grid services architecture for distributed systems integration. "— 2003.

[17] Platform globus toolkit. "— http://www.platform.com/.

[18] The shared hierarchical academic research computing network. "— www.sharcnet.ca.

[19] Univa. "— http://www.univa.com.