Реализация вычислений по схеме Master-Worker в ГРИД

О.В. Сухорослов

Центр Грид-технологий и распределенных вычислений ИСА РАН, Москва
os@isa.ru

Большинство решаемых в грид вычислительных задач допускает декомпозицию на множество независимых подзадач. Это позволяет реализовать параллельное решение задачи путем распределения подзадач между вычислительными узлами грид. При этом эффективность и время решения задачи в грид во многом определяется выбранной стратегией распределения подзадач.

Простейшая стратегия, заключающаяся в статическом назначении подзадач отдельным грид-заданиям, диспетчеризуемым с помощью штатного планировщика грид, часто приводит к неравномерной загрузке узлов, плохо контролируемому общему времени вычислений и медленному восстановлению после отказов отдельных заданий. Связано это может быть как с неравномерностью размера самих подзадач и невозможностью априори оценить этот размер, так и с неизбежной гетерогенностью и ненадежностью узлов грид. Существенную роль также играет высокая латентность запуска заданий в грид, составляющая от нескольких до десятков минут.

Более эффективной на практике является стратегия динамического распределения подзадач в соответствии с известной схемой "управляющий-рабочие" (master-worker). В рамках грид данная схема может быть реализована следующим образом (Рис. 1). Запускается некоторое число грид-заданий, которые представляют собой рабочие процессы. Данные процессы связываются по сети с управляющим процессом, который выполняется на выделенной машине вне грид. Управляющий процесс хранит список заданий (подзадач исходной задачи) и распределяет их между рабочими процессами.

Для балансировки нагрузки между рабочими процессами может использоваться простой и в то же время эффективный механизм распределения заданий, заключающийся в активном "вытаскивании" заданий рабочими процессами. В этом случае управляющий процесс играет пассивную роль, обрабатывая приходящие запросы рабочих процессов. При регистрации рабочий процесс запрашивает у управляющего процесса задание, выполняет его, передает полученный результат управляющему процессу, запрашивает новое задание и так далее, до тех пор, пока список невыполненных заданий не будет исчерпан.

Поскольку узлы грид являются ненадежными, а грид-задания имеют ограниченное время выполнения, то рабочие процессы могут непредсказуемым образом выходить из процесса вычислений. Таким образом, также необходим механизм обнаружения и восстановления после отказов рабочих процессов.

Очевидно, что механизмы запуска рабочих процессов в грид, их взаимодействия с управляющим процессом, распределения заданий и обработки отказов в рамках описанной схемы являются в значительной мере общими и слабо зависят от решаемой задачи. Это позволяет создать универсальный каркас (framework) для разработки подобных грид-приложений, содержащий готовые реализации данных механизмов и допускающий их настройку и расширение.

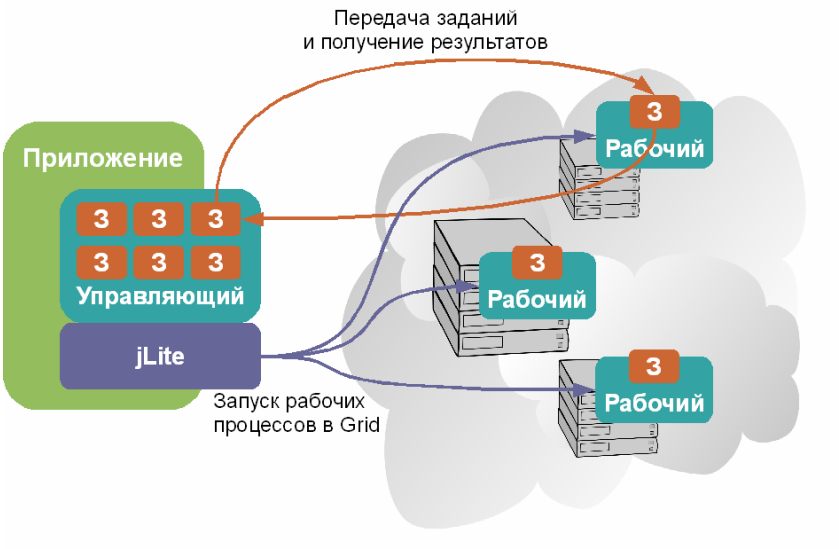


Рисунок 1. Схема работы MaWo.

Разрабатываемый автором инструментарий MaWo реализует каркас для организации вычислений по схеме “управляющий-рабочие” в грид-инфраструктуре EGEE [1] (Рис. 1). Инструментарий реализован на языке Java. Для запуска рабочих процессов в Grid используется библиотека jLite [2-3].

Создаваемое с помощью MaWo грид-приложение состоит из двух частей: управляющий компонент и вычислительный модуль. Управляющий компонент приложения, вызываемый управляющим процессом, осуществляет разбиение исходной задачи на подзадачи и последующую обработку результатов подзадач для получения окончательного результата. Вычислительный модуль, загружаемый и запускаемый рабочими процессами на узлах грид, реализует алгоритм решения отдельной подзадачи.

Текущая версия MaWo поддерживает реализацию управляющего компонента на языке Java. Также существует готовая реализация данного компонента, считывающая входные файлы заданий из указанной директории и записывающая в другую директорию результаты выполненных заданий. В качестве вычислительных модулей решения задач могут использоваться любые исполняемые файлы. Таким образом, для использования MaWo разработчику грид-приложения в общем случае не требуется владеть языком программирования Java.

При запуске грид-приложения на основе MaWo автоматически выполняются следующие действия:

- Запускается сервер с управляющим процессом;
- Запускается требуемое число грид-заданий с рабочими процессами (адрес управляющего процесса передается в аргументах грид-задания);
- Генерируется список заданий путем вызова управляющего компонента приложения;
- Задания помещаются в очередь управляющего процесса;
- Управляющий процесс приступает к обработке вызовов от рабочих процессов.

Период времени между запуском грид-задания и началом его выполнения на узле в инфраструктуре EGEE составляет не менее нескольких минут и может заметно варьироваться от задания к заданию в зависимости от загрузки соответствующих ресурсных центров грид. Таким образом, в начале работы приложения наблюдается некоторая пауза, после которой рабочие процессы из грид начинают постепенно подключаться к приложению. Данную паузу можно было бы устранить путем предварительного запуска грид-заданий и поддержания в гриде готового к работе пула рабочих процессов. Од-

нако при этом занимаемые ресурсы грид фактически бы простаивали до начала вычислений, будучи недоступными для других пользователей грид, что противоречит принципам справедливого разделения ресурсов. Корректным способом устранения возникающей паузы может быть быстрый запуск дополнительных рабочих процессов на локальных ресурсах. В любом случае, вычисления начинаются при появлении первого рабочего процесса.

После запуска рабочий процесс выполняет следующие действия:

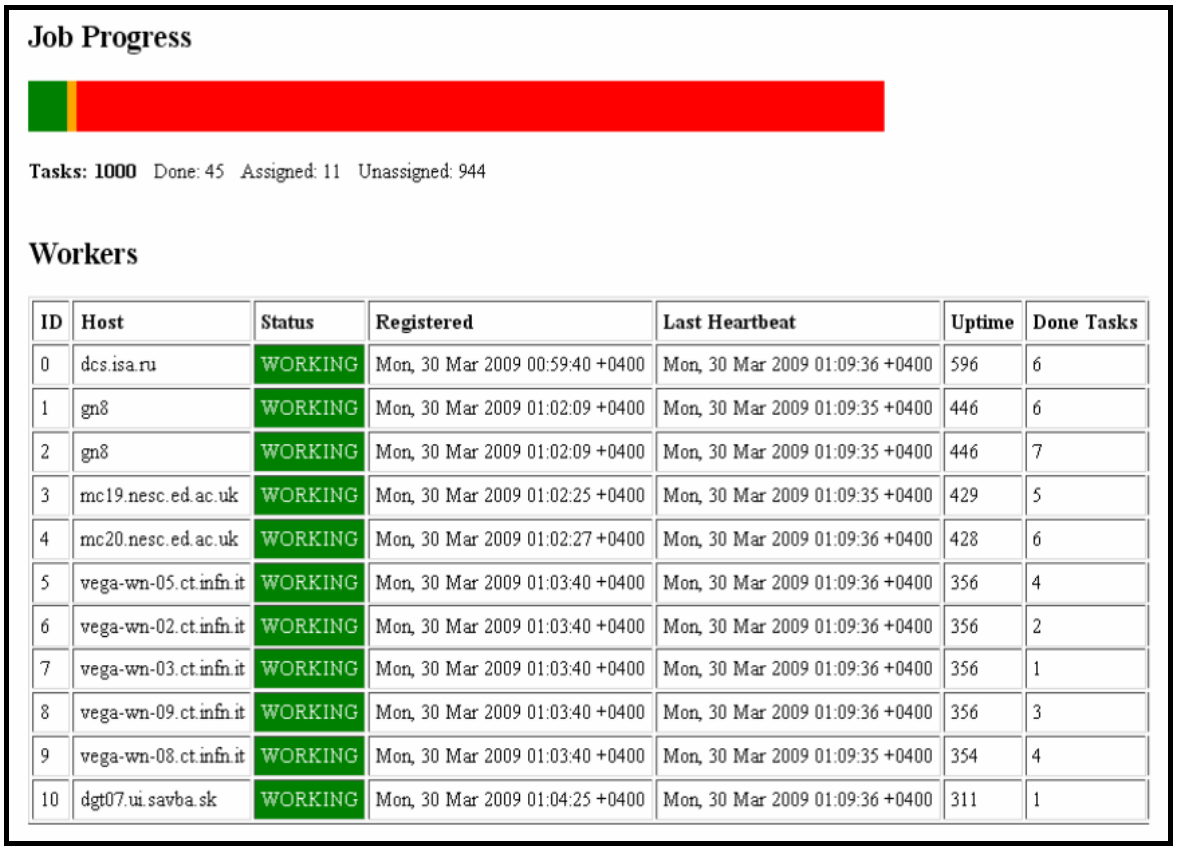
- Устанавливает соединение с управляющим процессом и инициирует регистрацию нового рабочего процесса, передавая информацию о платформе и характеристиках узла;
- Запрашивает у управляющего процесса общие файлы вычислительного модуля приложения (для оптимизации последующих передач файлов заданий) и сохраняет их на локальном диске;
- Приступает к выполнению основного, рабочего цикла: загрузка задания, выполнение задания, передача результатов управляющему процессу.

Выполнение рабочего процесса продолжается до наступления одного из трех событий: выполнены все задания, исчерпан лимит по времени выполнения грид-задания, потеряна связь с управляющим процессом.

Для контроля процесса вычислений и обнаружения отказов рабочие процессы, одновременно с выполнением основного цикла, периодически отправляют управляющему процессу служебные сообщения (heartbeat) с информацией о своем состоянии. Если в течение определенного промежутка времени от рабочего процесса не поступало служебных сообщений, то данный процесс помечается как отказавший и выполняемые им задачи помещаются обратно в очередь управляющего процесса. Также при отказе рабочего процесса автоматически происходит запуск нового грид-задания с тем, чтобы поддерживать заданное в начале вычислений число рабочих процессов. Исключением является ситуация, когда вычисления близки к завершению, и есть свободные рабочие процессы, - в этом случае новое задание не запускается.

Для отображения информации о ходе вычислений предусмотрен Web-интерфейс (Рис. 2), встроенный в сервер с управляющим процессом. Доступна следующая информация:

- Адрес управляющего процесса;
- Время начала и окончания вычислений, полное время вычислений;
- Полное число заданий, число выполненных и выполняющихся заданий;
- Для каждого рабочего процесса: доменное имя узла, текущий статус, время регистрации, время последнего полученного heartbeat-сообщения, время доступности (с момента регистрации до текущего момента или момента отключения), число выполненных заданий, среднее время выполнения задания, полное время выполнения заданий и эффективность, вычисляемая как отношение полного времени выполнения заданий к времени доступности.



Job Progress

Tasks: 1000 Done: 45 Assigned: 11 Unassigned: 944

Workers

ID	Host	Status	Registered	Last Heartbeat	Uptime	Done Tasks
0	dcs.isa.ru	WORKING	Mon, 30 Mar 2009 00:59:40 +0400	Mon, 30 Mar 2009 01:09:36 +0400	596	6
1	gn8	WORKING	Mon, 30 Mar 2009 01:02:09 +0400	Mon, 30 Mar 2009 01:09:35 +0400	446	6
2	gn8	WORKING	Mon, 30 Mar 2009 01:02:09 +0400	Mon, 30 Mar 2009 01:09:35 +0400	446	7
3	mc19.nesc.ed.ac.uk	WORKING	Mon, 30 Mar 2009 01:02:25 +0400	Mon, 30 Mar 2009 01:09:35 +0400	429	5
4	mc20.nesc.ed.ac.uk	WORKING	Mon, 30 Mar 2009 01:02:27 +0400	Mon, 30 Mar 2009 01:09:36 +0400	428	6
5	vega-wn-05.ct.infn.it	WORKING	Mon, 30 Mar 2009 01:03:40 +0400	Mon, 30 Mar 2009 01:09:36 +0400	356	4
6	vega-wn-02.ct.infn.it	WORKING	Mon, 30 Mar 2009 01:03:40 +0400	Mon, 30 Mar 2009 01:09:36 +0400	356	2
7	vega-wn-03.ct.infn.it	WORKING	Mon, 30 Mar 2009 01:03:40 +0400	Mon, 30 Mar 2009 01:09:36 +0400	356	1
8	vega-wn-09.ct.infn.it	WORKING	Mon, 30 Mar 2009 01:03:40 +0400	Mon, 30 Mar 2009 01:09:36 +0400	356	3
9	vega-wn-08.ct.infn.it	WORKING	Mon, 30 Mar 2009 01:03:40 +0400	Mon, 30 Mar 2009 01:09:35 +0400	354	4
10	dgt07.ui.savba.sk	WORKING	Mon, 30 Mar 2009 01:04:25 +0400	Mon, 30 Mar 2009 01:09:36 +0400	311	1

Рисунок 2. Фрагмент Web-интерфейса с отображением информации о ходе вычислений.

Для сетевого взаимодействия между управляющим и рабочими процессами используется технология промежуточного ПО Ice [4-5]. Отметим, что в рассмотренной реализации сетевое соединение всегда инициируется рабочим процессом. Вычислительные узлы грид, как правило, находятся за межсетевым экраном, запрещающим входящие соединения. Тем не менее, после установления соединения возможна передача вызовов в направлении от управляющего к рабочему процессу с помощью механизма двусторонних (bidirectional) соединений Ice.

Как уже отмечалось, рабочие процессы MaWo могут быть запущены в ручном режиме на локальных ресурсах. Это позволяет использовать для проведения вычислений не только узлы грид, но и ресурсы доступных пользователю рабочих станций и вычислительных серверов. Кроме того, для упрощения отладки и тестирования приложения предусмотрена возможность запуска рабочих процессов в отдельных потоках на локальной машине.

Кратко остановимся на сравнении MaWo с аналогичными разработками. Одной из первых реализаций схемы “управляющий-рабочие” для Grid явился пакет Condor MW [6-7], предоставляющий каркас для разработки приложений на языке C++ в рамках системы Condor. Существует несколько подобных разработок, поддерживающих запуск рабочих процессов в инфраструктуре EGEE. Пакет DIANE (DIstributed ANalysis Environment) [8] реализует схему “управляющий-рабочие” на языке Python. Инструментарий ProActive [9] содержит обширный набор средств реализации параллельных и распределенных вычислений на базе платформы Java, в том числе каркас для реализации схемы “управляющий-рабочие”. В отличие от указанных инструментариев, MaWo не только содержит каркас для разработки приложений на одном языке программирования (Java), но и предоставляет готовые средства для интеграции в приложение произ-

вольных исполняемых файлов, не требующие знания языка Java. Кроме того, в отличие от DIANE и ProActive, MaWo не требует доступа к пользовательскому окружению gLite для запуска грид-заданий, что значительно упрощает разработку и развертывание приложения.

В рамках тестирования MaWo было проведено две серии вычислительных экспериментов в грид. Первая серия экспериментов состояла в выполнении большого количества синтетических вычислительных задач с изменяемыми параметрами, такими как время вычислений и размер данных задания. Вторая серия экспериментов состояла в реализации распределенного рендеринга трехмерных сцен и анимации с помощью алгоритма трассировки лучей. Результаты экспериментов подтверждают эффективность разработанного инструментария для организации подобного рода вычислений. В дальнейшем планируется реализовать ряд оптимизаций описанной схемы вычислений, а также предоставить возможность подключения произвольных реализаций планировщика заданий. Помимо этого планируется реализовать с помощью MaWo решение других вычислительных задач.

Литература

1. EGEE: [http://www.eu-egee.org/], 01.09.2009.
2. O.V. Sukhoroslov. jLite: A Lightweight Java API for gLite // Proceedings of the 3rd International Conference "Distributed Computing and Grid-technologies in Science and Education", GRID'2008, Dubna, Russia, 30 June - 4 July, 2008.
3. jLite: [http://jlite.googlecode.com/], 01.09.2009.
4. ZeroC Ice: [http://www.zeroc.com/], 01.09.2009.
5. Сухорослов О.В. Промежуточное программное обеспечение Ice // Проблемы вычислений в распределенной среде / Под ред. А.П. Афанасьева. Труды ИСА РАН. Т. 32. - М.: Издательство ЛКИ, 2008. – 288 с. (с. 33-67)
6. Condor MW: [http://www.cs.wisc.edu/condor/mw/], 01.09.2009.
7. Goux, J.-P., Kulkarni, S., Linderoth, J., and Yoder, M. An enabling framework for master-worker applications on the computational grid // Proceedings of the Ninth International Symposium on High Performance Distributed Computing (Pittsburgh, Pennsylvania, Aug. 2000), pp. 43-50.
8. DIANE: [http://cern.ch/diane/], 01.09.2009.
9. ProActive: [http://proactive.inria.fr/], 01.09.2009.