

РАЗВИТИЕ МОДЕЛЕЙ ПРОГРАММИРОВАНИЯ В GRID

О.В. Сухорослов

Базовой моделью использования современной Grid-инфраструктуры является запуск заданий в пакетном режиме, во многом повторяющий способ взаимодействия пользователя с вычислительным кластером. Главным образом, эта модель используется для запуска большого числа однотипных, независимых заданий. Однако не всегда задача пользователя укладывается в подобную схему. Для реализации новых Grid-приложений требуются высокоуровневые средства и модели программирования в Grid. В первую очередь, речь идет именно о средствах и инструментариях, упрощающих программирование и скрывающих от пользователя динамические аспекты распределенной среды. Данная работа посвящена не только тому, что уже сделано в этом направлении, но и намечает дальнейшие пути развития моделей программирования в Grid.

Первым шагом в указанном выше направлении стала поддержка **запуска параллельных программ** на отдельных кластерах Grid. С одной стороны, подобная возможность дает пользователю максимальную свободу в реализации схемы взаимодействия параллельных процессов. С другой стороны, доступные такому заданию вычислительные ресурсы ограничены ресурсами только одного кластера, причем с учетом его загрузки заданиями других пользователей. Для преодоления указанного ограничения были созданы **распределенные реализации MPI** (MPICH-G2, PASCX-MPI, GridMPI), позволяющие запускать MPI-программы “поверх” нескольких кластеров. Данный подход имеет ряд проблем, обусловленных особенностями среды Grid. Глобальный характер Grid приводит к высоким накладным расходам при обмене данными между процессами на различных кластерах, что требует соответствующей модификации алгоритма и программы. Децентрализованная структура Grid затрудняет развертывание подобных реализаций MPI и синхронный запуск заданий на нескольких кластерах. Существенным недостатком подобных реализаций является отсутствие механизма восстановления после отказов, вызванных, например, преждевременным завершением задания на одном из кластеров.

Модель “мастер-работчие” применяется в тех случаях, когда необходимо организовать динамическую балансировку нагрузки между группой рабочих процессов, обрабатывающих отдельные подзадачи. Данная необходимость может быть обусловлена неравномерностью и динамичностью как подзадач, так и доступных вычислительных узлов. Заметим, что последнее обстоятельство ярко выражено в Grid. Казалось бы, можно запускать подзадачи в виде заданий и полагаться на балансировку нагрузки, осуществляемую сервисом управления заданиями Grid. Однако проведенные вычислительные эксперименты показали, что такой подход менее эффективен, чем собственная реализация модели “мастер-работчие” с запуском в Grid не самих подзадач, а рабочих. Вызвано это, главным образом, значительными накладными расходами по запуску заданий в Grid. Гораздо быстрее передать новую подзадачу уже запущенному в Grid заданию напрямую по сети, минуя промежуточное ПО Grid. Кроме того, сервис управления заданиями Grid осуществляет планирование выполнения каждого задания независимо от других заданий и не позволяет использовать собственную стратегию планирования.

Для упрощения разработки Grid-приложений, использующих модель “мастер-работчие”, требуется соответствующий программный инструментарий. Данный инструментарий должен содержать “заготовки” для мастера и рабочих, реализовывать динамический запуск рабочих в Grid и обеспечивать восстановление после отказов. Прототипом подобного инструментария может служить пакет Condor MW [1], реализующий модель “мастер-работчие” в системе Condor и поддерживающий разработку приложений на языке C++. Автором статьи разработан прототип инструментария gMW, предназначенного для реализации “master-worker” приложений на языке Java поверх Grid-инфраструктуры EGEE. Инструментарий gMW позволяет запустить в Grid необходимое число рабочих процессов и использовать их через стандартный интерфейс платформы Java ExecutorService. В ходе вычислений gMW контролирует статус рабочих процессов и автоматически осуществляет их перезапуск. Для автоматической балансировки нагрузки между рабочими процессами используется pull-схема, когда свободные рабочие сами извлекают задания из очереди мастера.

Workflow-модель применяется в случае, если требуется связать в единый процесс несколько заданий [2]. Зависимости между заданиями часто описываются в виде ориентированного графа, ребра которого определяют порядок выполнения заданий и потоки данных между заданиями. Workflow-система осуществляет запуск отдельных заданий, передачу данных между ними и контроль за выполнением процесса в соответствии с его описанием. В качестве примера такой системы можно привести входящий в состав пакета Condor компонент DAGMan. Потребность в подобной модели программирования в Grid велика, поскольку сложные процессы анализа и обработки данных могут состоять из сотен элементарных заданий, ручной запуск и координация которых практически невозможны.

Инфраструктура EGEE поддерживает запуск workflow-заданий, описываемых в виде ориентированного ациклического графа (ОАГ), вершинами которого являются простые Grid-задания. Выполнение таких заданий реализуется с помощью вышеупомянутого компонента DAGMan. Обмен данными между заданиями происходит через входные и выходные файлы заданий. Возможностей подобной модели во многих случаях достаточно. Тем не менее, можно выделить несколько направлений ее совершенствования.

Во-первых, не всегда выразительных возможностей ОАГ достаточно для описания желаемой структуры процесса. Затруднения возникают, например, при описании циклов и условных переходов. Наиболее часто предлагаемые решения: введение в ОАГ специальных вершин [3], применение формализма сетей Петри [4] или создание языков описания процессов [5]. Во-вторых, спектр механизмов взаимодействия между заданиями не должен ограничиваться файлами, причем доступными только после завершения задания. В зависимости от характера и относительного размещения заданий в Grid это может быть общая память, распределенная файловая система или TCP-канал. Подобные механизмы особенно эффективны в случае, если задания могут быть запущены одновременно в режиме “producer-consumer”. Примером реализации этой модели в распределенной среде может служить система Dryad [6]. Наконец, требуется реализовать динамическую коррекцию структуры процесса в зависимости от входных данных и количества доступных ресурсов. Это позволило бы не описывать в отдельности каждую из сотен однотипных задач, а ограничиться только описанием отдельных стадий процесса. Примером для подражания здесь также может служить система Dryad.

Grid-сервисы создаются для того, чтобы предоставить пользователям Grid доступ к некоторому ресурсу, будь то вычислительный пакет, база данных или экспериментальная установка. Речь идет о проблемно-ориентированных сервисах, расширяющих возможности и формирующих новое поколение Grid-систем [7]. Примером реализации подобного подхода может служить инструментарий IARnet [8], предназначенный для интеграции распределенных алгоритмических ресурсов при решении прикладных задач. Для работы Grid-сервисов необходима инфраструктура, обеспечивающая бесперебойный доступ к сервисам, хранение данных и требуемые для обработки запросов вычислительные мощности. В этом отношении Grid-инфраструктура может сыграть роль платформы для таких сервисов. Рассмотрим этот вопрос подробнее.

Не каждый разработчик имеет возможность разместить Grid-сервис на соответствующем сервере. Запускать сервис в Grid в виде задания не имеет смысла. Во-первых, время выполнения задания ограничено. Во-вторых, из соображений безопасности на рабочих узлах обычно запрещены входящие соединения, что не позволяет размещать на них серверные приложения. Таким образом, требуется дополнительная инфраструктура для хостинга Grid-сервисов. Примером для подражания здесь может служить сервис EC2 компании Amazon.

Более реалистичной сейчас является схема, когда Grid-сервис размещается на сервере разработчика, но для вычислений и хранения данных он использует Grid. В качестве примера реализации такой схемы можно привести разрабатываемый в рамках проекта “Электронная Земля” сервис обработки геоданных. Сервис GDPS предоставляет доступ к вычислительно сложным алгоритмам в области наук о Земле и поддерживает запуск соответствующих вычислений в Grid. Взаимодействие пользователя с сервисом осуществляется через Web-портал проекта. Пользователь формирует через интерфейс портала набор входных данных, выбирает из списка алгоритмов, предоставляемых сервисом, интересующий его алгоритм обработки данных, и указывает необходимые параметры алгоритма. Портал отправляет сформированное пользователем задание сервису GDPS. Сервис производит загрузку данных, находит требуемый алгоритм и осуществляет запуск задания в Grid. Пользователь может отслеживать статус своего задания через портал. После окончания выполнения задания, результаты задания загружаются из Grid, передаются на портал и отображаются пользователю. Отметим, что описанная схема работы пользователя с Grid не требует специальной подготовки и позволяет сконцентрироваться на решаемой задаче.

Работа выполнена при поддержке фонда РФФИ (грант № 08-07-00430-а), Президиума РАН (программа фундаментальных исследований 15П) и Федерального агентства по науке и инновациям (государственный контракт № 02.514.11.4053).

ЛИТЕРАТУРА:

1. Goux, J.-P., Kulkarni, S., Linderoth, J., and Yoder, M. An enabling framework for master-worker applications on the computational grid. In Proceedings of the Ninth International Symposium on High Performance Distributed Computing (Pittsburgh, Pennsylvania, Aug. 2000), pp. 43--50.
2. И.В. Лазарев, О.В. Сухорослов. Использование workflow-методологии для описания процесса распределенных вычислений. // Проблемы вычислений в распределенной среде: Модели обработки и представления данных. Динамические системы. Труды ИСА РАН. Том 14. - М.: КомКнига, 2005, с.26-70.
3. David Churches, Gabor Gombas, Andrew Harrison, Jason Maassen, Craig Robinson, Matthew Shields, Ian Taylor and Ian Wang. Programming Scientific and Distributed Workflow with Triana Services. In Concurrency and Computation: Practice and Experience (Special Issue: Workflow in Grid Systems), 18(10):1021-1037, 2006.

4. О.В. Сухорослов, И.В. Лазарев. Реализация службы управления сценариями в распределенной вычислительной среде. // Проблемы вычислений в распределенной среде: распределенные приложения, коммуникационные системы, математические модели и оптимизация. Труды ИСА РАН. Т. 25. - М.: КомКнига, 2006, с.77-98.
5. Gregor von Laszewski and Mihael Hategan. Workflow Concepts of the Java CoG Kit. Journal of Grid Computing, Volume 3, Numbers 3-4, September 2005 , pp. 239-258.
6. Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. European Conference on Computer Systems (EuroSys), Lisbon, Portugal, March 21-23, 2007.
7. Ian Foster. Service-Oriented Science. Science, 308(5723), 2005, pp. 814-817.
8. Alexander Afanasiev, Oleg Sukhoroslov, Mikhail Posypkin. A High-Level Toolkit for Development of Distributed Scientific Applications. // Victor E. Malyshev (Ed.): Parallel Computing Technologies, 9th International Conference, PaCT 2007, Pereslavl-Zalessky, Russia, September 3-7, 2007, Proceedings. Lecture Notes in Computer Science 4671 Springer 2007, ISBN 978-3-540-73939-5, pp. 103-110.