УДК 629.78.018

С.А. КУЛАНОВ

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Украина

АДАПТАЦИЯ СУЩЕСТВУЮЩИХ ПРИЛОЖЕНИЙ ДЛЯ ВЫПОЛНЕНИЯ В GRID С УЧЕТОМ ТИПОВ ДАННЫХ

Рассмотрены проблемы разработки GRID приложений, а также параметры, влияющие на механизм переноса существующих программ в GRID окружение. Определены типы данных в рамках механизма адаптации приложения под GRID. Рассмотрены способы передачи входных данных переносимых в GRID приложений, предложен механизм их расщепления на основе технологии SIMD (Single Instruction, Multiple Data).

GRID, SIMD, GLOBUS Toolkit, grid-адаптация приложений, GRID Application enablement

Введение

Основной целью GRID является интеграция большого количества распределенных ресурсов, которые могут взаимодействовать друг с другом для достижения определенной цели. Все большее количество организаций, исследовательских групп, ученых, вовлекаются в GRID инфраструктуру. Однако с переходом на новую платформу появился ряд проблем связанных с адаптацией для GRID большого количества существующих программ.

Существует достаточно большое количество определений GRID [1]. Однако, на наш взгляд, наиболее емким, и в тоже время достаточно четким определением термина GRID является определение GRID как системы, связанной с функциями интеграции, виртуализации и управления службами и ресурсами в распределенной, гетерогенной среде, которая поддерживает совокупность пользователей и ресурсов (виртуальные организации) на совокупности традиционных административных и организационных доменов (фактических организаций) [2].

Часто классифицируют GRID-системы согласно решаемым им задачам. Различают GRID[3]:

– вычислительные (Computing), в большинстве своем состоят из высокопроизводительных кластеров, и направлены на выполнение емких вычислений;

– колаборационные (Collaboration), состоят из различных типов компьютеров: от настольных до суперкомпьютеров и используются в основном для взаимодействия между различными виртуальными организациями и отдельными пользователями;

– хранилище данных (Data), представляют собой хранилища данных, которыми можно обмениваться вне зависимости от географического положения.

Исходя из определения, GRID можно представить в виде системы, объединяющей ресурсы (вычислительные, хранения, передачи, обработки данных) вне зависимости от их географического положения (рис. 1) и обеспечивающей прозрачный доступ пользователям к любому из ее компонентов.

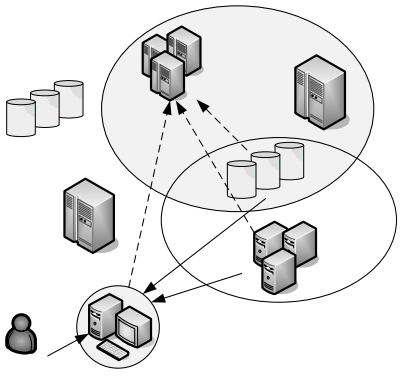


Рис. 1. GRID инфраструктура

GRID представляет собой многослойную архитектуру протоколов [4] (рис. 2), где приложение пользователя располагается на самом верху, что позволяет скрыть от него все подробности реализации нижних протоколов. Ввиду этого существует возможность адаптировать уже существующие приложения для GRID.

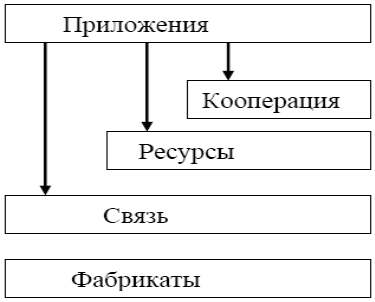


Рис. 2. Многоуровневая архитектура GRID

Поэтому возникает проблема, как существующее приложение, которое функционирует в рамках одного вычислительного ресурса, наиболее оптимально разместить в GRID; как определить сам факт возможности адаптации приложения, какие параметры влияют на этот факт.

Зачастую рассматриваются примеры приложений, которые разрабатываются для GRID с самого начала [3] – с нуля. Данные обзоры в основной своей массе посвящены практическим аспектам внедрения и написания программ под GRID инфраструктуру [5].

В большинстве своем разработчики GRID инфраструктуры, а также организации [6], разрабатывающие спецификации для систем такого рода, приводят примеры адаптации существующих приложений. Однако данные примеры носят зачастую рекомендательный характер [7] и пользователь должен интуитивно использовать тот или иной механизм адаптации.

Таким образом, необходимо построить адекватную модель, которая позволит наиболее оптимальным образом, на основании предъявляемых требований и критериев, предложить механизм адаптации существующего приложения для выполнения в GRID окружении.

В рамках данной статьи рассматриваются основные параметры, влияющие на выбор механизма адаптации. Также, в данной статье, в рамках адаптации, представлена методика расщепления данных в контексте GRID на основе их типа.

Понятие адаптации приложения

Под приложением будем понимать последовательность действий, которые необходимо выполнить для решения поставленной задачи. Также приложение и задача в рамках статьи будут употребляться как равнозначные понятия.

Под GRID приложением будем понимать приложение, которое необходимо выполнить в рамках GRID инфраструктуры. Таким образом, задача в GRID – это программа, выполняющаяся в определенной точке GRID, которая может производить расчеты, выполнять системные команды, перемещать или подготавливать данные, а также управлять различного рода техникой. Задача состоит из исполняемых файлов, а также входных и выходных данных. Для выполнения задачи требуются ресурсы: вычислительные, хранения, сетевые и т.д., которые в свою очередь формируются при помощи кластерного программного обеспечения в вычислительные комплексы.

Зачастую, для оптимального использования инфраструктуры GRID задачи разбиваются на подзадачи таким образом, чтобы позволить их параллельное выполнение на различных узлах. Существуют также задачи, параллельное выполнение которых не возможно ввиду определенных условий, например, необходимость в данных, которые могут быть получены только на предыдущем этапе алгоритма работы программы.

Таким образом, под адаптацией приложения для его последующего выполнения в GRID окружении, будем понимать процесс переноса приложения в инфраструктуру GRID.

Принцип адаптации

Пусть на вход адаптера M поступает задача (приложение) T, тогда на выходе необходимо получить механизм адаптации A. Задача T в свою очередь состоит из исполняемого кода и потока входных данных D для данной задачи. Каждая задача, поступающая на вход M, принадлежит определенному типу (классу) задач T_C; в зависимости от порядка обработки входных данных D для приложения T при разработке адаптера, необходимо также ввести понятие типа данных D_C. Таким образом, на основании класса приложения T_C, типа данных D_C, необходимо получить наиболее оптимальный механизм адаптации A_C (рис. 3).

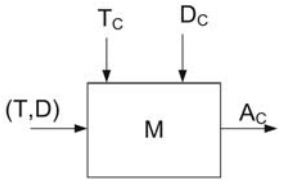


Рис. 3. Прототип модели адаптации приложений под GRID

Параметр тип данных D_C, будем рассматривать только в совокупности с классом приложения T_C, так как задача всегда требует данные для обработки, а данные могут поступать на вход адаптера M только вместе с задачей.

Анализ типов данных приложений D_C в контексте GRID-технологии

Под типом данных D_C будем понимать последовательность поступления данных для их последующей обработки, таким образом можно выделить два класса (типа) данных: с возможностью параллельной обработкой и последовательной. Процесс последовательной обработки данных представлен на рис. 4. Пусть существует некий источник данных S, с объемом данных D_i необходимой для обработки, тогда задача T будет последовательно обрабатывать данные и на выходе формировать результат R.

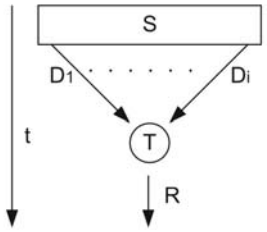


Рис. 4. Последовательная обработка данных

Пусть существует некий источник S, который содержит объем данных D_i необходимых для обработки задачей T (рис.5), тогда данные могут быть обработаны параллельно в том случае, если конечный результат R равняется сумме результатов R_i, полученных в результате параллельной обработки T_i параллельных задач. Иными словами

$$R = R_1 + R_2 + R_3 + \ldots + R_i,$$

где i – количество параллельно выполняемых задач.

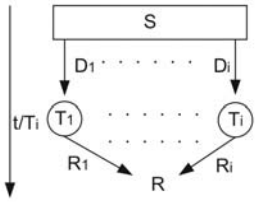
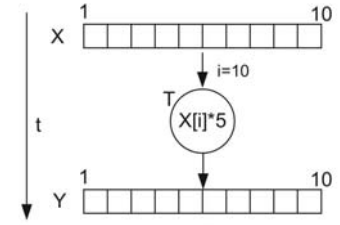
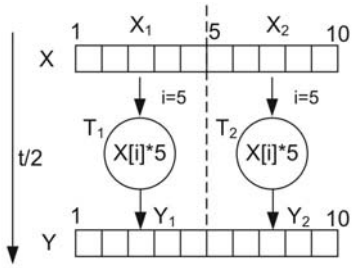


Рис. 5. Параллельная обработка данных

Рассмотрим пример (рис. 6). Пусть имеется массив X, состоящий из 10 элементов. Каждое значение элемента данного массива необходимо увеличить в 5 раз.



а



б

Рис. 6. Пример обработки данных:
а – последовательная; б – параллельная

Таким образом, одна задача T (рис. 6, а), будет выполнять данную работу в течении t времени. Если разбить массив X на 2 части (рис. 6, б) то получим соответственно массив X_1 и X_2, с размером по 5 элементов. Затем, значение элементов массивов X_1 и X_2 передавать в качестве данных на вход двух экземпляров задачи T_1 и T_2, в данном случае время обработки сократиться в два раза $t/2$.

Однако, $t/2$ – это идеальный случай, на самом деле необходимо учитывать время распределения данных и сбора результата, этот параметр необходимо учитывать, так как, возможно, результат работы приложения может передаваться в качестве входных данных для следующего этапа обработки. Например, на основании типа данных D_C принято

решение разделить входной поток данных представленный в виде файла F на 5 F_i частей (где i=1..5), для этого на удаленных узлах GRID было запущено на выполнение 5 T_i экземпляров задачи T (где i=1..5). Для передачи входных данных созданным экземплярам T_i было передано F_i частей файла, тогда время начала и окончания выполнения задачи может изменяться в зависимости от временных расходов на передачу (допускаем, что узлы имеют одинаковую вычислительную мощность – время вычисления, т.е. $tT_1 = tT_2 = tT_3 = tT_4 = tT_5$) (рис. 7). В нашем случае (рис. 7) время выполнения задачи в GRID равняется сумме времени передачи (t передачи), времени выполнения задачи в GRID t_T и времени сбора полученного результата (t сбора).

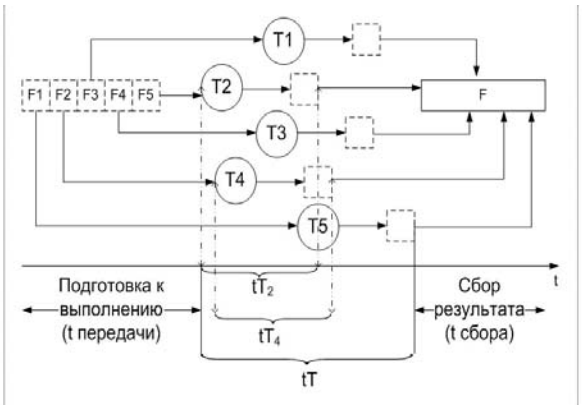


Рис. 7. Время выполнения в GRID адаптированной задачи на основе распараллеливания данных

Рассмотренный процесс распараллеливания данных также известен как технология **SIMD** (**S**ingle **I**nstruction, **M**ultiple **D**ata) [8].

Таким образом, мы определили два типа данных D_C с точки зрения адаптации приложения под GRID: с последовательной обработкой и с параллельной. Механизм расщепления входных данных необходимо производить только в том случае, если конечный результат выполнения задачи не измениться.

При переносе приложения в GRID окружение необходимо использовать возможность параллельной обработки данных и учитывать затраты на распараллеливание.

Выводы

В дальнейшем, необходимо определить, до какого уровня можно производить «дробление» или расщепление входных данных адаптируемого приложения, чтобы выигрыш от производительности был максимальным, определить факторы и параметры с точки зрения GRID-инфраструктуры, которые влияют на механизм расщепления данных и кто должен выполнять механизм расщепления, и провести классификацию приложений импортируемых в GRID.

Литература

1. Stockinger H. Defining the Grid: A Snapshot on the Current View. – Draft 1.0, 26 June 2006.

2. Jacob B., Luis F. Enabling Applications for Grid Computing with Globus. – IBM RedBooks, June 2003.

3. Foster I., Kesselman C., Tuecke S. The Anatomy of the Grid. Enabling Scalable Virtual Organizations. – Intl J. Supercomputer Applications, 2001.

4. Machado M. Enable existing applications for grid. – 22 Jun 2004, IBM Tech.

5. Foster I., Kesselman C. A Metacomputing Infrastructure Toolkit // International Journal of Supercomputer Applications. – 1997. – 11(2). – P. 115-128.

6. Flynn M. Some Computer Organizations and Their Effectiveness // IEEE Trans. Comput. – 1972. – Vol. C-21. – P. 948.

7. Chervenak A., Foster I., Kesselman C., Salisbury C., Tuecke S. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets // Journal of Network and Computer Applications (based on conference publication from Proceedings of NetStore Conference 1999). – 2001. – 23. – P. 187-200.

8. Jackson M., Antonioletti M., Chue Hong N.P., Hume A.C., Krause A., Sugden T., Westhead M. Performance Analysis of the OGSA-DAI Software // Proceedings of the UK e-Science All Hands Meeting 2004. – September 2004. – P. 256-258.

Поступила в редакцию 1.03.2007

Рецензент: д-р техн. наук, проф. А.В. Скатков, Севастопольский национальный технический университет, Севастополь.