УДК 519.685

ПРИМЕНЕНИЕ ИНТЕРПРЕТАТОРА СЦЕНАРИЯ GRAPHPLUS ДЛЯ УПРАВЛЕНИЯ РАСПРЕДЕЛЕННЫМИ ВЫЧИСЛЕНИЯМИ

© 2005 С.В. Востокин

Самарский государственный аэрокосмический университет

Рассматривается реализация интерпретатора сценария, разработанного в рамках проекта GraphPlus (graphplus.ssau.ru), предназначенного для управления GRID-системой распределенной пакетной обработки. Описан метод организации вычислений, архитектура интерпретатора и принцип интеграции с существующими GRID-системами.

Введение

Интенсивное развитие сетей общего пользования открывает возможности привлечения большого числа территориально и организационно разобщенных ресурсов для решения трудоемких вычислительных задач с использованием GRID-технологий. GRID система должна выглядеть прозрачно. Пользователь, выполнив регистрацию и передав в систему задание, не должен заботиться о том, на каком компьютере и в какой организации оно выполняется; какие действия предпринять в случае сбоя; как обеспечить целостность данных.

В настоящее время в наибольшей степени концепции прозрачного доступа к ресурсам в рамках GRID технологий отвечают системы пакетной обработки заданий [1]. Системы пакетной обработки предназначены для решения задач высокой вычислительной сложности из области High Throughput Computing (HTC). Для них характерно длительное время счета, а также автономность работы. Взаимодействие с программным окружением в таких приложениях сводится к чтению и записи файлов и использованию стандартных потоков ввода-вывода.

Для ускорения счета HTC-приложение должно быть описано как совокупность большого числа взаимосвязанных заданий. При этом актуальной является разработка удобных и гибких способов представления сценариев и интерпретаторов, формирующих по сценариям пакеты заданий. Рассматриваемая в работе система предназначена для динамической генерации заданий по их описанию на языке сценариев GraphPlus для систем пакетной обработки, работающих по принципу Master-Worker.

Метод

Архитектура Master-Worker представляет собой применение клиент серверного подхода для реализации распределенных вычислений. Сервер регистрирует подключения клиентов и, по их запросу, передает задания для вычислений. Клиенты рассматриваются как ненадежные и не требующие наличия постоянного соединения с сервером. В рамках данной архитектуры координацию вычислений выполняет сервер. Проблемой при этом является описание алгоритма работы сервера для каждой прикладной задачи таким образом, чтобы, во-первых, корректно организовать взаимодействие сотен вычислительных процессов и, во-вторых, совместить во времени передачу и обработку данных на клиентских компьютерах. Последнее свойство алгоритмов управления принципиально для GRID-приложений, так как в противном случае из-за высокой латентности сетей общего пользования резко снижается эффективность вычислений.

Для описания логики работы сервера приложения предлагается использовать специальный сценарий. Сценарий описывается на языке моделирования распределенных процессов GraphPlus [2]. Концептуальной основой данного языка является объектная модель, представляющая вычисления в виде дерева процессов и работ (рис.1).

Вычисления моделируются как обход

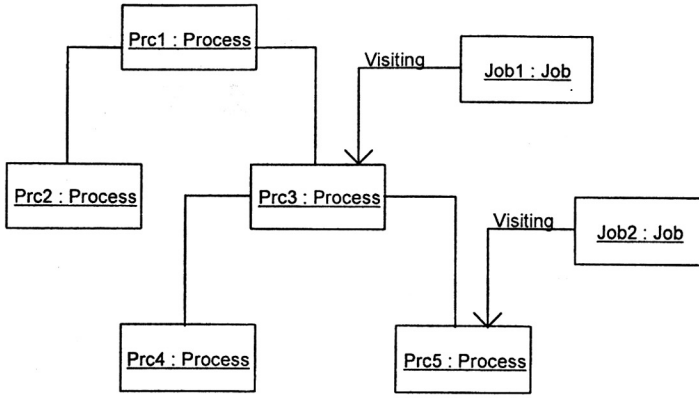

Рис. 1. Диаграмма объектов "дерево процессов и работ"

дерева процессов (PROCESS) специальными объектами-посетителями называемыми работами (JOB). При посещении JOB-объектами PROCESS-узлов выполняются вычисления, а также обмен информацией между объектами типа PROCESS и JOB. Параллелизм возникает за счет того, что в одно и тоже время на дереве процессов может присутствовать несколько активных JOB-объектов. Синхронизация вычислений реализуется посредством взаимоисключающего доступа JOB-объектов к PROCESS-объектам. Совмещение передачи и обработки данных на клиентских машинах будет организовано сервером автоматически, если сценарий описан таким образом, что в любой момент вычислений на дереве процессов имеется некоторое число JOB-объектов.

Для наглядного представления логики работы объектов типа PROCESS язык GraphPlus предлагает специальную графическую нотацию (рис. 2). В языке используется модульная организация кода, позволяющая скрыть реализацию типового сценария и обеспечить его повторное использование. Компактное описание деревьев процессов, состоящих из сотен узлов, возможно за счет применения рекурсии. Управление глубиной рекурсии достигается посредством модулей с параметрами. Глубина рекурсии определяет размер получающегося дерева процессов и степень масштабируемости алгоритма.

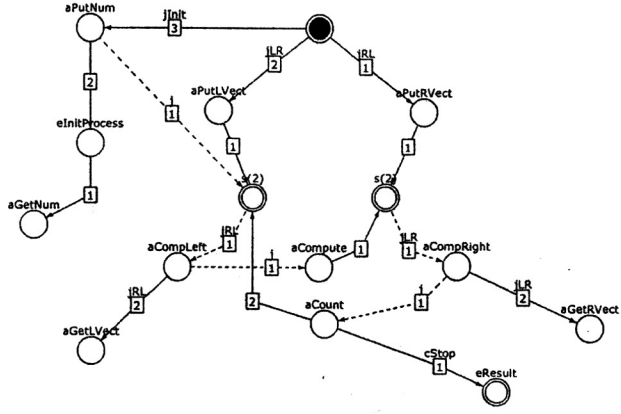

Рис. 2. Пример диаграммы процесса

Архитектура интерпретатора сценария

Общая архитектура системы пакетной обработки приведена на рис. 3. Как видно из рисунка, функция интерпретатора состоит в определении порядка запуска исполняемых файлов пакета заданий, в соответствии со сценарием. Файл сценария представляет собой закодированное представление дерева процессов и работ, построенное транслятором языка GraphPlus. Интерпретатор сценария формирует поток регистрации задач для Master-процесса системы пакетной обработки. При этом интерпретатор следит за тем, чтобы все задачи, одновременно находящие на обработке, были независимы по данным. Пополнение списка запущенных на обработку задач осуществляется интерпретатором по мере их завершения.

Отметим, что интерпретатор сценария и Master-процесс пакетной системы исполняются параллельно и взаимодействуют асинхронно. Как только в интерпретатор поступает информация о завершении задачи, он обновляет данные о состоянии вычислений и, если это возможно, добавляет новые задачи в систему пакетной обработки.

Рассмотрим, как объектная модель описания вычислений языка GraphPlus отображается на внутренние данные интерпретатора сценария. Структура дерева PROCESS-объектов, а также их текущее состояние хранится в файле данных prc.dat (рис.4). Все JOB-объекты для реализации обхода дерева хра-

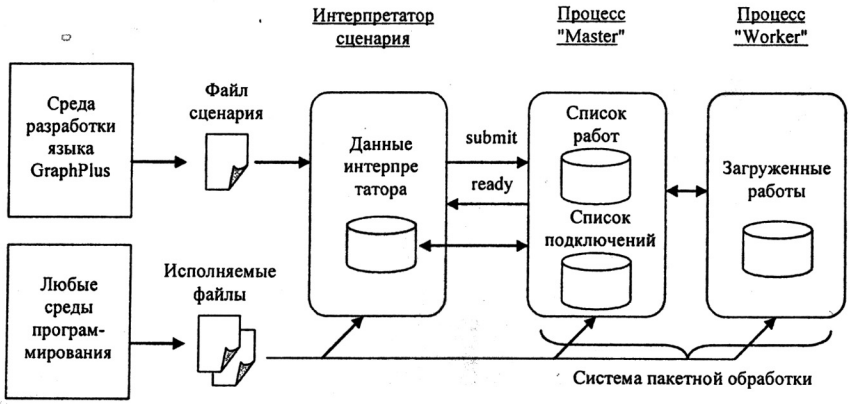

Рис. 3. Архитектура пакетной системы

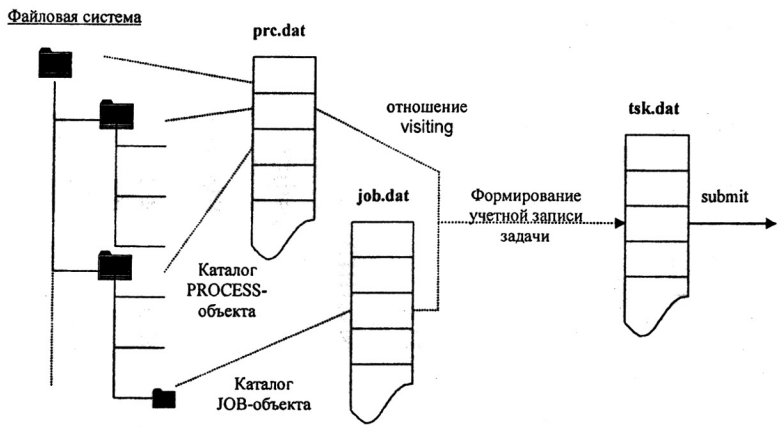

Рис. 4. Структуры данных интерпретатора

нятся в файле данных job.dat. Состояние задач, запускаемых по мере обхода дерева процессов, отслеживается в файле данных tsk.dat. Файлы данных представляют собой текстовые файлы с записями фиксированного размера. Транслятор адресует объекты по их номерам и находит объекты в базах, преобразуя номера в смещения относительно начала файлов. В тоже время, благодаря читаемому текстовому формату, имеется возможность отслеживать изменения баз данных при помощи любых программ для просмотра текстовых файлов. Эта возможность используется для отладки сценария.

Состояние PROCESS- и JOB- объектов, определяемое пользователем, хранится в дереве каталогов файловой системы. Переменные PROCESS- и JOB- объектов представлены файлами, размещенными в соответствующих подкаталогах. Методы PROCESS-объектов представляют собой исполняемые файлы, в которые при запуске в качестве параметров передаются имена обрабатываемых файлов. При регистрации задачи используется следующая информация: адрес исполнимого файла, список обрабатываемых файлов, адрес рабочего каталога, в котором размещены файлы задачи. Коллизии, которые потенциально могут возникнуть при асинхронном доступе к файловой системе, разрешаются интерпретатором сценария. Файловая система может быть локальной на машине интерпретатора, а также распределенной. В случае локального размещения файлов необходимое копирование исходных файлов и файлов с результатами вычислений выполняется Master-процессом пакетной системы. В случае распределенной системы Worker-процессы имеют непосредственный доступ к необходимым файлам, а кэширование осуществляется внутренними механизмами файловой системы.

Особенностью сценария вычислений является строго фиксированное число PROCESS- и JOB-объектов. Благодаря этому, при инициализации пакета можно сформировать структуру каталогов процессов и заполнить соответствующие базы данных. В ходе вычислений изменяется содержимое каталогов и записей управляющих баз данных, но не меняется их структура.

Все состояние вычислений хранится в каталоге файловой системы. В любой момент, по запросу пользователя, можно приостановить исполнение сценария и возобновить его исполнение с прерванного места. Таким образом, формируются контрольные точки для защиты от сбоев и для переноса процесса интерпретатора с одной машины на другую.

Интерпретатор реализован на языке C++ с использованием стандартной библиотеки шаблонов STL. Текущая реализация выполнена для операционной системы Windows. Имеется возможность переноса кода интерпретатора в другие операционные системы.

Аналогичные разработки

Известно несколько способов описания сценариев для систем пакетной обработки. Простейшим из них является перечисление всех возможных данных, поступающих на вход программы, при помощи шаблонов. Такая возможность реализована в языке ClassAd [3] системы Condor. Зависимости по управлению и по данным между программами пакета описываются при помощи ориентированных ациклических графов. Такой способ, например, реализован в утилите DAGMan [4] системы Condor. Недостатком является необходимость формирования большого числа исходных файлов и отсутствие возможности описания циклических процессов.

Имеется множество систем, использующих функциональную парадигму для описания вычислений, например, Chimera [5] и GRACE [6]. Метод параметризации модулей языка GraphPlus напоминает функции высших порядков, однако, имеет целью описание статических структур и не выражает никакой императивной семантики. Императивная семантика языка GraphPlus представляется объектной моделью, описываемой как взаимодействие JOB- и PROCESS- объектов.

Наиболее гибким способом является реализация сценария на известном процедурном или объектно-ориентированном языке (C/C++,Java). Такой метод используется в системе X-Com [7], системе XtremWeb [8]. Интерпретатор системы GraphPlus может быть интегрирован в эти системы.

Заключение

Описана реализация интерпретатора сценария, разработанного в рамках проекта GraphPlus (graphplus.ssau.ru). Достоинство предложенного метода заключается в использовании высокоуровнего языка, реализующего объектную парадигму, модульность, механизм шаблонов и визуальное программирование. Интерпретатор сценария легко интегрируется с имеющимися GRID-системами распределенной пакетной обработки.

СПИСОК ЛИТЕРАТУРЫ

1. Douglas Thain and Miron Livny, "Building Reliable Clients and Servers", in Ian Foster and Carl Kesselman, editors, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 2003, 2nd edition. ISBN: 1-55860-933-4.
2. *Востокин С.В.* Технология моделирования распределенных систем, основанная на визуальном языке и ее приложения //Известия СНЦ РАН. 2004. Т. 6, №1.
3. Condor® Version 6.6.8 Manual. Condor Team, University of Wisconsin–Madison. February 8, 2005.
4. Condor® Version 6.6.8 Manual. Condor Team, University of Wisconsin–Madison. February 8, 2005.
5. *Foster I., Vockler J., Wilde M. and Zhao Y.* Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation. 14th International Conference on Scientific and Statistical Database Management, Edinburgh, July 23, 2002.
6. *Абрамов С.М., Васенин В.А., Мамчиц Е.Е., Роганов В.А., Слепухин А.Ф.* Динамическое распараллеливание программ на базе параллельной редукции графов. Архитектура программного обеспечения новой версии Т-системы // Высокопроизводительные вычисления и их приложения. Труды Всероссийской научной конференции. Черноголовка, 2000.
7. *Воеводин Вл., Филамофитский М.* Суперкомпьютер на выходные//Открытые системы. 2003. №05.
8. *Cappello Fr., Djilali S., Fedak G., Herault Th., Magniette Th., Nüri V. and Lodygensky O.* Computing on Large Scale Distributed Systems: XtremWeb Architecture, Programming Models, Security, Tests and Convergence with Grid. Future Generation Computer Science, 2004. (http://www.elsevier.com/locate/future)

USING THE GRAPHPLUS SCRIPT INTERPRETER FOR DISTRIBUTED COMPUTATIONS CONTROL

© 2005 S.V. Vostokin

Samara State Aerospace University

The article discusses an implementation of the script interpreter, which was developed as a part of GraphPlus project (graphplus.ssau.ru). The program is a master process for GRID-enabled batch systems. Computational techniques, architecture, and method of integration with known batch systems are described.