УДК 004.3

ИНСТРУМЕНТАЛЬНАЯ СРЕДА МЕТАКОМПЬЮТИНГА ПРОЕКТА GRAPHPLUS

© 2006 С. В. Востокин

Самарский государственный аэрокосмический университет

Рассматривается реализация компактной инструментальной среды метакомпьютинга, разработанной в рамках проекта GraphPlus (graphplus.ssau.ru). Описан метод организации вычислений, состав инструментов и стадии трансляции исходного представления сценария в файл управления сервером. Приводится сравнение описанной инструментальной среды с проектами аналогичного назначения.

Введение

Актуальным направлением исследований в области параллельных вычислений в настоящее время являются метакомпьютинг и GRID-технологии [1]. Рост интереса к данной проблематике связан с активным развитием сетевой инфраструктуры общего пользования, представляемой сетью Internet и сетями нового поколения. Причиной, по которой высокопроизводительные вычисления на основе GRID-технологий стали развиваться относительно недавно, является сложность алгоритмов управления вычислениями по сравнению с широко используемыми суперкомпьютерными и кластерными системами. Это, прежде всего, касается методов защиты метакомпьютерных систем от злонамеренных действий, поддержания вычислений при возможных отказах или динамическом изменении конфигурации системы, а также повышения производительности вычислений за счет динамического переназначения процессов (load balancing). Решение перечисленных проблем открывает доступ к огромным вычислительным ресурсам, превосходящим по мощности одиночные суперкомпьютеры. Потребность в больших вычислительных мощностях, в свою очередь, обусловлена прогрессом в таких областях, как биотехнология, экономическое прогнозирование, моделирование физических процессов, где и возникают задачи большой трудоемкости.

В статье рассматривается инструментарий, автоматизирующий процесс разработки приложений для метакомпьютера, реализующего централизованное управление вычислениями.

Метод

Наиболее простым способом построения метакомпьютера является реализация клиент-серверной архитектуры, в которой сервер выполняет управление вычислениями, а в функцию клиентов входит прием заданий от сервера и отправка на сервер результатов. В рамках данной архитектуры функции организации сетевого взаимодействия, отказоустойчивости, управления нагрузкой выполняются сервером. Проблемой при этом является описание алгоритма работы сервера для каждой прикладной задачи таким образом, чтобы, во-первых, корректно организовать управление сотнями вычислительных процессов и, во-вторых, совместить во времени передачу и обработку данных на клиентских компьютерах. Последнее свойство алгоритма управления принципиально для метакомпьютерных приложений, так как в противном случае резко снижается их эффективность.

Для описания логики работы сервера метакомпьютерного приложения предлагается использовать файл сценария. Сценарий описывается на специально разработанном языке моделирования распределенных процессов GraphPlus [2]. Концептуальной основой данного языка является объектная модель, представляющая вычисления в виде дерева процессов рис. 1.

Вычисления моделируются как обход 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 то

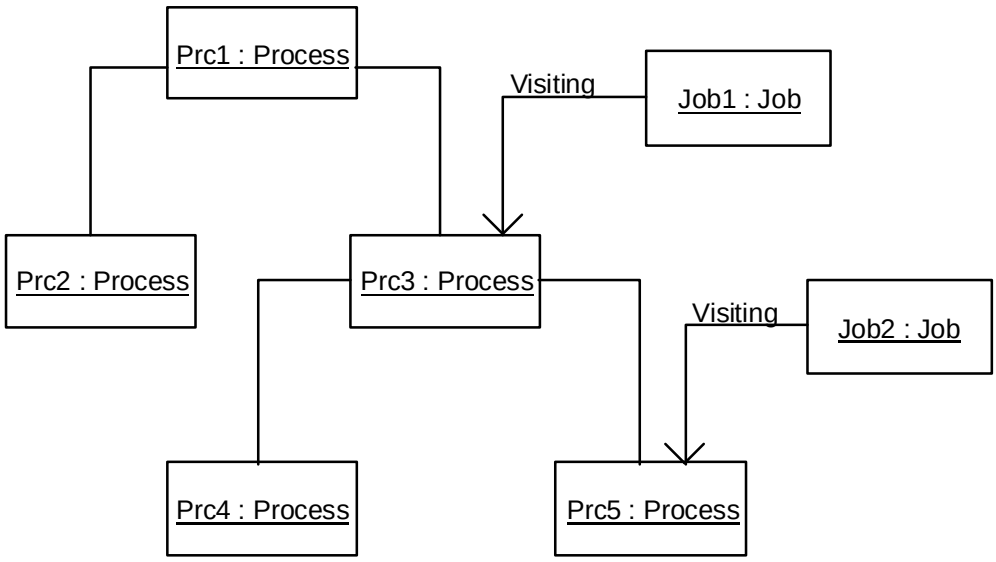

Рис. 1. Диаграмма объектов "дерево процессов и работ"

же 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. 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 рекурсии. Управление глубиной рекурсии достигается посредством применения модулей с параметрами. Глубина рекурсии определяет размер получающегося дерева процессов и степень масштабируемости алгоритма.

Инструментарий

Минимальный комплект инструментов разработки сценария, управляющего работой сервера, включает графический редактор, транслятор и отладчик. Для написания текстовых фрагментов сценария применяется штатный текстовый редактор. Содержательные части алгоритма организуются как исполнимые файлы, реализующие последовательные процедуры. Они могут быть написаны на любом языке программирования в соответствующей инструментальной среде и вызываются на исполнение на сервере либо на клиенте. При отладке сценария все вызовы выполняются отладчиком на локальной машине.

Файлы сценария, поступающие на обработку транслятора, показаны на рис. 2. Сценарий организован как совокупность модулей. Физически модуль представляет собой каталог в файловой системе. Каталог модуля обязательно содержит файл описания модуля и обычно содержит несколько файлов PROCESS-объектов. В файле описания модуля объявляются ссылки на все типы PROCESS-объектов, относящихся к данному модулю. Дополнительно файл модуля содержит объявления всех структурных элементов, из которых строятся процессы. Среди них объявления JOB-объектов и связей с внешними exe-модулями.

Описание каждого PROCESS-объекта может состоять из трех файлов. Обязательным является файл с текстовым описанием процесса на языке GraphPlus. Для процессов,

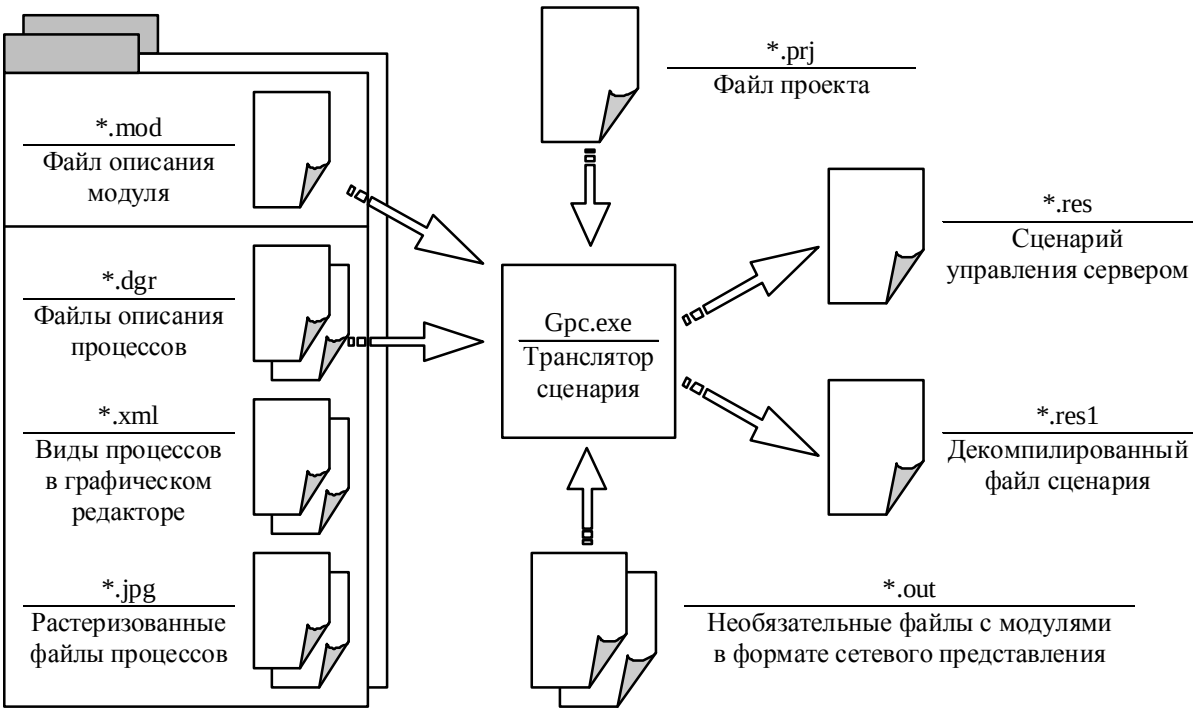

Рис. 2. Интерфейс транслятора сценария

имеющих простую структуру, данный файл может быть создан в штатном текстовом редакторе. Если PROCESS-объект сложный, то его структуру можно “нарисовать” в специализированном графическом редакторе. Редактор хранит изображение диаграммы процесса в XML-файле и автоматически формирует текстовое описание PROCESS-объекта на языке GraphPlus. При желании можно сохранять графический образ PROCESS-объекта в jpg-файле рис. 3.

Множество модулей, из которых формируется файл сценария, задается в файле проекта. Интерпретируя файл проекта, транс-

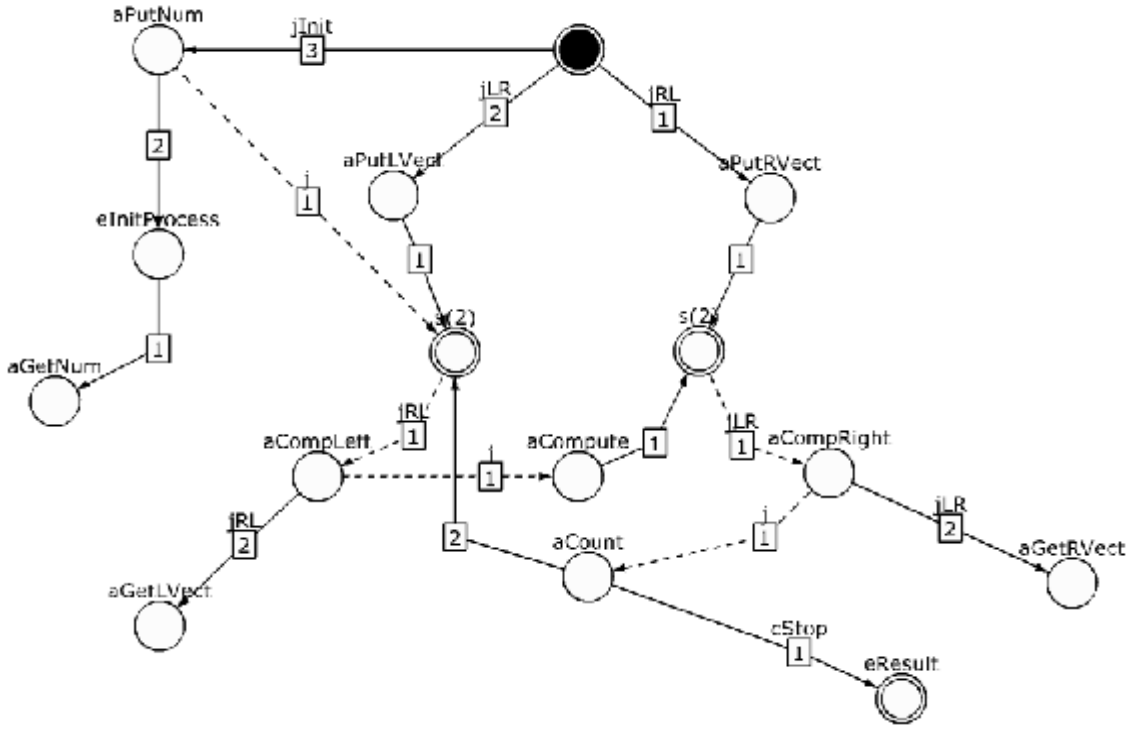

Рис. 3. Пример диаграммы процесса

лятор “узнает”, где размещается конкретный модуль и как находить файлы PROCESS-объектов для данного модуля. Часть модулей проекта может быть взята из библиотеки типовых шаблонов вычислительных процессов. Новый процесс может представлять собой специализацию уже готового шаблона. Шаблон вычислительного процесса на языке GraphPlus описывается как совокупность логически связанных модулей, инкапсулирующих как поведенческие, так и структурные свойства управляющих алгоритмов. Применение шаблонов позволяет упростить процесс программирования и повышает надежность кода.

Стадии трансляции показаны на рис. 4. Унифицированное взаимодействие между разными стадиями трансляции достигается за счет применения специального формата хранения модуля. Все промежуточные файлы – это модули, закодированные в виде сети объектов и атрибутов. Данная сеть описывается в текстовом файле как последовательность команд, выполнение которых позволяет восстановить сеть объектов и атрибутов модуля в памяти транслятора. Все преобразования в процессе трансляции происходят с сетевым представлением модуля. При необходимости сетевое представление модуля может быть преобразовано к исходному текстовому виду.

На первой стадии трансляции формируется сетевое представление для каждого модуля проекта. Полученные файлы помещаются в специальный каталог, указываемый в файле проекта. Далее, на второй стадии, происходит замена формальных параметров модулей фактическими параметрами. При этом один модуль с параметрами может породить несколько модулей без параметров с расширением *.bin. Модули, получающиеся на стадии связывания параметров, помещаются в специальный каталог, заданный в файле проекта. Файлы в каталоге различаются благодаря применению декорированных имен. Наконец, завершающей стадией трансляции является связывание модулей с предыдущей стадии в один модуль. На данной стадии разрешаются команды импорта и экспорта, содержащиеся в модулях. Результатом трансляции сценария является один модуль без параметров и команд импорта-экспорта. Для различения структурных элементов, унаследованных из разных модулей, используются декорированные имена. Заметим, что в процессе трансляции проверяется корректность

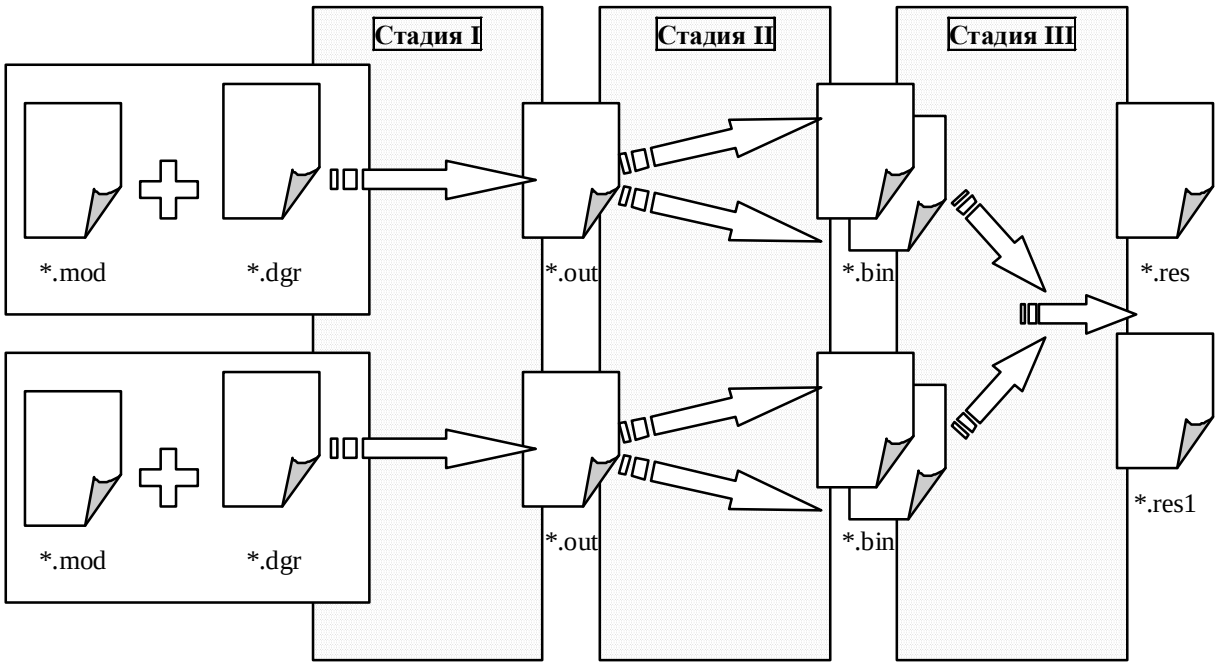

Рис. 4. Стадии трансляции сценария

выполнения каждой стадии, о чем выдаются соответствующие диагностические сообщения. В зависимости от настроек в файле проекта они могут выдаваться на русском или английском языках и направляться на консоль или в файл. В трансляторе также имеется возможность сокрытия реализации модулей путем подключения их в проект в виде готового сетевого представления.

Отладчик реализован как консольное приложение, выдающее сообщения о действиях, выполняемых JOB-объектами во время посещения PROCESS-объектов. Текущее состояние вычислений хранится в базах данных, реализованных в виде текстовых файлов. Наблюдать за изменениями состояния вычислений можно с использованием штатных средств операционной системы, например, в web-браузере.

Реализация инструментальных средств проекта GraphPlus выполнена на языке C++ с использованием стандартной библиотеки шаблонов STL. Зависимые от платформы участки кода, содержащиеся в отладчике, изолированы в специальных классах. Благодаря этому обеспечивается мобильность кода. Текущая реализация инструментальной среды выполнена для платформы Windows 98/NT/XP. Размер исходного кода составляет около 8000 строк.

Аналогичные разработки

Наиболее известными системами, реализующими архитектуру ведущий-ведомый (master-worker), являются системы Condor-MW, XtremWeb, Entropia, SETI@home, Nimrod и другие [3]. В отличие от них в системе GraphPlus основное внимание уделяется разработке новых методов программирования, ориентированных на различные парадигмы организации распределенных вычислений.

Среди отечественных разработок компактных инструментальных сред метакомпьютинга можно выделить проект X-Com НИВЦ МГУ [4], проект GRACE Центра телекоммуникаций и технологий Интернет МГУ [5], проект языка MC# [6].

В системе X-Com сервер управляется из пользовательского кода. Создание абстракций для описания распределенного алгоритма целиком лежит на программисте. Судя по описанию системы X-Com, возможно применение интерпретатора системы GraphPlus в качестве кода управления сервером.

Проект GRACE, как и большое число аналогичных проектов, реализует функциональную парадигму описания вычислений. 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.

Язык MC# – это расширение объектно-ориентированного языка C#, предназначенное для метакомпьютинга. Путем расширения синтаксиса организуется неявное встраивание в исходную программу кода взаимодействия с сервером на языке C#. Применение платформ с переносимым кодом Java и .NET, по-видимому, будет являться основой новых метакомпьютерных технологий. Язык GraphPlus и его модель вычислений предусматривает генерацию кода для данных платформ.

Заключение

Приведено описание компактной инструментальной среды метакомпьютинга, разрабатываемой в рамках проекта GraphPlus (graphplus.ssau.ru). Инструментарий предназначен для подготовки сценария работы сервера, выполняющего централизованное управление вычислениями, и основан на использовании высокоуровнего языка, реализующего объектную парадигму, модульность, механизм шаблонов и визуальное программирование.

Список литературы

1. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления //СПб.: БХВ-Петербург, 2002.

2. Востокин С. В. Технология моделирования распределенных систем, основанная на визуальном языке и ее приложения // Известия СНЦ РАН, том 6, №1(10), 2004. - С. 185-193.

3. 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.

4. Воеводин Вл., Филамофитский М. Суперкомпьютер на выходные // Открытые системы, № 5, 2003.

5. 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 // Труды Всероссийской научной конференции «Высокопроизводительные вычисления и их приложения», Черноголовка, 2000.

6. Сердюк Ю., Гузев В. Асинхронный параллельный язык программирования для платформы Microsoft .NET // 7-ая Международная Конференция PACT'2003, Нижний Новгород, Россия, 15-19 Сентября 2003, Lecture Notes in Computer Science, v. 2763, pp. 236 -243, Springer, 2003.

METACOMPUTING TOOL PLATFORM IN THE CONTEXT OF GRAPHPLUS PROJECT

© 2006 S. V. Vostokin

Samara State Aerospace University

The paper discusses the realization of metacomputing compact tool platform developed in the context of Graphplus project (graphplus.ssau.ru). The procedure of computations, instrument composition and the stages of transmitting scenario original presentation to the file of server management are described. The instruments described are compared with projects with a similar purpose.