АВТОМАТИЗАЦИЯ РАЗРАБОТКИ ГРИД-ПРИЛОЖЕНИЙ

А. А. Бабошин, В. И. Воробьев

Санкт-Петербургский институт информатики и автоматизации РАН

СПИИРАН, 14-я линия ВО, д. 39, Санкт-Петербург, 199178

<vvi@iias.spb.su>, <andrey.baboshin@gmail.com>

УДК 004.8

Бабошин А. А., Воробьев В. И. **Автоматизация разработки грид-приложений** // Труды СПИИРАН. Вып. 7. — СПб.: Наука, 2009.

Аннотация. *В работе проводится анализ существующих средств для автоматизации разработки (и отладки) параллельных программ. Дается обзор формальных моделей, на которых основаны описываемые средства. Делаются предложения по применению существующих моделей для получения средств автоматизации разработки параллельных программ на базе грид.* — Библ. 11 назв.

UDC 004.8

Baboshin A. A., Vorob'ev V. I. **Automation of grid-application's development** // SPIIRAS Proceedings. Issue 5. — SPb.: Nauka, 2009.

Abstract. *It develops an analysis of a current tools for automation of parallel programs development (and debugging). It reviews formal models for those tools. It submits a proposal for applying current models for development tools for automation of parallel grid-based programs development.* — Bibl. 11 items.

1. Введение

Практика распараллеливания показала, что существует два основных подхода к обмену данными между параллельными процессами: а) обмен сообщениями (message passing) и б) разделяемая память (shared memory).

В то же время найдены следующие формы организации параллельных задач, основанные на вышеупомянутых подходах: а) итеративный параллелизм для реализации нескольких процессов, каждый из которых содержит циклы; б) рекурсивный параллелизм для программ с одной или несколькими рекурсивными процедурами, вызов которых независим (технологии «разделяй-и-властвуй» или «перебор-с-возвращением»); в) производители и потребители, осуществляющие взаимодействие неравноправных процессов: одни из процессов «производят» данные, другие их «потребляют»; г) клиент-сервер-модель; д) «управляющий и рабочие» — модель организации вычислений, при которой существует поток, координирующий работу всех остальных потоков; е) взаимодействующие равные — модель, из которой исключен не занимающийся непосредственными вычислениями управляющий поток.

Описанные формы основаны на эвристических подходах, и сами по себе они не подходят для построения формальных моделей. Для автоматизации необходимы формальные модели. Такие модели, например, описаны в работах ([1], [2], [3]).

2. Модели представления параллельных программ

Потоковые модели [3]. Потоковые вычисления реализуются по мере готовности данных (можно сравнить с «ленивыми» вычислениями). Распространенная форма представления — граф (сеть) потоков данных, существенное

свойство которого — однократное присваивание (отсутствие повторной сходимости). Потоковая модель распределенной программы рассматривается как множество процессов, взаимодействующих посредством асинхронной посылки сообщений через входные, а иногда и выходные буферы и инициируемых готовностью входных сообщений.

Проблема реализуемости потоковых моделей заключается в согласовании параметров очередей сообщений (входных и выходных буферов) таким образом, чтобы учесть все допустимые истории процессов программы, когда ни один из них не блокируется. Для решения этой проблемы вводятся графы специального вида — маркированный потоковый граф.

Потоковые модели не дают всей полноты описания параллельных процессов, т.е. не позволяют определить детерминированность процессов, поэтому в дополнение к ним используются **событийные модели** [1, 2]. С точки зрения событийных моделей процессы могут быть *детерминированными* или *недетерминированными* и *однозначными* или *неоднозначными*. Детерминированный процесс — каждому начальному событию соответствует не более одного конечного события. В случае недетерминированных процессов — каждому начальному событию соответствует множество конечных событий. Процесс однозначный — существует отображение множества входных во множество выходных сообщений процессов.

Уже на этапе проектирования можно проанализировать программу, отнести ее к одному из классов и предпринять необходимые действия при обнаружении проблем. В следующем разделе будет показано, как рассмотренные модели поддерживаются программными средствами.

3. Системы автоматизации

Наиболее развитыми на сегодняшний день являются средства Valgrind [6] и V-Ray [5]. Valgrind лицензирован под GNU GPL. Он позволяет самостоятельно определять ошибки в управлении памятью и обработке потоков, включает средства для детального исследования производительности программ, позволяет разрабатывать на основе себя новые средства анализа параллельных программ.

Valgrind содержит средства анализа по следующим основным направлениям: а) обработка ошибок работы с памятью; б) обработка ошибок работы с потоками; в) профилирование работы с кэшем; г) профилирование работы с «кучей»; д) генерация графа вызовов (для визуализации результатов используется KcacheGrind [7]).

В рамках данной работы нам наиболее интересен д)-пункт. Граф вызовов конструируется в процессе выполнения программы. Для последующего анализа доступна следующая информация: отношение функций вызывающий-вызываемый, количество вызовов, номера исполняемых инструкций (и их связь с исходным кодом программы), характеристика доступа к памяти. Valgrind позволяет получить распределение временных затрат на выполнение программы по различным её потокам/функциям. Valgrind позволяет получить информацию об ошибках синхронизации в программах на языках C, C++ и Fortran. Обнаруживаются следующие типы ошибок: а) deadlocks; б) data races (повреждение данных); в) неправильное использование API для межпроцессного взаимодействия (например, рекурсивное блокирование нерекурсивного мьютекса). Данные ошибки обнаруживаются за счет отслеживание состояния памяти програм-

Труды СПИИРАН. 2008. Вып. 7. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2008. Issue 7. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

мы, мониторинга различных связанных с выполнением потоков событий. Например, два различных потока пытаются одновременно вывести на экран какую-либо информацию — это может привести к повреждению данных.

Для визуализации полученных при анализе с помощью valgrind данных используется ряд сторонних программ. Например, выше упомянутый Kcachegrind отображает полученные данные о графе вызова и позволяет осуществлять навигацию по исходному коду программы. Более подробно ознакомиться с механизмами работы valgrind можно в работе [8].

V-Ray, разработанный НИВЦ МГУ, является средством для анализа и преобразования программ, включает в себя инструменты для статического и динамического анализа, макро- и микроанализа. Предоставляются следующие возможности анализа:

1. Статический микроанализ применяется для статического анализа структуры подпрограмм и включает в себя следующие средства: а) иерархический граф управления; б) pardo циклы; в) обход иерархии; г) связь с исходным текстом; д) разные термины представления; е) разные типы зависимостей.
2. Статический макроанализ применяется для статического анализа структуры проекта в целом и включает в себя следующие средства: а) граф вызовов; б) связь с исходным текстом; в) разные методы представления; г) вложенность циклов; д) операции: рост, из, в; е) циклический профиль; ж) граф использования общей памяти.
3. Динамический анализ: а) представление процессов; б) оерации взаимодействия; в) масштабирование; г) различные способы представления трасс; д) статистика; е) фильтрация; ж) синхронизация; з) сопоставительный анализ.

Резюмируя рассмотренные инструменты, приведем их краткий сравнительный анализ (табл. 1).

Таблица 1

Сравнительный анализ Valgrid и V-Ray

	Valgrid	**V-Ray**
Статический анализ	+	+
Динамический анализ	+	+
Открытость разработки	GPL	Закрытая разработка
Основное направление использования	Параллельные программы на множестве платформ	суперкомпьютеры
Поддерживаемые API	POSIX-совместимые	Специфичное для суперкомпьютеров
Поддерживаемые ОС	POSIX-совместимые	Специфичное для суперкомпьютеров

На основе сравнения, приведенного в таб. 1, можно сделать вывод, что V-Ray и Valgrind предоставляют сопоставимые возможности для анализа программ во время выполнения. Однако Valgrind обладает следующими преимуществами: а) открытая лицензия и как следствие большое сообщество пользователей и разработчиков; б) поддержка большего количества программных и аппаратных платформ. Эти преимущества делают его лучшим выбором для анализа программ на этапе выполнения.

Выше были рассмотрены средства автоматизации анализа уже разработанных программ. Кроме них нас интересуют средства для анализа программ на этапе разработки. Fraunhofer Resource Grid [10] предлагает пользователю язык GADL для описания грид-приложений. GADL (Grid Application Definition

Труды СПИИРАН. 2008. Вып. 7. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2008. Issue 7. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Language) основан на XML и включает в себя языки для описания сценариев (GJobDL), интерфейсов (GInterfaceDL), данных (GDataDL) и ресурсов (GResourceDL). Пользователю предоставляется инструмент для задания грид-приложений с помощью сетей Петри. Он написан на языке Java и выполняется в браузере, что освобождает пользователя от необходимости устанавливать на свой компьютер дополнительное программное обеспечение. После составления сценария он передается в грид-среду, будучи описанным на языке Petri Net Markup Language. Teuta [11] позволяет в визуальном режиме разрабатывать приложение с использованием workflow и проверять модель на завершенность.

4. Таксономическое описание параллельных процессов на грид-системах

Поскольку мы не можем предложить формальную модель для общего случая, то будем рассматривать таксономическое описание параллельных процессов на базе онтологий. Ограничимся рассмотрением генерации исходного кода для грид-приложений. В качестве исходного описания будем использовать диаграммы действий и диаграммы классов из языка UML.

Выберем типы диаграмм, необходимые (и достаточные) для полноценного описания распределенного приложения. С помощью диаграммы классов зададим статическое описание приложения. Классы определяются атрибутами, методами и связями с другими классами. Грид-сервисы описываются (в gLite[9]) с помощью языка WSDL, основанного на XML. Сервис может содержать в себе один или несколько методов, доступных для вызова пользователем. Специфицируем их при помощи методов класса на UML-диаграмме. С помощью диаграммы объектов опишем объекты, которые затем будут задействованы в диаграмме вариантов использования (use-case). Диаграммы состояний опишут состояния объектов и условия переходов из одного состояния в другое. Диаграммы вариантов использования позволит отразить отношения между объектами и их действиями (прецедентами). Для описания взаимодействия объектов выбирается диаграмма последовательностей (sequence). Она позволит отразить взаимодействие объектов упорядоченно по времени их появления.

Следующим шагом является построение онтологического описания на основе UML-диаграммы. Под термином *“онтология”* понимается формальная спецификация разделяемой концептуализации в некотором контексте предметной области. Она описывается множеством классов, множеством атрибутов классов, множеством доменов атрибутов и множеством ограничений. Множество ограничений включает в себя: 1) отношение (класс, атрибут, домен); 2) таксономические (быть экземпляром) и иерархические (быть частью) отношения между классами; 3) отношения совместимости классов; 4) ассоциативные отношения между классами; 5) ограничения на число классов в подмножестве классов; 6) функциональные ограничения.

Использование онтологий для описания грид-приложений открывает возможность использования методов анализа онтологий. Например, на основе выделения семантически связанных групп сущностей можно выделить параллельные ветви приложения. Используя редуцирование промежуточных графов, можно оптимизировать результирующий код. Кроме того, промежуточное звено упростит добавление результирующих систем, для которых будет генерироваться код.

Труды СПИИРАН. 2008. Вып. 7. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2008. Issue 7. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Опишем представление грид-приложения посредством онтологий. Диаграмма классов будет отражена классами онтологии и отношениями между ними. Диаграмма объектов будет отражена с помощью объектов онтологии. Состояния из диаграммы состояний выразим через классы, а правила перехода между ними выразим посредством введения нового типа связей – “переходов”. Прецеденты из диаграммы использований отразим специальными классами, а отношения между объектами распределенного приложения и этими прецедентами отразим посредством введения нового типа связей.

Диаграмма последовательностей потребует ввести новый тип связей, который будет связывать объекты друг с другом, кроме того, этот тип связей должен нести относительные временные отметки для задания порядка вызова, также необходимо отразить тип связей – синхронный, асинхронный, группы параллельных вызовов.

Диаграмма процесса трансформации будет выглядеть следующим образом (рис. 1):

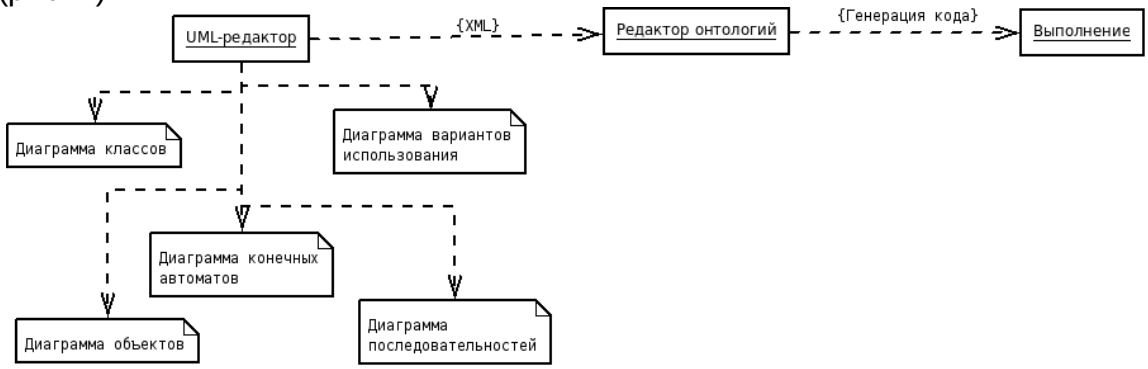

Рис. 1. Процесс преобразования UML-диаграмм в исполняемый код.

В работе предложена гибридная технология автоматизации разработки грид-приложений с использованием распространенных технологий разработки. Была разработана модель таксономического описания параллельных процессов.

Литература

1. *Воеводин В. В., Воеводин Вл. В.* Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.
2. *Воеводин В. В.* Математические проблемы освоения суперЭВМ. Вычис. проц. и сист., М.: Наука, 1984. Вып. 2. 10 с.
3. *Бурцев В. С.* Параллелизм вычислительных процессов и развитие архитектуры суперЭВМ. М.: ИВВС РАН, 1997. 152 с.
4. *Dijkstra E. W.* Hierarchical ordering of sequential processes // Acta Informatica 1971. Vol. 1. P. 115-138.
5. *Воеводин В. В.* V-Ray project [Электронный ресурс] // <http://parallel.ru/v-ray/> (по состоянию на 11.12.2008)
6. Valgring [Электронный ресурс] // <http://valgrind.org/> (по состоянию на 11.12.2008)
7. KcacheGrind [Электронный ресурс] // <http://kcachegrind.sourceforge.net> (по состоянию на 11.12.2008)
8. Nethercote N., Seward J. Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation // ACM. 2007. Vol. 42, issue 6. P. 89-100.
9. gLite [Электронный ресурс] // <http://glite.web.cern.ch> (по состоянию на 11.12.2008)
10. *Hoheisel A.* Grid Application Definition Language — GADL 0.2, Technischer Report, Fraunhofer FIRST, 2002.
11. Austrian Grid [Электронный ресурс] // <http://www.austriangrid.at> (по состоянию на 11.12.2008)