Информатика

УДК 004.272:004.4

ПРОГРАММНЫЙ КОМПЛЕКС ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ GRAPHPLUS TEMPLET

С. В. Востокин, А. Р. Хайрутдинов, В. Г. Литвинов

Самарский государственный аэрокосмический университет им. ак. С. П. Королёва (национальный исследовательский университет),
443086, Самара, Московское ш., 34.

E-mails: easts@mail.ru, khairutdinov@yandex.ru, doom-black@mail.ru

Описан программный комплекс для автоматизации параллельного программирования GraphPlus templet. Рассмотрены цели проектирования, архитектура, эффективность применения при решении задач численного моделирования, особенности в сравнении с аналогичными подходами. Описан состав комплекса программ, приведена оценка сложности параллельного программирования численных методов при его использовании.

Ключевые слова: *комплекс программ, параллельное программирование, модель вычислений, схемы программ, доменно-ориентированный язык, автоматизация программирования, численное моделирование.*

Последние 10 лет развития компьютерной индустрии характеризуются широким использованием параллельных вычислительных архитектур. Наряду с доступностью и широтой применения можно отметить их разнообразие: многоядерные настольные системы, кластерные системы и суперкомпьютеры, распределённые системы в сети интернет, процессоры CUDA. Развитие аппаратных архитектур порождает спрос на программное обеспечение, позволяющее эффективно использовать новые параллельные вычислительные ресурсы.

На основании опыта параллельного программирования можно выделить следующие актуальные проблемы, на преодоление которых направлен рассматриваемый в работе комплекс параллельного программирования.

Практика показывает, что программист средней квалификации обычно не может самостоятельно разработать параллельный алгоритм решения задачи. Более того, при наличии точного описания алгоритма возникают проблемы его реализации требуемыми средствами. Это кардинально отличается от последовательного программирования, где процесс перехода от спецификации к реализации в настоящее время достаточно формализован. Выходом является повторное использование имеющихся решений. Известно около двух десятков типовых решений в области параллельного программирования. Требуется предложить способ удобного описания и конструирования программ на

Сергей Владимирович Востокин (д.т.н., доц.), профессор, каф. информационных систем и технологий. *Андрей Ринатович Хайрутдинов*, аспирант, каф. информационных систем и технологий. *Владимир Геннадьевич Литвинов*, аспирант, каф. информационных систем и технологий.

их основе. Причём такие типовые решения должны быть по возможности языково-нейтральными.

Существует ряд проблем, обусловленных неудобством традиционных методов программирования параллельных вычислений. В первую очередь — проблема надёжности кода. Известно, что недетерминированный характер вычислительного процесса затрудняет выявление ошибок методами отладки. Также возникают трудности при переносе программ в различные параллельные архитектуры: например, многопоточные программы обычно не удаётся сделать распределенными, если это не предусматривалось изначально при проектировании. Представляет большой интерес разработка масштабируемых программ, не теряющих эффективности при увеличении аппаратных ресурсов. Общий подход в решении перечисленных проблем — использование формальной модели вычислений.

При наличии понимания особенностей параллельного алгоритма, методов его программирования встаёт вопрос о стоимости и сроках разработки программного продукта. Современные программисты привыкли работать в интегрированных средах. Процесс разработки и отладки параллельных программ в настоящее время отличается более низкой автоматизацией. Поэтому необходимо встраивание системы параллельного программирования в известные процессы разработки программного обеспечения и интегрированные среды разработки, поддерживающие их. Таким образом, проектирование программного комплекса `Graphplus templet` выполнено с учётом следующих требований:

1) возможность работы с типовыми решениями на разных языках программирования;
2) независимость прикладного кода от API управления вычислениями, а также взаимозаменяемость API без переработки приложений;
3) совместимость с коммерческими интегрированными средами разработки, в частности, с целью удобства отладки параллельных приложений стандартными средствами.

Семантическая основа языка системы `Graphplus templet` — модель процессов диффузного типа. В данной модели конструктивными элементами являются пассивные объекты — процессы и активные объекты — двунаправленные каналы, соединяющие их. Объекты выполняют свои методы при поступлении сообщений по каналам и в ответ могут сформировать произвольное число сообщений. Данная модель удобнее модели многопоточного программирования, позволяет гибко управлять состоянием вычислений, эффективно реализуется на компьютерах с распределенной и общей памятью.

Особенностью предлагаемого подхода является то, что семантика модели программирования передается не на специальном языке, а на традиционном языке программирования `C++` без использования примитивов для задания параллелизма. Это делается следующим образом. Предполагается, что весь код программы разбит на модули. Каждый модуль представляет собой один или несколько файлов. В программе различаются модули, код которых интерпретируется в терминах описанной вычислительной модели. Интерпретируемость означает, что в коде можно выделить участки, соответствующие элементам модели. Такие модули имеют строго определённую структуру, нарушать которую запрещено. Также в программе имеется специальный модуль с предопределённой структурой, описывающий механизм исполнения.

Исполнение программы является недетерминированным. Наличие нескольких возможных вариантов развития вычислительного процесса передаётся в модуле механизма исполнения с использованием генератора случайных чисел. Остальные модули могут содержать код произвольной структуры. Таким образом, программа в терминах модели исполнения является последовательной, построенной специальным образом с учётом ограничений на структуру кода, и передаёт функциональные аспекты поведения аналогичной параллельной программы. Далее требуется определить два алгоритма:

- алгоритм проверки интерпретируемости программы в терминах модели;
- алгоритм распараллеливания — преобразования исходной последовательной программы в параллельную программу с теми же функциональными свойствами.

Для этого вместо процедур лексического и синтаксического анализа используется процедура генерации. Каждый интерпретируемый модуль снабжается `XML`-спецификацией, описывающей объекты модели программирования, содержащиеся в нём. Код модуля генерируется по этой спецификации. При этом код состоит из чередующихся фрагментов — блоков кода, соответствующих элементам модели, а также тел методов и определённых пользователем структур данных. Границы таких фрагментов аннотированы при помощи специальных комментариев, указывающих, к какому структурному элементу модели относится код, написанный пользователем. Алгоритм генерации кода по `XML`-спецификации на известном языке является эталонной реализацией, определяющей смысл модели. Алгоритм основан на преобразованиях по шаблонам, так что связь между частями `XML`-спецификации и сгенерированным кодом легко обнаруживается программистом. Для дополнительного удобства в генерируемый код переносятся комментарии из `XML`-спецификации.

Алгоритм распараллеливания кода в общем случае заключается в извлечении блоков пользовательского кода из исходной программы; генерации новой программы с синхропримитивами параллелизма по `XML`-спецификации; включения извлеченных пользовательских блоков в код сгенерированной параллельной программы. Таким образом, примененный подход является простым, а следовательно, надёжным; языково-нейтральным, так как не требует лексического и синтаксического анализатора для целевого языка; позволяет точно сформулировать операционную семантику модели программирования в терминах последовательного языка, не прибегая к дополнительным математическим нотациям; программа отлаживается обычным способом.

Алгоритм генерации кода реализован в препроцессоре `templet`, совместная работа которого с интегрированной средой разработки `Microsoft Visual Studio` организуется, как описано ниже. Пусть код программы пишется на языке `C++`, что предусмотрено в текущей реализации. Модулем в данном случае является пара файлов: заголовочный файл с расширением `h`; файл реализации с расширением `cpp`. Код этих файлов помещается в общее пространство имён. Каждому модулю в проекте интегрированной среды разработки соответствует файл с `XML`-спецификацией, который тоже включается в проект (обычно в папку ресурсов приложения). Для файла `XML`-спецификации в проекте указывается способ его обработки. Перед сборкой проекта (pre-built) такой файл должен быть обработан `templet`-препроцессором, который генерирует фрагменты кода, соответствующие модели программирования в файлах модуля. Также для файла `XML`-спецификации указывается редактор. Им

может быть как штатный текстовый или XML-редактор, так и специальный графический редактор, визуализирующий модель программирования при помощи графических обозначений.

В процессе итеративной разработки, состоящей из чередующихся шагов редактирования, сборки и запуска программы, пользователь редактирует как XML-спецификацию модуля, так и код внутри модуля. Препроцессор синхронизирует XML-спецификацию и код модуля при каждой сборке. Вопрос контроля ошибок при такой синхронизации решен следующим образом. Сначала проверяется, является ли XML-файл правильно построенным. Далее в процессе разбора SAX-парсером проверяются правила вложенности для тегов, описывающих структурные элементы модели программирования. Атрибуты могут быть пропущены и автоматически заменены значениями по умолчанию, если в препроцессоре активирована функция автоматического ввода (auto-complete). В препроцессоре имеется опция преобразования кода XML-спецификации, восстанавливающая все атрибуты и форматирующая файл XML-спецификации с учётом вложенности тегов. Никакого контроля при генерации больше не выполняется. Семантика модели (например, соблюдение протокола канала) проверяется во время исполнения путем генерации соответствующих assert-инструкций в коде модуля. Если пользователь сформировал блоки, не соответствующие структурным элементам модели, они выносятся в отдельное место в файле модуля. Если в XML-спецификацию был добавлен новый элемент, для которого не было пользовательского блока, то генерируется пустой блок с комментарием, указывающим на необходимость его определения. Таким образом, основной контроль выполняется компилятором и системой исполнения (в случае assert-вызовов) целевого языка программирования. В текущей реализации — компилятор языка C++.

Описанная схема работы обеспечивает уверенность в правильном функционировании кода параллельной программы, так как весь критический код является открытым для анализа и отладки, а технически сложные части анализа выполняются надежным штатным компилятором.

Состав библиотеки типовых решений программного комплекса показан в табл. 1. Первые три схемы демонстрируют возможности языка templet при описании логики процессов. Для решения прикладных задач предназначены схемы 4–6. Применение данных схем не требует от пользователя специальных знаний в области параллельного программирования.

Косвенную оценку трудоемкости программирования при использовании программного комплекса Graphplus templet даёт сопоставление размеров рукописного и сгенерированного кода, представленное в табл. 2 и 3.

Размер рукописного кода относится только к алгоритму схемы, а не к численному методу. По таблицам видно, что управляющий алгоритм составляет значительную долю кода. Эффект от автоматизации заключается в его повторном использовании.

При работе над прикладным алгоритмом пользователь системы работает только с кодом, размер которого указан в табл. 2. Из сравнения табл. 2 с табл. 3 видно, что размер последовательного кода, который приходится анализировать при отладке, несколько меньше. Однако следует учитывать, что код из табл. 3 содержит системные вызовы синхронизации. Его ручная разработка требует квалификации в области параллельного программирования.

Часть кода программ пишется на языке разметки XML. Этот код при ра-

Таблица 1

Состав библиотеки программного комплекса Graphplus templet

№ п/п	Кодовое имя схемы	Описание
Схемы с детальным описанием процессов и каналов на языке `templet XML`		
1	Map	Схема «применить ко всем». Фиксированное число независимых подзадач со сборкой результата.
2	TaskBag	Схема «портфель задач». Разновидность схемы «управляющий – рабочие» с пассивным управляющим процессом.
3	Pipeline	Схема типа «конвейер» с двунаправленным потоком данных между ступенями.
Проблемно-ориентированные схемы		
4	TBag	«Портфель задач» для оптимизационных алгоритмов. Используется для исследования алгоритмов балансировки нагрузки имитационным методом.
5	ADI	Схема, реализующая численный метод переменных направлений. Используется для моделирования диссипативных систем.
6	PLine	Схема «конвейер» для моделирования электромагнитных процессов путем численного решения уравнений Максвелла.

Таблица 2

Размеры последовательных версий схем программ, байты

№ п/п	Кодовое имя схемы	Размер кода на `C++`			Размер кода на `XML`	
		в модуле, сгенерированном по `XML`	в библиотеке времени исполнения	всего	без учёта функции автозавершения	с учётом функции автозавершения
1	Map	7994	3050	11044	1820	496
2	TaskBag	12175	3050	15225	2933	845
3	Pipeline	6999	3050	10049	2962	733
4	TBag	2899	1417	3040	213	92
5	ADI	3032	1679	4711	229	106
6	PLine	1652	1100	2752	213	90

Таблица 3

Состав библиотеки программного комплекса Graphplus templet, всего в байтах

№ п/п	Кодовое имя схемы	Размер кода на `C++` для `API Win32`	Размер кода на `C++` для `API POSIX`
1	Map	12823	12779
2	TaskBag	17004	16960
3	Pipeline	11828	11784
4	TBag	4502	4504
5	ADI	7705	8237
6	PLine	3906	4309

боте в интегрированной среде (например, `Visual Studio`) либо генерируется по графическому образу процессов и каналов в специальном редакторе, либо пишется в `XML`-редакторе с поддержкой схем `XML`-документов. В данном случае редактор автоматически генерирует имена тегов, атрибутов и разделители. Размер частей `XML`-файла с учётом возможностей автоматического завершения приведён в последнем столбце табл. 2.

В основе представленной системы автоматизации параллельного программирования лежит система визуального программирования `GraphPlus` [1]. Усовершенствования коснулись модели программирования, в которую была добавлена концепция каналов, используемая в операционной системе `Singularity` и языке программирования `Sing#` [2]. Подвергся пересмотру метод определения семантики `XML`-спецификации. В отличие от работы [3], где семантика модели описана в терминах темпоральной логики, применен метод эталонной реализации [4], упрощающий понимание системы программистом. В язык `XML`-спецификации введена возможность описания типовых решений (паттернов, шаблонов) [5].

В данной системе используется парадигма языково-ориентированного программирования [6], в рамках которой система может рассматриваться как пример доменно-ориентированного языка для параллельных вычислений. Система предназначена для реализации и использования паттернов параллельного программирования [7] в более гибкой форме, чем в соответствующих каркасных библиотеках. В частности, как в работе [8], единая вычислительная модель служит целям композиции типовых решений. Параллелизм в представленной системе вводится не как расширение языка или собственный язык, а в форме библиотеки времени исполнения [9], однако для обеспечения переносимости определяется лишь эталонная реализация. Система занимает промежуточное положение между языком и каркасной библиотекой, построена на основе препроцессора и в этом похожа на систему программирования `Qt` [10]. Однако она не имеет жёсткой привязки к языку программирования `C++`, не выполняет даже лексический разбор языковых конструкций, что значительно упрощает дизайн.

Общеупотребительными являются несколько способов введения семантики параллельного исполнения в язык программирования. Каркасная библиотека, например `TBB` на языке `C++` [11], жёстко связывает решение с языком программирования. Расширение последовательного языка программирования требует написания как минимум лексического анализатора, что может оказаться неприемлемо сложным для практического применения, хотя в некоторых случаях удается построить компактные расширения, например `T++` [12]. Использование распараллеливающих директив в форме комментариев, например `OpenMP` [13] или `DVM` [14], тоже предполагает наличие сложного компилятора, но кроме этого заставляет пользователя вручную приводить структуру кода в соответствие с моделью программирования этих средств. Использование приема генерации исходного кода вместо его анализа снимает данную проблему.

Семантика кода в представленной системе основана на модели процессов. Данный выбор обусловлен широкими описательными возможностями, императивностью, удобством декомпозиции сложных систем на её основе. Несмотря на достоинства функциональной модели, и в том числе возможности описания некоторых паттернов параллелизма в её терминах [15], мы

считаем наш подход более универсальным. Использование модели процессов диффузного типа [16] позволяет выполнить её реализацию базовыми средствами процедурно-ориентированных языков и избегать манипулирования со стеком вызовов.

Эксперименты с созданной системой программирования `Graphplus templet` подтверждают эффективность и возможность практического применения изложенных в работе решений. В программный комплекс `Graphplus templet` входит транслятор модели / препроцессор, модуль с отладочным последовательным кодом системы исполнения, модуль многопоточного исполнения для `API Win32` и `API POSIX`, примеры тестовых приложений и опционный графический редактор файлов `XML`-спецификации. Система реализована на языке `C/C++`, использует `SAX`-парсер `Expat 2.0.1` для разбора файлов `XML`-спецификации. В настоящий момент `Graphplus templet` поддерживает программирование приложений на языке `C++`. Код системы зарегистрирован Федеральным институтом промышленной собственности и открыт для некоммерческого применения на сайте Самарского государственного аэрокосмического университета (СГАУ) по адресу `http://graphplus.ssau.ru/`. Система используется на суперкомпьютере «Сергей Королёв», установленном в СГАУ.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Востокин С. В.* Графическая объектная модель параллельных процессов и её применение в задачах численного моделирования. Самара: СНЦ РАН, 2007. 286 с. [*Vostokin S. V.* Graphical object model of parallel processes and its application in numerical modeling. Samara: SNC RAN, 2007. 286 pp.]
2. *Hunt G. C., Larus J. R.* Singularity: Rethinking the Software Stack // *SIGOPS Oper. Syst. Rev.*. Vol. 41, no. 2. Pp. 37–49.
3. *Востокин С. В.* Спецификация визуальной модели параллельных и распределённых вычислений на основе логики TLA / В сб.: *Параллельные вычисления и задачи управления*: Тр. Четверт. Международн. конф-ции (PACO'2008, 27–29 октября 2008 г., Москва). М.: ИПУ РАН, 2008. С. 1338–1348. [*Vostokin S. V.* Visual model specification of parallel and distributed computing based on TLA logic / In: *Parallel Computations and Control Problems*. Moscow: IPU RAN, 2008. Pp. 1338–1348].
4. *Dalci E., Fong E., Goldfine A.* Requirements for GSC-IS Reference Implementations. National Institute of Standards and Technology, Information Technology Laboratory, 2003.
5. *Gamma E., Helm R., Johnson R., Vlissides J.* Design Patterns: Elements of Reusable Object-Oriented Software. New York: Addison-Wesley, 1995. 416 pp.; русск. пер.: *Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.* Приёмы объектно-ориентированного проектирования. Паттерны проектирования. Спб.: Питер, 2003. 368 с.
6. *Дмитриев С.* Языково-ориентированное программирование: следующая парадигма // *RSDN Magazine*, 2005. № 5. [*Dmitriev S.* Language Oriented Programming: The Next Programming Paradigm // *RSDN Magazine*, 2005. no. 5].
7. *Schmidt D. C., Huston S. D.* C++ Network Programming. Vol. 2: Systematic Reuse with ACE and Frameworks. New York: Addison-Wesley, 2002. 384 pp.; русск. пер.: *Шмидт Д., Хьюстон С.* Программирование сетевых приложений на C++. Т. 2. М.: Бином-Пресс, 2004. 400 с.
8. *Бергизияров П. К.* Программирование на типовых алгоритмических структурах с массивным параллелизмом // *Вычислительные методы и программирование*, 2001. Т. 2, № 2. С. 1–16. [*Bergiziyarov P. K.* Programming model of algorithmic structures with massive parallelism // *Vychislitel'nye metody i programmirovanie*, 2001. Vol. 2, no. 2. Pp. 1–16].
9. *Stroustrup B.* The C++ Programming Language / 3rd edition. New York: Addison-Wesley, 1997. 1040 pp.; русск. пер.: *Страуструп Б.* Язык программирования C++ / 3-е изд-ние. Спб.: Невский диалект; М.: Бином, 1999. 991 с.

10. *Земсков Ю. В.* Qt 4 на примерах. СПб.: БХВ-Петербург, 2008. 608 с. [*Zemskov Yu. V.* Qt 4 with examples. St. Petersburg: BHV-Peterburg, 2008. 608 pp.]
11. *Reinders J.* Intel Threading Building Blocks: Outfitting C++ for Multi-core Processor Parallelism. Sebastopol, CA: O'Reilly Media, 2007. 334 pp.
12. *Абрамов С. М., Адамович А. И., Инюхин А. В., Московский А. А., Роганов В. А., Шевчук Ю. В.* Т-система с открытой архитектурой / В сб.: *Суперкомпьютерные системы и их применение*: Тр. международн. научн. конф-ции (SSA'2004, 26–28 октября 2004 г., Минск). Минск: ОИПИ НАН Беларуси, 2004. С. 18–22. [*Abramov S. M., Adamovich A. I., Inyuhin A. V., Moscovskiy A. A., Roganov V. A., Shevchuk Yu. V.* T-System with open architecture / In: *Supercomputer Systems and Their Applications*. Minsk: OIPI NAN Belarusi, 2004. Pp. 18–22].
13. *Quinn M. J.* Parallel Programming in C with MPI and OpenMP. New York: McGraw-Hill, 2004. 544 pp.
14. *Крюков В. А., Удовиченко Р. В.* Отладка DVM-программ // *Программирование*, 2001. № 3. С. 19–29. [*Kryukov V. A., Udovichenko R. B.* Debugging DVM Programs // *Programmirovanie*, 2001. no. 3. Pp. 19–29].
15. *Московский А. А., Первин А. Ю., Сергеева Е. О.* Первый опыт реализации шаблона параллельного программирования на основе Т-подхода / В сб.: *Программные системы: теория и приложения*: Тр. Междунар. конф-ции. Т. 1. М.: Наука, Физматлит, 2006. С. 245–255. [*Moskovskiy A. A., Pervin A. Yu., Sergeeva E. O.* First experience of implementing parallel programming skeletons using T-approach / In: *Software Systems: Theory and Applications*. Vol. 1. Moscow: Nauka, Fizmatlit, 2006. Pp. 245–255].
16. *Dijkstra E. W., Scholten C. S.* Termination detection for diffusing computations // *Information Processing Letters*, 1980. Vol. 11, no. 1. Pp. 1–4.

Поступила в редакцию 21/IX/2011;
в окончательном варианте — 21/X/2011.

MSC: 68N19

PARALLEL PROGRAMMING SOFTWARE PACKAGE GRAPHPLUS TEMPLET

S. V. Vostokin, A. R. Khayrutdinov, V. G. Litvinov

S. P. Korolyov Samara State Aerospace University
(National Research University),
34, Moskovskoe sh., Samara, 443086, Russia.

E-mails: easts@mail.ru, khairutdinov@yandex.ru, doom-black@mail.ru

The parallel programming automation software package named Graphplus templet is considered. Its design purposes, architecture, efficiency of applying for numerical modeling problems solutions, features against similar approaches are discussed. Composition of the software package is described in details. Programming complexity of the numerical simulation applications coded with help of the software is considered.

Key words: *software, parallel programming, computational model, program scheme, domain specific language, automation of programming, numerical simulation.*

Original article submitted 21/IX/2011;
revision submitted 21/X/2011.

Sergey V. Vostokin (Dr. Sci. (Tech.)), Professor, Dept. of Information Systems & Technologies.
Andrey R. Khayrutdinov, Postgraduate Student, Dept. of Information Systems & Technologies.
Vladimir G. Litvinov, Postgraduate Student, Dept. of Information Systems & Technologies.