УДК 004.5; 004.4:004.7

М.К. Буза, О.М. Кондратьева

ПРОГРАММНАЯ СРЕДА ПРОЕКТИРОВАНИЯ ПАРАЛЛЕЛЬНЫХ ПРИЛОЖЕНИЙ С ОБЩЕЙ ПАМЯТЬЮ

Исследуются средства поддержки разработки параллельных программ на языке C/C++. Предлагаются методика и программная среда, которые могут быть использованы для автоматизации процесса проектирования параллельных приложений.

Введение

Разработка эффективных программ для параллельных компьютеров является сложной задачей. В настоящее время ведется значительная работа по упрощению процесса проектирования и реализации параллельных приложений. Существующие средства разработки параллельных программ весьма разнообразны [1]. Представляет интерес разработка инструментальных систем, облегчающих создание и проектирование параллельных программ, а именно объектно-ориентированных библиотек и фреймворков.

При выборе инструментария для разработки параллельной программы хотелось бы руководствоваться возможностью не только создания эффективной программы как главной цели распараллеливания, но и сохранения ее эффективности при переносе с одного целевого компьютера на другой. Однако перечисленные критерии имеют во многом противоречивый характер [2].

В настоящей работе представлена программная среда, которая может быть использована для проектирования параллельных программ с общей памятью (многопоточных программ) на языке C/C++.

1. Средства поддержки проектирования параллельных приложений

Параллельная программа может быть написана на специальном языке или расширении существующего языка, а также на стандартном языке с использованием библиотеки. Одним из преимуществ последнего варианта является то, что среда разработки привычна для программиста. В настоящее время существует ряд кроссплатформенных многопоточных библиотек для языка программирования C++: Boost Thread, C++ Standard Library Thread, Parallel Patterns Library, Threading Building Blocks [3]. Перечисленные программные продукты представляют собой объектно-ориентированные библиотеки и позволяют использовать преимущества объектно-ориентированного и обобщенного программирования языка C++ при разработке параллельных приложений.

Известный пример обобщенного программирования – стандартная библиотека шаблонов C++. Она предоставляет пользователю высококачественные алгоритмы и структуры данных. В стандарте языка C++, принятом в 2011 г. (C++11), появилась развитая поддержка многопоточности [4]. Возможно, многопоточные C++11-программы и уступают по эффективности низкоуровневым средствам, однако явно превосходят их по скорости создания и мобильности.

Одним из механизмов, которые могут увеличить качество и скорость разработки параллельной программы, служат паттерны параллельного программирования. Паттерн в разработке программного обеспечения – это повторимая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста [5]. Обычно паттерн является не законченным образцом, который может быть прямо преобразован в код, а лишь примером решения задачи, который можно использовать в различных ситуациях. И если в известной книге [6], посвященной паттернам параллельного программирования, представлены примеры решения задач, то в настоящее время речь идет уже о «алгоритмических скелетах», которые являются высокоуровневыми моделями программирования для

параллельных и распределенных вычислительных систем [7]. В качестве примеров можно назвать такие известные паттерны, как master-slave, pipe, for, while, if, map, divide-conquer.

Объектно-ориентированные библиотеки часто включают реализации различных паттернов. Обычно говорят, что библиотека содержит алгоритмы. Например, Parallel Patterns Library от Microsoft содержит for, for_each, map и другие алгоритмы, а Threading Building Blocks от Intel – for, reduce, do, scan, while, pipeline, sort.

Таким образом, можно констатировать, что параллельное программирование уже накопило общие приемы и превратилось из искусства в технологию. Оно может стать простым и безопасным с помощью предварительно запрограммированных паттернов. Для проектирования параллельной программы потребуется взять готовый и проверенный временем шаблонный код и адаптировать к нему свой алгоритм.

Настоящее исследование направлено на то, чтобы процесс проектирования параллельных программ стал простым и понятным для разработчика. Цель работы – предложить доступную методику проектирования параллельных программ и создать программный инструмент для ее поддержки. В отличие от существующих аналогов здесь используются более общие модели многопоточных программ и существенно упрощается процесс проектирования. При этом предложенная методика ориентирована не только на создателей параллельного программного обеспечения, но и на студентов при обучении их проектированию параллельных программ.

2. Состав среды проектирования

Рассмотрим компоненты разработанной среды проектирования параллельных приложений (рис. 1).

Рис. 1. Компоненты среды проектирования

Общие паттерны параллельных программ. Авторы представляют паттерн как каркас программы, реализующей некоторую модель создания и функционирования потоков. К рассматриваемым моделям относятся: модель делегирования, модель с равноправными узлами, конвейер, изготовитель-потребитель [8]. Каждая модель характеризуется собственной декомпозицией работ, которая определяет, кто отвечает за создание потоков, при каких условиях они создаются, как потоки взаимодействуют. При разработке параллельной программы предлагается выбрать подходящий паттерн. Решение задач, связанных с коммуникацией, синхронизацией и балансировкой нагрузки, уже реализовано в выбранном паттерне.

Специализированные паттерны параллельных программ. Каждый специализированный паттерн является конкретизацией общего паттерна для решения определенного множества задач. Например, имеющиеся паттерны для решения матричных задач реализуют различные общие паттерны, а также позволяют определить способ распределения данных (полосы или бло-

ки, непрерывный или циклический), размерность матрицы (одномерная, двухмерная, трехмерная), вариант заполнения матрицы (генератор случайных чисел, из файла).

База данных параллельных приложений содержит готовые решения разнообразных задач параллельного программирования: матричных вычислений, решений систем линейных уравнений, сортировки, обработки графов, уравнений в частных производных, многоэкстремальной оптимизации.

Потокобезопасные контейнеры. Для многопоточных программ существует проблема разделения данных между потоками. Операция «читать – модифицировать – писать» должна быть атомарной в качестве общего решения этой проблемы. Предлагаемая среда проектирования содержит потокобезопасные контейнеры: вектор, очередь и стек. Названные контейнеры используются в паттернах параллельных программ, например в моделях, которые работают с очередью заданий. Пользователи могут напрямую включать их в свои программы.

Мастер создания остова (каркаса) приложения представляет собой оконный интерфейс пользователя, посредством которого он выбирает и настраивает паттерн параллельной программы.

Скрипты для проведения экспериментов автоматизируют запуск приложений, сохраняют замеры времени и рассчитывают показатели эффективности.

3. Методика проектирования параллельных приложений

В основу предлагаемой методики проектирования параллельных программ положена PCAM (Partitioning, Communication, Agglomeration, Mapping) [9], которая включает следующие этапы:

– разделение вычислений на независимые части;

– выделение информационных зависимостей;

– масштабирование набора подзадач;

– распределение подзадач между вычислительными элементами (в многопоточных программах обычно выполняется операционной системой).

Предлагаемая среда проектирования поддерживает все этапы разработки параллельной программы. Поскольку выделение информационных зависимостей и разделение вычислений на независимые части существенно зависят от рассматриваемой задачи, соответствующие этапы частично реализуются при создании паттернов, а частично − при адаптации (конкретизации) паттерна к решаемой задаче. Рассмотрим процесс разработки с использованием среды (рис. 2).

Рис. 2. Взаимодействие пользователя со средой проектирования

Первым этапом разработки параллельного приложения является *создание остова приложения*. Для этого используется мастер, который включает следующие шаги:

Шаг 1. Определение типа приложения:

C++11 (технология высокого уровня) – паттерны, написанные на стандарте C++11 языка C++, в который включена поддержка параллелизма [4];

WinAPI (технология низкого уровня) – встроенные потоки операционной системы Microsoft Windows [10];

PThread (технология низкого уровня) – потоки POSIX Thread Library [10];

Author Thread (технология высокого уровня) – собственная объектно-ориентированная библиотека, которая инкапсулирует интерфейс низкого уровня.

Шаг 2. Выбор общего паттерна параллельной программы. Предлагаются общие паттерны, реализующие следующие модели создания и функционирования потоков:

– модель делегирования в двух вариантах: управляющий поток создает новый поток для каждого нового запроса и управляющий поток создает пул потоков, которые обрабатывают все запросы;

– модель с равноправными узлами в двух вариантах: общий поток входных данных и собственный входной поток у каждого потока;

– конвейер;

– модель «изготовитель – потребитель».

Шаг 3. Пользователь может для решения своей задачи подобрать специализированный паттерн, который реализует выбранную на предыдущем шаге модель для определенного класса задач. Предлагаются варианты:

– матричные задачи;

– анализ данных из файлов.

Шаг 4. Выбор конкретного приложения из базы данных параллельных приложений, если оно подходит для решения задачи пользователя. Это приложение удовлетворяет параметрам выбора, установленным на предыдущих шагах.

Шаг 5. Установка дополнительных параметров. Предлагается включить в приложение замеры времени работы, определить количество потоков, задать способ подготовки входных данных для тестирования параллельной программы: ручной или автоматический (генерация файлов).

После завершения работы мастера создания остова приложения получаем каркас или уже законченное многопоточное приложение. Этапы проектирования (декомпозиция вычислений, их координация и масштабирование) нашли свое отражение в тех паттернах, которые выбирал пользователь. Каркасы используют в качестве параметра количество потоков, обеспечивая зависимость правил масштабирования от количества вычислительных элементов. Также есть возможность использовать как высокоуровневые, так и низкоуровневые средства.

Второй этап проектирования параллельного приложения – *ручная конкретизация остова приложения*. Полученный с помощью мастера каркас приложения вручную дорабатывается пользователем. Трудоемкость этого этапа существенно зависит от того выбора, который был сделан на первом этапе. Если пользователь на четвертом шаге подобрал подходящую программу из базы данных параллельных приложений, то ручная конкретизация не требуется. Если подходящая программа не была найдена, но на третьем шаге был выбран специализированный паттерн, то требуется определить функцию, которую будет выполнять каждый поток. Если пользователь выбрал только общий паттерн, то требуется максимально трудоемкая доработка: определить функцию потока, конкретизировать типы данных. Ясно, что ручная конкретизация сильно зависит от типа приложения.

Третий этап проектирования – *проведение вычислительных экспериментов* с целью определения эффективности разработанной программы. Для этого выполняют последовательную и параллельную версии программы, измеряют время выполнения и определяют показатели эффективности. При отсутствии последовательной версии программы можно воспользоваться параллельной версией, установив количество потоков равным единице.

Одной из главных характеристик параллельной программы является ускорение:

$$S_p(n) = T_1(n) / T_p(n),$$

где $T_1(n)$ − время выполнения последовательной версии программы; $T_p(n)$ − время выполнения параллельной программы на p-процессорной вычислительной системе.

Еще одна важная характеристика параллельной программы – эффективность:

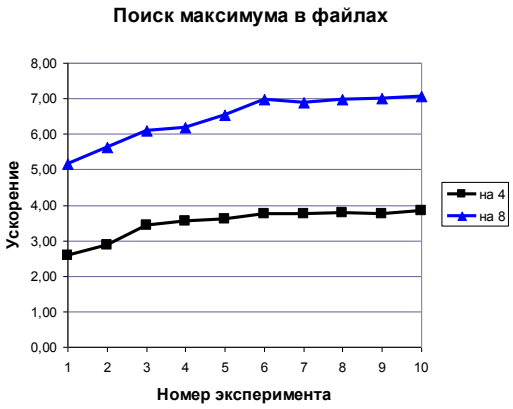
$$E_p(n) = T_1(n) / (pT_p(n)) = S_p(n) / p.$$

Для помощи пользователю написаны скрипты, которые автоматизируют запуск многопоточной программы с различным количеством потоков и для различных входных данных, сохранение замеров времени в файле и вычисление показателей эффективности.

Результаты экспериментов можно просмотреть как в текстовом, так и графическом виде. В качестве примера на рис. 3 показан фрагмент данных, которые были получены при решении задачи поиска максимума в файлах. Для каждого теста (входного набора данных) указываются результаты выполнения в виде списка значений «время_выполнения (количество_потоков)».

```
Тест 1:     3,1(1),    1,2(4),    0,6(8)
Тест 2:    12,2(1),    5,3(4),    2,7(8)
Тест 3:    31,1(1),    9,1(4),    5,1(8)
Тест 4:   148,7(1),   42,0(4),   24,0(8)
Тест 5:   298,0(1),   82,5(4),   45,6(8)
Тест 6:   600,1(1),  160,0(4),   86,0(8)
Тест 7:  1197,0(1),  318,0(4),  174,0(8)
Тест 8:  2420,0(1),  639,0(4),  347,0(8)
Тест 9:  4837,0(1), 1290,0(4),  692,0(8)
Тест 1:  9760,0(1), 2551,0(4), 1380,0(8)
```

а)



б)

Рис. 3. Результаты эксперимента: *а)* фрагмент текстового файла;
б) графики ускорений для четырех и восьми потоков

Четвертый этап проектирования – *анализ результатов*. Процесс проектирования имеет итеративную природу. Если в результате анализа эффективности параллельной программы разработчик будет неудовлетворен результатом, то он может перепроектировать свою программу, вернувшись на предыдущие этапы. Существует важное преимущество предлагаемого подхода – легкость перехода от одной модели функционирования потоков к другой, эти изменения выполняются автоматически.

4. Апробация и области применения среды проектирования

Исследование среды проектирования проводилось посредством использования предлагаемого инструмента для проектирования некоторых распространенных задач параллельного программирования. В таблице представлены основные результаты экспериментов – среднее время, затраченное на разработку параллельного приложения для решения типовой задачи с использованием разных средств проектирования. Разработка приложений осуществлялась студентами третьего курса факультета прикладной математики и информатики, специализация «Прикладная информатика».

На новом уровне был продолжен эксперимент по распараллеливанию сложной прикладной программы − пакета моделирования методом молекулярной динамики XMD (Molecular

Dynamics for Metals and Ceramics). В работе [11] к названному пакету были добавлены многопоточные версии функций вычисления сил в случае полупроводниковых ковалентных материалов. С помощью среды проектирования легко были получены несколько вариантов распараллеливания: модели с равноправными узлами и модели делегирования. Эти модели обычно дают показатели эффективности, близкие по значению. Однако существует вероятность того, что на разных вычислительных системах могут быть получены существенно разные показатели эффективности.

Среднее время разработки параллельного приложения, ч

Задача	Используемая модель	Встроенные потоки	Общий паттерн	Специализированный паттерн
Вычисление значения определенного интеграла	Модель с равноправными узлами	2,5	1,25	0,7
Поиск максимума в матрице	Модель делегирования	2,5	1,25	0,7
Умножение матриц	Модель с равноправными узлами	2,5	1,25	0,7
Внутренняя сортировка	Модель с равноправными узлами	3	2	1
Внешняя сортировка	Модель делегирования	4,5	2,5	1,5
Внешняя сортировка	Конвейер	5	3	1,7

Полученные результаты иллюстрируют снижение трудоемкости разработки, а также подтверждают, что абстрагирование и использование паттернов позволяют ускорить и улучшить процесс проектирования параллельного кода.

Обобщенные средства разработки, как правило, порождают не самый эффективный код. Этот недостаток может быть ослаблен на этапе ручной конкретизации остова приложения.

Универсальные средства программирования обычно ориентированы на ограниченный класс приложений. В том случае если приложение не вписывается в модель программирования, принятую в системе разработки, программист не может ею воспользоваться. Эта проблема решается программистом за счет введения собственных расширений.

Рассмотрим возможные способы применения разработанной системы. Наиболее органичным способом является использование ее при обучении технологиям параллельного программирования, в первую очередь при освоении тем «Распараллеливание последовательных программ» и «Технологии параллельного программирования с использованием шаблонов» [12].

Применение разработанной среды проектирования повышает культуру программирования. То, что может написать начинающий программист, существенно отличается от параллельных приложений, созданным опытным разработчиком. Примером может служить «простая реализация параллельной версии аккумулятора» [4].

Заключение

В настоящей работе представлена авторская инструментальная система, предназначенная для проектирования параллельных программ с общей памятью на языке C/C++. При проектировании программ используются готовые модели, что делает процесс написания программ более простым. Использованы преимущества объектно-ориентированного и обобщенного программирования.

Разработанная программная среда автоматизирует процесс проектирования и исследования эффективности параллельной программы, избавляет программиста от рутинной работы и позволяет писать многопоточный код проще и с меньшим количеством ошибок. Поддерживается возможность получать оценки применения различных моделей при решении конкретных задач.

Полученные при апробации среды результаты иллюстрируют снижение трудоемкости разработки, а также подтверждают, что избранная методика позволяет ускорять и улучшать процесс проектирования параллельного кода, при этом несущественно влияя на эффективность исполнения созданных программ.

Разработанная среда может быть использована при обучении технологиям параллельного программирования, позволяя выбирать и оценивать тот или иной подход для создания собственных эффективных параллельных приложений.

Проект выполнен при частичной поддержке ГПНИ «Информатика и космос».

Список литературы

1. Средства разработки параллельных программ [Электронный ресурс]. – Режим доступа : https://parallel.ru/tech/tech_dev. – Дата доступа : 15.06.2016.

2. Воеводин, В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – СПб. : БХВ – Петербург, 2002. – 608 с.

3. List of C++ multi-threading libraries [Electronic resource]. – Mode of access : https://en.wikipedia.org/wiki/List_of_C%2B%2B_multi-threading_libraries. – Date of access : 15.06.2016.

4. Уильямс, Э. Параллельное программирование на C++ в действии. Практика разработки многопоточных программ / Э. Уильямс. – М. : ДМК Пресс, 2012. – 672 с.

5. Шаблон проектирования [Электронный ресурс]. – Режим доступа : https://ru.wikipedia.org/wiki/Шаблон_проектирования. – Дата доступа : 15.06.2016.

6. Mattson, T.G. Patterns for Parallel Programming / T.G. Mattson, B.A. Sanders, B.L. Massingill. – Addison Wesley, 2002. – 300 p.

7. Algorithmic skeleton [Electronic resource]. – Mode of access : https://en.wikipedia.org/wiki/Algorithmic_skeleton. – Date of access : 15.06.2016.

8. Хьюз, К. Параллельное и распределенное программирование с использованием C++ / К. Хьюз, Т. Хьюз. – М. : Вильямс, 2004. – 672 с.

9. Foster, I. Designing and Building Parallel Programs: Concepts and Tools for Parallel / I. Foster. – Addison Wesley, 1995. – 430 p.

10. Буза, М.К. Проектирование программ для многоядерных процессоров : учеб.-метод. пособие для студентов факультета прикладной математики и информатики / М.К. Буза, О.М. Кондратьева. – Минск : БГУ, 2013. – 48 с.

11. Буза, М.К. Параллельная реализация метода молекулярной динамики на многоядерном процессоре / М.К. Буза, О.М. Кондратьева // Информатика. – 2011. – № 1(29). – С. 44–51.

12. Технологии параллельного программирования II (2012871) [Электронный ресурс]. – Режим доступа : http://open.ifmo.ru/wiki/Технологии_параллельного_программирования_II_(2012871). – Дата доступа : 15.06.2016.

Поступила 08.08.2016

Белорусский государственный университет,
Минск, пр. Независимости, 4
e-mail: bouza@bsu.by,
kondratjeva@bsu.by

M.K. Bouza, O.M. Kondratjeva

SOFTWARE FOR DESIGNING PARALLEL APPLICATIONS

The object of research is the tools to support the development of parallel programs in C/C ++. The methods and software which automates the process of designing parallel applications are proposed.