ISBN 5-94052-127-4 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ. ПЕРЕСЛАВЛЬ-ЗАЛЕССКИЙ, 2006

УДК 004.272:004.4

А. А. Московский, А. Ю. Первин, Е. О. Сергеева

Первый опыт реализации шаблона параллельного программирования на основе T-подхода

Аннотация. Целью этой работы является разработка и реализация шаблонов параллельного программирования на основе библиотеки T-Sim. Шаблоны параллельного программирования — подход, родственный широко используемому в настоящее время подходу «обобщенного программирования». В данном подходе, создаётся абстрактный «скелет» или «шаблон» параллельной программы, описывающий способ распараллеливания вычислений. В то же время, пользователю предоставляется возможность реализовать часть приложения, специфичную для решаемой задачи. В статье приводится краткий обзор существующих разработок в области шаблонов. Проводится анализ наиболее часто реализуемых шаблонов. В проекте ставится и решается задача обеспечения наиболее удобного пользовательского интерфейса, совместимого с библиотекой стандартных шаблонов (STL). В качестве примера описана реализация шаблона «map» (отображение), самый простой и наиболее часто реализуемый шаблон. В качестве более низкоуровневого средства параллельного программирования используется библиотека шаблонных классов C++ T-Sim, реализующая подход T-системы к распараллеливанию вычислений.

Ключевые слова и фразы: Шаблоны параллельного программирования, Map, Reduce, скелет параллельного программирования, T-Sim, распараллеливание вычислений.

1. Базовые сведения о шаблонах параллельного программирования

Эффективные программы для высокопроизводительных параллельных компьютеров обычно обладают высокой степенью сложности реализации. Это связано с тем, что прикладному программисту, разрабатывающему параллельную программу, приходится решать задачи далекие от той предметной области, в которой проводится исследование. Так, например, одни библиотеки, поддерживающие стандарт Message Passing Interface (MPI) содержат десятки

Работа выполнена при поддержке программы фундаментальных исследований Президиума РАН «Разработка фундаментальных основ создания научной распределенной информационно-вычислительной среды на основе технологий GRID», а также гранта РФФИ № 05–07–08005–офи_а.

функций, доступных пользователю. Параллельные программы, разработанные с использованием MPI, трудны для понимания и требуют значительного времени для отладки и тестирования. Другой подход к разработке параллельных приложений связан с использованием автоматических средств для распараллеливания. Пользователь вносит в свою программу соответствующие изменения (например, помечает некоторые переменные специальным префиксом), в результате чего приложение начинает эффективно работать на параллельных ЭВМ. Однако, практика показывает, что освоение автоматических средств такого рода также может быть сопряжено с большими затратами времени.

Шаблоны параллельного программирования (ШПП) — это коллекция полезных техник параллельного программирования, которая может быть организована в виде иерархии классов, функций высших порядков или C++-шаблонов. Как правило, ШПП опираются на некоторую библиотеку, позволяющую организовать параллельный счет. При этом вся сложность использования библиотеки скрывается за простым и гибким интерфейсом шаблона. Мы используем библиотеки TSim[1] для разработки шаблонов параллельного программирования. В общем виде ШПП — это заготовка, пригодная для организации логики параллельной программы. Для того, чтобы воспользоваться ШПП, программисту необходимо снабдить абстрактные конструкции данными о решаемой задачи. Например, для того, чтобы воспользоваться параллельной версией алгоритма сортировки, программисту может понадобится указать тип сортируемых элементов.

Важной чертой ШПП является возможность многократного использования. Библиотека шаблонов, как правило, подразуемвает наличие достаточно большого набора типовых алгоритмов, типов данных, абстрактный конструкций, позволяющих реализовывать определенный круг задач.

2. Исследование существующих разработок в области ШПП

2.1. COOPPS. Разработанный в университете провинции Канады — Альберты, пакет COOPPS [2] призван решить две часто встречающиеся проблемы средств параллельного программирования:

[1] T-Sim — одна из последних разработок Института программных систем РАН в области параллельного программирования [1], [6], [7].

- Недостаточно высокая производительность.

  Обобщенные средства разработки, как правило, порождают далеко не самый эффективный код. В COOPPS этот недостаток позволяет решить последовательный тюнинг кода (incremental code tuning).
- Недостаточная универсальность.

  Такие средства программирования обычно ориентированы на небольшой класс приложений. В том случае если приложение не вписывается в модель программирования, принятой в системе разработки, программист не сможет ею воспользоваться. В COOPPS эта проблема решается введением возможности для определение программистом собственных расширений.

Таким образом COOPPS имеет следующие ключевые особенности:

- параметризируемые шаблоны (parameterized frameworks);
- поддержка нескольких уровней абстракции;
- инструмент для создания расширений.

Для решения задачи программисту предлагается выбрать удовлетворяющий его задаче шаблон, уточнить его некоторыми параметрами и затем написать последовательный код, реализующий логику приложения. Это единственная точка соприкосновения прикладной программы со средой COOPPS. Всю работу по коммуникации, синхронизации, и балансировке нагрузки COOPPS берет на себя, реализуя их в соотвествии с выбраным шаблоном.

Шаблоны используются в COOPPS до генерации «каркасного» кода. Так, например, в шаблоне Wavefront («фронт волны») необходимо уточнить множество зависимостей (dependency set) и конфигурацию данных (data shape). Гибкость настройки шаблона позволяет покрыть большое множество прикладных задач.

Пакет COOPPS имеет иерархическую структуру, призванную, работая на верхнем уровне, оградить пользователя от возможности написать некорректную параллельную программу, и, работая на нижнем уровне, довести программу до приемлемых показателей производительности (тюнинг).

Структура пакета состоит из трех уровней:

- Уровень шаблонов.

  На этом уровне программисту доступна только библиотека шаблонов и параметры, передаваемые этим шаблонам. Пакет COOPPS генерирует объектно-ориентированный каркас приложений, в соответствии с выбранным программистом шаблоном. Каркас приложения состоит из абстрактных классов, реализующих корректную работу выбранного алгоритма и конкретных подклассов, которые используются для описания логики приложения.
- Промежуточный уровень.

  Этот уровень предоставляет объектно-ориентированный язык программирования с конструкциями явного параллелизма. Пользователь имеет возможность изменять сгенерированную структуру, создавать новый программный код или отлаживать ранее созданный.
- Уровень кода.

  На этом уровне программный код с предыдущего уровня транфсформируется в объектно-ориентировнный язык (такой как Java или C++). На этом уровне пользователю доступны все библиотеки необходимые для реализации верхних уровней.

2.2. DPnDP. Проект DPnDP идеологически напоминает среду COOPPS: расширяемый набор параметризируемых шаблонов, скрывающих низкоуровневые вопросы синхронизации/коммуникации.

Пакет DPnDP поддерживает следующие шаблоны проектирования: task-farm, pipeline, fan-out structure, divide and conquer, process replication.

Программа, разработанная с использованием пакета DPnDP, может быть представлена в виде графа из модулей и используемых шаблонов. Модули могут взаимодействовать между собой и шаблонами посредством унифицированных интерфейсов. Шаблоны в свою очередь могут быть вложенным и содержать другие модули или шаблоны.

Пакет DPnDP состоит из нескольких компонент:

- пользовательский интерфейс;
- библиотека шаблонов (DPL);
- модуль генерации каркасного кода (CSG);
- другие библиотеки поддержки.

Пользовательский интерфейс служит для выбора и уточнения подходящего шаблона.

DPL содержит ШПП, написанные в виде C++ классов. Эти шаблоны описывают структуру, поведение и архитектурную информацию. Все шаблоны параметризованы. Так, например, в библиотеке представлен шаблон N-stage pipeline (ступенчатый конвейер), описывающий соответственно конвейеры 2-й, 3-й и т. д. ступени.

Все шаблоны имеют унифицированную форму объявления и определения. Каждый шаблон наследуется от базового, содержащей некоторый общий для всех шаблонов интерфейс.

Модуль CSG переводит шаблон из промежуточного представления DPL в удобно организованную структуру каталогов с инструкциями для сборки результирующего приложения. Отделение модуля CSG от пакета DPL предоставляет возможность создавать приложения из шаблонов на разных языках, для разных интерфейсов передачи данных и операционных систем без изменения пакета DPL.

2.3. Язык SKIL. Язык SKIL [3] — это процедурный язык, основанный на C, расширенный некоторой дополнительной функциональностью, что позволяет эффективно использовать его для разработки параллельных приложений. Основная идея языка состоит, с одной стороны, в поддержке полиморфизма, определении и использовании распределенных структур данных и функций высших порядков, а с другой стороны, в упрощении базового языка, посредством устранения некоторых менее важных технических деталей.

Расширения языка:

- Переменные типа (type variables) для поддержки полиморфизма.

  Переменные типа, описываются с помощью C++-шаблонов и могут быть инстанцированы практически любым типом. Исключение составляет тип void, а также типы функций. Переменные типа могут быть использованы при объявлении сложных структур данных распределенных данных.
- Модификатор типа «pardata».

  Это модификатор предназначен для создания распределенных типов данных. Модификатор тип pardata может иметь переменную типа в качестве аргумента.

- Поддержка функций высших порядков.

  Язык SKIL поддерживает функции высших порядков, аргументами которых или результатом работы может быть другая функция. Однако, на функциональные выражения существует ряд ограничений. Так, например, оно не может быть аргументом следующих операторов: cast, sizeof, assignment (=), conditional (?:), sequence (,);
- Преобразование операторов в функциях.

  В языке SKIL введена аналогичная Haskell поддержка инфиксных операторов, указываемых при вызовах функций.

2.4. DatTel. DatTeL — это библиотека, основанная на параллелизме по данным, которая позволяет создавать эффективные программы для разных параллельных архитектур, не отступая от парадигм классического C++ программирования. Для переносимости и простоты в использовании DatTel имеет трехъярусную архитектуру.

- Первый ярус.

  Аналогично STL, библиотека DatTel предоставляет программисту широкий спектр структур и алгоритмов, основанных на параллелизме по данным. В DatTeL реализованы контейнеры и итераторы, принципы работы с которыми аналогичны принципам STL. Другой важный компонент этой библиотеки — это непосредственно шаблоны параллельного программирования. Это функции высшего порядка, широко известные в функциональном программировании — zip, reduce, scan и другие. Параллелизм спрятан в реализации этих алгоритмов и остается прозрачным для пользователя.
- Второй ярус (слой абстракции).

  На втором ярусе реализованы примитивы, на которых базируются компоненты из первого яруса. Этот слой позволяет изолировать DatTeL код от различных пересылок между узлами и архитектуры вычислительной системы, на которой работает библиотека, и позволяет коду оставаться общим и абстрактным.
- Третий ярус.

  В конце концов, DatTeL основан на низкоуровневых библиотеках, таких, как PThreads или MPI, которые обеспечиваю функциональность DatTeL. И третий слой обеспечивает единый интерфейс для работы с ними.

2.5. Map-Reduce. Модель Map-Reduce [4] — это теоретическая модель программирования и связанная с ней реализация для обработки большого количества данных. Она используется в одной из лидирующих технологических компаний — Google — для распараллеливания вычислений на кластерах рекордного размера — так, например, установки Google состоят из более чем десяти тысяч узлов.

Программист использует функцию «map» (отображение), чтобы обработать множество пар ключ-значение, и функцию «reduce» (упростить), которая каким-либо образом (при помощи заданной операции) объединяет все значения со сходными ключами. Как оказалось, удивительно большое количество реальных задач естественно выражается в терминах этой модели. Программы, написанные в подобном функциональном стиле, автоматически распараллеливаются и запускаются на кластере. Система сама обеспечивает распределение входных данных, планирует распределение задач по машинам, отслеживает системные сбои и управляет необходимыми пересылками данных между узлами. Реализация модели Map-Reduce обеспечивает высокий уровень масштабируемости.

3. Требования к разрабатываемому шаблону

Несмотря на то, что существует большое количество шаблонов, многие часто встречающиеся алгоритмы ещё не нашли воплощения в виде высокоуровневых библиотек. Конечной целью нашей работы является создание шаблона генетического алгоритма, предоставив пользователям интерфейс в виде шаблонных функций C++. Отличительной чертой нашей работы является использование динамического распараллеливания. Это позволяет надеятся на возможность эффективного использования, когда остальные библиотеки окажутся неэффективны. Например, в гетерогенных кластерах или вычислительных сетях (грид-сетях), использующих простаивающие компьютеры. На основании анализа изложенных выше материалов легко сформулировать предъявляемые к шаблонам параллельного программирования требования:

- Возможность многократного использования.

  Предполагается реализовывать шаблоны в виде параметризуемых функций или классов C++. Кроме того, шаблоны должны реализовывать часто используемые основные операции, что позволит программисту без затруднений применять один и тот же шаблон для решения различных классов задач.
- Удобный (интуитивно понятный) пользовательский интерфейс.

  Желательно иметь интерфейс, сходный с интерфейсом стандартной библиотеки шаблонов (STL).
- Автоматическое распараллеливание.

  Определение тяжести гранулы параллелизма не должно быть заботой пользователя.

4. Реализуемый шаблон

Одним из наиболее удобных шаблонов среди рассмотренных является Map. По принципам работы он похож на шаблон stl::transform, который применяет функцию к диапазону элементов и сохраняет результаты. Он, очевидно, применим в достаточно широком классе задач, и интуитивно понятен опытному программисту, умеющему работать с STL.

В реализованной в данной работе версии этого шаблона пользовательский вызов Map имеет следующий вид:

Map<Type, MyOp> (IteratorBegin, IteratorEnd);

где *Type* — пользовательский тип, *IteratorBegin* и *IteratorEnd* — итераторы произвольного доступа, позволяющие получить доступ к содержимому конейнеров. Они задают диапазон, в котором будет применяться операция. *MyOp* — определяемый пользователем класс, который обязательно должен иметь открытый метод *op*, задающий унарную операцию. Именно она и будет применяться к элементам в заданном диапазоне.

Если эта операция вычислительно сложная, то при вызове шаблона можно добиться ускорения вычислений, за счет использования параллелизма. Проведено предварительное тестирование, которое показало, что если операция по сложности эквивалентна 10^6 операций по извлечению квадратного корня из чисел с плавающей точкой, то ускорение приближается к линейному. Тестирование проводилось на кластере «Первенец-М» семейства «СКИФ» [5] под операционной системой Red Hat.

5. Псевдокод, демонстрирующий применение шаблонной функции Map

Как уже было сказано выше, функция Map может быть применима в достаточно широком классе задач. Рассмотрим пример применения этой функции в стандартном генетическом алгоритме. В настоящее время генетические алгоритмы широко применяются при решении различных задач глобальной, дискретной оптимизации в инженерии, научных исследованиях.

```
НАЧАЛО /* генетический алгоритм */
    <Создать начальную популяцию>
    <Оценить приспособленность каждой особи>
    ПОКА <НЕ получена искомая особь> ВЫПОЛНЯТЬ
    НАЧАЛО /* создать популяцию нового поколения */
        ПОВТОРИТЬ (размер_популяции) РАЗ
        НАЧАЛО /* цикл воспроизводства */
            <Выбрать две особи с высокой приспособленностью
            из предыдущего поколения для скрещивания>
            <Скрестить выбранные особи и получить новую особь>
            <При скрещивании могут возникнуть мутации>
        КОНЕЦ
        <Map<Тип особи, оценка приспособленности>
            (начало_популяции, конец_популяции)>
        <Поместить потомков в новое поколение>
    КОНЕЦ
КОНЕЦ
```

Приведем некоторые пояснения к рассмотренной схеме.

Оценка приспособленности потомков в большинстве генетических алгоритмов — наиболее вычислительно сложное место. Кроме того, приспособленность каждого потомка оценивается независимо, поэтому выполнение этого шага состоит из некоторого количества частей, которые могут выполняться одновременно и независимо друг от друга. Этот факт во многом определяет возможность эффективной параллельной реализации генетических алгоритмов.

Как мы видим из рассмотренной схемы, программист вовсе не должен вдаваться в тонкости параллельного программирования. Он пишет программу на знакомом ему последовательном C++, а в наиболее вычислительно сложном месте алгоритма прибегает к использованию шаблона параллельного программирования.

6. Выводы, перспективы, направления дальнейшей работы

Параллельное программирование до сих пор считается одной из наиболее трудных дисциплин в компьютерных науках. Существующие технологии нуждаются в упрощении (при сохранении эффективности), поскольку сейчас возникает по-настоящему массовый спрос на параллельные приложения. Шаблоны параллельного программирования могут послужить одним из путей решения этой проблемы.

Разработанные нами реализации шаблонов демонстрируют, что их использование действительно позволяет добиться сокращения времени счета программы на кластере. При этом интерфейс к шаблону формулируются в терминах простых функциональных вызовов, освоение которых не требует длительной подготовки прикладного программиста. При реализации использовалась библиотека T-Sim шаблонных классов C++, реализующая подход T-системы к распараллеливанию вычислений, что дает нам преимущества динамического распараллеливания.

Дальнейшие исследования шаблонов параллельного программирования планируется вести по следующим основным направлениям:

- Разработка динамических адаптивных схем балансировки нагрузки.

  Такое решение, основанное на быстрых предварительных прогонах, позволило бы автоматически сгруппировывать легкие гранулы параллелизма в более сложные конструкции. В конечном счете, адаптивные схемы снизят накладные расходы на запуски легких гранул и увеличит масштабируемость шаблона, а также позволят прикладным программистам меньше заботиться о тяжести гранулы.
- Создание других шаблонов.

  Мы ведем работы по созданию шаблона Reduce и других шаблонов, на основе хорошо известных в функциональном программировании алгоритмов. В дальнейшем планируется разработка библиотеки шаблонов и конкретных алгоритмов, демонстрирующих их возможности (например, генетических алгоритмов)

Список литературы

[1] T-Sim — библиотека для параллельных вычислений на основе подхода T-системы: Международная конференция «Программные системы: теория и приложения», 2006.

[2] Schaeffer, Szafron Correct Object-Oriented Pattern-based Programming System. — http://www.cs.ualberta.ca/~systems/cops/index.html: Department of Computing Science University of Alberta.

[3] Home Page of the Skil Project. — http://www-i2.informatik.rwth-aachen.de/Skil/.

[4] Dean J., Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters. — http://labs.google.com/papers/mapreduce-osdi04.pdf.

[5] Суперкомпьютерная программа «СКИФ» Союзного государства: Разработка и освоение в серийном производстве семейства высокопроизводительных вычислительных систем с параллельной архитектурой (суперкомпьютеров) и создание прикладных программно-аппаратных комплексов на их основе. — http://skif.pereslavl.ru/skif.

[6] Абрамов С. М., Роганов В. А., Московский А. А. О задачах выравнивания нагрузки для сред с динамическим распараллеливанием вычислений (реализаций Т-системы): III Международная конференция «Параллельные вычисления и задачи управления», 2006.

[7] Отчет о научно-исследовательской работе за 2005 год (по проекту Функционально-ориентированные Т-суперструктуры как эффективное средство для построения высокопроизводительных распределённых приложений и сервисов): Исследовательский центр мультипроцессорных систем, Институт программных систем РАН, 2005.

[8] Воеводин В. В., Воеводин В. Параллельные вычисления: БХВ-Петербург, 2002, с. 609.

Институт программных систем Российской академии наук, Исследовательский центр мультипроцессорных систем

A. A. Moskovsky, A. Y. Pervin, E. O. Sergeeva. *First experience of implementing parallel programming skeletons using T-approach.* (in Russian.)

Abstract. The goal of this article is development and implementation of parallel programming skeletons based on tsim library. User instantiate the skeletons with applications specific part of code. At the same time, the opportunity to realize specific part of the solving problem application is given to user.

This paper considers review of various currently used developments of parallel programming skeletons. This work describes and solves the problem of development the suitable user interface, compatible with Standard Templates Library (STL). An an example, this article describes implementation of «map» template. This is one of most widely used template. The template class library TSim, that follows the T-approach to parallel computing, is employed as low-level programming tool.