Ю.В. Савинкова, С.В. Востокин

ПРИМЕНЕНИЕ ПОТОКОВОГО ПУЛА ДЛЯ РЕАЛИЗАЦИИ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ С СЕМАНТИКОЙ ПЕРЕДАЧИ СООБЩЕНИЙ НА ЯЗЫКЕ C++

(Самарский государственный аэрокосмический университет имени академика С.П. Королева (национальный исследовательский университет))

Модель потоков исполнения является самой распространённой моделью реализации параллелизма в библиотеках современных языков программирования, так как её семантика наиболее соответствует современной многоядерной архитектуре процессоров, а методы синхронизации потоков хорошо изучены. Однако использование потоков в программировании является сложным в силу необходимости решения проблемы конкурентного доступа к данным при реализации каждого конкретного алгоритма. Одним из подходов к решению данной проблемы является применение высокоуровневых моделей параллелизма, более близких к предметной области алгоритма по сравнению с исходной многопоточной моделью. В работе исследуется одна из моделей данного класса, основанная на обмене сообщениями между процессами.

Обычно для программирования на C++ выбираются сложные библиотеки с высокоуровневыми моделями параллелизма, например, Intel Threading Building Blocks. Этот выбор, в частности, мотивирован тем, что он гарантирует эффективную кросс-платформенную реализацию алгоритма. Библиотека содержит алгоритмы и структуры данных, позволяющие программисту избежать многих сложностей, возникающих при использовании традиционных реализаций потоков, таких как POSIX Threads, Windows threads или Boost Threads, в которых создаются отдельные потоки исполнения, синхронизируемые и останавливаемые вручную. Библиотека TBB абстрагирует доступ к отдельным потокам. Все операции трактуются как «задачи», которые динамически распределяются между ядрами процессора. Кроме того, достигается эффективное использование кэша [1].

Внедрение нового стандарта C++11 со встроенной поддержкой многопоточности позволяет компактно реализовать высокоуровневую модель без использования дополнительных библиотек, подобных Intel Threading Building Blocks. В данной работе рассматривается вопрос эффективности такой реализации на примере реализации поддержки времени выполнения в языке разметки Templet [2] и возможные дополнительные преимущества предлагаемого подхода.

В первой части доклада описаны интерфейсы исходной многопоточной модели и предлагаемой модели сообщений в терминах функций и структур данных языка C++.

Основные операции многопоточной модели выполнения в виде функций API представляются следующим образом:

```
struct thread{void(*tfunc)(thread*);};//структура
 // потока: локальное хранилище и функция потока
struct mutex{};     // мьютекс и функции
void lock(mutex*);  // для реализации
void unlock(mutex*);// конкурентного взаимодействия
struct event{};     // событие и функции
void wait(event*,mutex*); // для реализации
void notify(event*);// кооперативного взаимодействия.
```

Основные операции исследуемой модели выполнения языка разметки Templet представляются в виде функций API описываются следующим образом:

```
struct chan{}; // канал – моделирует сообщение
struct proc{void(*recv)(chan*,proc*);};// процесс,
// обрабатывающий сообщение с функцией обработки
void send(chan*,proc*);  // функция отправки сообщения
bool access(chan*,proc*);// функция проверки возможности
// доступа к сообщению из указанного процесса.
```

Далее представлено описание четырёх алгоритмов, основанных на паттерне «потоковый пул» [3], реализующих исполнение предлагаемой модели. Первый алгоритм реализует модель на однопроцессорной ЭВМ. Второй – процесс логической отладки на однопроцессорной ЭВМ и моделирует эффект недетерминированного выполнения. Третий – дискретно-событийное моделирование вычислений и позволяет оценить эффективность распараллеливания на последовательной ЭВМ. Четвёртый – выполнение на параллельной ЭВМ с разделяемой памятью. Обсуждаются критерии оценки эффективности для каждого алгоритма.

Тестирование алгоритмов проведено на задаче, имитирующей решение дифференциального уравнения в частных производных методом Гаусса-Зеделя. Построен алгоритм в терминах модели сообщений и выведено аналитическое выражение ускорения в зависимости от параметров теста. Описаны две программы нагрузочного тестирования: с использованием интерфейса к предлагаемой модели передачи сообщений и интерфейса tbb::graph библиотеки Intel TBB. По результатам тестирования даются рекомендации по практическому применению модели передачи сообщений и реализующих её алгоритмов.

Литература

1. Reinders, James Intel Threading Building Blocks. Outfitting C++ for Multi-core Processor Parallelism [Текст] / James Reinders. – Sebastopol.: O'Reilly Media, 2007.-336 с.

2. Востокин, С.В. Препроцессор языка Templet: инструмент программирования в терминах модели «процесс-сообщение» [Текст] / С.В. Востокин // Вестн. Сам. гос. техн. ун-та. Сер. Физ.-мат. Науки, 3(36) (2014), 169–182.

3. Schmidt, Douglas C., et al. Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects [Текст] / D.C. Schmidt, M. Stal, H. Rohnert, F. Buschmann. – Vol. 2. John Wiley & Sons, 2013.