УДК 004.432, 004.434:5,
004.4’42, 004.451.45

На правах рукописи

Арыков Сергей Борисович

Язык и система фрагментированного параллельного программирования задач численного моделирования

05.13.11 – математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ
диссертации на соискание учёной степени
кандидата физико-математических наук

Новосибирск – 2010

Работа выполнена в Учреждении Российской академии наук Институте вычислительной математики и математической геофизики Сибирского отделения РАН.

Научный руководитель: доктор технических наук,
профессор,
Малышкин Виктор Эммануилович

Официальные оппоненты: доктор физико-математических наук,
Андрианов Александр Николаевич

кандидат физико-математических наук,
доцент,
Скопин Игорь Николаевич

Ведущая организация: Нижегородский государственный университет им. Н.И. Лобачевского

Защита состоится 17 декабря 2010 г. в 15 час. 30 мин. на заседании диссертационного совета ДМ003.032.01 в Учреждении Российской академии наук Институте систем информатики им. А.П. Ершова Сибирского отделения РАН по адресу: 630090, г. Новосибирск, пр. Акад. Лаврентьева, д. 6.

С диссертацией можно ознакомиться в библиотеке Института систем информатики Сибирского отделения РАН.

Автореферат разослан 15 ноября 2010 г.

Ученый секретарь
диссертационного совета,
канд. физ.-мат. наук Мурзин Ф. А.

Общая характеристика работы

Актуальность работы. Роль численного моделирования в решении научных и прикладных задач продолжает возрастать. Для моделирования с высокой точностью необходимы огромные вычислительные ресурсы, поэтому для большинства задач разрабатываются параллельные программы, позволяющие выполнять моделирование на суперкомпьютерах.

Разработка хорошей параллельной программы в настоящее время остаётся сложной задачей. Параллельная программа должна обладать рядом динамических свойств (динамической настройкой на доступные ресурсы, динамической балансировкой загрузки, динамическим учётом поведения численной модели), которые существующими системами программирования в полной мере не обеспечиваются, а их ручная реализация требует серьёзных знаний теории параллельного программирования, архитектуры ЭВМ, программных особенностей средств разработки. В параллельных программах встречаются специфические ошибки, которые сложно обнаружить и устранить в силу недетерминизма исполнения таких программ. Наконец, перенос параллельной программы с одного вычислителя на другой часто также сопряжён со значительными трудностями.

Таким образом, специалистам-прикладникам приходится уделять слишком большое внимание проблемам параллельного программирования. В связи с этим, одним из перспективных направлений является создание систем параллельного программирования высокого уровня, которые в значительной степени скрывают от пользователя технические трудности разработки параллельных программ и позволяют сосредоточиться на алгоритме решения задачи.

Значительным шагом на этом пути можно считать такие системы, как HPF, DVM, mpC. Они берут на себя заботу о реализации коммуникаций, синхронизаций и некоторых других проблемах, однако, в силу излишне жестко

заданного управления (что характерно для всех императивных языков), автоматически выявить внутренний параллелизм алгоритма часто не удаётся.

С повсеместным распространением многоядерных процессоров задача разработки параллельной программы ещё более усложняется, поскольку эффективно распределить доступные ресурсы между крупными взаимодействующими последовательными процессами затруднительно. Это породило множество исследовательских проектов, направленных на поиск более подходящей модели вычислений. Среди российских разработок можно отметить проект МАРС (языки Барс и Поляр), систему НОРМА, систему OpenTS; среди зарубежных — библиотеку PLASMA, системы ALF, Charm++, SMP superscalar и другие.

Одним из новых подходов, альтернативных программированию на основе параллельно-последовательной модели вычислений, является фрагментированное программирование. К его основным преимуществам можно отнести возможность автоматической генерации параллельной программы, возможность автоматически обеспечить программам ряд динамических свойств, лучшую переносимость между вычислителями различной архитектуры, а также возможность выполнить формальный анализ параллельной программы. Этот подход в настоящее время активно развивается, и в существующих проектах (PLASMA, ALF, Charm++) реализованы лишь его отдельные элементы. Поэтому дальнейшее развитие средств фрагментированного программирования является актуальной задачей.

Объект и предмет исследования. Объектом исследования является фрагментированное программирование. Предмет исследования — языки и системы фрагментированного программирования.

Цель работы. Целью диссертационной работы является *разработка языка и системы фрагментированного параллельного программирования задач численного моделирования*.

Для достижения поставленной цели были сформулированы следующие задачи:

1. Анализ известных моделей, методов и программных средств параллельного программирования и формулирование требований к системе фрагментированного параллельного программирования.

2. Разработка модели вычислений, учитывающей особенности фрагментированного представления алгоритмов численного моделирования.

3. Разработка языка программирования, позволяющего представлять фрагментированные алгоритмы в терминах предложенной модели.

4. Реализация разработанных моделей, методов и алгоритмов в виде системы фрагментированного параллельного программирования для архитектур с общей памятью.

Методы исследования. В работе использовались результаты теории алгоритмов, теории множеств, теории формальных языков. В качестве формальной модели параллелизма взята асинхронная модель вычислений. В экспериментальных исследованиях применялись методы системного, объектно-ориентированного и параллельного программирования.

Научная новизна. Научной новизной обладают следующие результаты диссертационной работы, полученные автором лично:

1. Асинхронная модель вычислений с управлением на основе строгого частичного порядка, ориентированная на фрагментированное программирование численных задач, и алгоритмы её преобразования в асинхронную модель с управляющими операторами.

2. Совокупность способов задания управления во фрагментированной программе, включая массовое управление.

3. Алгоритмы реализации системы фрагментированного программирования Аспект для архитектур с общей памятью, а также сама реализация, в том числе:

   – алгоритмы генерации управляющих операторов, поддерживающие массовое управление;

   – методика представления асинхронной программы на языке C++.

Практическая значимость. Результаты работы могут лечь в основу широкого спектра средств разработки фрагментированных параллельных программ и применяться для решения прикладных задач численного моделирования. Язык Аспект позволяет начать накопление библиотеки фрагментированных алгоритмов, а возможность замены разработанной исполнительной подсистемы на другую обеспечивает переносимость Аспект-программ.

Система программирования Аспект используется в отделе МО ВВС Института вычислительной математики и математической геофизики СО РАН для выполнения исследований по различным проектам, а также в учебном процессе Новосибирского государственного университета и Новосибирского государственного технического университета для обучения студентов параллельному программированию.

На защиту выносятся:

1. Асинхронная модель вычислений с управлением на основе строгого частичного порядка, ориентированная на фрагментированное программирование численных задач, и алгоритмы её преобразования в асинхронную модель с управляющими операторами.

2. Язык фрагментированного программирования Аспект.

3. Система фрагментированного программирования Аспект и алгоритмы её реализации.

Апробация работы. Основные результаты диссертационной работы докладывались и обсуждались:

- на Конференции молодых ученых ИВМиМГ СО РАН (Новосибирск, 2005 и 2006 гг.);
- на VIII Всероссийской конференции молодых ученых по математическому моделированию и информационным технологиям (Новосибирск, 2007 г.);
- на Международной научной конференции «Параллельные вычислительные технологии (ПаВТ'2008)» (Санкт-Петербург, 28 января — 1 февраля 2008 г.);
- на VI Всероссийской конференции студентов, аспирантов и молодых ученых «Молодёжь и современные информационные технологии» (Томск, 26-28 февраля 2008 г.);
- на Международной научной конференции «Параллельные вычислительные технологии (ПаВТ'2009)» (Нижний Новгород, 30 марта — 3 апреля 2009 г.);
- на Международной научной конференции «Parallel Computer Technologies (PaCT-2009)» (Новосибирск, 31 августа — 4 сентября 2009 г.);
- на Международной научной конференции «Параллельные вычислительные технологии (ПаВТ'2010)» (Уфа, 29 марта — 2 апреля 2010 г.);
- на научных семинарах «Математическое и архитектурное обеспечение параллельных вычислений» (отдел МО ВВС Института вычислительной математики и математической геофизики СО РАН), «Архитектура,

системное и прикладное программное обеспечение кластерных супер-ЭВМ» (расширенный семинар ССКЦ, НГУ, Центра Компетенции СО РАН — INTEL), «Системное программирование», «Конструирование и оптимизация программ» (объединённые семинары кафедры программирования НГУ и Института систем информатики имени А.П. Ершова СО РАН).

Публикации. По теме диссертации опубликовано 11 работ, в том числе 3 статьи в рецензируемых журналах и 4 статьи в сборниках трудов международных конференций.

Личный вклад автора. Выносимые на защиту результаты получены автором лично. В опубликованных совместных работах участие автора заключалось в разработке и исследовании алгоритмов и их программной реализации. Постановка и исследование задач осуществлялись совместными усилиями соавторов. Создание и внедрение прикладных разработок осуществлено автором лично.

Структура и объем диссертации. Диссертационная работа состоит из введения, пяти глав, заключения, списка литературы из 125 наименований и пяти приложений. Общий объем работы составляет 195 страниц, из них 151 страница основного текста, включая 13 таблиц и 23 рисунка. Общий объем программного кода составляет 9702 строки.

Содержание работы

Во введении обоснована актуальность диссертационной работы, сформулированы цель и научная новизна исследований, показана практическая значимость полученных результатов, представлены результаты, выносимые на защиту.

В первой главе излагаются различные подходы к разработке параллельных программ и формулируются требования к системе программирования.

В разделе 1.1 приводится классификация средств разработки параллельных программ на основе метода разработки. Выделяются следующие методы: распараллеливание последовательных программ, использование параллельных библиотек, поддержка распараллеливающих конструкций в существующих языках, программирование на основе шаблонов, системы поддержки времени выполнения, новые языки программирования, визуальное проектирование параллельных программ. Описание каждого метода сопровождается краткой характеристикой, указаниями на сложности, возникающие при его применении, а также ссылками на известные проекты.

Раздел 1.2 посвящен детальному анализу средств разработки асинхронных программ. Рассматривается пять проектов: МАРС (Вычислительный центр СО АН СССР), OpenTS (Исследовательский центр мультипроцессорных систем ИПС РАН), Charm++ (Иллинойский университет в Урбана-Шампейн), Граф Плюс (Самарский государственный аэрокосмический университет) и PLASMA (совместный проект университета штата Теннесси, Калифорнийского университета в Беркли и университета штата Колорадо в Денвере). Для каждого проекта описывается модель вычислений, язык программирования и особенности реализации, анализируются достоинства и недостатки, приводится пример программы.

В разделе 1.3 на основе анализа рассмотренных проектов формулируются требования к перспективной системе параллельного программирования, включающие фрагментированное представление алгоритмов, явное описание управления, асинхронность, высокий уровень программирования, специализированность.

Вторая глава посвящена исследованию теоретических основ фрагментированного подхода к программированию.

В разделе 2.1 обсуждаются теоретические (синтез параллельных программ на вычислительных моделях), практические (метод «частицы-в-ячейках») и экспериментальные (оптимизация под существующие архитектуры) предпосылки возникновения фрагментированного программирования, после чего на примере задачи умножения матриц неформально поясняются основные понятия этого подхода.

Определение 1. *Фрагментированное программирование* — это представление алгоритма решения задачи в виде множества *фрагментов данных* и множества *фрагментов кода*. Фрагмент кода получает на вход набор *входных фрагментов данных*, на основе которых вычисляет набор *выходных фрагментов данных*. Подстановка фрагментов данных в качестве параметров фрагмента кода называется *применением* фрагмента кода к фрагментам данных (один и тот же фрагмент кода может применяться к различным фрагментам данных). Совокупность фрагмента кода и его входных и выходных фрагментов данных называется *фрагментом вычислений*. На множестве фрагментов вычислений задаётся строгий частичный порядок.

Определение 2. *Управление* — это множество ограничений на порядок выполнения фрагментов вычислений.

Раздел 2.2 посвящён разработке подходящей модели вычислений для фрагментированного программирования численных задач. В качестве базовой выбрана асинхронной модель вычислений. Даётся формальное определение простой асинхронной модели, асинхронной модели со структурными A-блоками, асинхронной модели с массовыми A-блоками. В этих моделях управление скрывается внутри управляющих операторов, что делает его недоступным для анализа, который необходим для автоматического/автоматизированного распределения ресурсов. Поэтому предлагается асинхронная модель вычислений с управлением на основе строгого частичного порядка, которая представляет информацию об управлении на уровне модели.

Определение 3. *Асинхронная программа с управлением на основе строгого частичного порядка* представляет собой набор $P = (M, A)$, где M – конечное множество *переменных*, $M = \{x_1, x_2, \ldots, x_m\}$; A – конечное множество *A-блоков*, $A = \left\{A_k \mid k \in \overline{1,n}\right\}$.

Множество M называется *памятью*. Память $M = X \cup Y$ состоит из множества $X = \{x, y, \ldots, z\}$ *простых* переменных и множества $Y = \{\bar{x}, \bar{y}, \ldots, \bar{z}\}$, разбитого на конечное число счётных, непересекающихся, линейно упорядоченных подмножеств, которые называются *массивами*, или *структурными переменными*. Элементы массива $\bar{x} = \{x_1, x_2, \ldots, x_n\}$ называются *компонентами* $\bar{x}$; компонент $\bar{x}_i$ обозначается $\bar{x}[i]$. $X \cap Y = \emptyset$. Запись нового значения в переменную стирает предыдущее значение.

Пусть $\mathbb{N}$ — множество натуральных чисел, $G \subseteq A \times \mathbb{N}$. Элемент $(A_k, i) \in G$ обозначается A_k^i и называется i-м экземпляром А-блока A_k. Пусть $N_{A_k} = \{i \in \mathbb{N} \mid (A_k, i) \in G\}$. А-блок A_k называется *простым*, если N_{A_k} — одноэлементное множество, и *массовым*, если N_{A_k} содержит более одного элемента. Множество N_{A_k} называется *областью применимости* А-блока A_k.

Экземпляр А-блока A_k^i описывается четвёркой $(M_k^i, T_k^i, O_k^i, R_k^i)$, где M_k^i — подмножество памяти M, используемое экземпляром А-блока A_k^i, причём $M_k^i = T_{k_{In}}^i \cup O_{k_{In}}^i \cup O_{k_{Out}}^i$; T_k^i — *триггер-функция*, представляющая собой предикат $T_k^i(x_1, x_2, \ldots, x_l)$, $x_s \in T_{k_{In}}^i$, $s = \overline{1,l}$; O_k^i — *операция*, которая по значениям *входных* переменных $x_1, x_2, \ldots, x_m$ вычисляет значения *выходных* переменных $y_1, y_2, \ldots, y_n$, $x_s \in O_{k_{In}}^i$, $y_t \in O_{k_{Out}}^i$, $s = \overline{1,m}$, $t = \overline{1,n}$, при этом количество входных и выходных параметров всех экземпляров одного А-блока совпадает: $(|O_{k_{In}}^p| = |O_{k_{In}}^q|) \wedge (|O_{k_{Out}}^p| = |O_{k_{Out}}^q|)$, $p \in \mathbb{N}$, $q \in \mathbb{N}$, $p \neq q$; R_k^i есть строгий частичный порядок, заданный на подмножестве C_k^i экземпляров А-блоков А-программы A: $\langle C_k^i, R_k^i \rangle$, который будем также обозначать $\langle C_k^i, < \rangle$. $R_k = \bigcup\limits_i R_k^i$, $R = \bigcup\limits_k R_k$.

Операция O_k^i есть либо множество частично рекурсивных функций $f_1, f_2, \ldots, f_n$, каждая из которых вычисляет одну выходную переменную

$y_1 = f_1(x_1, x_2, \ldots, x_m), y_2 = f_2(x_1, x_2, \ldots, x_m,), \ldots, y_n = f_n(x_1, x_2, \ldots, x_m)$, $x_s \in O^i_{k_{In}}$, $y_t \in O^i_{k_{Out}}$, либо А-программа $P' = (M', A')$, которая вычисляет все переменные из $O^i_{k_{Out}}$. А-блок A_k называется *атомарным*, если $O^i_k = \{f_1, f_2, \ldots, f_n\}$, и *структурным*, если $O^i_k = P'$.

Определение 4. *Асинхронная система* есть набор правил, позволяющих произвести вычисления для заданной асинхронной программы P:

1. При запуске А-программы на исполнение формируется множество A' *готовых* к исполнению экземпляров А-блоков A^i_k: $A' = \{A^i_k \mid \neg\exists(B^p_q, A^i_k) \in R \wedge T^i_k = \text{ИСТИНА}\}$.

2. Из множества готовых экземпляров A' выделяется некоторое подмножество A'', экземпляры которого запускаются на исполнение. Исполнение экземпляра А-блока A^i_k состоит в вычислении его операции O^i_k, проверке значений триггер-функций всех экземпляров B^p_q, у которых после вычисления O^i_k все предшествующие экземпляры А-блоков оказались вычисленными, и добавлении в множество A' тех B^p_q, у которых триггер-функции имеют значение ИСТИНА. Вычисление операции O^i_k состоит либо в вычислении всех функций $f_1, f_2, \ldots, f_n$, либо в вычислении А-программы P', которая вычисляется по тем же правилам, что и исходная А-программа P. В последнем случае $O^i_{k_{In}} \subseteq M'$, $O^i_{k_{Out}} \subseteq M'$, а операция O^i_k считается вычиленной, когда А-программа P' завершила своё исполнение. Экземпляры исполняются независимо друг от друга, каждый экземпляр исполняется не более одного раза.

3. Исполнение А-программы завершается, когда ни один экземпляр А-блока не исполняется и множество готовых экземпляров пусто: $A' = \emptyset$.

Если в асинхронной модели с управлением на основе строгого частичного порядка рассматривать каждую переменную (простую либо компонент

массива) как фрагмент данных, каждый А-блок — как фрагмент кода, а каждый экземпляр А-блока — как фрагмент вычислений, то представление алгоритма в асинхронной модели будет соответствовать фрагментированному представлению.

Теорема 1. Пусть операции экземпляров А-блоков реализуют лишь простейшие вычислимые функции (следования, выбора и нулевую функцию). Тогда класс асинхронных программ с управлением на основе строгого частичного порядка позволяет определить любую частично рекурсивную функцию.

В разделе 2.3 рассматриваются вопросы конструирования асинхронных программ. Предложены два формальных определения управляющих операторов: на основе признаков завершения/готовности и на основе счётчиков зависимостей. Основным достоинством первого подхода является более простая реализация, тогда как второй подход позволяет исполнять зависимые экземпляры А-блоков по мере вычисления значений отдельных выходных переменных предшествующих экземпляров А-блоков, что особенно важно в случае реализации операции экземпляра А-блока с помощью А-программы.

На основе предложенных формальных определений разработаны алгоритмы генерации управляющих операторов:

Алгоритм 1 позволяет сгенерировать управляющий оператор для экземпляра А-блока A_k^i асинхронной программы P с управлением на основе строгого частичного порядка используя формальное определение управляющих операторов на основе признаков завершения А-блоков.

Алгоритм 2 позволяет сгенерировать управляющий оператор для экземпляра А-блока A_k^i асинхронной программы P с управлением на основе строгого частичного порядка используя формальное определение управляющих операторов на основе счётчиков зависимостей по переменным.

Применяя алгоритм 1 или алгоритм 2 ко всем экземплярам А-блоков, можно преобразовать асинхронную программу с управлением на основе

строгого частичного порядка в асинхронную программу с управлением на основе управляющих операторов.

Третья глава посвящена описанию языка программирования Аспект, позволяющего разрабатывать фрагментированные параллельные программы.

В разделе 3.1. рассмотрены ключевые особенности языка: явное описание зависимостей между операциями, частичное распределение ресурсов, разделение управления и вычислений, ориентация на регулярные структуры данных, поддержка структурных фрагментов кода, представление вычислений внутри фрагментов кода с помощью внешнего языка (например, C++). При проектировании языка Аспект учтены особенности задач численного моделирования, что нашло отражение в поддерживаемых структурах данных и способах задания управления.

Если в императивных языках все команды по умолчанию исполняются последовательно, а участки программы, пригодные для параллельного исполнения, необходимо явно указать, то в языке Аспект используется противоположный подход: по умолчанию считается, что все фрагменты вычислений могут исполняться параллельно, а если необходим порядок (например, из-за зависимости по данным), то его необходимо задать явно.

В разделе 3.2 рассматривается формальный синтаксис и семантика языка Аспект. Пример простой Аспект-программы показана на рис. 1.

Раздел `preface` позволяет задать определения внешнего языка. Раздел `data fragments` позволяет определить фрагменты данных, раздел `code fragments` — фрагменты кода, раздел `task data` — данные задачи (статические и динамические), раздел `task computations` — фрагменты вычислений, раздел `task control` — управление, в котором слева от символа «<» указывается предшествующий экземпляр фрагмента вычислений, а справа — зависимый экземпляр фрагмента вычислений.

Множество номеров экземпляров массового фрагмента вычислений

```
program MultMatrix
preface {
   const int M = 3, N = 3;
};
data fragments
   double Matrix[M][M];
code fragments
   Mult(in Matrix X, Matrix Y, Matrix Z; out Matrix Z) {
      for(int i=0; i<M; ++i)
         for(int k=0; k<M; ++k)
            for(int j=0; j<M; ++j)
               Z[i][j] += X[i][k]*Y[k][j];
   };
task data
   Matrix A[N][N], B[N][N], C[N][N];
task computations
   S[i][j][k]: Mult(A[i][k], B[k][j], C[i][j], C[i][j])
      where i: 0..N-1, j: 0..N-1, k: 0..N-1;
task control
   S[i][j][k] < S[i][j][k+1];
end
```

Рис. 1. Аспект-программа умножения матриц

задаётся декартовым произведением индексов. Множество значений каждого индекса определяется в разделе `task computations` после ключевого слова `where` указанием нижнего и верхнего пределов диапазона натуральных чисел с шагом, равным единице. При определении одного индекса допускается использовать значение другого, например, j: i+1..N-1, но граф зависимостей между индексами должен быть ациклическим.

Управление может быть задано статически (с помощью раздела `task control`) и динамически (с помощью триггер-функций, записанных на внешнем языке). Статическое управление возможно задать как между разными фрагментами вычислений (например, A < B), так и между экземплярами одного и того же фрагмента вычислений (см. рис. 1). Поддерживается простое управление (между конкретными экземплярами), например, A[5] < B[5]; сложное управление (с использованием логических операций), например, (A | (B & C)) < D; и массовое управление (между всем экземплярами сразу),

например, A[i] < B[i]. Для обозначения всех экземпляров фрагмента вычислений используется пустой индекс [].

Для указания связи между предшествующим и зависимым экземплярами в случае массового управления допускается использовать выражения вида $y = x \pm b$, где x — идентификатор, а b — константа. Множество значений идентификатора можно ограничить с помощью предиката.

Раздел 3.3 посвящен программированию на языке Аспект. Обсуждаются преимущества фрагментированного программирования и сложности, возникающие при его применении. Рассматриваются способы записи типовых схем управления (последовательное, косое, конвейер и другие), разбирается решение двух задач: LU-разложение матриц и вычисление чисел Фибоначчи.

В четвёртой главе описывается реализация системы фрагментированного программирования Аспект и порядок работы с системой.

Раздел 4.1 посвящён программной архитектуре. В её основе лежит разделение системы на два компонента: транслятор с языка Аспект и исполнительную подсистему. Транслятор осуществляет перевод Аспект-программы в асинхронную программу с массовыми A-блоками на языке C++, а исполнительная подсистема обеспечивает поддержку времени выполнения (управление потоками, памятью, очередями готовых A-блоков).

Процесс разработки программы в системе Аспект показан на рис. 2. На

Рис. 2. Разработка программы в системе Аспект

вход транслятору подаётся текст программы на языке Аспект, на выходе генерируется асинхронная программа на C++, которую необходимо скомпилировать вместе с исполнительной подсистемой компилятором C++.

В разделе 4.2 рассматривается реализация транслятора с языка Аспект. Лексический и синтаксический анализы реализованы на основе общеизвестных подходов, поэтому основное внимание уделено генератору кода.

При генерации выделяются следующие этапы: генерация основной программы, генерация структуры А-программы и вспомогательных элементов, генерация переменных, генерация А-блоков, генерация управляющих операторов, генерация процедуры aplMain и генерация функции aplRun. В свою очередь, генерация управляющего оператора разделяется на генерацию заголовка, генерацию проверки условной области, генерацию множества потенциально готовых экземпляров, генерацию захватов примитивов синхронизации, генерацию изменения признака завершения, генерацию проверки потенциально готовых экземпляров и генерацию освобождения примитивов синхронизации.

Генератор кода создаёт один параметризованный шаблон управляющего оператора для всех экземпляров одного А-блока. Генерация кода основывается на алгоритмах, учитывающих массовое и сложное управление языка Аспект:

Алгоритм 3 позволяет для заданного А-блока A_k сгенерировать код, формирующий список потенциально готовых к исполнению (после завершения любого экземпляра A_k) экземпляров А-блоков.

Алгоритм 4 позволяет сгенерировать код, проверяющий экземпляр А-блока на готовность к исполнению.

Алгоритм 5 позволяет сгенерировать код, добавляющий в очередь все экземпляры А-блоков, которые могут быть исполнены сразу после инициализации А-программы.

В разделе 4.3 рассматривается реализация исполнительной подсистемы, ориентированная на архитектуру с общей памятью. Её структура представлена на рис. 3.

Слой абстрагирования от ОС включает в себя все подпрограммы, в которых используется API операционной системы (определение доступных

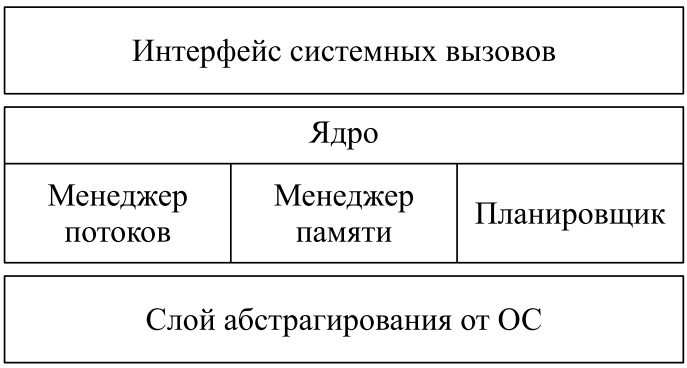

Рис. 3. Архитектура исполнительной подсистемы

ресурсов, работа с потоками, функции для работы со временем и другие). Это позволяет переносить исполнительную подсистему из одной ОС в другую заменой лишь одного, выделенного слоя.

Менеджер потоков управляет распределением работы между процессорами (ядрами). Менеджер памяти управляет распределением памяти в системе, осуществляет её выделение/освобождение для очередей и А-программ. Планировщик управляет набором очередей, каждая из которых имеет определённый приоритет и реализована в виде монитора.

Идея алгоритма планирования заимствована из реализации планировщика ядра ОС Linux версии 2.6. Алгоритм имеет сложность $O(1)$ и заключается в следующем: при каждом вызове метода get() очереди просматриваются в порядке убывания приоритетов; если в текущей очереди имеется готовый экземпляр А-блока, он запускается на исполнение. Таким образом, экземпляры А-блоков из очередей с более высоким приоритетом всегда будут запущены на исполнение раньше, чем экземпляры из очередей с более низким приоритетом. Если в очереди с некоторым приоритетом имеется несколько экземпляров, выбирается первый из них.

В разделе 4.4 рассмотрен порядок работы с системой. Описывается способ трансляции Аспект-программы и её окончательной сборки в исполняемый файл. Рассматриваются возможности системы по отладке Аспект-программ.

В пятой главе приводятся результаты практических испытаний системы программирования Аспект. Все тесты разделены на три группы: специальные задачи, модельные задачи и прикладные задачи.

В разделе 5.1 описывается тестовая среда и методика испытаний.

В разделе 5.2 рассматривается тестирование исполнительной подсистемы на специальных тестах, разработанных для проверки её характеристик. Накладные расходы при исполнении Аспект-программы складываются из накладных расходов исполнительной подсистемы и накладных расходов на исполнение управляющих операторов. Для каждого типа расходов разработаны отдельные тесты, результаты которых подтверждают основные характеристики, заложенные в систему Аспект при проектировании: сложность планировщика $O(1)$, стабильная работа при большом (миллион и более) количестве фрагментов вычислений, линейный рост накладных расходов на исполнение управляющего оператора при линейном увеличении числа зависимых фрагментов.

В разделе 5.3 рассматривается тестирование системы Аспект на модельных задачах, в качестве которых выбраны задачи умножения матриц, LU-разложения, и явная разностная схема, активно используемые в численных методах. Полученные результаты подтвердили хорошую масштабируемость фрагментированных программ. В этом же разделе приводится пример использования сторонних библиотек для реализации фрагментов кода.

В разделе 5.4 описываются результаты испытаний системы Аспект на двух прикладных задачах.

Первая задача состояла в оценке математических ожиданий аддитивных функционалов от траекторий диффузионных процессов и решалась методом Монте-Карло. Результаты измерений на системе с 4 процессорами Intel Xeon X7350 (16 ядер) и 256 Гбайт RAM для 64 млн. смоделированных траекторий приведены в таблице 1. Видно, что ускорение практически линейное, что

достигается за счёт независимого расчёта концентрации в каждом интервале, а также хорошей локальности данных, обеспеченной фрагментированным подходом.

Таблица 1. Результаты тестирования программы, реализующей задачу оценки математических ожиданий аддитивных функционалов от траекторий диффузионных процессов

Характеристика	Количество ядер				
	1	2	4	8	16
Время счёта, с	23805,50	12051,00	5939,00	2990,00	1510,68
Ускорение, раз	1	1,97	4	7,96	15,75

Вторая задача состояла в моделировании взаимодействия короткого лазерного импульса с плазмой и решалась методом «частицы-в-ячейках». Результаты измерений на сервере с 2 процессорами Intel Xeon E5520 (8 ядер, HT отключена) и 24 Гбайт RAM для 5 шагов моделирования приведены в таблице 2.

Таблица 2. Результаты тестирования программы, реализующей задачу моделирования взаимодействия короткого лазерного импульса с плазмой

Характеристика	Количество ядер			
	1	2	4	8
Время счёта, с	21,92	11,03	6,14	3,48
Ускорение, раз	1	1,99	3,57	6,30

Особенность задачи состоит в том, что расчёт электромагнитных полей распараллеливается хуже (из-за интенсивного обмена с памятью), чем расчёт движения частиц. Поэтому чем больше частиц используется при моделировании, тем лучше показатель ускорения. В то же время большее количество частиц приводит к лучшей точности.

В заключении кратко перечисляются основные результаты диссертации.

В приложении А приведены примеры асинхронных программ на различных языках программирования.

В приложении Б приведены допустимые лексемы и грамматика языка программирования Аспект.

В приложении В приведены тексты Аспект-программ.

В приложении Г приведены параметры командной строки для управления транслятором, а также сообщения об ошибках, выдаваемые им при трансляции программ.

В приложении Д приведено описание программного интерфейса (API) исполнительной подсистемы.

Основные результаты работы

1. Выполнен аналитический обзор существующих инструментов параллельного программирования, на основе которого сформулированы требования к перспективной системе параллельного программирования.

2. Предложена асинхронная модель вычислений с управлением на основе строгого частичного порядка, ориентированная на фрагментированное представление численных алгоритмов. Доказано, что в ней можно определить любую частично рекурсивную функцию, если операции экземпляров А-блоков реализуют лишь простейшие вычислимые функции. Разработаны алгоритмы преобразования этой модели в асинхронную модель с управляющими операторами.

3. Предложен язык параллельного программирования Аспект, позволяющий разрабатывать фрагментированные программы. Аспект реализует идею разделения управления и вычислений, поддерживает развитые сред-

ства задания управления и не содержит технических деталей реализации параллельной программы, что позволяет накапливать с его помощью фонд переносимых фрагментированных алгоритмов.

4. На языке Аспект реализованы фрагментированные версии алгоритмов умножения матриц, LU разложения, явной разностной схемы, оценки математических ожиданий аддитивных функционалов от траекторий диффузионных процессов, взаимодействия короткого лазерного импульса с плазмой, на которых продемонстрирована запись типовых схем управления.

5. Разработаны алгоритмы генерации управляющих операторов, поддерживающие многомерные массовые A-блоки, сложное и массовое управление языка Аспект.

6. Разработана и реализована система программирования Аспект, основными компонентами которой являются:

   – транслятор, обеспечивающий перевод программы с языка Аспект на C++;

   – исполнительная подсистема, обеспечивающая поддержку времени выполнения.

7. Проведено тестирование системы Аспект на специальных, модельных и прикладных задачах, подтвердившее основные характеристики, заложенные при её проектировании и показавшее высокую эффективность предложенных подходов на выбранном классе задач.

Публикации по теме диссертации

Публикации в журналах, входящих в перечень ВАК:

1. Арыков С. Б., Малышкин В. Э. Алгоритмы конструирования асинхронных программ заданной степени непроцедурности методом группировки // Вестн. Новосиб. гос. ун-та. Серия: Информационные технологии. 2009. Т. 7, № 1. С. 3–15.

2. Арыков С. Б. Язык программирования Аспект // Известия Томского политехнического университета. 2008. Т. 313, № 5. С. 89–92.

3. Арыков С. Б., Малышкин В. Э. Система асинхронного параллельного программирования «Аспект» // Вычислительные методы и программирование. 2008. Т. 9, № 1. С. 205–209.

Публикации в трудах международных конференций:

4. Арыков С. Б. Асинхронное программирование численных задач // Труды международной научной конференции «Параллельные вычислительные технологии (ПаВТ'2010)». Челябинск: Изд. ЮУрГУ, 2010. С. 28–39.

5. Arykov S., Malyshkin V. Asynchronous Language and System of Numerical Algorithms Fragmented Programming // Proceedings of the 10th International Conference on Parallel Computing Technologies (PaCT'2009). Berlin: Springer-Verlag, 2009. LNCS 5698. Pp. 1–7.

6. Арыков С. Б. Группировка данных в системе асинхронного параллельного программирования Аспект // Труды международной научной конференции «Параллельные вычислительные технологии (ПаВТ'2009)». Челябинск: Изд. ЮУрГУ, 2009. С. 357–363.

7. Арыков С. Б., Малышкин В. Э. Система асинхронного параллельного программирования «Аспект» // Труды Международной научной конференции «Параллельные вычислительные технологии (ПаВТ'2008)». Челябинск: Изд. ЮУрГУ, 2008. С. 290–295.

Прочие публикации:

8. Арыков С. Б. Представление алгоритма умножения матриц на языке программирования «Аспект» // Сборник трудов VI Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых «Молодежь и современные информационные технологии». Томск: Томский политехнический университет, 2008. С. 80–81.

9. Арыков С. Б. Архитектура системы асинхронного параллельного программирования «Аспект» // Материалы Всероссийской научно-технической конференции студентов, аспирантов и молодых ученых «Молодежь и наука: начало XXI века»: в 4 ч. Ч.1. Красноярск: Сибирский федеральный университет; Политехнический институт, 2007. С. 6–8.

10. Арыков С. Б. Реализация фрагментированной асинхронной модели вычислений в системе параллельного программирования // Тезисы докладов VIII Всероссийской конференции молодых ученых по математическому моделированию и информационным технологиям. Новосибирск: Ин-т выч. техн. СО РАН, 2007. С. 84–85.

11. Арыков С. Б. Некоторые подходы к разработке асинхронного языка программирования «Аспект» // Труды конференции молодых ученых. Новосибирск: ИВМиМГ СО РАН, 2005. С. 13–23.

Арыков С. Б.

Язык и система фрагментированного параллельного программирования задач численного моделирования

Автореферат

Подписано в печать 12.11.2010 г. Объем , уч.-изд. л.

Формат бумаги 60·90 1/16 Тираж 100 экз.

Отпечатано в ЗАО РИЦ «Прайс-курьер»

630128, г. Новосибирск, ул. Кутателадзе, 4г, тел. 330-72-02

Заказ №