

На правах рукописи

ГОРСКИЙ СЕРГЕЙ АЛЕКСЕЕВИЧ

**Инструментальный комплекс для организации
параллельных вычислений в интеллектуальных пакетах
прикладных программ**

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

АВТОРЕФЕРАТ

диссертации на соискание ученой степени
кандидата технических наук

Иркутск – 2008

Работа выполнена в Институте динамики систем и теории управления Сибирского отделения Российской академии наук (ИДСТУ СО РАН).

Научный руководитель: кандидат технических наук, доцент
Феоктистов Александр Геннадьевич

Официальные оппоненты: доктор технических наук, профессор
Массель Людмила Васильевна

кандидат технических наук, доцент
Хмельнов Алексей Евгеньевич

Ведущая организация: **Институт вычислительного
моделирования СО РАН**
(г. Красноярск)

Защита состоится 25 декабря 2008 г. в 10 часов на заседании диссертационного совета Д 003.021.01 при Институте динамики систем и теории управления СО РАН по адресу: 664033, г. Иркутск, ул. Лермонтова, 134.

С диссертацией можно ознакомиться в библиотеке ИДСТУ СО РАН.

Автореферат разослан 24 ноября 2008 г.

Ученый секретарь
диссертационного совета
д.ф.-м.н.

А.А. Щеглова

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. Бурный прогресс в области разработки программно-аппаратного обеспечения и коммуникационных сетей, наблюдаемый в последние годы, привел к появлению высокопроизводительных вычислительных сред (кластеров, метакомпьютеров, сетей грид). С точки зрения научно-исследовательской деятельности основным направлением применения такого рода сред является решение задач математического моделирования, требующих параллельного выполнения многовариантных вычислительных экспериментов. Решение такого класса задач в распределенной среде позволяет в значительной степени сократить общее время проведения расчетов. При этом для того чтобы на полную мощь использовать ресурсы среды, пользователю необходимо научиться управлять этими ресурсами и осуществить переход от последовательного программирования к параллельному.

На сегодняшний день разработаны разные модели, методы, алгоритмы и средства, ориентированные на поддержку различных этапов создания и выполнения параллельных программ (см., например, работы С.М. Абрамова, О.Л. Бандман, А.Б. Барского, В.В. Воеводина, Вл.В. Воеводина, И.Б. Задыхайло, В.Д. Корнеева, В.А. Крюкова, А.О. Лациса, И.А. Легалова, В.Э. Малышкина, Г.А. Опарина, Т.П. Плакса, В.В. Топоркова, Д. Ивенса, Э. Гейтса, Н. Макдоналда, Д. Гелернтера, Г.Р. Эндрюса, Ф.Г. Энслоу и др.). Однако массовое создание и использование таких программ сдерживается рядом нерешенных до конца проблем параллельного программирования, возникающих перед рядовым пользователем. В числе таких проблем: сложность выявления параллелизма алгоритма решаемой задачи и учета всех специфических особенностей аппаратного обеспечения и коммуникационных сред, возникающая в процессе разработки параллельной программы, выбора нужного языка (или системы) параллельного программирования и дальнейшего написания текста программы; необходимость обеспечения эффективного выполнения параллельной программы и ее переносимости; специфичный по сравнению с ПЭВМ интерфейс для доступа к вычислительной установке и управления процессом решения задачи.

Следует отметить, что появление высокопроизводительных вычислительных сред привело к значительному усложнению используемого в них программного обеспечения и вызвало необходимость существенного повышения уровня профессиональной квалификации рядового пользователя (специалиста-исследователя предметной области). Поэтому актуальным вопросом с точки зрения системного программирования становится создание для пользователя специальных средств выявления параллелизма алгоритма и составления схем решения задач, конструирования параллельных программ, доступа к вычислительной установке и выполнения на ней разработанных параллельных программ. Такие средства позволят существенно расширить круг потенциальных пользователей, снизив требования к уровню их программистской подготовки и тем самым обеспечив им возможность полностью сосредоточиться на решении своих прикладных задач. Одним из подходов, позволяющих частично реализовать эти средства,

является применение пакетов прикладных программ¹ для организации параллельных и распределенных вычислений в рамках технологии модульного программирования.

Естественной формой развития пакетов программ можно считать интеллектуальные пакеты², включающие средства синтеза в общем случае параллельных планов решения задач. Такие интеллектуальные пакеты кроме знаний о предметной области и алгоритмах решения задач способны накапливать знания о методах и средствах создания расчетных программ по планам решения этих задач, о свойствах и правилах включения в расчетную программу модулей, задающих содержательную интерпретацию вычислительных моделей, о способах управления вычислительным процессом.

Эффективная реализация перечисленных выше возможностей интеллектуальных пакетов для подготовки и выполнения параллельных программ требует создания для них системных компонентов, основанных на гибком сочетании уже существующих и разработке новых методов, моделей и языков в рамках парадигмы параллельного программирования.

Цель работы состоит в разработке и реализации для интеллектуальных пакетов методов и инструментальных средств конструирования асинхронных параллельных программ, обеспечивающих возможность поэлементной обработки структур данных в процессе проведения многовариантных расчетов при решении прикладных задач математического моделирования на вычислительных кластерах.

Объектом исследования являются теория и практика параллельного программирования.

Предметом исследования являются методы и инструментальные средства построения асинхронных параллельных программ.

Методы исследования. Для решения поставленных задач использованы методы объектно-ориентированного и сборочного программирования, теории построения трансляторов, баз данных, создания языков и инструментальных средств параллельного программирования.

Научная новизна диссертационной работы заключается в разработке для интеллектуальных пакетов моделей, алгоритмов и программных средств их реализации, предназначенных для создания и выполнения параллельных программ и базирующихся, в отличие от известных, на использовании как параллелизма по управлению (традиционно присущему такого рода пакетам), так и параллелизма по данным. В целом в диссертации представлен новый подход к конструированию асинхронных параллельных программ, обеспечивающих возможность поэлементной обработки параллельных структур данных. Характерной особенностью данного подхода является тесная интеграция средств и методов объектно-ориентированного и сборочного программирования, автоматизации подготовки и выполнения параллельных программ.

¹ Горбунов-Посадов М.М. Системное обеспечение пакетов прикладных программ / М.М. Горбунов-Посадов, Д.А. Корягин, В.В. Мартынюк. – М.: Наука, 1990. – 208 с.

² Инструментальные средства построения и эксплуатации пакетов знаний / Г.А. Опарин, А.Г. Феоктистов, Д.Г. Феоктистов, А.Е. Журавлев // Управляющие системы и машины. – 1997. – №1-3. – С.138-143.

Практическая значимость. Разработанное программное обеспечение позволяет повысить эффективность и сократить сроки разработки параллельных программ. Применение представленного в диссертационной работе инструментального комплекса при решении ряда задач показало достаточную простоту и легкость использования входящих в этот комплекс средств, а также обеспечило значительное сокращение трудозатрат на этапах создания и эксплуатации программного обеспечения сложных информационно-вычислительных моделей и методов их исследования.

Созданные программные средства зарегистрированы в Федеральной службе по интеллектуальной собственности, патентам и товарным знакам и применяются для проведения экспериментальных расчетов по плановым НИР в Институте динамики систем и теории управления СО РАН, а также в учебном процессе в Институте математики, экономики и информатики Иркутского государственного университета.

Исследование, разработка и применение рассматриваемых в диссертации программных средств выполнялись в рамках проекта СО РАН 3.2.6 «Интегрированные информационно-вычислительные и коммуникационные ресурсы: интеллектуальные методы организации, автоматизации разработки и применения» (2004-2006 гг., блок 1 «Распределенная вычислительная САТУРН-среда»); проекта СО РАН «Разработка научных основ распределенной информационно-аналитической системы на основе ГИС и Веб-технологий для междисциплинарных исследований» междисциплинарной программы 4.5.2 (2007-2009 гг., блок 2 «Интеллектуальные методы и инструментальные средства разработки и комплексирования распределенных информационно-вычислительных ресурсов»); проекта РФФИ № 06-01-00340 «Разработка и исследование булевых моделей предметной области в задаче планирования при синтезе программ».

Достоверность и эффективность полученных в работе результатов подтверждена опытом практической эксплуатации разработанного автором диссертационной работы инструментального комплекса и созданных с его использованием двух интеллектуальных пакетов, а также успешным решением ряда известных в литературе типовых примеров из области параллельного программирования.

Апробация. Основные результаты работы были представлены на V – VIII Школах-семинарах молодых ученых «Математическое моделирование, управление и информационные технологии» (Иркутск, 2004 г., 2005 г., 2006 г.), на XLIII Международной научной студенческой конференции «Студент и научно-технический прогресс» (Новосибирск, 2005 г.), на III Всероссийской молодежной конференции «Под знаком Σ » (Омск, 2005 г.), на V и VI Межрегиональных школах-семинарах «Распределенные и кластерные вычисления» (Красноярск, 2005 г., 2006 г.), на Всероссийском конкурсе инновационных проектов аспирантов и студентов по приоритетному направлению развития науки и техники «Информационно-телекоммуникационные системы» (Москва, 2005 г.), на XI Байкальской Международной конференции «Информационные и математические технологии в научных исследованиях» (Иркутск, 2006 г.), на III Междуна-

родной конференции «Параллельные вычисления и задачи управления (РАСО)» (Москва, 2006 г.), на Второй школе-семинаре молодых ученых «Управление большими системами» (Воронеж, 2007 г.), на Международных научных конференциях «Параллельные вычислительные технологии (PAVT)» (Челябинск, 2007 г.; Санкт-Петербург, 2008 г.), а также неоднократно на семинарах ИДСТУ СО РАН.

Публикации и личный вклад автора. Результаты диссертации отражены в 14-ти научных работах (в том числе 2 статьи в журналах, рекомендованных ВАК для опубликования основных научных результатов диссертации на соискание ученой степени доктора или кандидата наук). В перечисленных публикациях все результаты, связанные с алгоритмизацией, программной реализацией и вычислительным экспериментом на ЭВМ, получены автором лично. Результаты по моделям и методам организации интеллектуальных пакетов для вычислительных кластеров получены совместно с Феоктистовым А.Г. и являются неделимыми. Из совместных работ с Опариным Г.А. и Новопашиным А.П. в диссертацию включены только те результаты, которые принадлежат лично автору.

Структура работы. Диссертация состоит из введения, четырех глав, заключения, библиографии из 91 наименований и 6 приложений. Общий объем работы – 145 страниц, из которых 117 страниц основного текста, включающего 30 рисунков и 7 таблиц.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении обоснована актуальность темы диссертации, сформулирована цель исследования, показана научная новизна и практическая значимость полученных результатов, приведена структура работы.

Первая глава диссертации посвящена вопросам организации параллельных вычислений в системах модульного программирования.

В первом разделе главы обсуждаются технологические аспекты модульного программирования, приводятся основные понятия этого подхода к созданию прикладного программного обеспечения, рассматриваются средства, методы создания и примеры систем модульного программирования. Модульное программирование является развитием и совершенствованием двух направлений программирования: процедурного и сборочного. В первом случае имеется в виду декомпозиция сложной проблемы на подзадачи и модуляризация программы. Во втором случае подразумевается создание программного обеспечения путем подбора, настройки и организации совместного применения широко используемых модулей. Под модулем здесь следует понимать конструктивный программный элемент, используемый на различных этапах проектирования, разработки и выполнения программы. Основные подходы к созданию модульной системы представлены на рис. 1. Среди них следует выделить пакетный подход, позволяющий, во-первых, интегрировать преимущества других подходов к созданию модульных систем и, во-вторых, разрабатывать программное обеспечение для решения не одной, а целого класса взаимосвязанных задач.

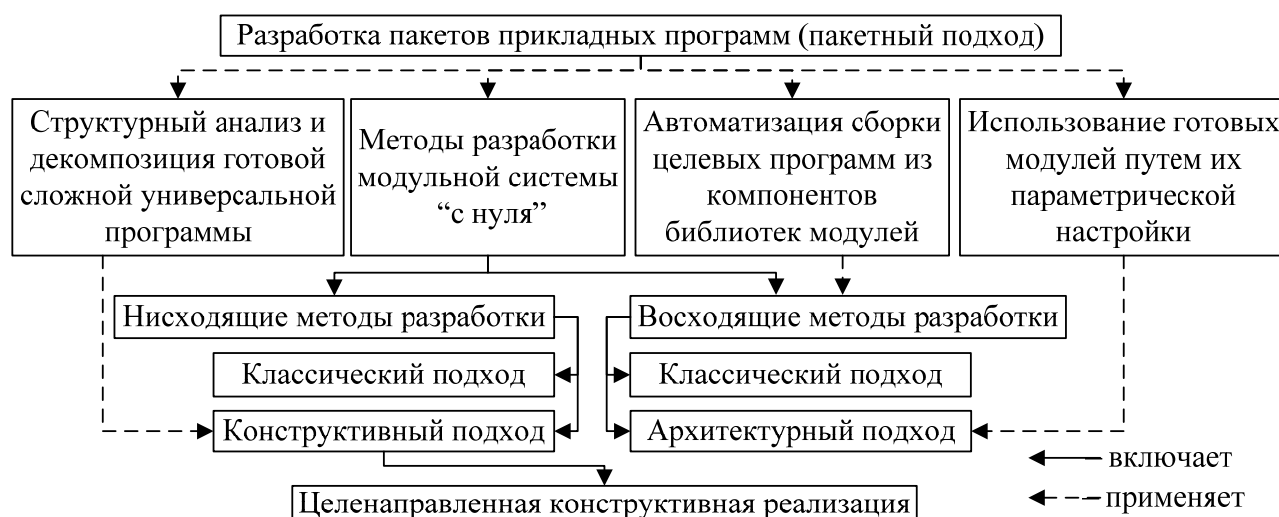


Рис. 1. Подходы к созданию модульных систем

Во втором разделе рассмотрена технология построения интеллектуальных пакетов, включающая следующие основные этапы: определение множества отношений между объектами исследуемой предметной области и формирование ее концептуальной схемы; формирование библиотеки вычислительных модулей, реализующих абстрактные операции над объектами предметной области; проведение анализа концептуальной схемы с целью выявления ее внутреннего параллелизма; формирование постановок исследовательских задач и синтез параллельных планов их решения; построение параллельных управляющих программ, реализующих эти планы.

Модели предметных областей, создаваемые в рамках подобных пакетов, относятся к классу вычислительных моделей³ и позволяют описывать вычислительные возможности пакета при решении определенного класса задач. Такую вычислительную модель можно определить как совокупность значимых величин (параметров) предметной области и информационно-логических связей между ними, реализуемых модулями пакета. Таким образом, вычислительная модель, по сути, определяет правила применения и сочетания модулей в процессе решения задачи и позволяет автоматически осуществлять планирование вычислений по непроцедурной постановке задачи вида: по заданным значениям параметров $x_1, x_2, \dots, x_n$ вычислить значения параметров $y_1, y_2, \dots, y_m$.

В простейшем случае концептуальная схема предметной области интеллектуального пакета определяется структурой⁴ $S = \langle Z, W, F \rangle$, где $Z = \{z_1, z_2, \dots, z_m\}$ – множество параметров предметной области; $W = \{w_1, w_2, \dots, w_k\}$ – множество допустимых типов данных; $F = \{f_1, f_2, \dots, f_n\}$ – множество вычислительных модулей⁵ предметной области. С каждым модулем f_i связано два множества параметров $IN^i, OUT^i \subset Z$, называемых соответственно его входом и выходом. Вход модуля IN^i определяет параметры, значения которых необходимо задать, чтобы

³ Тыгу Э.Х. Концептуальное программирование / Э.Х. Тыгу. – М.: Наука, 1984. – 256 с.

⁴ Опарин Г.А. Булево моделирование планирования действий в распределенных вычислительных системах / Г.А. Опарин, А.П. Новопашин // Теория и системы управления. – 2004. – № 5. – С.105-108.

⁵ Далее в диссертации модуль рассматривается как объект концептуальной схемы предметной области.

получить значения параметров, представленных выходом OUT^i . План решения задачи строится автоматически по непроцедурной постановке задачи, формулируемой на S , и является в общем случае параллельным.

В третьем разделе первой главы проведен анализ существующих методов и средств поддержки параллельного программирования с точки зрения их использования для достижения поставленной в рамках диссертационной работы цели. Из известных моделей параллельного программирования (модели передачи сообщений, использования общей памяти, прямого доступа к удаленной памяти) наиболее эффективной является модель, основанная на использовании общей памяти. Однако вычислительный кластер представляет собой вычислительную установку с распределенной памятью (его вычислительные узлы могут быть представлены SMP-машинами, но на сегодняшний день, как правило, с ограниченным числом процессоров). В этом случае наиболее целесообразным является выбор модели, базирующейся на передаче сообщений. В большинстве случаев реализация данной модели осуществляется с помощью одной из двух наиболее популярных коммуникационных библиотек: MPI или PVM. Обе библиотеки имеют свои сильные и слабые стороны. Существенными преимуществами PVM, с точки зрения решения прикладной задачи, являются наличие динамической системы управления вычислительными процессами и поддержка межязыкового взаимодействия между C и Fortran. Последнее обстоятельство особенно важно для пользователя, так как именно эти языки использованы в большинстве библиотек алгоритмов и программ прикладного характера.

Анализ систем, предназначенных для автоматизации параллельного программирования, показывает, что для интеллектуальных пакетов требуется создание специализированных средств решения данной проблемы. Это продиктовано тем, что одни системы (например, CODE, HeNCE, Grade, EDPEPPS, T-система) нацелены в основном на описание алгоритма решения отдельно взятой задачи, что делает их мало пригодными в рамках технологии интеллектуальных пакетов, а другие системы (например, НОРМА или СиКруС) не обладают достаточными возможностями для эффективного построения и выполнения асинхронной параллельной программы.

Последняя задача заключается в распределении n_{job} взаимосвязанных заданий (модулей) на n_{pr} процессоров вычислительной установки и в общем случае относится к классу NP-трудных задач⁶. Поэтому представляется целесообразной разработка эвристических алгоритмов, ориентированных на минимизацию времени планирования загрузки ресурсов. Такое планирование может осуществляться заранее (статически) или в процессе исполнения параллельной программы (динамически). При статическом планировании эффективной загрузки ресурсов необходимо знать их доступное количество, число взаимосвязанных подзадач, предназначенных для запуска на этих ресурсах, и предположительное время их выполнения, что не всегда возможно. С этой точки зрения более предпочтительным является динамическое планирование, при котором можно принять во

⁶ Гэри М. Вычислительные машины и трудно решаемые задачи / М. Гэри, Д. Джонсон. – М.: Мир, 1982. – 416 с.

внимание текущие изменения вычислительной среды⁷. Различные подходы к реализации механизмов динамического планирования без учета времени выполнения заданий используют такие факторы, как предшествование между заданиями, потребности в ресурсах, критичности работ и др. Достаточно популярными являются методы динамического управления заданиями, основанные на статических или динамических приоритетах. Следует заметить, что большинство этих алгоритмов не применимы в «чистом» виде в рассматриваемых интеллектуальных пакетах, ввиду специфики представления и использования имеющейся информации о предметной области, постановках задач и вычислительных процессах. Следовательно, необходима разработка специализированного алгоритма, учитывающего особенности интеллектуальных пакетов.

Итак, в первой главе диссертации на основе анализа методов и средств модульного и параллельного программирования были определены основные аспекты технологии конструирования и выполнения параллельных программ в интеллектуальных пакетах. Такая технология ориентирована на класс задач вычислительного эксперимента, в которых анализируются свойства некоторого объекта посредством проведения параллельных многовариантных расчетов над его математической моделью. В частности, были выделены основные этапы построения интеллектуального пакета и выбраны программно-аппаратные и языковые составляющие технологии параллельного программирования в этой системе: *вычислительный кластер + модель передачи сообщений + библиотека PVM + языки программирования C и Fortran + авторская система автоматизации параллельного программирования и статико-динамический алгоритм выполнения асинхронного вычислительного процесса.*

Во второй главе представлены разработанные автором методы, алгоритмы и языковые средства, предназначенные для организации параллельных вычислений в интеллектуальных пакетах.

В первом разделе второй главы множество допустимых типов данных W концептуальной схемы предметной области S расширяется новым типом данных *параметр-список* (здесь термин *список* понимается в смысле Д. Кнута⁸), предназначенным для определения параллельных структур данных⁹. Параметр-список создается на основе одного параметра любого типа (скалярного, векторного или матричного) и включает множество вариантов значений этого параметра. Число элементов параметра-списка задается параметром-скаляром целого типа. Основное отличие параметра-списка от параметра-массива заключается в способе обработки элементов параметров таких типов. Параметр-массив целиком передается в вычислительный модуль, в котором далее осуществляется его обработка, в то время как параметр-список может обрабатываться поэлементно (в общем слу-

⁷ Гершуни Д.С. Планирование вычислений в системах жесткого реального времени (обзор и перспективы) / Д.С. Гершуни // Вычислительная техника. Системы управления. – 1991. Вып. 6. – С. 4-51.

⁸ Кнут Д. Искусство программирования / Д. Кнут. – М.; СПб.; Киев: ИД «Вильямс», 2000. Т. 1–3.

⁹ Под параллельной структурой данных здесь понимается способ организации данных, позволяющий выполнять независимую параллельную обработку элементов этой структуры в рамках конкретной вычислительной системы (например, параллельный список в языке программирования Пифагор [Легалов А.И. Функциональный язык для создания архитектурно-независимых параллельных программ / А.И. Легалов // Вычислительные технологии. – 2005, №1(10). – С.71-89.].

чае параллельно) множеством экземпляров вычислительного модуля на разных процессорах. Наличие в концептуальной схеме предметной области параметров-списков приводит к появлению в процессе вычислений новых объектов (элементов параметров-списков и обрабатывающих их экземпляров модулей) и требует построения модели с более высоким уровнем детализации объектов предметной области и отношений между ними. Определим модель вычислений в виде структуры $CM = \langle S, PE, FE \rangle$, где S – это рассмотренная выше концептуальная схема предметной области, $PE = \{pe_1, pe_2, \dots, pe_v\}$ – множество элементов параметров-списков, $FE = \{fe_1, fe_2, \dots, fe_u\}$ – множество экземпляров модулей предметной области, $v, u \in N$. Для формального описания модели вычислений применим следующую нотацию¹⁰: $R(O_1(n_1, n_2):O_2(n_3, n_4)/c)$, где O_1, O_2 – множества объектов предметной области, связанные бинарным отношением R ; n_1, n_2 – числа, которые означают соответственно минимальное и максимальное число элементов O_1 , связанных с элементом O_2 ; n_3, n_4 – числа, которые означают соответственно минимальное и максимальное число элементов O_2 , связанных с элементом O_1 ; c – ограничение целостности, накладываемое на элементы O_2 , участвующие в отношении R . Тогда связи между множествами основных объектов модели вычислений зададим следующими отношениями¹¹ (рис. 2):

- 1) $R_1(Z(1, m):W(1, 1))$ – типизированные параметры;
- 2) $R_2(Z(1, m):F(1, n)/c_1)$ – входные параметры из $ZI \subset Z$ для модулей из F ; c_1 – ограничение на множество входных параметров модуля ($IN^i \cap OUT^i = \emptyset$);
- 3) $R_3(Z(1, m):F(1, n)/c_1)$ – выходные параметры из $ZO \subset Z$ для модулей из F ; c_1 – ограничение на множество выходных параметров модуля ($OUT^i \cap IN^i = \emptyset$);
- 4) $R_4(ZB(1, 2):ZA(1, ma)/c_2)$ – параметры из $ZB \subset Z$, задающие границы изменения индексов параметров-массивов из $ZA \subset Z$, $1 \leq ma < m$; c_2 – ограничение, требующее, чтобы эти параметры были скалярами целого типа;
- 5) $R_5(ZL(1, 1):ZU(1, 1)/c_3)$ – параметры-списки из $ZL \subset Z$, создаваемые на основе параметров из $ZU \subset Z$; c_3 – ограничение, накладываемое на тип параметра, на основе которого создается список (он может быть параметром любого типа, кроме списка);

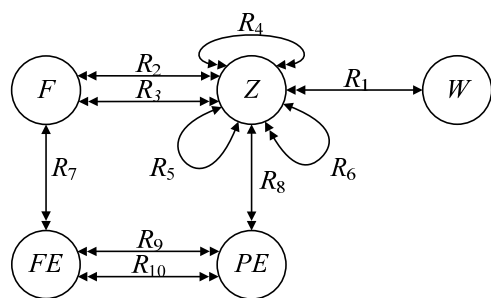


Рис. 2. Модель вычислений

- 6) $R_6(ZB(1, 1):ZL(1, ml)/c_2)$ – параметры из $ZB \subset Z$, задающие число элементов параметров-списков из $ZL \subset Z$, $1 \leq ml < m$; c_2 – ограничение целостности, требующее, чтобы эти параметры были скалярами целого типа;

- 7) $R_7(F(1, 1):FE(1, r))$ – экземпляры модуля, $r \in N$;

- 8) $R_8(ZL(1, 1):PE(1, s))$ – элементы параметров-списков, $s \in N$;

¹⁰ Цикритзис Д. Модели данных / Д. Цикритзис, Ф. Лоховски. – М.: Финансы и статистика, 1985. – 344 с.

¹¹ Следует отметить, что для задания некоторых из этих отношений используются вспомогательные объекты предметной области.

9) $R_9(FE(1, r):PE(1, s)/c_4)$ – входные элементы параметров-списков экземпляра модуля, $r, s \in N$; c_4 – ограничение, требующее, чтобы индексы этих элементов и обрабатывающего их экземпляра модуля совпадали;

10) $R_{10}(FE(1, r):PE(1, s)/c_5)$ – выходные элементы параметров-списков экземпляра модуля, $r, s \in N$; c_5 – ограничение, требующее, чтобы индексы этих элементов и обрабатывающего их экземпляра модуля совпадали.

В процессе вычислений параметры-списки могут обрабатываться как неделимые структуры данных. В первом случае обработка параметра-списка осуществляется аналогично обработке параметра-массива. Во втором случае под поэлементной обработкой параметров-списков подразумевается следующее: пусть x', y' – параметры-списки, созданные на основе параметров x и y , модуль $f_i: IN^i \rightarrow OUT^i$ предназначен для их поэлементной обработки, $x \in IN^i, y \in OUT^i$. Интерпретация модуля f_i выполняется следующим образом: 1) происходит параллельный запуск r экземпляров (r – размерность списков x', y') модуля f_i , j -му экземпляру модуля f_i^j передается j -й элемент списка x' ; 2) результат присваивается j -му элементу списка y' .

Рассмотрим основные принципы функционирования разработанного для интеллектуальных пакетов статико-динамического алгоритма выполнения асинхронного вычислительного процесса. Реализация алгоритма включает четыре способа исполнения плана решения задачи. Способ исполнения указывается пользователем на стадии компиляции параллельной программы.

Первый способ основан на случайном выборе модулей. Модули размещаются в очереди на выполнение в случайном порядке. Диспетчер осуществляет просмотр очереди, начиная с ее головы, и направляет первый готовый к запуску модуль на свободный процессор.

Второй способ предусматривает предварительное (до начала вычислений) назначение пользователем приоритетов модулей на основе своих экспертных знаний. Модули размещаются в очереди на выполнение в порядке убывания их приоритетов. Дальнейшая обработка очереди осуществляется аналогично первому способу.

Третий способ базируется на использовании эвристики, разработанной с учетом отсутствия информации о времени выполнения модулей и их приоритетах. На статической фазе работы алгоритма определяется степень влияния каждого модуля на другие модули (количество модулей, транзитивно зависящих от данного), и в соответствии с этой степенью влияния инструментальный комплекс назначает приоритеты модулям. Дальнейшее формирование очереди и ее обработка осуществляется аналогично второму способу.

Четвертый способ реализуется с учетом информации о времени выполнения модулей. Для планирования частично упорядоченной последовательности выполнения модулей на статической фазе вызывается планировщик системы крупноблочного синтеза параллельных программ¹², после чего в тело управ-

¹² Новопащин А.П. Методы и инструментальные средства крупноблочного синтеза параллельных программ для вычислительных кластеров. Автореф. дис. канд. техн. наук. Иркутск: ИДСТУ СО РАН, 2005. – 26 с.

ляющей программы (головного модуля) вставляются вызовы вычислительных модулей для реализации полученного расписания.

Условия, возникающие в процессе вычислений, при которых возможен запуск модуля или одного из его экземпляров, определим следующим образом. Пусть m и n задают, соответственно, число параметров и модулей в модели вычислений CM . Введем в рассмотрение следующие множества: $C^i = \{c_1, c_2, \dots, c_g\}$ – множество входных параметров модуля f_i , значение которых вычислены на данный момент времени, $C^i \subseteq IN^i$, $i \in \{1, 2, \dots, n\}$; $P^i = \{p_1, p_2, \dots, p_h\}$ – множество входных параметров-списков модуля f_i , обрабатываемых экземплярами этого модуля f_i^j поэлементно, $P^i \subseteq ZL \cap IN^i$; $E^i = \{e_1, e_2, \dots, e_q\}$ – множество индексов элементов параметров-списков из P^i , значения которых вычислены для каждого параметра-списка из P^i , $e_k \in N$. Тогда можно сформулировать условия, при которых возможен запуск модуля f_i или его экземпляра f_i^j .

Условие 1. Модуль f_i может быть запущен на выполнение, если $(|P^i| = 0) \wedge (|IN^i \setminus C^i| = 0)$.

Условие 2. f_i^j (j -й экземпляр i -го модуля) может быть запущен на выполнение, если $(|P^i| \neq 0) \wedge (IN^i \setminus C^i \subseteq P^i) \wedge (\exists k: e_k \in E^i) \wedge (j = e_k)$.

Определим следующие структуры данных: булеву матрицу предшествования Y размерности $n \times n$, $y_{ij} = 1$ означает, что модуль f_j предшествует модулю f_i (вычисляет значение хотя бы одного из его входных параметров); булеву матрицу зависимостей D размерности $n \times n$, $d_{ij} = 1$ означает, что модуль f_i транзитивно зависит от модуля f_j ; матрицу PR размерности $2 \times n$ (первая строка содержит номера модулей, вторая – соответствующие им приоритеты). Приведем основные этапы работы алгоритма выполнения асинхронного вычислительного процесса. После наименования этапа в скобках указаны номера способов выполнения плана решения задачи, использующих данный этап алгоритма.

Статическая фаза – инструментальный комплекс ORLANDO TOOLS, стадия компиляции.

- I. Инициализация структур данных Y, D, PR (сп. 1, 2, 3).
- II. Составление расписания (сп. 4).
- III. Построение матрицы предшествования Y (сп. 3).
- IV. Построение матрицы зависимостей D (сп. 3).
- V. Заполнение матрицы PR (сп. 1, 2, 3).
- VI. Выполнение упорядочения матрицы PR с помощью алгоритма быстрой сортировки по приоритетам (сп. 2, 3).

Динамическая фаза – параллельная программа, стадия выполнения.

- VII. Выполнение вычислительного процесса.
 - а) Выбор готового к запуску модуля с наивысшим приоритетом (сп. 1, 2, 3).
 - б) Запуск модуля (сп. 1, 2, 3).
 - в) Вызов модулей в соответствии с составленным планом (сп. 4).

VIII. Если все модули выполнены, то происходит завершение работы, иначе – ожидание завершения какого-либо модуля с помощью функции блокирующего приема. После завершения работы какого-либо модуля осуществляется прием данных от этого модуля и переход на этап VII (сп. 1, 2, 3).

Во втором разделе разработана структура параллельной программы, генерируемой в интеллектуальном пакете, и определены ее основные компоненты: вычислительные модули, базовые модули, головной (управляющий) модуль. Базовый модуль предназначен для организации порождения дочернего параллельного процесса выполнения вычислительного модуля из родительского процесса, реализуемого головным модулем. Базовый модуль выполняет следующие функции: прием значений входных параметров от головного модуля; вызов вычислительного модуля; отсылка значений выходных параметров головному модулю. Головной модуль предназначен для управления процессом решения задачи и выполняет следующие функции: ввод (вывод) значений входных (целевых) параметров; хранение информации о параметрах и их значениях; хранение информации о процессе вычислений; динамическое выполнение вычислительного процесса; проверка готовности входных данных модулей; запуск базовых модулей.

В третьем разделе второй главы разработаны средства создания системных контрольных точек, обеспечивающие отказоустойчивость вычислительного процесса решения прикладной задачи и базирующиеся на методологии конкретизирующего программирования¹³.

В последнем разделе второй главы разработан и реализован специализированный непроцедурный язык ORLANDO (Objects Relations in LANguage of DescriptiOns), предназначенный для спецификации задач вычислительного характера, решение которых может быть представлено в виде композиции функциональных блоков из заранее определенного базиса. Язык ORLANDO обеспечивает описание объектов предметной области: параметров, модулей, операций¹⁴ и их взаимосвязей, а также постановок задач. Все конструкции языка носят декларативный характер и задают правила вычисления значений параметров (модель вычислений). При спецификации предметной области на языке ORLANDO не требуется указывать порядок выполнения модулей, участвующих в решении задачи. Все информационно-логические связи выявляются и учитываются на этапах анализа содержимого файла спецификации и формирования модели вычислений. Язык позволяет сформировать запрос на решение задачи на модели вычислений, не уточняя, каким образом следует организовать вычислительный процесс.

В третьей главе определены необходимые функциональные и системные требования к инструментальному комплексу ORLANDO TOOLS, разработана и реализована его программная архитектура.

¹³ Ершов А.П. Научные основы доказательного программирования / А.П. Ершов // Вестник АН СССР. – 1984, № 10. – С. 9-19.

¹⁴ Операция – абстрактная процедура, реализующая отношение вычислимости между двумя подмножествами параметров предметной области, которое отражает возможность вычисления модулем искомого значений параметров первого подмножества, когда известны значения параметров второго подмножества.

Инструментальный комплекс ORLANDO TOOLS ориентирован на вычислительные системы, работающие под управлением ОС Linux. Архитектура инструментального комплекса (рис. 3) включает следующие основные компоненты: многооконный текстовый редактор, транслятор, подсистему компиляции, подсистему запуска, базы расчетных данных. Все эти компоненты устанавливаются и запускаются на рабочей станции пользователя.

Многооконный текстовый редактор предназначен для ввода описания вычислительной модели предметной области интеллектуального пакета на специализированном входном языке ORLANDO. Текстовый редактор обеспечивает набор функций, необходимых для быстрого и удобного формирования языковых конструкций ORLANDO, и взаимодействие с другими компонентами ORLANDO TOOLS: транслятором, подсистемами компиляции и запуска, базой расчетных данных.

Транслятор осуществляет разбор описания предметной области на языке ORLANDO и его перевод на язык C++. При этом используется разработанный автором синтаксический анализатор языка ORLANDO, созданный с помощью инструментального средства создания синтаксических анализаторов ANTLR.

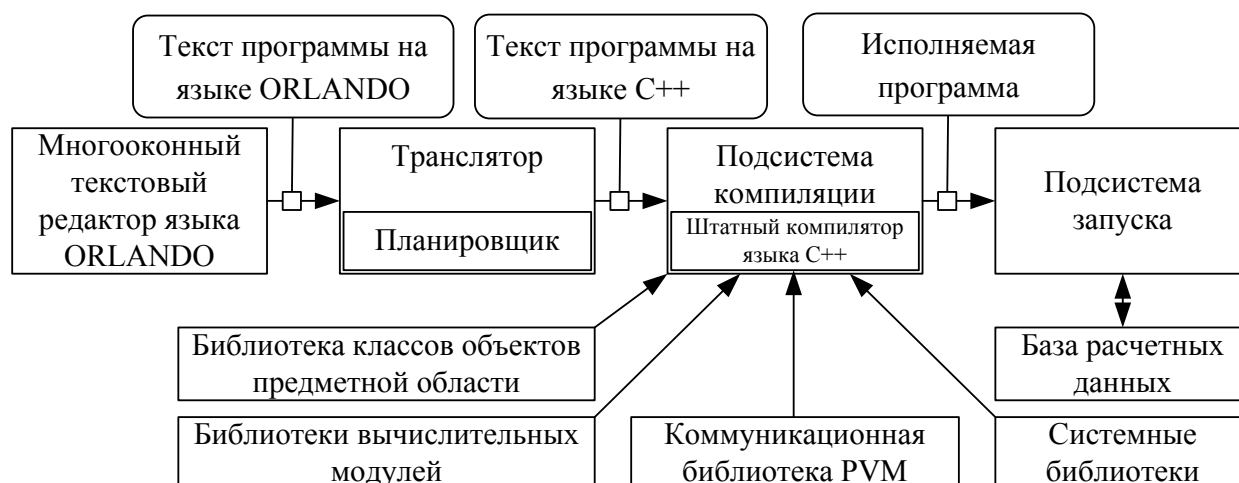


Рис. 3. Архитектура инструментального комплекса ORLANDO TOOLS

Подсистемы компиляции и запуска программ реализованы в виде единого программного приложения, которое может быть запущено из текстового редактора ORLANDO TOOLS. Подсистема компиляции параллельных программ выполняет следующие основные функции: осуществляет подключение к вычислительному кластеру; копирует исходные файлы программы на языке C++ на вычислительный кластер; производит трансляцию и компиляцию основной программы и программ модулей с использованием штатных транслятора и компилятора языка C++ (в случае возникновения ошибок пользователю выводятся соответствующие сообщения). При компиляции используются необходимые программные библиотеки: библиотека классов объектов предметной области и системные библиотеки, разработанные автором; библиотека вычислительных модулей пользователя; коммуникационная библиотека PVM.

Библиотека классов объектов предметной области предназначена для реализации объектов модели вычислений. Эта библиотека может использовать-

ся автономно от ORLANDO TOOLS в программах, работающих под управлением ОС семейства UNIX. Системные библиотеки обеспечивают выполнение следующих функций: работа с внутренними структурами данных; асинхронное исполнение плана решения задачи; управление запуском вычислительных модулей; ввод/вывод данных. Использование библиотеки классов объектов предметной области и системных библиотек позволяет также получить более компактный текст генерируемой программы и повысить его читабельность. Библиотеки вычислительных модулей включают функции пользователя, которые создаются с помощью сторонних средств разработки. Коммуникационная библиотека PVM обеспечивает поддержку параллельного программирования.

Подсистема запуска параллельных программ выполняет следующие основные функции: копирует файлы, содержащие значения входных параметров; осуществляет запуск программы на целевом кластере; производит мониторинг работы программы пользователя; по завершении вычислений копирует файлы, содержащие значения целевых параметров в файловую базу расчетных данных.

База расчетных данных предназначена для поддержки процесса проведения вычислительного эксперимента. Основной ее особенностью является возможность хранения расчетных данных по вариантам.

В четвертой главе рассматриваются примеры применения инструментального комплекса ORLANDO TOOLS и используемых в его работе методов и средств управления вычислительным процессом.

В первом разделе данной главы приводится пример, демонстрирующий работу статико-динамического алгоритма выполнения асинхронного вычислительного процесса. Пусть множество модулей модели вычислений задано следующим образом: $F = \{f_1(;z_1), f_2(z_6, z_{11}, z_{12};), f_3(z_1; z_2), f_4(z_2; z_3), f_5(z_3; z_4), f_6(z_3; z_5), f_7(z_4; z_7), f_8(z_7; z_9), f_9(z_4; z_6), f_{10}(z_3; z_{13}), f_{11}(z_5, z_{13}; z_8), f_{12}(z_8, z_9; z_{10}), f_{13}(z_8, z_9; z_{11}), f_{14}(z_{10}; z_{12})\}$, где запись $f_i(IN^i; OUT^i)$ определяет множество входных и выходных параметров модуля. Схема информационно-логических зависимостей между модулями базируется на графе, построенном для задачи упорядочения последовательности асинхронных работ¹⁵. Время выполнения модулей: $t_{f1}=0, t_{f2}=0, t_{f3}=1, t_{f4}=2, t_{f5}=2, t_{f6}=3, t_{f7}=2, t_{f8}=5, t_{f9}=1, t_{f10}=2, t_{f11}=2, t_{f12}=1, t_{f13}=4, t_{f14}=2$. Модули f_1 и f_2 моделируют постановку задачи «по заданному значению параметра z_1 вычислить значения параметров z_6, z_{11}, z_{12} ».

Был промоделирован процесс решения данной задачи на 2, 4 и 10 процессорах (для тестирования алгоритма как в случае ограниченности вычислительных ресурсов, так и в противоположном случае). В табл. 1 представлены результаты вычислительных экспериментов, полученные различными способами выполнения плана решения задачи, реализованными в инструментальном комплексе ORLANDO TOOLS. В табл. 1 приведены следующие характеристики: № – это номер вычислительного эксперимента; n_{pr} – число процессоров; Seq – вариант последовательности модулей в очереди (приоритеты модулей указаны в круглых скобках после их номеров) для третьего способа выполнения задания со временем решения задачи t , равным t_a ; t_{min} и t_{max} – соответственно минимальное и максимальное время

¹⁵ Лорьер Ж.-Л. Системы искусственного интеллекта / Ж.-Л. Лорьер. – М.: Мир. – 1991. – 568 с.

решения задачи. Значения t_{min} и t_{max} были получены в результате выполнения всех возможных вариантов планов решения задачи, что стало возможным вследствие небольшой размерности задачи. В табл. приведены варианты, актуальные с точки зрения демонстрации различных способов выполнения алгоритма. Вычислительные эксперименты для вариантов №№ 5-8 проводились с включением в модель вычислений модулей, предназначенных для поэлементной обработки параметров-списков. Номера таких модулей выделены в табл. 1 жирным шрифтом.

Таблица 1. Результаты вычислительных экспериментов

№	n_{pr}	Seq	t_{min}	t_a	t_{max}
1	2	1(13),3(12),4(11),5(7),6(6),10(5),7(5), 8(4) , 11(4) ,12(2),9(1),13(1),14(1),2(0)	16	16	19
2	2	1(13),3(12),4(11),5(7),6(6),10(5),7(5), 11(4) , 8(4) ,12(2),9(1),13(1),14(1),2(0)	16	17	19
3	4	1(13),3(12),4(11),5(7),6(6),10(5),7(5), 8(4) , 11(4) ,12(2),9(1),13(1),14(1),2(0)	16	16	16
4	4	1(13),3(12),4(11),5(7),6(6),10(5),7(5), 11(4) , 8(4) ,12(2),9(1),13(1),14(1),2(0)	16	16	16
5	2	1(13),3(12),4(11),5(7),6(6), 7(5) , 10(5) , 8(4) , 11(4) ,12(2),9(1),14(1), 13(1) ,2(0)	15.6	15.6	20
6	2	1(13),3(12),4(11),5(7),6(6), 10(5) , 7(5) , 8(4) , 11(4) ,12(2),9(1),14(1), 13(1) ,2(0)	15.6	16.6	20
7	4	1(13),3(12),4(11),5(7),6(6),10(5),7(5), 8(4) , 11(4) ,12(2),9(1),14(1), 13(1) ,2(0)	15	15	15.8
8	10	1(13),3(12),4(11),5(7),6(6),10(5),7(5), 8(4) , 11(4) ,12(2),9(1),14(1), 13(1) ,2(0)	15	15	15.4

Для четвертого способа работы алгоритма время выполнения плана решения задачи t равняется t_{min} . Это обусловлено тем, что в случае наличия информации о времени выполнения модулей для данной задачи можно построить оптимальный план выполнения вычислительного процесса. В случае, когда информация о времени выполнения модулей не известна, применяются оставшиеся три способа выполнения алгоритма. Для первого способа работы алгоритма время t изменяется в пределах от t_{min} до t_{max} , в зависимости от случайного порядка модулей в очереди. Для данного примера это может давать погрешность, равную 22% относительно t_{min} . При задании приоритетов модулей пользователем (второй способ выполнения алгоритма) время t зависит от его экспертных знаний с точки зрения логики схемы решения задачи. Для третьего способа $t=t_a$. Следует отметить, что порядок модулей в очереди может быть неоднозначным для данного способа выполнения алгоритма из-за того, что приоритеты разных модулей могут совпадать. При этом может изменяться время счета (см. варианты №№ 1, 2 и 5, 6; модули с одинаковыми приоритетами, перестановка которых в очереди влияет на время решения задачи, выделены подчеркиванием).

Анализ результатов вычислительных экспериментов показывает, что при увеличении количества процессоров t_{max} стремится к t_{min} , соответственно погрешность для первого, второго и третьего способов стремится к нулю (см. варианты №№ 2-3, 6-8). Отметим, что порядок модулей в очереди не имеет значения, когда количество процессоров больше степени максимального параллелизма плана решения задачи, так как отсутствуют задержки, связанные с ожиданием модулем свободного процессора, и все четыре способа выполнения алгоритма дают одинаковый результат. Примером тому служат варианты №№ 3 и 4, в которых использовались четыре процессора, в то время как степень макси-

мального параллелизма плана решения задачи в них равна трем. При отсутствии информации о времени выполнения модулей применение четвертого способа выполнения вычислительного процесса невозможно. Третий же способ позволил добиться вполне допустимой погрешности для всех вычислительных экспериментов, не превышающей в общем случае 7% при всех возможных вариантах распределения модулей по процессорам. Это позволяет рекомендовать этот способ выполнения алгоритма в случае отсутствия информации о времени выполнения модулей и числе процессоров меньшем, чем степень максимального параллелизма плана решения задачи.

Во втором разделе главы с целью иллюстрации возможности эффективного применения предложенных структур данных и средств их обработки для организации параллельных вычислений рассматривается разработанный автором диссертации пакет ГРАДИЕНТ, предназначенный для поиска глобального минимума многоэкстремальной функции $f(x)$ с помощью известного метода мултистарта.

Пакет ГРАДИЕНТ использовался для поиска глобального минимума ряда многоэкстремальных функций (Катковника, Растригина, Griewank), относящихся к классу сложных задач оптимизации. Изображение поверхностей этих функций представлено на рис. 4.

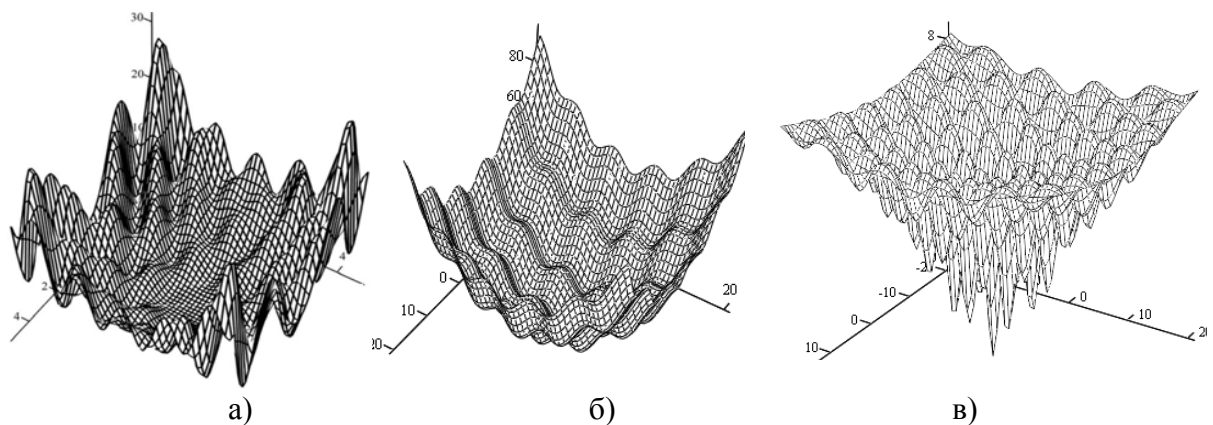


Рис. 4. Графики функций: Катковника (а), Растригина (б), Griewank (в)

1) Функция Катковника (рис. 4, а):

$$f(x,y)=0.5(x^2+y^2)(1+0.5\cos(1.5x)\cos(\pi y)+0.5\cos(\sqrt{5}x)\cos(3.5y)).$$

2) Функция Растригина (рис. 4, б):

$$f(x,y)=0.1x^2+0.1y^2-4\cos(0.8x)-4\cos(0.8y)+8.$$

3) Функция Griewank (рис. 4, в):

$$f(x,y)=\frac{-10}{0.005 \cdot (x^2+y^2) - \cos(x) \cdot \cos\left(\frac{y}{\sqrt{2}}\right) + 2} + 10$$

Все функции имеют глобальный экстремум в начале координат, равный 0.

Информационный двудольный ориентированный граф, представляющий модель вычислений пакета ГРАДИЕНТ, изображен на рис. 5.

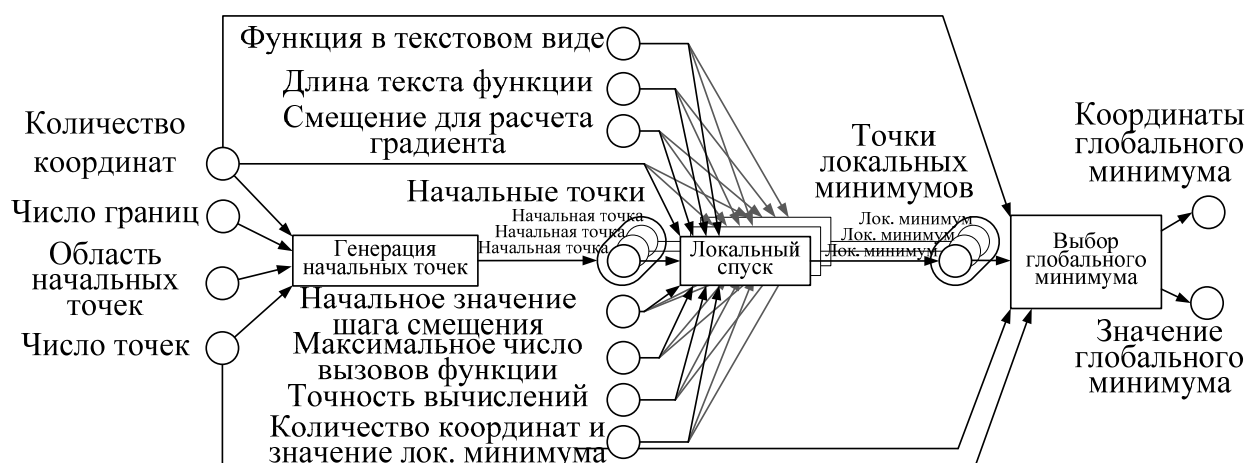


Рис. 5. Информационный двудольный ориентированный граф

Вычислительные эксперименты были проведены на вычислительном кластере МВС-1000М/16 ИДСТУ СО РАН. Для проведения расчетов на вычислительном кластере создавались параллельные виртуальные машины, включающие от 1 до 8 двухпроцессорных узлов кластера. Результаты расчетов представлены в табл. 2.

Таблица 2. Время поиска глобального минимума многоэкстремальных функций

№	Функция	Область (x,y)	Число точек	Время в секундах/Эффективность распараллеливания в %				
				Число процессоров				
				1	2	4	8	16
1	Катковника	$[-20;20]*[-20;20]$	1000	490	245/100	123/99,6	64/95,7	31/95
2	Растригина	$[-200;200]*[-200;200]$	10000	426	213/100	112/95	60/89	34/78
3	Griewank	$[-20;20]*[-20;20]$	100000	10796	5398/100	5048/53	4838/28	4316/8
4	Griewank	$[-20;20]*[-20;20]$	100000	626	313/100	157/99,7	82/95,4	42/93,2

Проведенные вычислительные эксперименты позволяют сделать следующие выводы. Методы, алгоритмы и средства, реализованные в инструментальном комплексе ORLANDO TOOLS, обеспечивают достижение хорошей эффективности при распараллеливании процесса вычислений для научных прикладных задач, позволяющих осуществлять их разбиение на независимые части с большой вычислительной мощностью (см. табл. 2, варианты №№ 1, 2, 4).

При недостаточной вычислительной мощности модуля резко возрастает процент накладных расходов, связанных с запуском модуля и передачей данных, что влечет за собой падение эффективности распараллеливания (см. табл. 2, вариант № 3). Для достижения необходимой мощности отдельной подзадачи декомпозицию данных на блоки требуемого размера можно осуществлять в рамках описания предметной области на языке ORLANDO путем применения параметров-списков, элементами которых являются векторные и матричные параметры нужных размерностей (см. табл. 2, вариант № 4).

В третьем разделе главы представлен пакет R-SIM, используемый при моделировании экономических показателей процесса реорганизации объектов ком-

мерческой недвижимости предприятия. Модель вычислений пакета R-SIM позволяет предметному специалисту предприятия (не программисту) сформулировать ряд постановок задач, автоматически сгенерировать для них целевые параллельные программы и выполнить с их помощью требуемые многовариантные расчеты. По результатам эксплуатации пакета R-SIM получен акт об его использовании.

Приложения содержат описания предметных областей пакетов ГРАДИЕНТ и R-SIM, библиотеки классов объектов предметной области, тексты сгенерированных на основе этих описаний параллельных программ и другие дополнительные материалы.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ ДИССЕРТАЦИИ

На защиту выносятся следующие основные результаты:

1. Модель вычислений, ориентированная на представление и поэлементную обработку параллельных структур данных в рамках интеллектуальных пакетов прикладных программ.

2. Язык ORLANDO и библиотека классов объектов предметной области, обеспечивающие удобное описание предметной области в соответствии с предложенной моделью вычислений и эффективную реализацию (автоматическое конструирование и последующее выполнение) параллельных управляющих программ по этому описанию.

3. Статико-динамический алгоритм выполнения асинхронного вычислительного процесса для решения задач, сформированных на языке ORLANDO, с поэлементной обработкой параллельных структур данных.

4. Системная архитектура и алгоритмы функционирования инструментального комплекса ORLANDO TOOLS и его программная реализация.

СПИСОК ПУБЛИКАЦИЙ ПО ТЕМЕ ДИССЕРТАЦИИ

1. Инструментальный комплекс ORLANDO TOOLS / Г.А. Опарин, А.Г. Феоктистов, А.П. Новопашин, С.А. Горский // Программные продукты и системы. – 2007. – № 4. – С. 63-65.
2. Феоктистов А.Г. Реализация метода мультистарта в пакете Градиент / А.Г. Феоктистов, С.А. Горский // Вестник НГУ. Сер.: Информационные технологии. – 2007. – Т. 5, вып. 2. – С. 78-82.
3. Горский С.А. Динамический алгоритм выполнения асинхронного вычислительного процесса / С.А. Горский // Прикладные алгоритмы в дискретном анализе: Сб. науч. тр. / Под ред. Ю.Д. Королькова. – Иркутск: Изд-во Иркутского гос. ун-та, 2008. – С. 4-22.
4. Горский С.А. ORLANDO TOOLS: Свидетельство об официальной регистрации программы для ЭВМ № 2007611625 / С.А. Горский, А.Г. Феоктистов. – М.: Федеральная служба по интеллектуальной собственности, патентам и товарным знакам, 2007.
5. Высокопроизводительные вычисления в пакетах прикладных программ / Г.А. Опарин, А.Г. Феоктистов, А.П. Новопашин, С.А. Горский // Парал-

- лельные вычислительные технологии: Тр. Междунар. науч. конф. (РАУТ'2007). – Челябинск: Изд-во ЮУрГУ, 2007. – Т. 2. – С. 193-200.
6. Феокистов А.Г. Работа с параллельными структурами данных в инструментальном комплексе ORLANDO TOOLS / А.Г. Феокистов, С.А. Горский / Параллельные вычислительные технологии (РАУТ'2008): Тр. Междунар. науч. конф. – Челябинск: Изд-во ЮУрГУ, 2008. – С. 482-487.
 7. Горский С.А. Создание пакетов программ в инструментальном комплексе ORLANDO TOOLS / С.А. Горский, А.Г. Феокистов // Сб. тр. II школы-семинара молодых ученых «Управление большими системами». – Воронеж: Научная книга, 2007. – Т. 2. – С. 33-39.
 8. Феокистов А.Г. Технология построения пакетов прикладных программ для вычислительных кластеров / А.Г. Феокистов, С.А. Горский // Параллельные вычисления и задачи управления: Тр. III Междунар. конф. РАУО'2006. – М.: ИПУ РАН, 2006. – С. 498-504.
 9. Горский С.А. Средства автоматизации процесса создания параллельных программ / С.А. Горский // Информационные и математические технологии в научных исследованиях: Тр. XI Байкальской междунар. конф.; в 2-х ч. – Иркутск: ИСЭМ РАН, 2006. – Ч. 2. – С. 20-26.
 10. Горский С.А. Исполнение параллельных программ в инструментальном комплексе ORLANDO / С.А. Горский // Математическое моделирование и информационные технологии: Материалы VIII школы-семинара молодых ученых. – Иркутск: ИДСТУ СО РАН, 2006. – С. 50-53.
 11. Горский С.А. Инструментальные средства для построения и применения параллельных пакетов прикладных программ / С.А. Горский // Студент и научно-технический прогресс. Информационные технологии. Материалы XLIII Междунар. науч. студ. конф. – Новосибирск: НГУ, 2005. – С. 15-17.
 12. Горский С.А. Языковые средства для построения и применения параллельных пакетов прикладных программ / С.А. Горский // Под знаком Σ : III Всерос. науч. молодежной конф. – Омск: ОНЦ СО РАН, 2005. – С. 232-233.
 13. Горский С.А. Инструментальные средства для построения параллельных программ / С.А. Горский // Распределенные и кластерные вычисления: Избранные материалы V школы-семинара / Под ред. В.В. Шайдурова. – Красноярск: ИВМ СО РАН, 2005. – С. 36-39.
 14. Горский С.А. Методы и инструментальные средства автоматизации конструирования параллельных программ / С.А. Горский // Материалы Всерос. конкурса инновационных проектов аспирантов и студентов по приоритетному направлению развития науки и техники Информационно-телекоммуникационные системы / Под ред. А.О. Сергеева. – М.: ГНИИ ИТТ Информика, 2005. – С. 30-31.

Редакционно-издательский отдел
Института динамики систем и теории управления СО РАН
664033, Иркутск, Лермонтова, 134
Подписано к печати 20.11.2008 г.
Формат бумаги 60×84 1/16, объем 1,2 п.л.
Заказ №8. Тираж 100 экз.

Отпечатано в ИДСТУ СО РАН

