Вестник УГАТУ
Уфа : УГАТУ, 2011 Т. 15, № 5 (45). С. 78–86



УДК 004.42

**А. П. Новопашин, И. А. Сидоров, С. А. Горский**

# ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ОРГАНИЗАЦИИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В ПАКЕТАХ ПРИКЛАДНЫХ ПРОГРАММ

Представлена базовая модель планирования параллельных крупноблочных вычислений, в качестве основного формализма которой используется аппарат булевых уравнений. На основе этой модели разработаны и программно реализованы инструментальные средства, предназначенные для создания и применения параллельных и распределенных пакетов прикладных программ. *Пакеты прикладных программ*; *инструментальные средства*; *параллельные и распределенные вычисления*; *планирование вычислений*

## ВВЕДЕНИЕ

Происходящее в последнее десятилетие широкомасштабное внедрение в практику расчетных работ параллельной вычислительной техники (в особенности кластерных архитектур) способствует возрождению интереса к пакетной проблематике, что влечет, в свою очередь, необходимость реконструкции старых и создания новых средств и методов организации параллельных вычислений с использованием пакетов прикладных программ (ППП).

В качестве самостоятельного научного направления пакетная проблематика сложилась в 70–80-х годах прошлого столетия. Если первые ППП представляли собой простые тематические подборки программ для решения отдельных задач в той или иной предметной области, то современные ППП являются сложными программными комплексами, включающими:

- прикладное программное обеспечение (функциональное наполнение) пакета в виде библиотеки прикладных программных модулей;
- системное программное обеспечение для организации решения пользовательских задач с использованием функционального наполнения, включающее средства автоматизации планирования вычислений (в том числе параллельных);
- специализированные языковые средства описания предметной области и постановки пользовательских задач.

Изначальная ориентация на широкий класс задач предметной области, в которой работает прикладной специалист, является одной из ключевых особенностей ППП. Сочетание в пакете разнообразных моделей и алгоритмов достигается путем использования принципа модульной организации функционального наполнения пакета. Входящий в состав ППП модуль представляется, как правило, в виде автономной программной единицы, записанной на традиционном языке программирования и обеспечивающей решение некоторой самостоятельной задачи.

Предполагается, что взаимодействие модулей возможно только на уровне их входных и выходных параметров. Использование принципа модульности позволяет заменить написание программы (в традиционном понимании) ее конструированием из готовых программных блоков крупного размера. Расширение класса задач, решаемых пакетом, может достигаться за счет подключения к ППП вновь создаваемых модулей.

В настоящее время серьезное внимание уделяется проблеме создания так называемых интеллектуальных ППП, позволяющих конечному пользователю формулировать свою задачу в содержательных терминах без указания алгоритма ее решения. Синтез схемы решения и сборка целевой программы производятся автоматически, при этом детали вычислений остаются скрытыми от пользователя.

Такой способ организации вычислений, берущий свое начало в работах Э. Х. Тыугу [1], С. С. Лаврова [2], Г. А. Опарина [3] и других, известен как концептуальное (структурное, сборочное) программирование. Он предполагает наличие вычислительной модели (схемы) предметной области, позволяющей описывать вычислительные возможности ППП для решения задач определенного класса. Такую вычислительную

Контактная информация: apn@icc.ru

Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии 2011».

Работа выполнена при частичной финансовой поддержке программы фундаментальных исследований Президиума РАН № 13 (проект СО РАН № 3).

модель можно определить как совокупность значимых величин (параметров) предметной области и функциональных отношений между ними. 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 $x_1$, $x_2$, …, $x_k$ вычислить значения параметров $y_1$, $y_2$, …, $y_r$».

Большое разнообразие разработанного прикладного программного обеспечения требует создания универсальных (изначально не привязанных к конкретной предметной области) инструментальных средств, позволяющих с наименьшими трудозатратами объединять имеющийся программный код в рамках одного ППП. Кроме этого, такой инструментарий должен обеспечивать корректность и отказоустойчивость вычислений в рамках ППП, эффективность использования функционального наполнения пакета, высокий уровень интеллектуализации с учетом современных требований, включая автоматизацию процессов выявления внутреннего параллелизма вычислительной модели, синтеза параллельных планов решения задач и генерации на основе этих планов параллельных прикладных программ. В настоящее время наблюдается дефицит инструментальных средств, обладающих перечисленными характеристиками.

Следующие разделы статьи посвящены описанию базовой вычислительной модели планирования параллельных вычислений и использующих эту модель инструментальных средств разработки параллельных и распределенных пакетов прикладных программ.

## БАЗОВАЯ МОДЕЛЬ ПЛАНИРОВАНИЯ ВЫЧИСЛЕНИЙ

В качестве базы знаний планировщика используется структура KB = ($F$, $Z$, $T$, $Y$, In, Out, Com), где:

$F = \{F_1,\ldots,F_n\}$ – множество имен программных модулей, реализующих операции предметной области;

$Z = \{Z_1,\ldots,Z_m\}$ – множество имен параметров (значимых величин) предметной области, являющихся входными или выходными параметрами модулей из $F$; с каждым модулем $F_i$ связано два множества параметров $A_i$, $B_i \subset Z$, называемых соответственно входом и выходом;

$T = \{T_1,\ldots,T_r\}$ – множество типов параметров;

$Y = \{Y_1,\ldots,Y_p\}$ – множество узлов вычислительной системы (ВС);

$\mathrm{In} \subset F \times Z$, $\mathrm{Out} \subset F \times Z$ – отношения, отражающие взаимосвязь модулей с данными соответственно по входу и выходу;

$\mathrm{Com} \subset F \times Y$ – отношение, определяющее статические связи между программными модулями и узлами ВС.

Предполагается, что база знаний $KB$ обладает высоким уровнем внутреннего параллелизма и является избыточной в том смысле, что для решения задачи вычисления требуемого набора целевых параметров $B_0$ по заданному множеству параметров – исходных данных $A_0$ используется только часть модулей из $F$, и/или эта задача имеет несколько альтернативных планов решения.

Отношения In, Out и Com представлены в виде трех булевых матриц $A$, $B$ и $C$ размерности $n \times m$, $n \times m$ и $n \times p$ соответственно. Элементы матриц формируются следующим образом: $a_{ij} = 1$ ($b_{ij} = 1$), если параметр $Z_j$ является входным (выходным) для модуля $F_i$; $c_{ij} = 1$, если модуль $F_i$ установлен в узле $Y_j$. Строки и столбцы матриц $A$, $B$ и $C$ являются двоичным представлением соответствующих подмножеств множеств $F$, $Z$ и $Y$, связанных отношениями In, Out и Com.

План вычислений представляется в виде матрицы $X$ $k \times n$ булевых переменных $x_{ij}$. Значение $x_{jj}$ равное единице означает, что программный модуль $F_j$ выполняется на $t$-м шаге плана $X$, где $t \in N = \{1, 2,...\}$ играет роль дискретного времени. Общая длина плана равна $k$, его строка задает множество параллельно исполняемых модулей, а столбцы соответствуют множеству модулей $F$.

Информационно-логические связи между параметрами и модулями, входящими в план, можно представить в виде двудольного ориентированного графа, включающего два непересекающихся множества вершин (рис. 1): множество вершин-параметров и множество вершин-модулей. Вершины-параметры, как и вершины-модули, являются попарно несмежными.

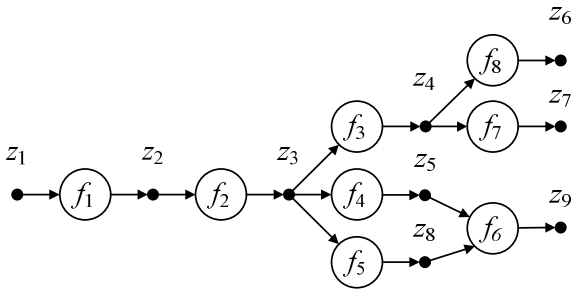


**Рис. 1.** Пример двудольного ориентированного графа, отражающего информационно-логические зависимости между параметрами и модулями ППП

К значениям элементов матрицы $X$ предъявляются ограничения, которые записываются в форме булевых уравнений и представляют собой требования к искомому параллельному плану, синтезируемому в процессе решения системы таких уравнений. К числу обязательных требований относятся следующие:

- допустимость (модули в плане должны быть упорядочены таким образом, чтобы каждый из них к моменту своего запуска был обеспечен необходимыми входными данными);
- бесповторность (каждый модуль может включаться в план не более одного раза);
- неизбыточность (исключение любого модуля из плана приводит к недопустимому плану);
- оптимальность (длина плана должна быть равна заданной величине $k$).

Кроме обязательных требований, при необходимости могут быть сформулированы и добавлены в общую систему ограничений дополнительные условия, позволяющие при построении плана учитывать временные задержки, связанные с исполнением программных модулей, ресурсные ограничения ВС (число узлов или процессоров), привязку модулей к узлам ВС, требование синхронности запуска модулей и другие ограничения. В [4] для большинства из перечисленных ограничений получены формулировки в виде булевых уравнений.

Преимущества такого подхода к планированию крупноблочных вычислений, при котором условия задачи формулируются в виде системы булевых уравнений (ограничений), а искомый план исполнения модулей является ее решением, состоит в том, что он позволяет, во-первых, получать параллельные планы требуемой длины, во-вторых, учитывать разнообразные ограничения, предъявляемые к плану, и, наконец, в-третьих, использовать существующие эффективные решатели булевых уравнений (или SAT-решатели), которые в ряде случаев обгоняют по быстродействию специализированные алгоритмы планирования.

С практической точки зрения можно выделить два типовых подхода к организации функционального наполнения пакета и распределения модулей по узлам ВС:

1. Все вычислительные модули содержатся в единой библиотеке, установленной на рабочей станции разработчика пакета. По непроцедурной постановке задачи производится автоматическое планирование параллельной схемы вычислений (с соблюдением установленных ограничений). В соответствии с полученной схемой осуществляется синтез целевой параллельной программы, ее компиляция и запуск на заданном числе узлов однородной ВС.

2. Вычислительные модули размещаются в узлах ВС, отличающихся программно-аппаратными характеристиками, что накладывает дополнительные требования к организации функционального наполнения ППП. При этом часть модулей может быть неотчуждаема от владельцев узлов, допускается множественность установки модулей в узлах ВС. Постановка задач осуществляется в процедурном виде, а исполнение построенных схем решения выполняется в режиме интерпретации.

Далее в статье рассматриваются инструментальные средства, реализующие описанные подходы к организации функционального наполнения параллельных и распределенных пакетов прикладных программ.

## ИНСТРУМЕНТАЛЬНЫЙ КОМПЛЕКС ORLANDO

Данный инструментарий ориентирован на создание ППП для однородных UNIX-кластеров с возможностью автоматической генерации параллельных программ для решения вычислительных задач в виде композиции готовых функциональных блоков. На рис. 2 представлена архитектура инструментального комплекса (ИК) ORLANDO, которая включает следующие основные компоненты: многооконный текстовый редактор, синтаксический анализатор, планировщик, генератор, подсистему компиляции, подсистему запуска, базу расчетных данных.

Высокоуровневый язык ORLANDO [5] предоставляет средства для описания объектов предметной области (параметров и операций), отношений между ними и обеспечивает на основе этих описаний постановку вычислительных задач в непроцедурном виде. Конструкции языка носят исключительно описательный (декларативный) характер и не предполагают указания порядка исполнения модулей, участвующих в решении задачи. Все информационно-логические связи между модулями выявляются на этапе трансляции описания предметной области. Текстовый редактор предоставляет пользователю набор функций, необходимых для быстрого и удобного формирования описания предметной области и постановок задач с использованием языковых конструкций ORLANDO, а также функций для взаимодействия с другими компонентами инструментального комплекса – транслятором, подсистемами компиляции и запуска, базой расчетных данных.


**Рис. 2.** Архитектура ИК ORLANDO

Синтаксический анализатор осуществляет разбор описания предметной области на языке ORLANDO, проверку корректности и целостности такого описания и его перевод во внутреннее представление системы планирования.

Планировщик, реализованный в соответствии с рассмотренной выше базовой моделью планирования, строит по непроцедурной постановке задачи с учетом имеющихся ограничений параллельный (в общем случае) план ее решения в виде частично-упорядоченного набора имен вычислительных модулей из функционального наполнения пакета.

Полученная таким образом абстрактная программа с использованием генератора преобразуется в программу на языке C++, состоящую из двух частей – управляющей и вычислительной. Управляющая часть программы оформляется в виде управляющего (головного) модуля, отвечающего за исполнение построенного параллельного плана решения задачи. Управляющий модуль реализует следующие функции: ввод (вывод) значений входных (выходных) параметров, хранение значений параметров в процессе вычислений, хранение информации о ходе вычислений, запуск отдельных вычислительных модулей на исполнение с использованием возможностей коммуникационной библиотеки PVM [6]. Вычислительная часть программы формируется из оберточных модулей. Для каждого вычислительного модуля, входящего в план решения задачи, создаются специальные программы-обертки, отвечающие за прием значений входных параметров от управляющего модуля, вызов соответствующей подпрограммы из функционального наполнения ППП, отсылку полученных значений выходных параметров управляющему модулю.

Подсистема компиляции параллельных программ выполняет следующие основные функции: обеспечивает подключение пользователя к вычислительному кластеру, копирует исходные файлы программ на языке C++ в пользовательский директорий на управляющем узле кластера, производит компиляцию главной программы и вспомогательных модулей с использованием штатных средств компилятора языка C++ (в случае возникновения ошибок выводятся

соответствующие сообщения). Для компиляции используются следующие библиотеки: библиотека классов языка C++ для программной реализации объектов предметной области; библиотека системных функций управляющего модуля; библиотека вычислительных модулей, представляющая собой функциональное наполнение ППП; коммуникационная библиотека PVM.

Подсистема запуска параллельных программ копирует файлы с наборами исходных данных, осуществляет запуск программы на целевом кластере, производит мониторинг работы программы пользователя, по завершении вычислений копирует файлы, содержащие значения целевых параметров, в базу расчетных данных. Главной особенностью такой базы является возможность хранения результатов по вариантам вычислительных экспериментов.

Порядок работы ИК ORLANDO предполагает, что на начальном этапе пользователь формирует описание предметной области, используя выразительные средства языка ORLANDO. Далее он приступает к решению задачи, формулируя ее в непроцедурной форме «исходные параметры => цель расчета» в виде указания планировщику. Построенный планировщиком параллельный план решения задачи на этапе трансляции преобразуется в параллельную программу на языке C++, после чего готовая параллельная программа перемещается на целевой вычислительный кластер, где компилируется штатными средствами. С помощью подсистемы запуска из базы расчетных данных на кластер копируются необходимые исходные данные и осуществляется запуск программы на счет.

## ИНСТРУМЕНТАЛЬНЫЙ КОМПЛЕКС DISCOMP

Инструментальный комплекс DISCOMP предназначен для поддержки основных этапов разработки и применения распределенных пакетов прикладных программ (РППП), ориентированных на работу в гетерогенной (многоплатформенной) распределенной вычислительной среде, которая может включать вычислительные кластеры различных конфигураций. Вычислительные модули, составляющие функциональное наполнение РППП, представляют собой исполняемые программы, которые могут быть реализованы на различных языках программирования (например, C, Fortran, Pascal и др.) и быть платформо-зависимыми. Допускается включение в состав функционального наполнения РППП не тиражируемых программных комплексов, размещенных в строго определенных узлах ВС, а также унаследованного программного обеспечения, переставшего соответствовать современным требованиям, но до сих пор эксплуатируемого в связи с трудоемкостью его модификации или замены. Удаленный запуск модулей, обмен данными между модулями через файлы и мониторинг узлов ВС реализуются средствами системной части РППП.

Для ИК DISCOMP в приведенной выше базовой модели планирования множество $T$ включает следующие типы параметров: тип *file*, используемый для описания параметров неопределенной структуры (блоков произвольного текста большого размера); тип *filelist*, предназначенный для поддержки распараллеливания по данным (параллельный список параметров типа *file*); тип *fileconst*, введенный с целью сокращения объемов передачи данных в ВС (значение параметра типа *fileconst* один раз передается узлу ВС и затем может неоднократно использоваться при запуске модулей, размещенных в данном узле). Содержательно модуль $f_i \in F$ реализует возможность вычисления множества целевых (выходных) параметров модуля по множеству заданных (входных) параметров модуля. Поэлементная обработка параметра $z_j$ типа *filelist* модулем $f_i$ выполняется таким образом, что каждый элемент параметра $z_{jd}$ обрабатывается соответствующим экземпляром модуля $m_{id}$.

Постановка содержательной задачи для структуры *KB* задается пользователем пакета в процедурном виде и в общем случае формулируется следующим образом: «зная *KB*, выполнить *Q*», где $Q$ представляет собой частично упорядоченную последовательность модулей $Q_i \subset F$. В процессе установления частичного порядка множество $Q$ разбивается на $k$ непустых подмножеств. Упорядочение подмножеств осуществляется по степени готовности модулей к исполнению. В рамках каждого подмножества входящие в него модули могут выполняться независимо друг от друга в любой последовательности или параллельно.

Под схемой решения задачи (СРЗ) в ИК DISCOMP понимается модель крупноблочной программы, отражающая информационно-логическую структуру вычислений в терминах предметной области. СРЗ строится автоматически в параллельно-ярусной форме из элементов следующих множеств: множества параметров $Z$; множества модулей $F$; множества событий $E$, возникающих в процессе выполнения СРЗ; множества операций $H$, предназначенных для управления процессом выполнения СРЗ; множества специальных операторов $O$ = {*START, STOP, READ* <список параметров>, *WRITE*

<список параметров>, *CALL* <имя модуля>, *FORK, JOIN, TERMINATE* <список модулей>}.

Построение СРЗ осуществляется в два этапа:

- разработчик РППП формирует ярусы СРЗ, размещает на этих ярусах модули из *F* и определяет события из *E*, при возникновении которых необходимо выполнить те или иные управляющие функции из *H*;
- транслятор постановки задачи на основе информационно-логических связей между параметрами и модулями структуры *KB* определяет множества входных параметров $A_0$ и целевых параметров $B_0$ СРЗ, выполняет проверку необходимых ограничений (допустимость, бесповторность, неизбыточность) и дополняет постановку задачи специальными операторами из *O*.

Выполнение СРЗ в режиме интерпретации представляет собой процесс параллельно-последовательного выполнения ее операторов в соответствии с порядком их вхождения в управляющий граф (рис. 3), вершинами которого являются операторы вызова модулей (или их экземпляров) из *F*, а также специальные операторы. Передача данных (значений параметров) между модулями выполняется в соответствии с информационным графом (рис. 4).

На рис. 5 приведен пример процедурной постановки задачи в формате, разработанном на основе расширяемого метаязыка XML. Данная постановка соответствует приведенным выше информационному и управляющему графам СРЗ и включает четыре яруса. На первом ярусе выполняется модуль декомпозиции, осуществляющий разбиение входного параметра $z_1$ на множество блоков (элементов параметра списка $z_2$). На следующем ярусе производится поэлементная обработка параметра-списка $z_2$ экземплярами модуля *solver*, которые могут выполняться параллельно на доступных узлах ВС. На третьем ярусе полученные результаты (элементы параметра-списка $z_3$) передаются на вход модулю *analyse*, результатом выполнения которого является параметр $z_4$. И на завершающем ярусе параллельно запускаются модули *plot_graph* и *statistics*, которые получают на вход значения параметров, вычисленных на предыдущих ярусах, и производят формирование значений параметров $z_5$ и $z_6$.

К основным составляющим программно-аппаратной архитектуры инструментального комплекса DISCOMP относятся: система управления ВС, набор вычислительных клиентов ВС, система хранения данных и средства доступа пользователей к РППП. Схема взаимодействия основных компонентов представлена на рис. 6.

Система управления ВС (серверная часть ИК DISCOMP) поддерживает взаимодействие с подсистемами хранения данных и доступа пользователей к пакету, а также обеспечивает централизованное управление узлами ВС.

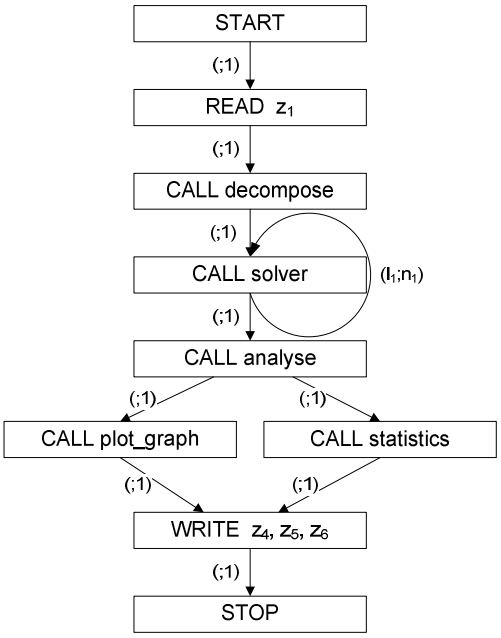


**Рис. 3.** Пример управляющего графа СРЗ

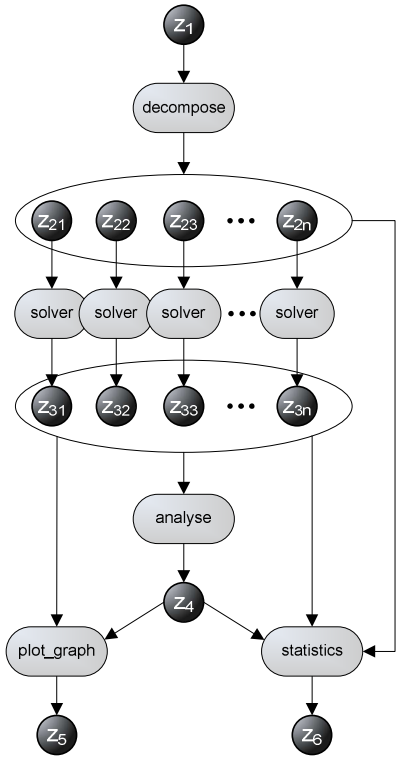


**Рис. 4.** Пример информационного графа СРЗ

Серверная часть ИК DISCOMP включает системное ядро, менеджеры вычислительных процессов и ресурсов, диспетчер очереди задач, подсистему журнализации и исполнительную подсистему.

Вычислительный клиент реализует процесс выполнения модуля в узле ВС и осуществляет следующие функции: организацию соответствующей среды для работы модуля (создание временных директорий, файлов входных и выходных параметров модулей, задание значений переменных окружения, перенаправление ввода / вывода и т. д.); получение значений входных параметров модуля из управляющего узла; запуск модуля; контроль процесса его выполнения; отсылку значений выходных параметров в управляющий узел после завершения работы модуля.

```
<?xml version="1.0"?>
<scheme>
  <stage>
    <module name='decompose'/>
  </stage>
  <stage>
    <modulelist name='solver'/>
  </stage>
  <stage>
    <module name='analyse'/>
  </stage>
  <stage>
    <module name='plot_graph'/>
    <module name='statistics'/>
  </stage>
</scheme>
```

**Рис. 5.** Пример процедурной постановки задачи

Система хранения данных используется для структурированного размещения описаний предметной области, постановок задач в формате XML и расчетных данных в виде файлов, а также для предоставления доступа к ним из различных подсистем ИК DISCOMP. Синхронизация процессов чтения / записи данных осуществляется посредством файловых блокировок.

Средства доступа пользователей к РППП обеспечивают взаимодействие пользователя с пакетом, управление вычислительными процессами, ввод данных, получение результатов вычислений. ИК DISCOMP предоставляет два способа взаимодействия пользователя с РППП: с помощью набора утилит командной строки и посредством интерактивного пользовательского web-интерфейса.

Связь распределенных компонентов ИК DISCOMP между собой реализована на основе специально разработанного протокола сетевого взаимодействия, поддерживающего обмен сообщениями, удаленный вызов процедур и передачу больших объемов бинарных данных. Протокол сетевого взаимодействия обеспечивает высокопроизводительный обмен данными между компонентами ИК DISCOMP за счет установки между ними непрерывного соединения по TCP-каналу, использования многопоточного режима работы серверной части ИК DISCOMP и минимизации объемов управляющих данных в теле передаваемого сообщения.


**Рис. 6.** Схема взаимодействия компонентов ИК DISCOMP

Эффективность решения задач в РППП обеспечивается следующими особенностями работы ИК DISCOMP: возможностью максимально эффективного использования разнородных ресурсов ВС; поддержкой высокопроизводительного взаимодействия между распределенными компонентами ИК DISCOMP; гибкой диспетчеризацией очередей задач; наличием средств оптимизации объемов данных, передаваемых между модулями в процессе решения задач в ВС; возможностью динамического управления процессами решения задач.

Динамическое управление вычислительным процессом в ИК DISCOMP базируется на механизмах обработки событий, возникающих при выполнении СРЗ. Алгоритм процесса управления включает две основные фазы: 1) применение операций, предназначенных для обработки события и выбора необходимого воздействия на процесс вычислений; 2) проверку корректности выбранного воздействия системными функциями РППП (контроль выполнения множества ограничений для интерпретатора СРЗ). Управ-

ляющие операции задаются разработчиком РППП при описании предметной области, а их вызовы реализуются в виде функций на языке JavaScript и включаются в постановку задачи. Динамическое управление вычислительным процессом позволяет сократить общее время решения задачи за счет устранения вычислительной избыточности на основе анализа текущих результатов счета в узлах ВС. Более подробно данные механизмы динамического управления вычислительным процессом рассмотрены в [7].

## ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

Представленные в данной работе инструментальные средства получили широкое применение в Суперкомпьютерном центре коллективного пользования ИДСТУ СО РАН при решении прикладных задач из различных предметных областей: исследование биоресурсов озера Байкал; моделирование логистических складских систем; решение задач оптимального управления, решение систем булевых уравнений и других.

С помощью ИК ORLANDO был разработан пакет ГРАДИЕНТ, предназначенный для поиска глобального минимума многоэкстремальной функции высокой размерности с помощью метода мультистарта. Пакет позволяет варьировать управляющие параметры, влияющие на точность нахождения глобального минимума и время решения задачи. Возможности пакета испытывались на ряде многоэкстремальных функций (Катковника, Растригина, Griewank), относящихся к классу сложных задач оптимизации. Результаты вычислительного эксперимента приведены в табл. 1.

Таблица 1

**Время решения задачи поиска глобального минимума**

| Функция | Время решения задачи на 2 ядрах | Время решения задачи на 8 ядрах | Время решения задачи на 32 ядрах |
|---|---|---|---|
| Катковника | 8 ч | 115 – 125 мин | 30 – 35 мин |
| Растригина | 7 ч | 100 – 115 мин | 30 – 35 мин |
| Griewank | 10 ч | 150 – 170 мин | 40 – 45 мин |

Одним из показательных примеров применения ИК DISCOMP являются задачи обращения дискретных функций. Разработанный средствами ИК DISCOMP пакет D-SAT [8] реализует технологию крупноблочного распараллеливания SAT-задач. С применением этого пакета был успешно проведен распределенный криптоанализ генераторов двоичных шифров: Гиффорда, суммирующего и порогового. Вычислительные эксперименты проводились в распределенной ВС, включающей 20 двухпроцессорных узлов (всего 160 процессорных ядер). В табл. 2 приведены сравнительные результаты криптоанализа, выполненного на одном ядре и в распределенной ВС как с применением механизмов устранения вычислительной избыточности, так и без них.

Таблица 2

**Время решения SAT-задач**

| SAT-задача криптоанализа | Время решения SAT-задачи на одном вычислительном ядре | Время решения SAT-задачи в ВС из 160 ядер | Время решения SAT-задачи в ВС из 160 ядер с устранением вычислительной избыточности |
|---|---|---|---|
| Генератор Гиффорда | 14 – 30 ч | 1 – 2 ч | 30 – 60 мин |
| Суммирующий генератор | 20 мин – 2 ч | 10 – 30 мин | 2 – 6 мин |
| Пороговый генератор | ≥ 3 суток | 30 – 120 мин | 6 – 10 мин |

## ЗАКЛЮЧЕНИЕ

Представленный в данной работе подход к организации параллельных и распределенных пакетов прикладных программ допускает естественное развитие и обобщение для базовых технологий параллельных и распределенных вычислений и новых классов фундаментальных и прикладных исследовательских задач. Архитектура и принципы работы рассмотренных инструментальных средств обеспечивают широкий спектр функциональных возможностей по разработке и применению таких пакетов для вычислительных систем различного типа..

## СПИСОК ЛИТЕРАТУРЫ

1. **Тыугу Э. Х., Харф М. Я.** Алгоритмы структурного синтеза программ // Программирование. 1980. № 4. С. 3–13.

2. СПОРА – система программирования с автоматическим синтезом программ / И. О. Бабаев [и др.]. // Применение методов математической логики. Таллин: ИК АН ЭССР, 1983. С. 29–41.

3. **Опарин Г. А.** САТУРН – метасистема для построения пакетов прикладных программ // Разработка пакетов прикладных программ. Новосибирск, Наука, 1982. С. 130–160.

4. **Опарин Г. А., Новопашин А.П.** Булевы модели синтеза параллельных планов решения вычислительных задач // Вестник НГУ. Серия: Информационные технологии. 2008. Т. 6. Вып. 1. С. 53–59.

5. **Опарин Г. А., Феоктистов А. Г., Горский С. А.** Язык описания модели предметной области в пакетах прикладных программ // Программные продукты и системы. 2011. № 1.

6. PVM: Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing / G. Geist [et al.] // MIT Press, Cambridge, MA, USA, 1995. 273 p.

7. **Опарин Г. А., Феоктистов А. Г., Сидоров И. А.** Технология организации распределенных вычислений в инструментальном комплексе DISCOMP // Современные технологии. Системный анализ. Моделирование. 2009. № 2. С. 175–180.

8. **Заикин О. С., Семенов А. А.** Технология крупноблочного параллелизма в SAT-задачах // Проблемы управления. 2008. № 1. С. 43–50.

## ОБ АВТОРАХ

**Новопашин Алексей Петрович**, зав. лаб. параллельных и распределенных вычислительных систем ИДСТУ СО РАН, канд. техн. наук.

**Сидоров Иван Александрович**, мл. науч. сотр. той же лаб., канд. техн. наук.

**Горский Сергей Алексеевич**, той же лаб., канд. техн. наук.