УДК 519.685:681.3.068

ЯЗЫК ПАРСЕК: ПРОГРАММИРОВАНИЕ ГЛОБАЛЬНО РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ В МОДЕЛИ ИСЧИСЛЕНИЯ ДРЕВОВИДНЫХ СТРУКТУР

Ю. С. Затуливетер, А. В. Топорищев

Институт проблем управления им. В.А. Трапезникова, г. Москва

Для языка программирования ПАРСЕК, построенного в модели исчисления древовидных структур, разработаны функции управления глобально распределенными вычислениями. Рассмотрены особенности программирования в математически однородном поле компьютерной информации.

ВВЕДЕНИЕ

Современные подходы к организации обработки и хранения распределенных данных в ресурсах глобальной Сети представлены grid-технологиями [1, 2], которые сформировались и апробировались в ходе решения сложных задач обработки больших объемов экспериментальных данных, получаемых в рамках крупномасштабных научных проектов ядерной физики, биологии и др. За неполное десятилетие испытаний из нескольких независимых подходов и систем наибольшее распространение получила инструментальная система Globus Toolkit (версии GT1, GT2) [3].

Актуальность проблематики распределенных вычислений в ресурсах Сети выходит далеко за рамки крупномасштабных научных проектов. В связи с особым значением задач консолидации в ресурсах Сети корпоративных информационных пространств grid-технологии переходят в фазу коммерциализации. В 2003 г. вышла третья версия Globus Toolkit (GT3), которая разрабатывалась в интересах бизнес-приложений при участии ведущих компьютерных компаний [3]. Перспективы быстрого развития grid-технологий связываются с растущей актуальностью проблем функциональной интеграции Сети для разнообразных задач в крупных корпорациях и ведомствах (http://www.osp.ru/nets/2003/15/000_5.htm).

Между бизнес-приложениями и научными задачами, требующими распределенной обработки/хранения данных, имеется существенное отличие. Научные проекты – в большей своей части целевые и разовые. Средства распределенной обработки/хранения данных стационарно структурируются под ту или иную фундаментальную проблему и после завершения экспериментов утрачивают актуальность. В коммерческих применениях приложения существенно более динамичны и по структурам, и по требованиям реагирования и адаптации в реальном времени в ответ на быстрые изменения внешней среды. Особое значение для бизнес-приложений имеют простота, доступность средств программирования с глобальной интеграцией распределенных данных, программ и систем не только со сложными, но и существенно нестационарными — быстро меняющимися — структурами.

"Появление масштабных инфраструктурных проектов обусловило переход к новой фазе развития программного обеспечения grid. Сыгравший свою роль Globus Toolkit перестает быть единственным доступным средством создания grid... Новое поколение программного обеспечения grid развивается в форме платформ... В связи с появлением нескольких альтернативных платформ на передний план выходит задача обеспечения интероперабельности grid разных виртуальных организаций, использующих различные исполнительные среды. Уже имеющийся опыт... показывает, что этого недостаточно: требуются определение базового состава сервисов и *унификация их семантики*" [3].

Многоплатформенность коммерческого grid-инструментария обостряет и без того крайне болезненную проблему неоднородности глобального компьютерного информационного пространства. Разрозненность форм представления и способов обработки компьютерной информации обусловливает один из фундаментальных факторов [4] торможения интеграционных процессов в глобальной компьютерной среде. Причины его появления уходят корнями в неадекватность классической компьютерной аксиоматики в модели фон Неймана [5] требованиям глобальной интеграции данных, программ, процессов и систем [4, 6—8].

Изначальная несовместимость различных платформ grid-технологий показывает, что общий подход, который бы исчерпывающим образом ответил на острые вопросы функциональной интеграции глобального информационного пространства в интересах сверхдинамично развивающейся социальной среды, в том числе

бизнес-среды, все еще не сложился. Компьютерная индустрия продолжает опираться на классические постулаты универсально программируемых вычислений, изначально предложенных для автономных компьютеров [5]. По этой причине ей до сих пор не удается выявить и устранить фундаментальные препятствия на пути к широкодоступным средствам программирования и интеграции глобально распределенных вычислений [6]. В эпоху глобальной Сети лидеры индустрии программирования под новыми флагами корпоративной консолидации глобального информационного пространства продолжают по старинке осуществлять политику продвижения своих платформ, каждая из которых заведомо не пригодна к полномасштабной функциональной интеграции компьютерной среды, поскольку в своих логических основаниях опирается на постулаты автономного компьютера.

При этом все еще отсутствует единое стратегическое понимание главной общей задачи grid-технологий. Весьма показательно мнение руководителя группы баз данных CERN Джейми Шиерса — одного из самых опытных специалистов в сфере применения grid к сложным задачам. На вопрос "Что, по Вашему мнению, может стать "убойным" приложением для появления действительно глобальной, всепроникающей grid-среды?" (http://www.osp.ru/os/2004/12/034.htm) он ответил: "Я много думаю над этим. Крайне сложно предсказать заранее. Возможно, это приложение, которое сейчас повсеместно используется, но мы просто не видим его потенциала. Но это должно быть нечто, что действительно изменит нашу жизнь, как Internet, который мы используем всегда и везде. *Это не может быть задачей из какой-либо частной области, это должно быть приложение, которое реально повлияет на стиль нашей повседневной работы.* Честно говоря, пока я такого приложения не вижу. Видимо, нам еще предстоит его найти". Курсивом мы выделили ключевую, как нам представляется, мысль, указывающую правильное направление поиска такой задачи.

В работах [7, 8] сделана попытка дать максимально общее определение задачи, которая составит единое представление о целях и особенностях глобально распределенной обработки данных. Это задача кибернетизации в математически однородном поле компьютерной информации социосистемы в целом. Все решаемые с помощью глобальной Сети задачи (экономика, госуправление, здравоохранение, природопользование, наука и другие) так или иначе сводятся к процессам компьютерной кибернетизации социальной среды через все ее институты самообеспечения, саморегулирования и самоорганизации. В общем случае это непрерывный спектр компьютерных задач установления системных балансов, направленных на обеспечение управляемого развития ("sustainable development") человечества[1].

Показано [7, 8], что для решения общей задачи компьютерной кибернетизации социосистемы требуется наличие адекватного компьютерного инструментария для программирования глобально распределенных вычислений и процессов управления. К ключевым требованиям следует отнести способность к глобальной интеграции распределенных данных, программ, процессов и систем, а также простоту и широкую доступность средств программирования распределенных задач высокой структурной сложности.

Особо подчеркнем, что для создания инструментария полномасштабного решения и интеграции глобально распределенных задач в ресурсах Сети требуется переход от классических постулатов универсальных вычислений в автономных компьютерах к новой аксиоматике математически однородного поля компьютерной информации [4, 6—8].

В качестве прототипа, отвечающего этим требованиям, мы рассматриваем язык и систему программирования ПАРСЕК (ParSeq) [9], в которых воплощена аксиоматика математически однородного поля компьютерной информации, семантика которого определяется математически замкнутой моделью исчисления древовидных структур.

Цель построения и развития языка и системы программирования ПАРСЕК — формирование подхода к программированию в аксиоматике математически однородного поля компьютерной информации (программ и данных), которая открывает практически неограниченные возможности глобальной интеграции данных, программ, вычислительных процессов и систем в рамках математически замкнутых формальных моделей.

1. КРАТКОЕ ОПИСАНИЕ ЯЗЫКА ПАРСЕК

Язык ПАРСЕК [9] является процедурной реализацией исчисления древовидных структур — математически замкнутой модели вычислений на множестве вполне структурированной информации. Области применения языка и системы программирования ПАРСЕК — разнообразные задачи высокой структурной и динамической сложности, предназначаемые для компьютерной реализации. Вычислительная среда — ресурсы отдельного компьютера (система реализована на платформе win32), ресурсы локальных сетей и глобальная компьютерная среда (система в стадии разработки).

1.1. Особенности языка

Отметим некоторые важные особенности модели исчисления древовидных структур и языка ПАРСЕК. Отличительное свойство языка ПАРСЕК — сочетание математической замкнутости с процедурностью. Программирование осуществляется посредством замкнутого и полного набора операций с *древовидными и только древовидными структурами*, составляющего суть исчисления древовидных структур. При этом обеспечены следующие ключевые для широкого использования качества:

- система остается универсальной;
- математическая замкнутость исчисления древовидных структур позволила в процедурном исполнении построить автоматическое управление внутренними вычислительными ресурсами компьютера;
- система изолирует программиста от сложностей управления вычислительной средой и позволяет ему решать задачи в формальных математических конструкциях, не связанных с особенностями машинной среды.

Важно отметить, что исчисление древовидных структур вполне допускает расширение автоматического уп-

[1] Концепция "sustainable development" была сформулирована как стратегическая линия развития человечества на Конференции ООН по окружающей среде и развитию (ЮНСЕД) 1992 г., Рио-де-Жанейро.

равления вычислительными ресурсами с внутренних ресурсов одного компьютера до ресурсов внешних компьютеров, связанных между собой сетями. Такое свойство дает формальную основу для полнофункциональной интеграции вычислительных ресурсов локальных сетей и глобальной компьютерной среды в целом с формированием в ней математически однородного поля компьютерной информации.

Первая версия языка и системы программирования ПАРСЕК была построена и испытана на задачах в 1993—1994 гг. С тех пор система ПАРСЕК развивается в трех своих компонентах: системное окружение, библиотеки встроенных функций и классы решаемых задач.

В работе представлены результаты текущего этапа развития системы ПАРСЕК во всех трех компонентах. Оно направлено на расширение сфер применимости системы ПАРСЕК с вычислительных ресурсов отдельного компьютера на распределенные ресурсы сетей.

Описание языка ПАРСЕК приводится по тексту [9].

1.2. Первичные понятия

При описании информационных объектов и систем используются формулировки типа: *A* состоит из *a*, *b* и *c*. В теории множеств этой формулировке соответствует дефиниция $A = \{a, b, c\}$. В языке ПАРСЕК для "раскрытия" объекта используется элементарное дерево в геометрической форме:

```
┬A
├a
├b
└c
```

Дерево состоит из вершин. Вершина характеризуется содержимым вершины и связями с другими вершинами. Имеются три вида связей для каждой вершины:

— deep — связь с подчиненной по уровню вершиной;

— next — связь с нижеследующей вершиной одного уровня;

— prev — связь с предыдущей вершиной.

Пример дерева и анализ его структуры приведен в табл. 1.

Двоичный набор из признаков наличия или отсутствия каждого из трех видов связей (`prev`, `deep`, `next`) образует статус вершины. Статусы вершин (связи дерева) отображаются с помощью псевдографических символов ┬, ├, └, а также вспомогательного символа │, который дополняется в левой части строк, несущих изображение вершин старшего ранга.

Древовидная структура представляет те или иные вполне структурированные информационные объекты. Обработка объекта ведется последовательно по компонентам. Для выделения компонентов, которыми являются вершины, вводится понятие *курсор*.

Курсор — это переменная, которая принимает значение адреса (ссылки, указателя) некоторой вершины. На дереве можно определить один или более одного курсора. Процесс обработки начинается с открытия курсоров и перемещения их по дереву.

С курсором связан базисный набор функций для работы с деревьями. Они разделяются на следующие группы.

Одношаговое перемещение курсора

`deep()` — переход на подчиненную вершину,

форма вызова: `cur2 = deep(cur1)`, где `cur1` — адрес текущей вершины, `cur2` — адрес вершины, подчиненной текущей;

`next()` — переход на нижнюю вершину того же ранга,

форма вызова: `cur2 = next(cur1);`

`prev()` — переход на предыдущую вершину,

форма вызова: `cur2 = prev(cur1).`

Функции анализа и замены содержимого

`cont()` — прочитать содержимое вершины,

форма вызова: `c = cont(cur)` или `c = {cur}`, где `c` — содержимое вершины, `cur` — адрес вершины (курсор);

`re_cont()` — занесение нового содержимого вершины,

форма вызова: `re_cont(cur, NC)` или `{cur} = NC`, где `cur` — адрес вершины, `NC` — новое содержимое.

Создание и уничтожение вершин

`node_create()` — создать вершину,

форма вызова: `cur = node_create(NC)`, где `cur` — переменная (с произвольным именем), которая является вновь создаваемым курсором, указывающим на созданную вершину, `NC` — новое содержимое, каждая создаваемая вершина является изолированной;

`node_del()` — уничтожить вершину,

форма вызова: `node_del(cur)`, где `cur` — текущий курсор.

Редактирование связей

`link_next( , )` — связать две вершины на одном уровне;

`link_deep( , )` — связать две вершины так, чтобы вторая стала подчиненной первой;

`link_del( , )` — разорвать связь между вершинами,

формы вызова: `link_next(cur1, cur2)`, `link_deep(cur1, cur2)`, `link_del(cur1, cur2)`, где `cur1` — адрес первой вершины, `cur2` — адрес второй вершины.

1.3. Структура программы

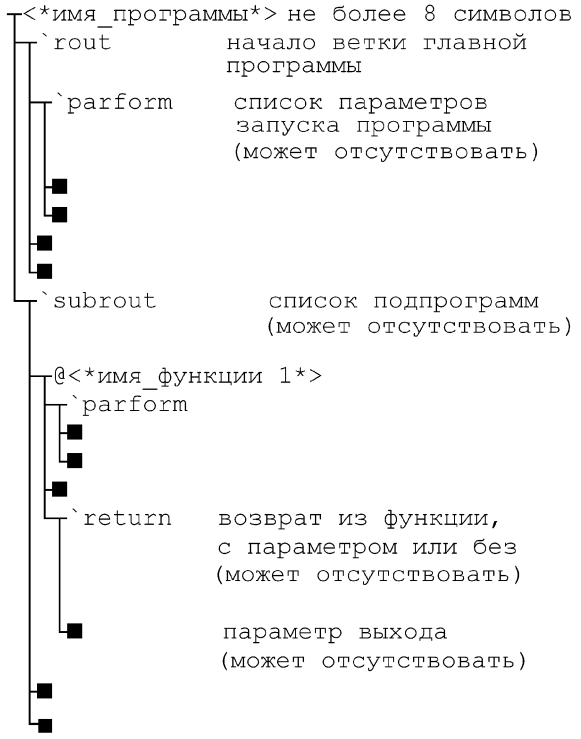
Структура ПАРСЕК-программ вполне классическая и типична для процедурного программирования.

Таблица 1

Определение структуры двоичного дерева

Дерево	Номер вершины	Содержимое вершины	Связь prev	Связь deep	Связь next
┬OBJ	1	OBJ	—	С вершиной 2	—
├A	2	A	С вершиной 1	С вершиной 3	С вершиной 5
├a1	3	a1	С вершиной 2	—	С вершиной 4
└a2	4	a2	С вершиной 3	—	—
└B	5	B	С вершиной 2	—	—

Особенность лишь в том, что тексты программ представляются в виде двоичного дерева с ключевыми словами:



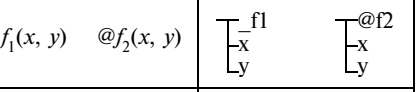
Символьные строки в правой части вершины дерева после первого пробела образуют комментарий (например: строка “начало ветки главной программы”).

Условные обозначения:

- ■ — поддерево или лист;
- ключевые слова в языке ПАРСЕК начинаются с символа “`” (обратный апостроф), т. е. <*ключевое_слово*> = `<*идентификатор*>, с помощью ключевых слов строятся все конструкции языка;
- идентификатор встроенных функций начинается с символа “_”, пользовательских – с символа “@”, т. е. <*встроенная_функция*> = _<*идентификатор*>, <*пользовательская_функция*> = @<*идентификатор*>.

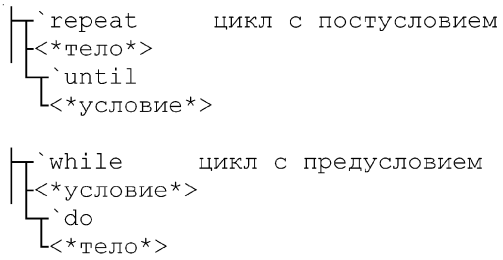
Таблица 2

Представление операций и выражений

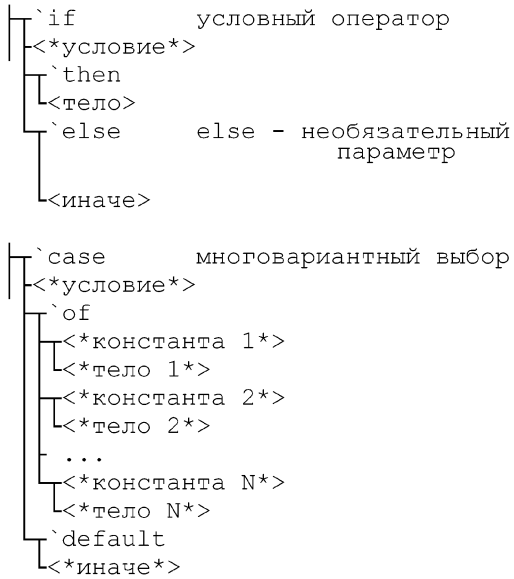
Конструкция	Формальный вид	Древовидная запись
Присваивание	$a = b$	
Операция	$1 + 3$	
Функция	$f_1(x, y)$ $@f_2(x, y)$	
Выражение	$a = 1 + 3$	

Базовые конструкции языка ПАРСЕК вполне традиционны для процедурного программирования:

Циклы



Условия



Операция присваивания, выражения и функции в древовидной записи представлены в табл. 2.

2. АРХИТЕКТУРА РАСПРЕДЕЛЕННОЙ ОБРАБОТКИ

Информационное пространство WWW построено по гипертекстовой модели в сетевой архитектуре “клиент—сервер”. Модель гипертекста не является моделью универсальных вычислений, поэтому изначально не поддерживает распределенные вычисления в ресурсах глобальной компьютерной среды.

Проблемы организации универсально программируемых распределенных вычислений в сетевой архитектуре P2P (Peer-to-Peer), в которых вычислительные и управляющие задачи могли бы реализовываться в виде распределенных параллельных процессов в ресурсах глобальной компьютерной среды, оказываются принципиально более сложными, чем в локальных сетях. Инструментарий параллельного программирования распределенных вычислений в локальных сетях и кластерах (PVM, MPI и др.) не обладает свойством произвольной масштабируемости в глобально распределенных вычислительных ресурсах. В настоящее время индустриальное программирование не имеет регулярных, легко масштабируемых средств программирования глобально распределенных вычислений и процессов, способных обеспечить массовые потребности в решении глобальных задач [6—8]. Глобальные задачи — это новый класс задач, решение которых предполагает согласованное использова-

ние (в автоматических режимах) вычислительных ресурсов компьютеров (в пределе — всех) глобальной Сети.

Стратегическая задача развития глобальной компьютерной среды состоит в структурной и функциональной интеграции ее вычислительных и информационных ресурсов, которая позволит всем компьютерам согласованно работать над одной или многими задачами в составе единого универсально программируемого глобального компьютера [6—8, 10, 11].

В связи с нарастающей глобализацией социальных процессов проблемы универсально программируемой распределенной обработки в глобально распределенных вычислительных ресурсах выходят на высшие приоритеты компьютерной индустрии [7, 8]. Их актуальность увеличивается по двум взаимосвязанным причинам.

Во-первых, из-за глобализации мировой экономики растет спрос на решение глобальных задач обработки и хранения данных в ресурсах Сети, которые отличаются высокой структурной и динамической сложностью, требуют переработки в реальном времени сверхбольших потоков разнообразной информации, обладающий свойствами глобальной связности.

Во-вторых, современные средства индустриального программирования, ориентированные на решение локализуемых задач, оказались не готовыми к требованиям глобальных задач, которые играют все более важную роль в глобальном информационном пространстве.

В данной работе развивается подход [4, 6—12] к реализации глобально распределенных вычислений в сетевой архитектуре P2P в вычислительной модели исчисления древовидных структур, реализованного в системе программирования ПАРСЕК [9]. Для исследования возможностей программирования в ресурсах глобальной компьютерной среды в системе ПАРСЕК строятся функции и среда программной эмуляции распределенных вычислений в ресурсах глобальной компьютерной среды. Архитектура среды такой эмуляции представлена далее.

2.1. Функционально полный набор действий по управлению распределенными ресурсами

В сетевой архитектуре P2P многие пары компьютеров связаны непосредственно между собой так, что в целом образуют *связную* сеть.

На рис. 1 показана отдельная пара связанных между собой компьютеров. Каждый из этих компьютеров может быть связан и с другими компьютерами. Конкретная топология таких сетей для дальнейшего рассмотрения несущественна.

Каждый компьютер сети имеет свой адрес, состоящий из двух компонентов: IP-адрес и IP-порт (в вырожденном случае сеть может быть “организована” на одном компьютере с адресом IP, но с разными портами). В каждом компьютере имеется два вида ресурсов памяти: оперативная память и дисковая память (на рисунке **Memory1**, **Disk1** и **Memory2**, **Disk2**, соответственно).

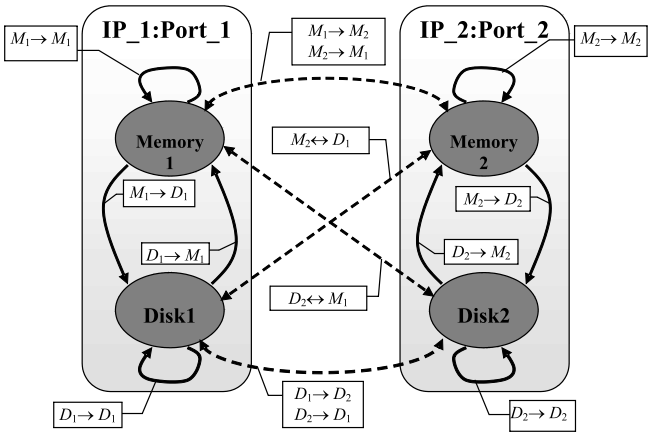


Рис. 1. Произвольная пара взаимосвязанных компьютеров с указанием направления действий (из $X \to$ в Y) между разными видами памяти

Стрелки с надписями показывают направления действий из одной памяти в другую. Оперативная память разных компьютеров обозначена буквами M_1 и M_2, дисковая — буквами D_1 и D_2 (первого и второго компьютеров, соответственно). В табл. 3 перечислены все направления и типы действий, показанные на рис. 1 стрелками.

В дисковой памяти, находящейся под управлением файловой системы, деревья хранятся в виде символьных файлов.

В оперативной памяти деревья представлены в виде внутренних структур. В языке ПАРСЕК функции ввода/вывода (_read_tree/_write_tree) обеспечивают преобразование деревьев, хранящихся в файлах, во внутренние структуры и наоборот.

В табл. 4 приведены все варианты адресации к внутри- и межкомпьютерным ресурсам памяти.

Одна из главных целей расширения системы ПАРСЕК на распределенные вычислительные ресурсы сетей − это достижение независимости программ от расположения древовидных структур в ресурсах всех компьютеров, связанных сетями и являющихся носителями математически однородного поля компьютерной информации.

Таблица 3

Возможные типы действий с использованием всех видов памяти

Внутрикомпьютерные из $X_i \to$ в Y_i, $i = 1, 2$			Межкомпьютерные (через сеть) из $X_i \to$ в Y_j, $i \neq j$, $i = 1, 2$		
$M_i \to M_i$	$D_i \to M_i$, $M_i \to D_i$	$D_i \to D_i$	$M_i \to M_j$	$M_i \to D_j$, $D_i \to M_j$	$D_i \to D_j$
Чтение/запись элементов дерева по адресу, обработка, присваивание	Ввод (read), вывод (write)	Управление файлами (create, del, copy, move, rename,...)	Чтение/запись элементов дерева по адресу, обработка, присваивание	Ввод (read), вывод (write)	Управление файлами (create, del, copy, move, rename,...)

Первый шаг в этом направлении — расширение областей применимости операций, функций и выражений языка ПАРСЕК с внутрикомпьютерных ресурсов оперативной и дисковой памяти на межкомпьютерные. При этом в ходе исполнения перечисленных действий обработка элементов деревьев должна осуществляться с автоматической локализацией мест расположения элементов деревьев, к которым они применяются.

В табл. 5 приведены инвариантные (относительно размещения элементов деревьев в ресурсах разных компьютеров) операторы всех трех видов действий.

Таким образом, рис. 1 и табл. 3—5 дают представление о функционально полном наборе функций управления распределенными ресурсами и принципах расширения действий языка ПАРСЕК с деревьев в ресурсах отдельного компьютера на ресурсы всех связанных компьютеров, являющихся носителями математически однородного поля компьютерной информации.

```
tree_remote_task
  IP_from          IP-адрес компьютера, выславшего задание
  port_from        порт компьютера, выславшего задание
  id_action        идентификатор: встроенная функция, имя exe-модуля
  ip_action        сетевой адрес отдаленного компьютера
  port_action      сетевой порт отдаленного компьютера
  `inp             входные параметры
    place_param_1  место расположения: адрес в ОЗУ, путь на диске
    ip_inp_1       сетевой адрес отдаленного компьютера
    port_inp_1     сетевой порт отдаленного компьютера
    place_param_2  место расположения: адрес в ОЗУ, путь на диске
    ip_inp_2       сетевой адрес отдаленного компьютера
    port_inp_2     сетевой порт отдаленного компьютера
    ■
  `out             выходные параметры
    place_param_1  место расположения: адрес в ОЗУ, путь на диске
    ip_out_1       сетевой адрес отдаленного компьютера
    port_out_1     сетевой порт отдаленного компьютера
    place_param_2  место расположения: адрес в ОЗУ, путь на диске
    ip_out_2       сетевой адрес отдаленного компьютера
    port_out_2     сетевой порт отдаленного компьютера
    ■
  `return          в левую часть выражений: только для _net_call
    place_param    место расположения: адрес в ОЗУ, путь на диске
    ip_return      сетевой адрес отдаленного компьютера
    port_return    сетевой порт отдаленного компьютера
```

Рис. 2. Структура дерева заданий для исполнения на отдаленных компьютерах

2.2. Структура распределенной обработки

Распределенная обработка осуществляется двумя способами управления обработкой деревьев, компоненты которых распределены по разным компьютерам. Первый — последовательная обработка компонентов деревьев, распределенных по разным компьютерам. Второй — параллельная обработка компонентов деревьев, распределенных по разным компьютерам посредством одновременного исполнения на разных компьютерах многих асинхронно взаимодействующих процессов.

Первый способ реализуется последовательным вызовом функции `_net_call(tree_remove_task)`. Дерево `tree_remove_task` — параметр, представляющий задание для исполнения на отдаленном компьютере (рис. 2).

Знак ■ на рис. 2 показывает, что число входных и выходных параметров не фиксируется (от нуля и более). Ветка `` `return `` может использоваться для функции `_net_call()`.

Второй способ реализуется с помощью основной пары функций `net_pid =_net_start(tree_remote_task)` и `_net_wait(net_pid)`, аналогами которых являются хорошо известные функции запуска процессов (`pid = start_process()`) с формированием `pid`- идентификаторов открытых процессов и ожидания их завершения (`wait(pid)`). С их помощью в существующих операционных системах осуществляется асинхронное управление многопроцессной обработкой.

В реализации многопроцессной обработки в системе ПАРСЕК добавлены также сопутствующие функции уп-

Таблица 4

Варианты адресации к ресурсам памяти

Область действия адресации	Виды памяти	Адресация к разным видам памяти	Форматы адресации	Примеры предоставления адресации деревом
Внутрикомпьютерная, $i = 1, 2$	M_i	adr_memory	Слово 32—64 (или более) двоичных разрядов	‘adr $123456789ABCDEF
	D_i	path_disk	Символьная строка-путь	‘path c:/dir_1/file
Межкомпьютерная (через сеть), $j = 1, 2, j \neq$ i	M_i	adr_memory_net	ip:port#adr_memory	‘net ip port ‘adr $123456789ABCDE
	D_j	path_disk_net	ip:port/path_disk	‘net ip port ‘path c:/dir_1/file

Таблица 5

Инвариантные операторы трех видов действий

Виды действий	Инвариантные относительно размещения элементов деревьев операторы (номера компьютеров $i, j = 1, 2$)	Y_j (ресурс для результирующих операндов)	X_i (ресурс для исходных операндов)	Область действия: $i = j$ – внутри компьютера; $i \neq j$ – между компьютерами
Присваивание, обработка	~Sign (Y_j, X_i), ~Exec (Y_j, X_i)	adr_memory	adr_memory	$i = j$
			adr_memory_net	$i \neq j$
		adr_memory_net	adr_memory	$i \neq j$
			adr_memory_net	$i \neq j$
Ввод-вывод деревьев	Y_j = _Read_tree (X_i)	adr_memory	path_disk	$i = j$
			path_disk_net	$i \neq j$
		adr_memory_net	path_disk	$i \neq j$
			path_disk_net	$i \neq j$
	_Write_tree (X_i, Y_j)	path_disk	path_disk	$i = j$
			path_disk_net	$i \neq j$
		path_disk_net	adr_memory	$i \neq j$
			adr_memory_net	$i \neq j$
Управление файлами	_Copy (X_i, Y_j)	path_disk	path_disk	$i = j$
			path_disk_net	$i \neq j$
		path_disk_net	path_disk	$i \neq j$
			path_disk_net	$i \neq j$
	_Move (X_i, Y_j)	path_disk	path_disk	$i = j$
			path_disk_net	$i \neq j$
		path_disk_net	path_disk	$i \neq j$
			path_disk_net	$i \neq j$
	_Rename (X_i, Y_j)	path_disk	path_disk	$i = j$
			path_disk_net	$i \neq j$
		path_disk_net	path_disk	$i \neq j$
			path_disk_net	$i \neq j$
	_Del (X_i)	нет	path_disk	$i = j$
			path_disk_net	$i \neq j$

Примечание. Операторы ~Sign (,), ~Exec (,) понимаются как обобщение всех действий языка ПАРСЕК, связанных с обработкой деревьев в оперативной памяти.

равления асинхронными процессами — уничтожение процесса, приостановка, диагностика текущих состояний, но в данной работе они не обсуждаются.

3. ПРИНЦИПЫ РЕАЛИЗАЦИИ РАСПРЕДЕЛЕННОЙ ОБРАБОТКИ

Распределенная обработка состоит из двух стадий: конфигурирования системы и исполнения заданий.

На стадии конфигурирования на всех компьютерах, связанных в сеть распределенной обработки, запускаются ОЗУ-резидентные модули программы приемника данных. Они обеспечивают прием заданий через указанный в параметрах запуска сетевой порт, а также возврат результата.

Стадия исполнения каждого задания состоит из следующих этапов:

- подготовка (оформление) задания (в формате, представленном на рис. 2) и вызов либо функцией `_net_call`, либо `_net_start`;
- доставка задания на отдаленный компьютер;
- исполнение задания на отдаленном компьютере;

- доставка результатов к местам, указанным в задании на обработку.

Реализация стадии исполнения подробнее рассматривается ниже.

3.1. Резидентные процессы приема заданий

На каждой машине сети, вовлеченной в процесс распределенной обработки, резидентно работают, как минимум, два процесса: приемник заданий (PS_Receiver) и интерпретатор команд языка ПАРСЕК (PSi_net).

Приемники заданий обрабатывают вычислительные запросы. Они связаны с одним или несколькими системными серверами управления доступными вычислительными ресурсами, которые следят за наличием свободных компьютеров. На стадии конфигурирования распределенной системы приемник сразу после запуска соединяется с одним из системных серверов и передает ему доступную для распределенной системы обработки конфигурацию компьютера, на котором он запущен, открывая, тем самым, доступ к выделенным вычислительным ресурсам своего компьютера. Вместе с этим приемник инициализирует системные переменные и буферы ОЗУ, необходимые ему для приема и запуска заданий, а также запускает интерпретатор языка ПАРСЕК.

При получении заданий от отдаленных компьютеров приемник определяет тип задания и передает его на обработку в интерпретатор. Компьютеру, инициировавшему выполнение задания, возвращается сообщение о его завершении и, если указано в задании, результат обработки.

Интерпретатор служит для обработки ПАРСЕК-команд, передаваемых ему приемником в задании. Задания передаются интерпретатору через буфер общей памяти с зарезервированным именем PS_RCV_SHARED_NAME. Передача данных защищена мьютексом с зарезервированным именем PS_RCV_SHARED_MUTEX и осуществляется по событию с зарезервированным именем PS_RCV_SHARED_EVENT.

3.2. Функция передачи данных из пользовательских процессов

Передача данных или заданий отдаленному компьютеру производится следующей функцией: `kod_z =_net_send(ip, port, string, len_string)`, где `ip` — IP-адрес отдаленного компьютера, в который передаются данные, `port` — IP-порт отдаленного компьютера, `string` — указатель на передаваемые данные в виде двоичной строки, `len_string` — размер передаваемой строки данных, `kod_z` — код завершения работы функции (0 — во время передачи произошла ошибка, 1 — в буфере `string` находится результат обработки отправленных данных, 2 — завершение без возвращения результата).

В общем случае строка `string` произвольного размера. В текущей версии размер строки не превышает размера буфера “общей памяти” (128 K) с зарезервированным именем PS_RCV_SHARED_NAME. Для отправки строки функцией передатчика буфер “общей памяти”, аналогичный тому, который открывается в приемнике, не требуется. Передающая программа не требует реализации в виде резидентного процесса (двух процессов, между которыми нужен буфер обмена данными, просто нет).

С помощью передающей функции `_net_send()` реализуются функции `_net_call()` и `net_start()`.

Вызов функции последовательного запуска действий на отдаленных компьютерах осуществляется в пользовательских программах и имеет следующий вид: `rez =_net_call(tree_remote_task)`, где `rez` — результат: либо дальняя (сетевая) адресная ссылка, либо двоичная строка, либо дерево (курсор корневой вершины), `tree_remote_task` — дерево подготавливаемого задания (см. рис. 2).

Запуск и синхронизация (ожидание завершения) параллельных процессов на отдаленных компьютерах осуществляется в пользовательских программах следующим образом.

Запуск:

```
net_pid_1=_net_start(tree_remote_task_1)
net_pid_2=_net_start(tree_remote_task_2)
...
```

Ожидание завершения:

```
_net_wait(net_pid_1)
_net_wait(net_pid_2)
...
```

Эти операторы реализуются средствами ПАРСЕК с помощью функции `_net_send(ip, port, string_task, len_string_task)`.

Здесь `net_pid_1`, `net_pid_2`, ... — *уникальные* идентификаторы процессов открываемых на отдаленных компьютерах, `tree_remote_task_1`, `tree_remote_task_2`, ... — деревья с указанием обратных адресов, действий, операндов их сетевых адресов (см. рис. 2).

3.3. Прием-исполнение заданий

На передающей стороне в пользовательских программах осуществляется строго последовательное обращение к передающей программе из пользовательского процесса, что обеспечивается вызовом функции `_net_send()` и завершением ее работы сразу после завершения передачи. Резидентные процессы на каждом принимающем компьютере работают следующим образом.

Процесс приемника PS_receiver создает (при запуске) или уничтожает (при закрытии) буфер “общей памяти” с зарезервированным именем PS_RCV_SHARED_NAME, создает (при запуске) или уничтожает (при закрытии) событие с именем PS_RCV_SHARED_EVENT, принимает из входного порта строку с данными или заданием, предает ее через буфер PS_RCV_SHARED_NAME в процесс интерпретатора PSi_net. После декодирования и исполнения задания интерпретатор возвращает сигнал завершения. Сигнал завершения может нести с собой результат обработки — строку с адресной ссылкой или с деревом результата, которые возвращаются из приемника передающей функции в компьютер-источник.

Процесс интерпретатора PSi_net обрабатывает деревья, передаваемые ему в строке, принятой приемником, через буфер “общей памяти”. Связь приемника PS_receiver с интерпретатором PSi_net через буфер “общей памяти” с зарезервированным именем PS_RCV_SHARED_NAME обеспечивает:

- передачу принимаемой строки через буфер в процесс PSi_net:
  - — целиком (для строк, размером не больших буфера);
  - — последовательно по частям (для строк, размером больших буфера);
- возврат процессу — приемнику PS_receiver строки результата:
  - — целиком (для строк, размером не больших буфера);
  - — последовательно по частям (для строк, размером больших буфера).

Синхронизация и корректность передачи данных через буфер обеспечивается событием с именем PS_RCV_SHARED_EVENT.

ЗАКЛЮЧЕНИЕ

Перспективные области применения предложенного подхода — решение задач глобально распределенной обработки информации, связанных с кибернетизацией социосистемы в целом [7, 8, 10, 11]. Имеются области, в которых создание высокоэффективных, безопасных и свободно масштабируемых глобально распределенных систем управления актуальны уже сейчас. Это — разнообразные сферы производства — потребления, а также госуправления (создание общегосударственных систем мониторинга и управления в сферах здравоохранения, социальной защиты, образования, науки и др. [13]). Полномасштабное решение многих глобально распределенных задач управления пока невозможно из-за отсутствия адекватного компьютерного инструментария.

Для глобализации функционального пространства компьютерной среды требуется единая модель универсально программируемых распределенных вычислений, сочетающая в себе следующие качества, распространимые на глобальную компьютерную среду:

- структурная сложность форм компьютерного представления данных и программ должна быть минимальной;
- модель не должна допускать стихийного воспроизводства в компьютерной среде разнородных, плохо совместимых форм представления данных, программ, процессов и систем;
- модель должна обеспечить простоту программирования, целостность и свободную масштабируемость глобально распределенной обработки информации.

Из-за отсутствия модели, отвечающей перечисленным требованиям, компьютерная среда оказалась неготовой к общему решению проблем глобализации распределенных вычислений.

Исчисление древовидных структур, реализованное в системе программирования ПАРСЕК, может рассматриваться как основа для построения практически значимой реализации универсальной модели распределенной обработки глобально распределенных данных, которая открывает пути к практическому воплощению перечисленных свойств.

Данная работа является одним из первых практических шагов к созданию простого и широко доступного компьютерного инструментария для воплощения в компьютерной среде глобально распределенных процессов управления, которые обеспечат практическое воплощение структурно сложных задач управления, решаемых с помощью математического аппарата, развивающего идеи графодинамики [14].

ЛИТЕРАТУРА

1. *Foster I., Kesselman C., Tuecke S.* The Anatomy of the Grid: Enabling Scalable Virtual Organizations // International Journal of High Performance Computing Applications. — 2001. — 15 (3) (www.globus.org/research/papers/anatomy.pdf).
2. *The Physiology* of the Grid: An Open Grid Services Architecture for Distributed Systems Integration / I. Foster, C. Kesselman, J. Nick, S. Tuecke (http://www.globus.org/research/papers/ogsa.pdf).
3. *Коваленко В., Корягин Д.* Организация grid: есть ли альтернативы? // Открытые системы. — 2004. — № 12 (http://www.osp.ru/os/2004/12/034.htm).
4. *Затуливетер Ю. С.* Информация и эволюционное моделирование // Труды междунар. конф. "Идентификация систем и задачи управления" SICPRO`2000, Москва, 26—28 сентября 2000 г. Ин-т проблем управления РАН. — М., 2000. — С. 1529—1573 (http://zvt.hotbox.ru/1529_.htm).
5. *Беркс А., Голдстейн Г., Нейман Дж.* Предварительное рассмотрение логической конструкции электронного вычислительного устройства // Кибернетический сборник. — 1964. — Вып. 9. — С. 7—67.
6. *Затуливетер Ю. С.* На пути к глобальному программированию // Открытые системы. — 2003. — № 3. — С. 46, 47 (http://www.osp.ru/os/2003/03/046.htm).
7. *Затуливетер Ю. С.* Проблемы глобализации парадигмы управления в математически однородном поле компьютерной информации. Ч. 1. Кибернетизация социосистемы // Проблемы управления. — 2005. — № 1. — С. 2—10 (http://www.ipu.ru/period/pu/docs/zatul.pdf).
8. *Затуливетер Ю. С.* Проблемы глобализации парадигмы управления в математически однородном поле компьютерной информации. Ч. 2. К единому функциональному пространству // Там же. — 2005. — № 2. С. 12—23.
9. *Затуливетер Ю. С., Халатян Т. Г.* ПАРСЕК — язык компьютерного исчисления древовидных структур с открытой интерпретацией. Стендовый вариант системы программирования. — М.: Ин-т проблем управления РАН, 1997. — 71 с.
10. *Затуливетер Ю. С.* Предпосылки появления глобального управляющего компьютера // Датчики и системы. — 2005. — № 4. С. 53—64.
11. *Затуливетер Ю. С.* Пути реализации глобального управляющего компьютера // Там же. — № 5. — С. 25—33.
12. *Затуливетер Ю. С.* Компьютерная информация в модели исчисления древовидных структур // Труды второй междунар. конф. "Идентификация систем и задачи управления" SICPRO`2003, Москва, 29—31 января 2003 г. / Ин-т проблем управления РАН. — М., 2003. — С. 790—858.
13. *Концепция* использования информационных технологий в деятельности федеральных органов государственной власти до 2010 года (http://www.it-gov.ru/site.shtml?id=76).
14. *Динамический* подход к анализу структур, описываемых графами (Основы графодинамики). I, II / М. А. Айзерман, Л. А. Гусев, С. В. Петров, И. М. Смирнова // Автоматика и телемеханика. — 1977. — № 7. — С. 135—151; № 9. — С. 123—136.

☎ *(095) 334-92-09*

E-mail: zvt@ipu.rssi.ru

E-mail: Artur.Toporischev@idm.ru ❑