Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
им. М.В.Келдыша
Российской Академии Наук

А.Г.Рубин, В.К.Смирнов, В.П.Тульский

ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС ДЛЯ ПРИКЛАДНЫХ ЗАДАЧ

Москва
2000

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проекты 99-01-00842 и 00-07-90353).

Пользовательский интерфейс для прикладных задач

А.Г.Рубин, В.К.Смирнов, В.П.Тульский

АННОТАЦИЯ

В работе обсуждаются проблемы разработки и реализации человеко-машинного интерфейса. Описывается интерфейс пользователя, предназначенный для облегчения подготовки и исполнения прикладных программ на параллельных вычислительных системах в среде Интернет.

Ключевые слова: интерфейс пользователя, человеко-машинный интерфейс, параллельные вычисления, Интернет, язык Норма.

This work was supported by Russian Foundation for Fundamental Investigations (Projects 99-01-00842 and 00-07-90353).

User Interface for Applied Problems

A.G.Rubin, V.K.Smirnov, V.P.Tulski

The preprint of the Keldysh Institute of Applied Mathematics,
Russian Academy of Sciences

ABSTRACT

The problems of development and implementation of human-computer interface are discussed. User interface project for application program design and execution on parallel computers in Internet environment is described.

Key words: user interface, computer-human interface (CHI), parallel computation, Internet, Norma.

Содержание

1. Введение 4
2. Параллельные вычисления 4
2.1. Параллельные машины 5
2.2. Параллельное программирование 6
2.3. Удаленный доступ 8
3. Язык Норма 8
4. Вопросы безопасности 10
5. Особенности системы 11
6. Структура интерфейса 13
6.1. Организация интерфейса 13
6.2. Главное окно интерфейса 14
6.3. Архив 18
6.4. Локальные и удаленные ресурсы 18
6.5. Паспорт задачи пользователя 18
7. Работа с интерфейсом 19
8. Средства реализации интерфейса 22
9. Заключение 27
Литература 28

1. Введение

Проект предусматривает создание мобильного интерфейса пользователя для решения сложных научно-технических задач. Интерфейс позволит осуществлять удобный доступ с любого, в частности, переносимого компьютера к удаленным ЭВМ (в том числе и параллельным суперкомпьютерам) практически из любой точки, где имеется телефон (обычный, сотовый или спутниковый). Интерфейс обладает средствами динамического управления ходом решения сложной научно-технической задачи.

Интерес к проблемам пользовательского интерфейса (ПИ) возник у авторов еще в 1992-1994 гг. и был тогда связан с возможностью применения системы символьной обработки (на базе языков Рефал и Лисп) в качестве инструментального средства для его эффективной реализации [1,2]. Позднее исследовались вопросы, связанные с выработкой общих принципов и средств построения интеллектуальных систем взаимодействия пользователя с прикладными программами для сложных предметных областей [5,7].

В течение 1995-1997 гг. в рамках проекта 67-94 проводились работы по созданию ПИ для пакета программ, предназначенного для нейтронно-физического расчета ядерного реактора. На основе опытной эксплуатации системы, созданной в рамках этого проекта, была предложена архитектура сетевого варианта интерфейса, который был обобщен и реализован в проекте РФФИ 98-07-90001 в 1998-1999 гг. Реализация проектов 67-94 и 98-07-90001 позволила выйти на использование современных аппаратных и программных средств (в частности, сетевых возможностей и средств визуального программирования), создать базовую систему и разработать предложения по сетевому варианту интерфейса [3-4,6-9]. В настоящем проекте, поддерживаемом Грантами РФФИ 99-01-00842 и 00-07-90353, предполагается дальнейшее развитие созданной архитектуры и использование разработанных для нее методов и механизмов.

2. Параллельные вычисления

Со времени создания и использования первых вычислительных машин и до настоящего времени проблема производительности была и остается важнейшей проблемой вычислительной техники. Решается она в основном двумя путями. Во-первых, постоянным увеличением быстродействия элементов вычислительных машин. Второй путь – это использование параллелизма в работе компьютеров. Если

первый путь может использоваться без заметных изменений архитектуры машин, то второй, как правило, связан с применением специальных архитектурных решений, и, следовательно, новых методов создания программ для параллельных машин. Поэтому введение параллелизма в вычислительные машины связано как с созданием новых аппаратных средств, так и с разработкой новых средств программирования.

2.1. Параллельные машины

Можно выделить три уровня реализации параллелизма в современных вычислительных средствах: суперкомпьютеры, вычислительные кластеры и метакомпьютинг.

Суперкомпьютеры начали создаваться уже на ранних этапах развития вычислительной техники. Однако они всегда были дорогостоящим средством, а поэтому доступным лишь небольшому числу крупных организаций. Суперкомпьютер требует не только больших начальных затрат - дорого обходится и его эксплуатация. Не прост он и в использовании.

Сейчас наибольшее распространение имеют векторно-параллельные (Cray SVI, NEC SX-5) и массивно-параллельные (Cray T3E,Origin 2000) суперкомпьютеры. В современном суперкомпьютерном мире доминируют массивно-параллельные системы. В качестве производителя лидирует фирма SGI/Cray. Производительность таких систем перевалила за 1 TFLOPS, число процессоров достигает 5 - 9 тысяч. В этих системах используются стандартные массовые процессоры, такие как Pentium или Alpha с частотой 200 - 600 MHz. В России массивно-параллельные машины представлены системами МВС-100 (на базе процессора i860) и МВС-1000 (на базе процессора Alpha).

В последнее время все большую популярность получают более "демократичные" параллельные системы - вычислительные кластеры и метакомпьютинг.

В рамках проекта Beowulf, зародившегося в научно-космическом центре NASA, были собраны 16-процессорный кластер на процессорах 486DX4/100MHz, а позднее более мощный кластер HIVE (Highly-parallel Integrated Virtual Environment), содержащий 64 узла по 2 процессора Pentium Pro/200MHz. В Лос-Аламосской национальной лаборатории был построен суперкомпьютер Avalon, представляющий собой Linux-кластер на базе 140 процессоров DEC Alpha/533MHz. В настоящее время он широко используется в астрофизических, молекулярных и других научных вычислениях. В 1998 году создатели Avalon заслужили премию по показателю

цена/производительность. Среди других достижений в этой области - в NASA создается крупнейший в мире параллельный кластер из рабочих станций на базе Windows NT, а также устанавливается 128-процессорный Linux-кластер, свой Linux-кластер с суперкомпьютерной производительностью продемонстрировала фирма IBM, кластер фирмы Compaq побил рекорд производительности в сортировке. Наряду с Fast Ethernet для взаимодействия узлов кластера стали использоваться более производительные каналы связи, в частности, SCI (Scalable Coherent Interface) со скоростью передачи данных порядка 400 Mbit/sec. В качестве примера такой системы в России можно привести кластер в НИВЦ МГУ (г.Москва).

Резервом существенного снижения стоимости параллельных вычислений является перспектива использования сетей ЭВМ в качестве суперкомпьютеров. Эта идея, получившая наименование метакомпьютинг, предполагает использование обычных компьютеров, подключенных к сети, для образования параллельной вычислительной системы большой мощности. Такая система может создаваться динамически, и в нее могут включаться компьютеры разных платформ.

Метакомпьютинг обещает в будущем дать вычислительные мощности, необходимые для работы систем, обеспечивающих национальную безопасность, моделирующих экологические задачи, природные катаклизмы, предсказывающих погоду. Гранты под подобные проекты выделяют такие солидные организации, как Национальный научный фонд, NASA, DARPA и Министерство энергетики США. Масштабы метакомпьютерной деятельности можно представить на примере известного проекта GUSTO. К началу 1998 года в него входило 17 узлов (США, Гавайи, Швеция, Германия), 330 компьютеров с 3600 процессорами, имеющих суммарную производительность 2 TFLOPS.

2.2. Параллельное программирование

Важное значение для параллельных вычислений имеют инструментальные средства создания параллельных программ.

Одной из самых заметных работ в этом направлении стал проект PVM (Parallel Virtual Machine), в котором участвовали Ок-Риджская Национальная Лаборатория (Oak Ridge National Laboratory) и ряд университетов США. Аналогичные проекты велись в Аргоннской Национальной Лаборатории (система P4), Йельском университете (система Linda) и ряде фирм США - Para Soft Corp. (Express) и др. Сейчас PVM перенесена на большинство известных платформ от ПК и рабочих станций до параллельных суперкомпьютеров, а также на многие ОС (в том числе Unix и Windows). С использованием PVM созданы библиотеки линейной алгебры

(например, SCALAPACK) и численных методов, параллельный Fortran (FORGE90), расширения языков Java (JPVM) и Perl (Perl-PVM), средства параллельного программирования и отладки, средства визуального программирования и пользовательский интерфейс.

Крупнейшим достижением стала стандартизация интерфейса передачи сообщений в проекте MPI (Message Passing Interface). Набор функций этого интерфейса вобрал в себя лучшие черты своих предшественников P4 и PVM. Сейчас MPI является, пожалуй, наиболее распространенным интерфейсом параллельного программирования. Библиотеки MPI реализованы практически на всех современных суперкомпьютерах. Известно несколько распространенных пакетов на основе MPI, например, LAM MPI и MPICH, разработанный в Аргоннской Национальной Лаборатории. Об актуальности технологий MPI и PVM говорят регулярно проводимые международные конференции EuroPVM/MPI и HiPer.

Уровень PVM/MPI представляет собой значительный шаг вперед на пути создания инструментальных средств параллельного программирования. Однако их использование требует больших усилий, затрачиваемых на отладку программ. Поэтому на базе PVM/MPI в основном стали базироваться библиотеки и системы поддержки параллельных языков программирования.

Реальным инструментом пользователя стали языки программирования, основанные на параллелизме данных. Первый из них, Fortran-D, появился в 1992 г. На смену ему пришел High Performance Fortran (HPF), представляющий собой расширение языка Fortran 90 и требующий от пользователя лишь указания распределения данных по процессорам. Транслятор с этого языка может генерировать обращения к библиотеке передачи сообщений (например, MPI).

Несколько библиотек и систем параллельного программирования разработано и в России, например, mpC (ИСП РАН) и Т-Система (ИПС РАН), НОРМА (ИПМ РАН) и др. В языке программирования mpC - расширении ANSI Си пользователь распределяет не только данные, но и вычисления. Этот язык позволяет избежать трудностей отладки, характерных для библиотек передачи сообщений.

С появлением таких инструментальных средств, как PVM, MPI, параллельных версий языков Си и Фортран, ситуация в области параллельного программирования улучшилась, тем не менее, разработка и отладка программ с параллельными процессами все еще чрезвычайно затруднена. Это основное препятствие для расширения сферы применения сетевых вычислений.

2.3. Удаленный доступ

Ведущие суперкомпьютерные центры мира предоставляют удаленный доступ к своим компьютерам. Ряд крупных вычислительных центров России также предоставляют удаленный доступ к своим вычислительным мощностям через сеть Интернет. К таким центрам относятся:

- МСЦ (Межведомственный Суперкомпьютерный Центр) Миннауки РФ и РАН (г.Москва);
- ИВВиБД (Институт Высокопроизводительных Вычислений и Баз Данных) Миннауки РФ (г. С.Петербург);
- ИОХ (Институт Органической Химии) РАН (г. Москва);
- НИВЦ (Научно-Исследовательский Вычислительный Центр) МГУ (г. Москва).

Возможность удаленного доступа к параллельным машинам разных платформ делает необходимым создание удобных средств общения с ними. Такие средства должны помочь удаленному пользователю:

- в освоении параллельных машин и упрощении работы с параллельными системами разных платформ;
- в использовании параллельных машин как для создания новых параллельных программ, так и перевода на эти машины уже существующих последовательных программ;

Для решения этих задач и предназначается описываемый здесь интерфейс.

3. Язык Норма

В ПИ на этапе подготовки программы можно воспользоваться языком Норма. Норма позволяет не только упростить написание программы, но и автоматизировать процесс ее распараллеливания, генерируя в результате эффективную мобильную параллельную программу. ПИ, естественно, позволяет создавать и программы на Фортране, но в этом случае заботы о распараллеливании алгоритма возлагаются на плечи самого прикладного программиста. Норма же позволяет переложить решение этой задачи на компилятор.

Язык Норма является специализированным языком программирования, способствующим реализации дружественного интерфейса с прикладным пользователем. Норма - это еще один шаг на пути к такому состоянию, когда вычислительная система сможет автоматически создавать программу по

спецификации в конкретной предметной области, а пользователь сможет решать задачу без программирования в традиционном понимании этого термина.

Язык Норма дает возможность пользователю (специалисту-прикладнику) работать в рамках единого подхода, не отвлекаясь на решение вопросов, не относящихся собственно к прикладной области, например, таких как:

- особенности устройства конкретных вычислителей;
- проблемы переноса программ между компьютерами;
- особенности компиляторов;
- особенности языков программирования;
- системы распараллеливания;
- организация системных библиотек и пр.

Значительное изменение архитектуры вычислительных систем требует создания нового программного обеспечения. Традиционные языки программирования здесь мало чем могут помочь, поскольку они были рассчитаны на однопроцессорные машины. Малоэффективными часто оказываются и попытки распараллелить программу на этапе трансляции. Поэтому пользователю имеет смысл программировать сразу параллельно на языках, специально для этого предназначенных. Для этого прикладнику необходимы средства, позволяющие ему в привычных терминах формулировать алгоритм решения задачи, не вникая в детали архитектуры машины, на которой эта задача будет выполняться. Таким образом, становится очевидной необходимость разработки новых специализированных языков, которые отвечали бы всем этим требованиям. Это, конечно, должны быть непроцедурные языки высокого уровня, обеспечивающие дружественный интерфейс с пользователем. Фактически, программирование в этом случае не должно требовать написания программ в традиционном смысле. Язык Норма [10,11] является попыткой создания такого языка.

4. Вопросы безопасности

Одна из важнейших проблем, с которой приходится сталкиваться при организации работы с удаленным компьютером через сеть это безопасность. Традиционные способы работы в сети Интернет используют протокол telnet для запуска программ на удаленном компьютере и протокол ftp для пересылки файлов. Однако регистрация пользователя в telnet и ftp имеет очевидный недостаток,

состоящий в том, что передача имени пользователя и его пароля по сети производится "открытым текстом" (без шифрования), что делает возможным их перехват по пути следования и использование для несанкционированного доступа к компьютеру.

В настоящее время получили широкое распространение криптографические методы защиты информации при работе в сети. Стандартом де-факто стали протоколы SSH и SCP, используемые соответственно для запуска программ и пересылки файлов. Они обычно входят в стандартную поставку Unix-подобных операционных систем, включая Linux или FreeBSD. В предлагаемом варианте ПИ защита информации строится на базе протокола SSH, поддерживаемого на клиентской стороне системой SecureCRT, работающей в ОС Windows 95/98/NT. Эта система позволяет использовать как протокол telnet для незащищенной связи, так и протокол SSH - для защищенной. SecureCRT позволяет работать как в диалоговом режиме, так и из командной строки. В последнем случае система предоставляет возможность пользоваться скриптами для управления взаимодействием компьютеров. Именно этот метод был выбран в качестве базового для реализации интерфейса с удаленной машиной. Выглядит это следующим образом: программа ПИ подготавливает командный файл, обеспечивающий выполнение нужной функции на удаленной машине, пересылает его на удаленную машину и с помощью команды SSH осуществляет его исполнение.

Использование протокола SCP решает проблему безопасности пересылки файлов непосредственно между пользователем-клиентом и сервером, обеспечивая шифрование передаваемой информации.

Для пересылки информации между пользователем-клиентом и удаленным сервером может также применяться промежуточный ftp-сервер. Так, если нужно переслать файл пользователя на удаленный компьютер, сначала командой ftp, исполняемой на клиенте, файл пересылается на этот промежуточный ftp-сервер, а затем с помощью протокола SSH выполняется команда ftp на удаленной машине, которая уже принимает файл с промежуточного ftp-сервера на удаленный компьютер. Аналогично пересылаются файлы в обратном направлении - с удаленной машины на машину клиента: сначала исполняется ftp на удаленной машине (с использованием протокола SSH) для пересылки файла на промежуточный ftp-сервер, а затем команда ftp на клиенте пересылает файл с промежуточного ftp-сервера пользователю.

Метод с промежуточным ftp-сервером имеет тот недостаток, что информация по сети между ним и пользователем-клиентом передается в незашифрованном виде, а предотвращается несанкционированный обмен информацией лишь с удаленной машиной. Сейчас уже имеются варианты ftp, в которых информация, передаваемая по сети, шифруется. Можно рассчитывать, что в будущем использование такого подхода позволит ограничиться при пересылке файлов одной командой ftp.

Если на машине клиента установлен собственный ftp-сервер, то его можно использовать для пересылки файлов вместо промежуточного ftp-сервера. Таким образом при этом решается проблема безопасности.

5. Особенности системы

Функциональные характеристики пользовательского интерфейса отвечают решению следующих задач:

- ускорению процесса разработки и отладки программ для параллельных ЭВМ;
- упрощению переносимости программ с одной вычислительной платформы на другую;
- облегчению эксплуатации сложных программных комплексов;
- обеспечению открытости системы в отношении возможности наращивания как новых технологий и инструментальных средств разработки программ, так и новых архитектур ЭВМ, становящихся доступными для исполнения программных модулей.

ПИ имеет сетевую ориентацию: все компьютеры, доступные через интерфейс, подключаются к сети. Взаимодействие между интерфейсом и компьютерами осуществляется на основе протоколов TCP/IP, что позволит пользователям работать как в локальных, так и в глобальных сетях (например, в Интернет). Использование стандартных сетевых протоколов, позволит адаптировать ПИ практически к любому типу компьютерной сети. Выход интерфейса в сеть возможен как непосредственно через сетевой адаптер, так и через модем и коммутируемую телефонную линию. В последнем случае взаимодействие будет осуществляться через протоколы SLIP/PPP.

В ПИ осуществляется поддержка подготовки программ, отладки и исполнения программных модулей, составляющих прикладную задачу, на следующих платформах:

- на персональных ПК типа IBM PC с ОС Windows 95/98/NT;
- на однопроцессорных ЭВМ с ОС типа Unix, доступных через локальную сеть или глобальную сеть Интернет;
- на параллельных суперкомпьютерах с ОС типа Unix, обладающих средствами защиты и доступных через локальную сеть или сеть Интернет.

Проект предполагает возможность переноса приложений на разные платформы практически без изменений. Для каждой из машин ПИ будет обеспечивать средства подготовки и средства исполнения для языков программирования Си (Си++) и Фортран. ПИ открыт для добавления новых современных языков параллельного программирования, основанных на параллелизме данных и вычислений. ПИ допускает работу с широким кругом научно-технических задач.

ПИ позволит запускать на удаленной машине как отдельные программные модули, так и цепочки программных модулей, связанных по данным. Будучи определенным образом сконфигурирован пользователем, ПИ позволит эффективно управлять имеющимися ресурсами, при этом отдельные модули могут исполняться на различных
машинах сети. Передача с машины на машину необходимых для работы данных будет осуществляться автоматически.

В предлагаемом варианте реализации основных функций ПИ в качестве модельной задачи используется система построения эффективных мобильных параллельных программ Норма (см. п.3), которая разрабатывается специалистами ИПМ им. М.В.Келдыша РАН. Эта система применяется для автоматического перевода вычислительных алгоритмов, записанных на языке Норма, в параллельные Фортран-программы, включающие обращения к библиотекам PVM или MPI (см. ниже), для последующего исполнения на различных параллельных платформах. Язык системы Норма является специализированным декларативным языком для спецификации задач вычислительного характера.

Мобильный интерфейс содержит следующие компоненты:

- PPP-сервер на базе ОС Linux или Windows NT, обеспечивающий выход с ПИ в сеть Интернет через телефонную линию;
- средства взаимодействия ПИ с параллельными ЭВМ как по сетевым линиям связи, так и по телефонным линиям через PPP-сервер;

- средства подготовки программ для параллельных ЭВМ, доступные через ПИ, включая такие современные инструментальные средства автоматического распараллеливания программ и повышения их мобильности, как PVM, MPI, параллельные версии языков Си и Фортран, средства языка непроцедурного программирования Норма и др.;
- унифицированный сетевой архив, предназначенный для хранения компонентов программ, данных и управляющей информации для удаленных машин, на которых решаются прикладные задачи;
- средства управления унифицированным сетевым архивом, обеспечивающие обмен информацией, содержащейся в архиве, между машинами, подключенными к сети, включая обмен промежуточными результатами;
- собственно интерфейс пользователя, обеспечивающий единообразие выполнения различных функций независимо от вычислительной платформы, на которой эти функции исполняются.

6. Структура интерфейса

Ниже кратко описаны особенности реализации интерфейса с точки зрения прикладного пользователя. Работа пользователя в среде ПИ демонстрируется на примере одной реальной задачи (см. п.7).

6.1. Организация интерфейса

В работе широко используются возможности известного оконного интерфейса, принятого в системе Windows 95/98/NT. Известно, что этот интерфейс был задуман как метафора рабочего стола с документами. В рамках упомянутой метафоры выбран некоторый общий изобразительный стиль, а также система конкретных интерфейсных элементов и их совокупностей (их набор, расположение, способ изображения), которыми необходимо овладеть пользователю.

При построении ПИ решалась задача организации естественной среды, обеспечивающей пользователю психологический комфорт. Преследовалась цель реализации простого управления системой. Для этого было необходимо обеспечить баланс между функциональными возможностями программы, возможностями управления ею и изобразительными средствами.

Следует отметить, что ПИ, конечно, осуществляет проверку правильности действий пользователя в ходе реализации сценария выполнения программы. В

случае обнаружения ошибки пользователя на экран ПК выдается соответствующее диагностическое сообщение.

Из перечня стандартных средств - приложений и технологий, предоставляемых ОС Windows, особенно широкое применение в данной реализации ПИ нашла программа Windows Explorer и технология drag&drop. Как известно, указанная технология предполагает возможность обработки файла путем буксировки его иконки на иконку желаемого исполнителя. Порядок действий обычно таков: нажимая на иконку файла, Вы перемещаете ее (удерживая кнопку мыши при ее перемещении) на иконку соответствующего исполнителя и оставляете ее там.

Очередной сеанс работы с ПИ начинается с активизации пиктограммы, именуемой "User Interface", расположенной на рабочем столе ПК пользователя. При этом открываются сразу четыре окна Windows Explorer (с определенными начальными настройками), каждое из которых имеет конкретное функциональное назначение в интерфейсе (см. рис.4).

В дальнейшем в соответствии со сценарием работы пользователя с прикладной программой на экране его ПК в нужный момент будут автоматически появляться и исчезать дополнительные окна. При реализации ПИ возникает проблема визуализации большого объема информации. Важное значение имеет то, сколько объектов попадает в поле видимости и как они структурированы.

Ниже описано назначение основных окон пользовательского интерфейса, а также содержащихся в них интерфейсных элементов - файлов и пиктограмм.

6.2. Главное окно интерфейса

Левое верхнее рабочее окно имеет наименование "Interface" - это главное окно Итерфейса (И). В нем в виде мнемонической схемы размещена вся совокупность значков-пиктограмм, как основных так и сервисных, обеспечивающих управление пользовательским интерфейсом. С каждой из них связан определенный исполнитель. Надписи под пиктограммами характеризуют их функциональное назначение. Таким образом, активизация любой из этих пиктограмм с помощью левой кнопки мыши вызовет ответную реакцию интерфейса. Информационные

пиктограммы в виде стрелок не несут какой-либо функциональной нагрузки, они просто отражают течение основных информационных потоков в процессе работы пользователя с задачей.

Верхний ряд пиктограмм окна И определяет перечень вычислителей-серверов, обслуживаемых интерфейсом. Этот ряд может быть расширен по мере включения в обслуживание новых компьютеров. Выбор конкретного вычислителя осуществляется нажатием курсора у соответствующего значка. Вслед за этим разворачивается основной сценарий работы пользователя с указанным вычислителем.
В настоящее время в интерфейсе реализована работа со следующими вычислителями:

- PC - ПК типа IBM PC с ОС Windows 95/98/NT;
- power.keldysh.ru - однопроцессорная ЭВМ Silicon Graphics с ОС типа Unix;
- gateway.kiam.ru - 29-процессорный вычислитель МВС-1000 с ОС типа Unix на базе рабочих станций Alpha, находящийся в ИПМ РАН;
- alpha.kiam.ru - 64-процессорный вычислитель МВС-1000 с ОС типа Unix на базе рабочих станций Alpha, находящийся в МСЦ Миннауки РФ и РАН.

Рассмотрим последовательно основные пиктограммы окна И, представляющие ключевые моменты реализации сценария работы пользователя с прикладной задачей (см. рис.4).

Пиктограмма "Norma on PC", расположенная слева, вызывает процедуру компиляции и конфигурации исходной Норма-программы непосредственно на ПК пользователя. Запуск этой процедуры осуществляется путем буксировки значка файла исходной Норма-программы из директории LOCAL (см. п.6.4) на эту пиктограмму.

Пиктограмма "F77 on Host", находящаяся внизу, связана с процедурой компиляции Фортран-программы на сервере (удаленном или локальном). Запуск указанной процедуры производится перемещением значка файла Фортран-программы из директории HOST ("зеркала" соответствующей директории вычислителя-хоста) на пиктограмму "F77 on Host".

Аналогично активизация значка "Run on Host" (наверху справа) обеспечивает запуск на сервере результирующего исполняемого модуля, если он правильно взят из директории HOST на ПК и размещен на пиктограмме "Run on Host".

В окне И, кроме того, имеется несколько сервисных значков, вызывающих вспомогательные процедуры, необходимые пользователю для успешного решения прикладной задачи.

Пиктограмма "Edit and View", расположенная в центре окна, при помещении на нее значка произвольного файла из любой директории (в частности, из директорий интерфейса LOCAL или HOST) вызывает для указанного файла текстовый редактор. Таким образом, в ПИ обеспечивается возможность просмотра или редактирования любого файла, интересующего пользователя.

Пиктограмма "Load and Save" (слева внизу) в случае ее активизации простым нажатием курсора вызывает на экран ПК вспомогательное окно Explorer. Указав в его адресной строке нужную директорию и тем самым настроив его на нужное место в файловой системе, а затем используя все ту же технологию drag&drop, пользователь может загрузить из этой директории любой необходимый ему в текущем сценарии файл в одну из стандартных рабочих директорий интерфейса - LOCAL или HOST. Таким же путем можно, наоборот, сохранить в выбранной директории (в окне Explorer) любой файл из стандартных рабочих директорий LOCAL или HOST. Эта процедура позволяет загружать из файловой системы ПК в среду ПИ тексты программ и исходные данные, а также сохранять в файлах ПК результаты вычислений, листинги, информацию о состоянии запущенных пользователем программ и пр.

Пиктограмма "Archives" (слева внизу) вызывает сервисную процедуру архивного исполнителя. Она позволяет, например, сохранить в архиве любой файл из стандартных рабочих директорий LOCAL или HOST. Для этого достаточно отбуксировать значок нужного файла на пиктограмму архивного исполнителя. Простое нажатие курсора на пиктограмме "Archives" вызовет выдачу оглавления текущего содержимого архива в образовавшееся на экране ПК информационное окно.

Проведение сложного вычислительного эксперимента может потребовать многочасовой работы удаленного вычислителя. В этом случае пользователю необходимо иметь возможность оперативного получения достоверной информации о состоянии вычислителя и текущем состоянии запущенной задачи. Пиктограмма (точнее соответствующая ей процедура), именуемая "How are You", обеспечивает пользователю возможность доступа к следующей информации, характеризующей текущее состояние его задачи:

- задача запущена;
- задача в очереди;
- задача в счете;
- задача завершена,

При этом предоставляется также возможность оперативного получения (для анализа) промежуточных и окончательных результатов счета. В реализации ПИ упомянутая информация добывается путем перехвата протоколов соответствующих команд управления удаленным вычислителем. Выдачу всей этой информации вызывает простое нажатие курсора на пиктограмме "How are You".

Довольно часто у пользователя может возникнуть необходимость удалить задачу из счета, например, в том случае, если ее исходные данные оказались ошибочными или полученные промежуточные результаты свидетельствуют о нецелесообразности ее дальнейшего исполнения. Пиктограмма "Kill on Host", точнее говоря, связанная с ней
процедура, позволяет принудительно завершить задачу, находящуюся
в счете или даже в очереди на исполнение.

В окне И интерфейса размещается также несколько информационных значков.

Пиктограммы-стрелки с именем "Norma" просто иллюстрируют пути прохождения информации в цикле отладки Норма-программы и при нажатии не вызывают какого-либо действия. В ходе отладки пользователь может попеременно передавать Норма-программу (из директории LOCAL) то процедуре Норма-компиляции, то процедуре текстового редактора (для исправления ошибок).
Аналогично пиктограммы-стрелки "Fortran" характеризуют пути прохождения информации в цикле отладки Фортран-программы. В ходе отладки может происходить попеременная передача файла с Фортран-программой (из директории HOST) то процедуре Фортран-компилятора, то процедуре текстового редактора (при обнаружении ошибок в Фортран-программе).

Пиктограмма-стрелка "Results" отражает этап передачи результатов вычислительного эксперимента, полученных на удаленном вычислителе (из директории HOST), текстовому редактору для просмотра и анализа.

Пиктограмма-стрелка "Protocols" соответствуют этапу передачи информации о текущем состоянии вычислителя или задачи пользователя, а возможно и промежуточных результатов вычислений от удаленного вычислителя (через директорию HOST) текстовому редактору для просмотра и анализа.

Непосредственное отношение к получению пользователем этой информации имеют пиктограммы "How are You" и "Kill on Host", а точнее говоря, связанные с ними процедуры.

6.3. Архив

Левое нижнее рабочее окно называется "Archives" - окно Архива (А). Оно обычно содержит пиктограмму текущего архива, в котором пользователь может сохранять информацию, возникающую в процессе выполнения сценария вычислений. Это окно также содержит паспорт текущей задачи пользователя. Пиктограмма окна И с одноименным наименованием "Archives" (см. выше) имеет непосредственное отношение к архивированию информации. В ПИ предполагается существование одного архивного файла для каждого сеанса работы пользователя над конкретной задачей.

6.4. Локальные и удаленные ресурсы

В правой половине экрана ПК постоянно размещаются два стандартных рабочих окна ПИ.

Нижнее окно LOCAL отражает текущее содержимое локальной директории ПИ, процессы, протекающие непосредственно на ПК, без обращения к удаленному вычислителю. Иначе говоря, в этом окне отображаются файлы, относящиеся к процедурам, исполняемым непосредственно на ПК пользователя без выхода во внешнюю сеть. В нашем примере это этап, касающийся компиляции и конфигурации Норма-программы.

Верхнее окно HOST является, так сказать, отражением, "зеркалом" содержимого директории, принадлежащей пользователю на удаленном вычислителе (удаленном или локальном сервере/хосте). Там, на удаленном сервере, выбранном пользователем в начале сеанса работы, выполняется его задача. В этом окне отображаются файлы, относящиеся к работе на удаленном вычислителе (в частности, им может оказаться и локальный ПК пользователя). В нашем примере это этапы, касающиеся компиляции Фортран-программы и запуска ее на счет на выбранном удаленном вычислителе.

6.5. Паспорт задачи пользователя

На каждую задачу пользователь предварительно оформляет так называемый паспорт задачи, содержащий следующую информацию:

- имя Норма-программы;
- имя Фортран-программы;

- имя результирующего исполняемого модуля задачи;
- имена стандартных рабочих директорий задачи (в частности, LOCAL и HOST);
- имена входных файлов задачи;
- метод распараллеливания программы (либо последовательный вариант);
- количество параллельных процессоров, необходимых задаче;
- максимальное время счета задачи и пр.

7. Работа с интерфейсом

В этом разделе рассматриваются основные приемы работы пользователя с интерфейсом. Рассмотрение ведется на примере подготовки и выполнения конкретной Норма-программы testd.hop на многопроцессорном вычислителе gateway.kiam.ru в параллельном Фортране MPI.

Типовой сценарий работы над программой состоит из нескольких характерных этапов.

На предварительном этапе осуществляется вызов системы ПИ и начало сеанса работы пользователя путем активизации пиктограммы "User Interface" на рабочем столе ПК.

1. На первом этапе после появления на рабочем столе четырех окон Windows Explorer - основных окон интерфейса (см. п.6.2), осуществляется активизация в окне И пиктограммы выбранного удаленного многопроцессорного вычислителя - gateway.kiam.ru.

2. На втором этапе производится активизация пиктограммы "Load and Save". В результате появляется дополнительное окно Explorer. В его адресной строке пользователь выбирает директорию "User". С помощью технологии drag&drop осуществляется загрузка из директории "User" пользователя в стандартную рабочую директорию интерфейса LOCAL исходных файлов, необходимых для решения данной

задачи. Это следующие файлы, назначение которых более подробно описано ниже:

testd.hop

test1inc.dat

test1inp.dat

test1inr.dat

test1inu.dat

riemann.for

Затем это дополнительное окно Explorer закрывается пользователем.

3. На третьем этапе осуществляется вызов Норма-компилятора (вместе с конфигуратором) посредством буксировки на пиктограмму "Norma on PC" значка с исходной Норма-программой testd.hop (из директории LOCAL на ПК). В результате образуется Фортран-программа testd.fmp, а также листинг результатов трансляции - testd.lst, которые по умолчанию помещаются в рабочую директорию LOCAL.

В соответствии с алгоритмом работы данной Норма-программы на этом этапе при построении соответствующей Фортран-программы testd.fmp используются следующие файлы исходных данных из директории LOCAL:

test1inc.dat

test1inp.dat

test1inr.dat

test1inu.dat

В соответствии с алгоритмом работы данной Норма-программы в ходе ее компиляции генерируется файл исходных данных NORMAIN.IN (он необходим для будущей результирующей программы, исполняемой на удаленном вычислителе), который также помещается в директорию LOCAL.

Затем на этом же этапе в соответствии с технологией подготовки Норма-программы осуществляется процесс конфигурации и создания результирующей параллельной Фортран-программы testd.f на ПК. Конфигурация представляет собой сборку модулей, относящихся к одной исполняемой единице, в один файл, Это собственно процесс сборки скомпилированной Фортран-программы с другими модулями, написанными непосредственно на Фортране.

В данном случае в ходе конфигурации на ПК создается новая Фортран-программа testd.f, скомпонованная из Фортран-программы testd.fmp, соответствующей исходной Норма-программе testd.hop и дополнительной исходной Фортран-программы riemann.for, подготовленной заранее на Фортране.

Параллельная Фортран-программа testd.f, предназначенная для непосредственного исполнения в Фортране MPI на удаленном многопроцессорном вычислителе gateway.kiam.ru, по умолчанию помещается в директорию HOST этого удаленного сервера.

4. На четвертом этапе с помощью технологии drag&drop файл исходных данных NORMAIN.IN (полученный на предыдущем этапе) переносится пользователем из директории LOCAL в директорию HOST удаленного сервера.

5. На пятом этапе выполняется компиляция полученной параллельной Фортран-программы testd.f на Фортране MPI на удаленном многопроцессорном

вычислителе gateway.kiam.ru, подключенном к сети Интернет. Для этой цели на пиктограмму "F77 on Host" перемещается значок с Фортран-программой testd.f (из директории HOST). В результате компиляции в директории HOST образуется файл с именем testd.out, представляющий результирующий загрузочный модуль, образованный на удаленном сервере и пригодный к исполнению (одновременно в директории HOST генерируется файл листинга testd.lst, см. рис.1).

6. На шестом этапе перед запуском задачи на счет, вообще говоря, полезно обратиться за справкой о состоянии загрузки вычислителя gateway.kiam.ru, что осуществляется посредством активизации пиктограммы "How are You" в окне И интерфейса. Если протокол, полученный после этого, показывает, что в распоряжении
пользователя имеется достаточное количество свободных процессоров (а для данной задачи их нужно 11 штук), то можно спокойно перейти к следующему этапу сценария (см. рис.2). В противном случае возможны следующие варианты: перейти к следующему этапу сценария, поставив задачу в очередь; ждать освобождения на gateway.kiam.ru
требуемого количества процессоров; перейти к первому шагу сценария и обратиться за обслуживанием к другому многопроцессорному вычислителю, например, к alpha.kiam.ru.

7. На седьмом этапе осуществляется запуск на счет исполняемого файла testd.out на многопроцессорном вычислителе gateway.kiam.ru. Для этого на пиктограмму "Run on Host" помещается значок исполняемого файла testd.out из директории HOST.

В результате счета задачи в директории HOST образуются файлы, представляющие результат данного вычислительного эксперимента:

test1rep

test1rer

test1reu

а также (при необходимости) записывается информация, направляемая пользователем в процессе счета задачи в системный вывод (в виде файлов с именами, соответствующими номерам процессоров от output0 до output10).

8. Наконец, на восьмом этапе результаты вычислений, а также любая другая интересующая пользователя информация из директорий LOCAL и HOST сохраняется (с применением технологии drag&drop) в выбранной пользователем директории. Для этой цели (как и на втором этапе) активизируется пиктограмма

"Load and Save" и вызывается окно Explorer, в адресной строке которого указывается имя директории
пользователя "User" (см. рис.3). После сохранения результатов это окно Explorer закрывается пользователем (см. рис.4).

Для завершения очередного сеанса работы ПИ пользователь должен также закрыть все основные окна пользовательского интерфейса.

8. Средства реализации интерфейса

В качестве инструмента при построении данного варианта ПИ использовались следующие средства:

- стандартные bat-файлы MS-DOS с использованием языка скриптов DOS Shell и встроенных команд MS-DOS на IBM PC;
- компилятор с языка Си при реализации подпрограмм разбора строчных аргументов на IBM PC;
- стандартные приложения Windows 95/98: программа Windows Explorer, текстовый редактор WordPad на IBM PC;
- технология WSH (Windows Scripting Host) для Windows со скриптами на языках JScript и VBScipt на IBM PC;
- стандартные для Windows 95/98 на IBM PC клиенты (утилиты сетевой коммуникации): ftp- и rexec-компоненты;
- программа SecureCRT (Windows-клиент, реализующий протоколы telnet и SSH) на IBM PC для поддержки Secure Shell со скриптами, написанными на языке VBScript;
- стандартная для SSH (Secure Shell) SCP-компонента (Secure Copy client) для Windows на IBM PC;
- возможности языка Shell и стандартные команды ОС Unix на удаленных серверах;
- возможности архиваторов ARJ, ZIP (в ОС MS-DOS) на IBM PC, а также GZIP и TAR (в ОС Unix) на удаленных серверах;
- компилятор и конфигуратор для языка Норма на IBM PC;
- компиляторы с языка Фортран на IBM PC и серверах;
- команды запуска и сопровождения задач пользователя на многопроцессорном параллельном вычислителе МВС-1000.

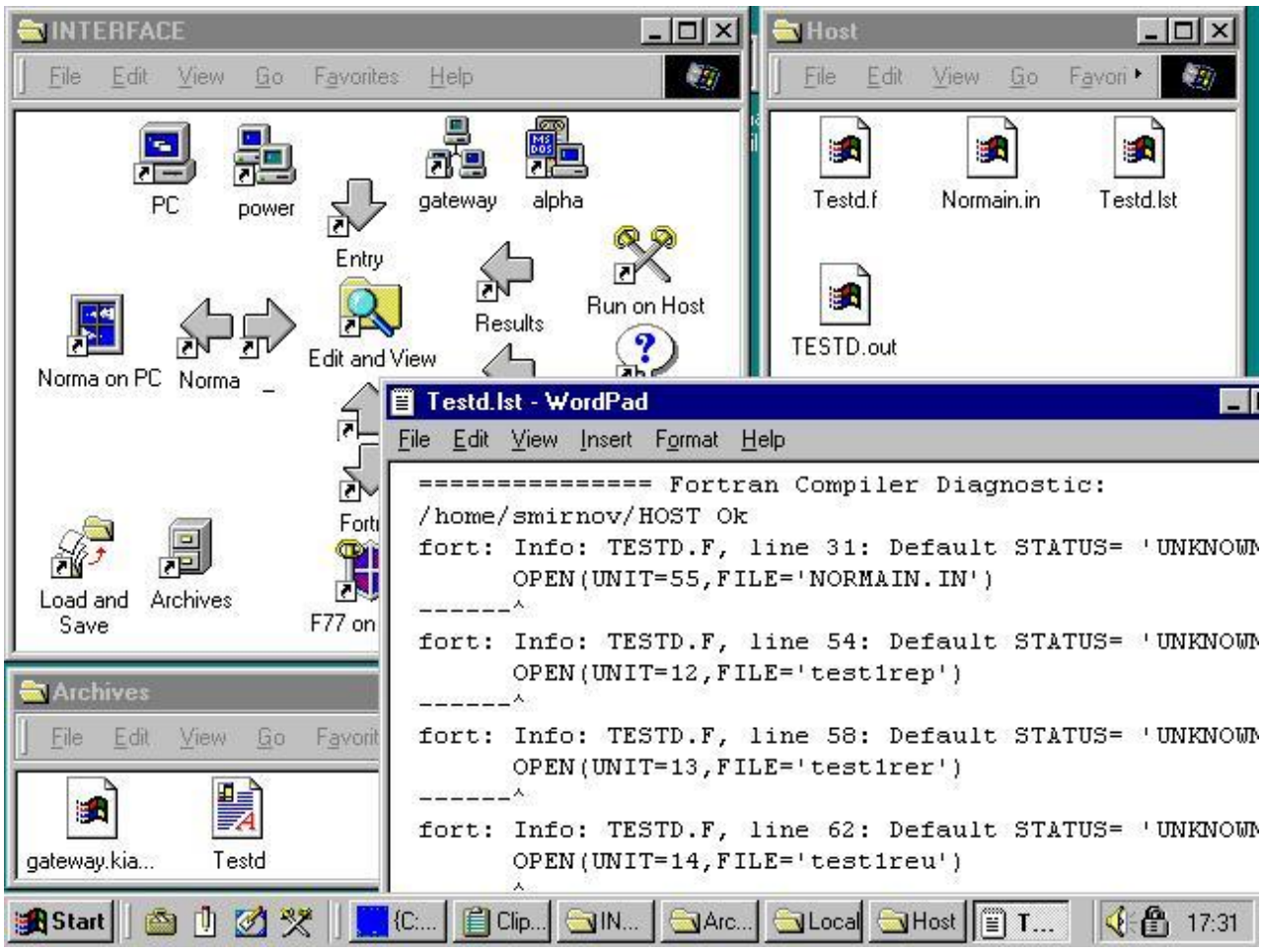

Рис.1. Просмотр диагностики после компиляции Фортран-программы

Рис.2. Контроль загрузки процессоров вычислителя

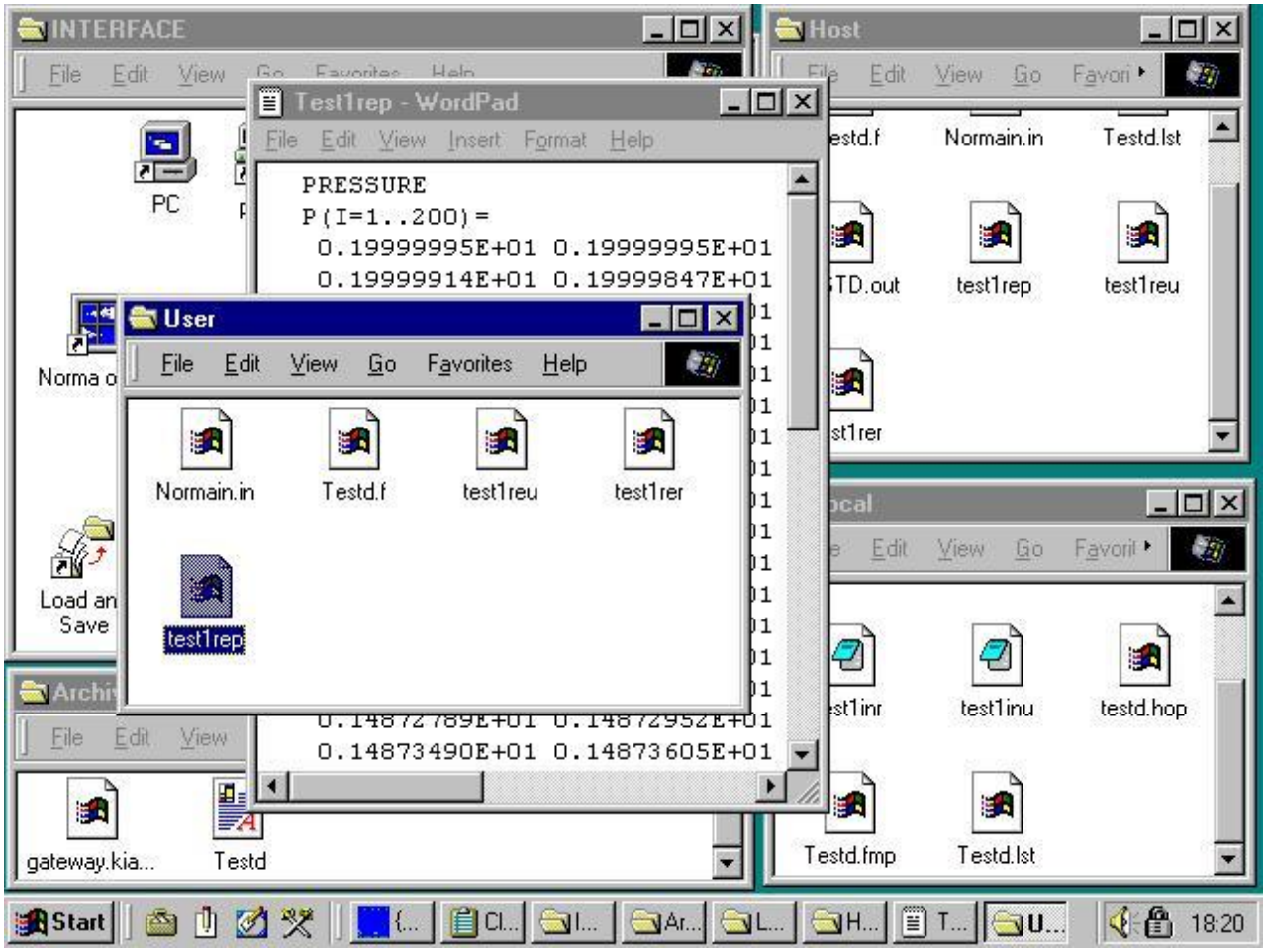

Рис.3. Сохранение результатов вычислений в директории пользователя

Рис.4. Вид рабочего стола после завершения сеанса вычислений

9. Заключение

Актуальность проекта определяется следующими обстоятельствами:

- повышением интереса к параллельным вычислениям благодаря развитию современных суперкомпьютеров с параллельной архитектурой, а также перспективе использования сетей ЭВМ в качестве параллельных вычислителей;
- появлением метакомпьютинга - направления, занимающегося поддержкой параллельных процессов в сети географически распределенных вычислительных ресурсов;
- бурным развитием методов коммуникации - глобальные сети, сотовые и спутниковые системы телефонии;
- необходимостью перевода традиционных вычислительных алгоритмов на параллельные компьютеры или сети;
- постоянным усложнением предметных областей и технологий решения научно-технических задач, предъявляющих особые требования к пользователям ЭВМ.

В связи с этим возникла острая необходимость в создании программных оболочек для облегчения процесса разработки, отладки и исполнения прикладных программ, использующих параллельные методы.

В рамках проекта мобильного пользовательского интерфейса были разработаны и созданы:

- PPP-сервер на платформе Linux или Windows NT;
- средства взаимодействия ПИ с удаленными однопроцессорными ЭВМ и параллельными суперкомпьютерами типа МВС-1000 (Unix-подобные ОС) как непосредственно через сеть Интернет, так и по телефонным линиям через PPP-сервер;
- интерфейс пользователя, функционирующий на персональной ЭВМ типа IBM PC под управлением ОС Windows 95/98/NT;
- средства подготовки программ для персональных машин IBM PC, удаленных компьютеров и суперкомпьютеров типа МВС-1000.

Литература

1. Головков С.Л., Наумов Н.А., Рубин А.Г., Смирнов В.К. Интеллектуальный интерфейс и средства его разработки. ИПМ РАН, Отчет 12-34, Москва. 1992. 32 стр.
2. Головков С.Л., Диев С.В., Рубин А.Г., Смирнов В.К. Макет интеллектуального интерфейса (проект). ИПМ РАН, Отчет. Москва. 1993. 32 стр.
3. Воронков А.В., Смирнов В.К. Интеллектуальный интерфейс для сложной научно-технической задачи. Пилотная модель - Архитектура. ИПМ РАН, Отчет 425-95, Москва, 1995.
4. Воронков А.В., Смирнов В.К., Тульский В.П. Интеллектуальный интерфейс для сложной научно-технической задачи. Взаимодействие между Интерфейсом Пользователя и сервером через сеть Internet. ИПМ РАН, Отчет, Москва, 1996.
5. Diev S.V., Rubin A.G. Ensembles of hyperelements: a notion base for activity representation. Preprint, Inst.Appl.Mathem., Russia Ac.of Sc., N 61, 1996.
6. Воронков А.В., Смирнов В.К., Тульский В.П. Интеллектуальный интерфейс для сложной научно-технической задачи. Интерфейс пользователя. Подготовка программ и данных. ИПМ РАН, Отчет, Москва, 1997.
7. Диев С.В., Рубин А.Г. Ансамбли: метод описания деятельности. Программирование, N 2, 1997, с.27-38.
8. Воронков А.В., Смирнов В.К., Тульский В.П. Интеллектуальный интерфейс для сложной научно-технической задачи. Заключительный отчет по Проекту 67-94. ИПМ РАН, Отчет, Москва, 1997.
9. Смирнов В.К., Тульский В.П. Сетевой интерфейс для научно-технической задачи. Всероссийская научная конференция "Научный сервис в Интернет", Тезисы докладов, Новороссийск, 20-25 сентября 1999 г. Изд-во Московского университета, 1999.
10. Андрианов А.Н., Ефимкин К.Н., Задыхайло И.Б. Непроцедурный язык для решения задач математической физики. Программирование, N 2, 1991, с.80-94.
11. Андрианов А.Н., Бугеря А.Б., Ефимкин К.Н., Задыхайло И.Б. Норма. Описание языка. Рабочий стандарт. Препринт ИПМ им. М.В.Келдыша РАН, N 120, 1995, 50 с.