На правах рукописи

Петрищев Дмитрий Вячеславович

ИССЛЕДОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА СИСТЕМНОГО СОПРОВОЖДЕНИЯ ВЫЧИСЛЕНИЙ ДЛЯ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ С АВТОМАТИЧЕСКИМ РАСПРЕДЕЛЕНИЕМ РЕСУРСОВ

Специальность: 05.13.11 – “Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей”

АВТОРЕФЕРАТ

диссертации на соискание ученой степени
кандидата технических наук

Москва – 2007

Диссертационная работа выполнена в Институте проблем информатики Российской академии наук.

Научный руководитель: доктор технических наук, профессор
Владимир Павлович Торчигин

Официальные оппоненты: доктор технических наук, профессор
Московского инженерно-физического
института (государственного университета)
Ярослав Афанасьевич Хетагуров

кандидат технических наук, доцент
Московского государственного
технического университета им. Н.Э. Баумана
Александр Петрович Ковтушенко

Ведущая организация: ОАО «Научно-исследовательский институт
супер ЭВМ»

Защита диссертации состоится “23” мая 2007 года в 16 часов на заседании диссертационного совета Д002.073.01 при Институте проблем информатики РАН по адресу: 119333, Москва, ул. Вавилова, д. 44, корп. 2.

С диссертацией можно ознакомиться в библиотеке Института проблем информатики РАН.

Автореферат разослан “20” апреля 2007 г.

Отзыв, заверенный печатью, просим отправлять в одном экземпляре по адресу: 119333, Москва, ул. Вавилова, д. 44, корп. 2.

Председатель диссертационного
совета Д002.073.01,
доктор технических наук И.А. Соколов

Общая характеристика работы

Актуальность темы диссертации

Современные высокопроизводительные вычислительные системы – суперЭВМ – играют значительную роль в ключевых областях науки и техники. Для решения научных и технических задач создаются многопроцессорные суперкомпьютеры с производительностью 10^{12}-10^{15} операций в секунду, которая достигается благодаря распараллеливанию вычислительных процессов. Дальнейшее повышение производительности возможно как за счет увеличения количества содержащихся в вычислительной системе процессоров или машин, так и за счет поиска более эффективных методов распараллеливания вычислений.

В настоящее время во всем мире основное внимание уделяется созданию кластерных и массивно-параллельных систем на основе процессоров общего назначения. Производительность подобных систем сильно зависит от класса решаемых задач. Так, на задачах пакета LINPACK лучшие из современных суперЭВМ показывают максимальную производительность, равную 60-80% от пиковой производительности. Вместе с тем, на многих других задачах их производительность существенно ниже и может составлять 10-20% от пиковой производительности. Основными причинами этого являются сложность балансировки нагрузки на блоки вычислительной системы, высокие накладные расходы на синхронизацию вычислительных процессов и низкая эффективность механизмов кэширования данных при определенных схемах доступа к памяти (например, при нерегулярном доступе к данным или при частом доступе к глобальным данным из параллельно выполняющихся вычислительных процессов).

Чтобы обойти ограничения, присущие традиционным архитектурам высокопроизводительных вычислительных систем, во всем мире делаются попытки исследования и создания новых, нетрадиционных архитектур: асинхронных, реляционных, рекурсивных, функциональных, потоковых и других. Весьма перспективными являются архитектуры, основанные на модели вычислений, управляемых потоком данных, работа над которыми проводится во многих странах мира, в том числе, в США, Японии, Великобритании и России. Одним из важнейших достоинств потоковых архитектур является возможность распараллеливания вычислений и балансировки загрузки процессоров на аппаратном уровне.

В отделе «Проблем построения информационно-вычислительных систем высокого параллелизма» Института проблем информатики Российской академии наук под руководством академика В.С. Бурцева были разработаны новые принципы организации вычислительного процесса и на их основе создана вычислительная система с автоматическим распределением ресурсов (ВСАРР), использующая гибридную модель вычислений, управляемых потоком данных, с динамически формируемым контекстом. Разработан инструментальный испытательный комплекс ВСАРР, включающий в себя имитационную модель системы, блочно-регистровую модель системы и макет

системы на ПЛИС. Созданы инструментальные средства, позволяющие вести программирование для ВСАРР с использованием языка ассемблера и языка повышенного уровня. Таким образом, можно говорить о перспективах применения существующих моделей и макетов в качестве «испытательной лаборатории» для исследования различных научно-технических решений, отработки подходов к практическому применению ВСАРР.

Эффективность практического применения высокопроизводительного вычислительного комплекса во многом зависит от его операционной системы. Основными функциями операционной системы являются управление задачами, управление памятью, организация ввода-вывода, поддержка файловой системы. Как правило, при выполнении своих функций операционная система полагается на имеющиеся в ее распоряжении аппаратные средства, так что структура и алгоритмы работы операционной системы в значительной степени зависят от структуры и алгоритмов работы аппаратных средств. Следует заметить, что модель вычислений, которая положена в основу ВСАРР, и архитектура ВСАРР обладают рядом принципиальных отличий от моделей вычислений и архитектур других вычислительных систем. Поэтому не представляется возможным использовать для организации вычислений в ВСАРР какую-либо из существующих операционных систем (пусть даже с некоторыми доработками). С другой стороны, некоторые из функций операционной системы являются в значительной степени нейтральными по отношению к модели вычислений и архитектуре ВСАРР. Например, механизмы удаленного доступа к вычислительной системе, реализованные во всех современных операционных системах, лишь в малой своей части привязаны к конкретным аппаратным средствам.

В данной диссертационной работе решается проблема создания программного комплекса системного сопровождения вычислений для вычислительной системы с автоматическим распределением ресурсов. Разрабатываемый программный комплекс предназначен для организации многозадачного многопользовательского режима работы ВСАРР с возможностью удаленного доступа к ВСАРР. Программный комплекс обеспечивает возможность одновременной работы с ВСАРР большого количества пользователей, которые могут запускать на ВСАРР различные вычислительные задачи. Программный комплекс выполняет те функции операционной системы, которые являются зависимыми от модели вычислений и архитектуры ВСАРР – управление задачами, управление памятью, ввод-вывод исходных данных и результатов вычислений. При этом остальные функции ввода-вывода и поддержка файловой системы возлагаются на обычную операционную систему – такую, как Windows или Linux, что позволяет существенно сократить размер и сроки создания программного комплекса.

Цель и задачи работы

Настоящая диссертационная работа посвящена исследованию и разработке принципов организации вычислений с использованием

вычислительной системы с автоматическим распределением ресурсов и созданию программного комплекса системного сопровождения вычислений для ВСАРР, позволяющего организовать эффективное практическое применение ВСАРР в составе традиционной вычислительной среды.

Для достижения поставленной цели в работе решаются следующие основные задачи:

1) Анализ различных вычислительных систем с моделью вычислений, управляемых потоком данных, с точки зрения роли системного программного обеспечения при организации вычислений.
2) Исследование модели вычислений и архитектуры вычислительной системы с автоматическим распределением ресурсов.
3) Разработка принципов функционирования и структуры программного комплекса системного сопровождения вычислений для ВСАРР.
4) Реализация компонентов программного комплекса системного сопровождения вычислений для ВСАРР.
5) Исследование выполнения различных программ на инструментальном испытательном комплексе ВСАРР с использованием программного комплекса системного сопровождения вычислений для ВСАРР.

Объект и предмет исследования

Объектом исследования является вычислительная система новой нетрадиционной архитектуры – вычислительная система с автоматическим распределением ресурсов, а также ее математическое обеспечение. Предметом исследования является программный комплекс системного сопровождения вычислений для ВСАРР.

Методы исследования

Исследования проводились с использованием основных положений теории параллельных вычислений, теории высокопроизводительных вычислений и теории построения операционных систем. Реализация программного комплекса системного сопровождения вычислений для ВСАРР велась с применением компонентного подхода.

Научная новизна

Впервые предложены программные средства, позволяющие организовать эффективное практическое применение ВСАРР в составе традиционной вычислительной среды.

Основные научные результаты работы таковы:

1) Предложены принципы функционирования программного комплекса системного сопровождения вычислений для ВСАРР, в том числе, принципы планирования задач в многозадачном режиме работы ВСАРР и принципы организации программной поддержки виртуальной памяти ВСАРР.

2) Предложена структура программного комплекса системного сопровождения вычислений для ВСАРР, позволяющая организовать многозадачный многопользовательский режим работы ВСАРР.
3) Созданы программные интерфейсы доступа к ВСАРР, позволяющие обеспечить удаленный доступ к ВСАРР и упростить интеграцию ВСАРР в традиционную вычислительную среду.
4) Исследовано выполнение различных программ на инструментальном испытательном комплексе ВСАРР с использованием программного комплекса системного сопровождения вычислений для ВСАРР, подтверждена работоспособность программного комплекса и получены оценки его эффективности.

Практическая значимость

Практическая значимость работы заключается в следующем:

1) Реализован программный комплекс системного сопровождения вычислений для ВСАРР.
2) Разработаны и апробированы различные способы доступа пользователей к программному комплексу системного сопровождения вычислений для ВСАРР.
3) Создана интегрированная среда разработки программ для ВСАРР.
4) Создан ряд прикладных программ для ВСАРР.

Положения, выносимые на защиту

1) Разработанные методы планирования задач в многозадачном режиме работы ВСАРР и программные механизмы поддержки виртуальной памяти ВСАРР.
2) Разработанная структура программного комплекса системного сопровождения вычислений для ВСАРР.
3) Разработанные программные интерфейсы доступа к ВСАРР.
4) Исследование выполнения различных программ на инструментальном испытательном комплексе ВСАРР с использованием программного комплекса системного сопровождения вычислений для ВСАРР.

Реализация результатов работы

Разработанные научно-технические решения были реализованы в Институте проблем информатики РАН в ходе исследования и разработки вычислительной системы с автоматическим распределением ресурсов, прошли апробацию на инструментальном испытательном комплексе ВСАРР и вошли в состав инструментального испытательного комплекса.

Апробация работы

Основные положения и результаты работы докладывались и обсуждались на научных семинарах в ИПИ РАН в 2003-2007 гг., а также на конференциях и научных форумах: на международной научной конференции «Искусственный интеллект – 2004. Интеллектуальные и многопроцессорные системы – 2004» (пос. Кацивели, Крым, 2004), на «II Научной сессии Института проблем информатики Российской академии наук: Проблемы и методы информатики» (Москва, 2005), на международной научно-технической конференции «Интеллектуальные и многопроцессорные системы – 2005» (пос. Дивноморское, 2005), на научно-практическом семинаре «Новые информационные технологии в автоматизированных системах» (Москва, 2006).

Исследование разработанных методик, алгоритмов и программных решений проводилось в Институте проблем информатики РАН на инструментальном испытательном комплексе ВСАРР.

Публикации

По материалам диссертационной работы опубликовано 7 печатных работ, список которых приводится в конце автореферата. Кроме того, по теме диссертации опубликованы материалы в научно-технических отчетах ИПИ РАН за 2003-2006 гг.

Структура и объем диссертации

Диссертационная работа состоит из введения, четырех глав, заключения и списка литературы. Общий объем диссертации составляет 138 страниц.

Содержание работы

Во введении обоснована актуальность темы диссертации, определены научная новизна и практическая значимость работы, сформулированы цель и основные задачи исследований, приведено краткое описание структуры диссертации.

В первой главе диссертационной работы приведен обзор различных вычислительных систем с моделью вычислений, управляемых потоком данных, рассмотрена роль системного программного обеспечения при организации вычислений.

Основные принципы модели вычислений, управляемых потоком данных (*dataflow*), были сформулированы в 1974-1975 годах Джеком Деннисом (США). Главная особенность этой модели вычислений заключается в том, что команда программы поступает на исполнение тогда, когда доступны (вычислены) все ее операнды. Для сравнения, традиционная модель вычислений фон-Неймана основывается на том, что команда программы поступает на исполнение тогда, когда на нее указывает счетчик команд.

По сравнению с традиционной моделью вычислений, модель вычислений, управляемых потоком данных, обладает рядом преимуществ. В частности, выявление параллелизма программ на уровне команд существенно упрощает решение проблемы балансировки нагрузки на процессоры вычислительной

системы. Вместе с тем, для модели вычислений, управляемых потоком данных, характерен собственный набор проблем, которые необходимо учитывать при разработке системного программного обеспечения.

Существуют различные виды моделей вычислений, управляемых потоком данных: статическая модель, динамическая модель, гибридная динамическая модель. С точки зрения создания вычислительных систем общего назначения, наиболее перспективными являются различные модификации гибридной динамической модели, которая позволяет рассматривать целые участки программы как команды, которые выполняются от начала и до конца на одном процессоре.

Одной из проблем вычислительных машин с архитектурой потока данных является проблема управления *памятью совпадений*, которая используется для определения готовности команд к выполнению и хранения операндов команд, не готовых к выполнению. Переполнение памяти совпадений в таких вычислительных машинах приводит к блокировке (*deadlock*) вычислительного процесса. Основной причиной переполнения памяти совпадений является то, что параллелизм программ во многих случаях оказывается настолько большим, что имеющегося объема памяти совпадений оказывается недостаточно для хранения всех операндов команд, не готовых к выполнению. В ряде известных проектов по созданию вычислительных машин с архитектурой потока данных (например, *MIT Tagged-Token Dataflow Architecture* и *Manchester Data-Flow Machine*) для преодоления данной проблемы применялось принудительное ограничение параллелизма программы. Чтобы уменьшить параллелизм программы, использовались различные ограничения, накладываемые на программу на этапе ее исполнения. Например, ограничивалось количество одновременно выполняемых процедур программы или количество одновременно выполняемых итераций циклов с целью исключить саму возможность переполнения памяти совпадений. Недостатком такого подхода являлось то, что параллелизм программы на этапе ее исполнения во многом зависит от входных данных и может варьироваться в широких пределах. Таким образом, чтобы исключить возможность переполнения памяти совпадений, приходилось накладывать на программу очень жесткие ограничения, которые существенно снижали эффективность ее исполнения.

Другой важной проблемой вычислительных машин с архитектурой потока данных является проблема распределения функций между аппаратурой и системным программным обеспечением. Во многих проектах потоковых вычислительных машин ряд низкоуровневых операций распределения процессоров, памяти и других ресурсов выполнялся программным образом, что приводило к возникновению значительных накладных расходов и существенному снижению производительности вычислительной машины. Производительность можно повысить, выполняя определенные операции распределения ресурсов на аппаратном уровне.

Во второй главе диссертационной работы исследованы модель вычислений и архитектура ВСАРР, определены принципы функционирования программного комплекса системного сопровождения вычислений для ВСАРР.

Модель вычислений ВСАРР

ВСАРР основана на гибридной модели вычислений, управляемых потоком данных, с динамически формируемым контекстом. Программу, написанную для выполнения на ВСАРР, можно представить в виде направленного графа потока данных. В узлах этого графа располагаются операторы – программы, предназначенные для обработки операндов, поступающих на входы узлов. Операторы могут быть простыми, то есть, выполнять над поступившими операндами одну операцию, или сложными – выполнять над поступившими операндами несколько операций. Дуги графа отражают зависимость операторов по данным.

Таким образом, основной структурной единицей потоковой программы является узел. С каждым узлом связана программа, которая называется программой узла. Выполнение потоковой программы представляет собой серию активаций программ узлов. Активация и выполнение программы узла происходит при поступлении на узел полного комплекта операндов. Как правило, в процессе вычислений программа узла активируется многократно. Активированные программы узлов выполняются параллельно и независимо друг от друга. Программа узла может выполняться одновременно над несколькими разными комплектами операндов.

Основными единицами информации в ВСАРР являются *токен* и *пара*. Токены являются носителями операндов. В основном, они предназначены для обмена информацией между узлами. Несколько полей токена объединяются в единую структурную единицу, называемую ключом токена. Ключ токена участвует в операции поиска в памяти совпадений для выявления парных токенов. При выявлении совпадающих по ключу токенов происходит их взаимодействие и формируется пара. Каждая пара включает в себя два операнда и адрес программы узла, обрабатывающей эти операнды. Таким образом, пара содержит комплект операндов, необходимый для активации программы узла.

Принципиальным отличием ВСАРР от других систем, основанных на гибридной модели вычислений, управляемых потоком данных, является возможность *динамического формирования контекста вычислений* программистом. Программа для ВСАРР может самостоятельно формировать содержимое поля контекста токенов в соответствии с алгоритмом своей работы, что устраняет накладные расходы на выделение и освобождение значений контекста и в некоторых случаях дает возможность использовать поле контекста в качестве носителя дополнительного операнда.

Проблема переполнения памяти совпадений в ВСАРР решается за счет введения *виртуальной памяти*. Функционирование виртуальной памяти основано на том, что все множество токенов задачи может быть разбито программистом на набор независимых подмножеств – подзадач, каждое из которых гарантированно умещается в доступный объем памяти совпадений. Если в процессе вычислений возникает угроза переполнения памяти совпадений, некоторые подзадачи могут быть откачаны во внешнюю память. В дальнейшем, когда память совпадений освободится, эти подзадачи могут быть

подкачаны обратно. Такой подход позволяет более эффективно использовать имеющийся ресурс памяти совпадений, что, в конечном счете, позволяет достичь более высокой производительности вычислительной системы в целом.

Архитектура ВСАРР

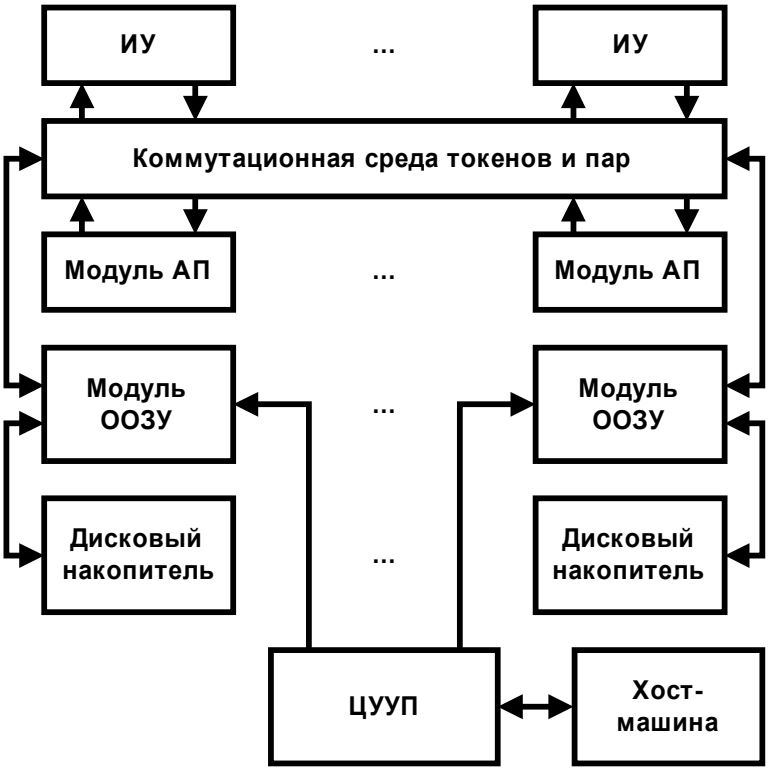

Рисунок 1. Структурная схема ВСАРР

Исполнительные устройства (ИУ) обрабатывают поступающие из коммутационной среды пары, выполняя соответствующие программы узлов. Как правило, результатом выполнения программы узла является один или несколько токенов, которые программа узла генерирует при помощи специальных команд. Через коммутационную среду эти токены направляются на модули ассоциативной памяти.

В качестве памяти совпадений используются модули *ассоциативной памяти* (АП), которые осуществляют хранение токенов и сопоставление хранимых токенов с токенами, поступающими из коммутационной среды. Результатом операции сопоставления может являться генерация одной или нескольких пар, которые через коммутационную среду направляются на исполнительные устройства.

Коммутационная среда осуществляет доставку пар к исполнительным устройствам и доставку токенов к модулям ассоциативной памяти.

Общее оперативное запоминающее устройство (ООЗУ) и диски в основном предназначены для хранения токенов и пар, откачанных из модулей ассоциативной памяти и внутренних буферов коммутационной среды.

Центральное устройство управления памятью (ЦУУП) поддерживает набор команд управления задачами (загрузка, запуск, останов, сбор

статистической информации и другие), а также поддерживает механизмы функционирования виртуальной памяти.

Х*ост-машина* представляет собой обычный персональный компьютер или сервер. Она предназначена для выполнения интерфейсных функций – организации взаимодействия между ВСАРР и другими вычислительными системами, а также, может быть использована для выполнения компонентов системного программного обеспечения ВСАРР.

Принципиальной особенностью ВСАРР является то, что большинство низкоуровневых операций по выделению и освобождению ресурсов выполняется на аппаратном уровне. Так, пары, формируемые в модулях АП, поступают на те ИУ, которые являются наименее занятыми. Токены, генерируемые в ИУ, распределяются между модулями АП в соответствии со значением *хэш-функции* от определенных полей токена. Токен, поступивший в модуль АП, записывается в любую свободную ячейку памяти. Стирание токена из модуля АП заключается в сбросе бита присутствия токена в памяти. Такой подход позволяет практически полностью устранить накладные расходы на распределение ресурсов вычислительной системы между задачами и между различными «ветвями» вычислительного процесса внутри каждой отдельно взятой задачи.

Программный комплекс системного сопровождения вычислений для ВСАРР

Системное программное обеспечение ВСАРР должно поддерживать многозадачный многопользовательский режим работы вычислительной системы, удаленный доступ к вычислительной системе, механизмы интеграции вычислительной системы в традиционную вычислительную среду и так далее. Некоторые из этих требований являются нейтральными по отношению к модели вычислений и архитектуре ВСАРР, что позволяет значительно сократить объем работ по созданию системного программного обеспечения ВСАРР, переложив выполнение некоторых функций на операционную систему хост-машины. Например, операционная система хост-машины может поддерживать различные файловые системы, механизмы предотвращения несанкционированного доступа к данным, сетевые протоколы для обмена данными с персональными компьютерами пользователей и многое другое. Фактически, чтобы продуктивно использовать ВСАРР в составе традиционной вычислительной среды, достаточно разработать набор компонентов системного ПО, позволяющий совместно с операционной системой хост-машины обеспечить выполнение вышеперечисленных требований.

Компоненты системного программного обеспечения ВСАРР могут выполняться непосредственно на потоковой вычислительной машине, на хост-машине и на персональных компьютерах пользователей. Непосредственно на потоковой вычислительной машине обязана выполняться только часть системного ПО – обработчики нештатных ситуаций и различные библиотеки. Остальные компоненты системного ПО могут выполняться на хост-машине; некоторые компоненты системного ПО (например, командные оболочки, различные утилиты) могут выполняться на персональных компьютерах

пользователей. Это дает возможность использовать для создания большой части системного ПО стандартные инструменты (языки программирования и среды разработки), так как и хост-машина, и персональные компьютеры пользователей работают под управлением стандартных операционных систем.

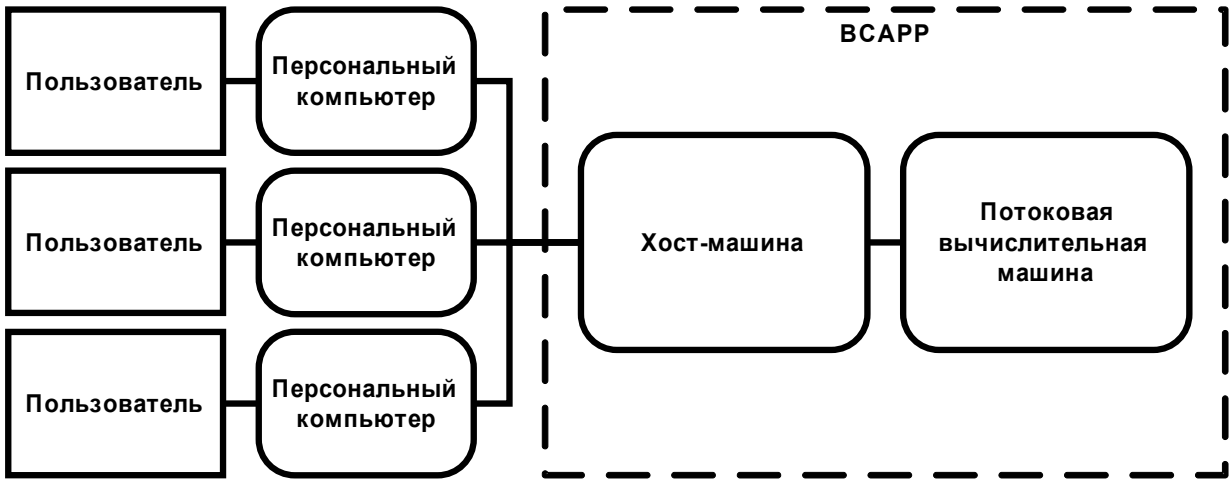

Рисунок 2. Схема доступа пользователей к BCAPP

Предполагается, что основной целью работы пользователей с BCAPP является запуск прикладных потоковых программ, позволяющих за сравнительно небольшое время выполнять необходимые пользователям вычисления. Таким образом, весь набор компонентов системного ПО, необходимый для продуктивного использования BCAPP в составе традиционной вычислительной среды, можно охарактеризовать как *программный комплекс системного сопровождения вычислений для BCAPP*. Данный программный комплекс должен предоставлять пользователям следующие основные возможности:

- Доступ к BCAPP с помощью локальной или глобальной сети со своего персонального компьютера.
- Хранение файлов с потоковыми программами и файлов с исходными данными и результатами вычислений в персональном рабочем пространстве и обмен информацией с другими пользователями.
- Запуск потоковых программ на BCAPP с определенными исходными данными и получение результатов вычислений.

Программный комплекс должен включать в себя три части:

- Резидентная часть – выполняется на потоковой вычислительной машине, включает в себя обработчики нештатных ситуаций и различные библиотеки.
- Операционная часть – выполняется на хост-машине, осуществляет обработку различных видов пользовательских запросов и контролирует использование ресурсов хост-машины и потоковой вычислительной машины.

- Клиентская часть – выполняется на персональном компьютере пользователя, организует взаимодействие между пользователем и операционной частью программного комплекса.

Потоковая программа в стадии выполнения представляет собой иерархически организованную группу задач, выполняющихся в общем окружении. Иерархия задач представляет собой дерево, построенное на основе отношения «родитель-ребенок». Корнем дерева является так называемая главная задача, которая запускается при вводе пользователем задания на выполнение потоковой программы. Главная задача может запускать дочерние задачи, те могут запускать свои дочерние задачи и так далее. Под окружением выполняющейся потоковой программы понимается совокупность различных системных параметров и таблиц. Поскольку каждая потоковая программа выполняется в собственном окружении, исключается возможность нарушения работы одной программы другой программой.

Управление задачами и памятью в ВСАРР

Работая в многозадачном многопользовательском режиме, программный комплекс системного сопровождения вычислений для ВСАРР должен осуществлять планирование задач, то есть, предоставлять задачам доступ к вычислительным ресурсам ВСАРР в соответствии с определенными правилами. Существует большое количество различных стратегий планирования, среди которых наиболее общей и универсальной является стратегия планирования в соответствии с приоритетами. Приоритет – некоторое числовое значение, которое пользователь, программист, администратор и системное программное обеспечение могут присваивать задачам. К алгоритмам планирования задач в соответствии с приоритетами предъявляются следующие основные требования:

Распределение ресурсов в соответствии с приоритетами – чем выше приоритет задачи, тем больший объем вычислительных ресурсов системы должен поступать в распоряжение задачи.

Предсказуемость – задачи должны регулярно получать доступ к вычислительным ресурсам системы.

Балансировка нагрузки – вычислительные ресурсы системы должны быть полностью загружены обработкой информации (если это возможно).

Важность выполнения тех или иных требований определяется типом вычислительной системы. Например, для систем реального времени и интерактивных систем чрезвычайно важным является требование предсказуемости, а для систем пакетной обработки – требование балансировки нагрузки.

Планирование задач в ВСАРР связано со следующими проблемами:

- Необходимость четко формулировать, какие виды ресурсов системы считаются *вычислительными* и должны предоставляться задачам в соответствии с выбранным алгоритмом планирования.
- Жесткие требования к скорости работы алгоритма планирования задач, возникающие ввиду ограниченного объема ассоциативной памяти.

Важно отметить, что принципиальной особенностью ВСАРР является то, что задача может выполняться только в том случае, когда ей выделен определенный объем ассоциативной памяти; если задача откачана во внешнюю память, ее выполнение приостанавливается. Таким образом, организация виртуальной памяти ВСАРР и планирование задач в ВСАРР тесно связаны между собой. Организация виртуальной памяти ВСАРР требует решения следующих проблем:

- Проблема распределения внешней памяти между одновременно выполняющимися задачами.
- Проблема выбора алгоритма откачки данных из ассоциативной памяти во внешнюю память и подкачки данных из внешней памяти в ассоциативную память.

В первую очередь, при решении этих проблем следует преследовать цель минимизировать простои вычислительной системы из-за ожидания завершения операции откачки или подкачки. Во вторую очередь, необходимо стремиться сократить объем различных накладных расходов, связанных с управлением памятью. Для этого необходимо искать оптимальное сочетание программных и аппаратных механизмов управления памятью.

Планирование задач в ВСАРР

Характерной особенностью ВСАРР является то, что процесс вычислений происходит как в исполнительных устройствах, в которых происходит обработка пар и генерация токенов, так и в модулях ассоциативной памяти, в которых происходит сопоставление токенов и генерация пар. Более того, в общем случае нельзя сказать, кто вносит основной вклад в получение конечного результата вычислений – ИУ или модули АП, поскольку в данном вопросе все зависит от алгоритма решаемой задачи и конкретных исходных данных. Таким образом, ИУ и модули АП являются *вычислительными* ресурсами ВСАРР, что должен учитывать алгоритм планирования задач.

Поскольку распределение доступного физического объема АП между задачами напрямую влияет на скорость выполнения задач, в процессе работы машины необходимо постоянно отслеживать загрузку АП токенами различных задач и перераспределять физический объем АП между задачами. Например, если низкоприоритетная задача захватила весь доступный физический объем АП, может возникнуть необходимость временно откачать ее внешнюю память, а впоследствии подкачать ее обратно.

Как показывают расчеты, частота принятия решений об откачке и подкачке задач может быть достаточно большой – порядка 10^6-10^7 раз в секунду, причиной чего является ограниченный объем АП. Действительно, чтобы обеспечить максимальную интенсивность генерации пар в модулях АП, необходимо поддерживать их в заполненном состоянии. Если заполненность модуля падает ниже определенного порога (допустим, 80%), должна начинаться подкачка токенов в модуль. Вместе с тем, если заполненность модуля поднимается выше определенного порога (допустим, 90%), необходимо начинать откачку токенов из модуля, чтобы избежать блокировки вычислительного процесса в ВСАРР. Таким образом, частота принятия

решений об откачке и подкачке определяется той скоростью, с которой меняется заполненность модуля. Данная скорость зависит как от постоянных факторов – то есть, объема модуля и темпа, с которым модуль принимает токены и генерирует пары, так и от переменных факторов – то есть, конкретных задач, которые выполняются в вычислительной системе на определенных наборах входных данных. Поскольку эффективная работа вычислительной системы должна обеспечиваться во всех ситуациях, целесообразно принятие решений об откачке и подкачке задач выполнять аппаратным образом. Эту функцию должно выполнять центральное устройство управления памятью ВСАРР. Системное программное обеспечение ВСАРР должно загружать в ЦУУП значения приоритетов задач, необходимые для принятия решений об откачке и подкачке.

Распределение ресурсов в соответствии с приоритетами в большинстве систем выполняется по следующему принципу: управление передается готовой к выполнению задаче с наивысшим приоритетом. Однако в этом случае необходимо предусматривать механизм предотвращения безостановочной работы такой задачи – например, уменьшать приоритет задачи с течением времени или ограничивать максимальный отрезок времени непрерывной работы задачи. Принципиально другим подходом является так называемое *гарантированное планирование*. Оно основывается на принципе передачи управления той задаче, которая за время своего предыдущего выполнения израсходовала меньше всего ресурсов из того лимита, который должен быть ей предоставлен в соответствии с ее приоритетом. Математически этот принцип может быть сформулирован следующим образом. Пусть в системе выполняется N задач. Обозначим через P_i абсолютный приоритет i-й задачи (присвоенный, допустим, администратором системы), через Q_i объем ресурсов, израсходованных i-й задачей за определенный промежуток времени. Q_i представляет собой некую интегральную величину, характеризующую загруженность ИУ и модулей АП обработкой токенов и пар i-й задачи, которая рассчитывается на основе значений аппаратных счетчиков активности задач. Относительный приоритет i-й задачи равен отношению P_i к сумме абсолютных приоритетов всех N задач. Относительный объем израсходованных i-й задачей ресурсов равен отношению Q_i к общему объему ресурсов, израсходованных всеми задачами. При постановке задач на выполнение предпочтение должно отдаваться тем задачам, у которых разность между относительным приоритетом и относительным объемом израсходованных ресурсов является наибольшей. Аппаратура ВСАРР (в частности, ЦУУП) может иметь различные встроенные средства, обеспечивающие распределение ресурсов между задачами в соответствии с их приоритетами. Вместе с тем, в силу ограничений реализации, аппаратные механизмы не всегда способны обеспечить достаточный уровень точности данного распределения. Эта проблема может решаться двумя способами.

Первый способ основан на использовании *динамических* приоритетов задач. Динамические приоритеты задач – те приоритеты, которые системное программное обеспечение ВСАРР загружает в ЦУУП и на основании которых

ЦУУП принимает решения об откачке и подкачке. Системное ПО должно отслеживать прохождение задач на вычислительной системе и корректировать динамические приоритеты задач, у которых относительный приоритет не совпадает с относительным объемом израсходованных ресурсов. Для этого можно использовать адаптивный алгоритм: если относительный приоритет задачи приблизительно совпадает с относительным объемом израсходованных ею за определенный промежуток времени ресурсов, то динамический приоритет задачи необходимо оставить неизменным; в противном случае, динамический приоритет задачи необходимо увеличить или уменьшить на определенную величину.

Второй способ основан на использовании *пакетов* задач. Системное программное обеспечение ВСАРР должно периодически набирать из всех готовых к выполнению задач определенный пакет и ставить его на выполнение. Выбирая задачи для очередного пакета, необходимо учитывать приоритеты задач и объемы израсходованных задачами ресурсов. Такой подход аналогичен подходу, используемому в традиционных вычислительных системах, и не предъявляет сколько-нибудь серьезных требований к реализации ЦУУП. Однако такой подход порождает дополнительные накладные расходы, связанные с необходимостью периодически снимать с выполнения предыдущий пакет задач и ставить на выполнение новый пакет. Кроме того, неудачно выбранный пакет может загрузить вычислительную систему не полностью.

Предсказуемость планирования для ВСАРР может быть обеспечена только непрерывной ротацией задач (так же, как и для традиционных вычислительных систем). При использовании механизма динамических приоритетов задач, ротацию должно осуществлять центральное устройство управления памятью. Основываясь на значениях динамических приоритетов задач, ЦУУП должно периодически снимать с выполнения одни задачи и ставить на выполнение другие задачи, стремясь обеспечить как распределение ресурсов в соответствии с приоритетами, так и полную загрузку вычислительной системы. При использовании механизма пакетов задач, ротацию осуществляет системное ПО, выбирая задачи для включения в очередной пакет.

Балансировка нагрузки для ВСАРР сводится к тому, что системное ПО должно постоянно отслеживать загрузку ИУ и модулей АП различными задачами. Для любой конфигурации ВСАРР могут быть подобраны так называемые несбалансированные задачи, которые нагружают ВСАРР неравномерно (недогружают ИУ при полной загрузке модулей АП, недогружают модули АП при полной загрузке ИУ, и так далее). Системное ПО может обнаруживать и соответствующим образом исправлять такие ситуации.

Организация виртуальной памяти в ВСАРР

Виртуальная память ВСАРР предназначена для решения проблемы блокировки вычислительного процесса при переполнении ассоциативной памяти. Переполнение АП может возникать как в многозадачном режиме (если доступного объема АП недостаточно для одновременного прохождения всех

выполняющихся задач), так и в монозадачном режиме (если доступного объема АП недостаточно для одновременного прохождения всех подзадач выполняющейся задачи). В случае угрозы переполнения АП часть токенов откачивается из АП в ООЗУ, что предотвращает блокировку вычислительного процесса. Поскольку объем ООЗУ также является ограниченным, в ВСАРР предусмотрена возможность откачки данных из ООЗУ в дисковую память при переполнении ООЗУ.

Обмены между АП и ООЗУ целесообразно проводить под контролем ЦУУП ввиду высокой частоты этих обменов. Алгоритм откачки и подкачки, заложенный в ЦУУП, может быть как достаточно простым, так и весьма сложным. Например, возможен такой простой алгоритм: при угрозе переполнения АП откачивать задачу с наименьшим приоритетом, при освобождении АП подкачивать задачу с наибольшим приоритетом. Такой алгоритм несложно реализовать в аппаратуре, однако ротацию задач в этом случае придется выполнять программным образом. Заложенный в ЦУУП более сложный алгоритм откачки и подкачки, поддерживающий ротацию задач на основе аппаратных счетчиков активности задач, может оказаться более эффективным с точки зрения использования ресурсов вычислительной системы.

Распределение ООЗУ между задачами и между подзадачами каждой отдельно взятой задачи может быть реализовано при помощи общепринятых механизмов страничной адресации памяти. ЦУУП поддерживает так называемую *аппаратную таблицу задач* (АТЗ), в которой хранится информация о задачах, выполняющихся в потоковой вычислительной машине. АТЗ позволяет сопоставить каждой задаче несколько сегментов в ООЗУ, предназначенных для откачки токенов задачи. Системное программное обеспечение ВСАРР должно загружать в АТЗ начальные виртуальные адреса этих сегментов. Если виртуальное адресное пространство ООЗУ достаточно велико, то сегменты всех задач могут быть расположены в одном виртуальном адресном пространстве, а для преобразования виртуальных адресов в физические адреса потребуется одна таблица страниц. В противном случае, необходимо предоставлять каждой задаче свое виртуальное адресное пространство и свою таблицу страничного преобразования адресов. В любом случае, системное ПО должно осуществлять заполнение таблиц страниц и гарантировать, что таблица страниц каждой задачи содержит достаточное количество заполненных элементов (то есть, отслеживать количество страниц, используемых задачей, и заблаговременно выделять задаче дополнительные страницы).

Откачка данных из ООЗУ на диск может производиться под контролем системного ПО ввиду того, что решения об откачке на диск можно принимать достаточно редко. Для этого необходимо, чтобы системное ПО отслеживало заполненность ООЗУ и откачивало целую группу страниц на диск тогда, когда заполненность ООЗУ превышает определенный порог. Дополнительным преимуществом такого подхода является то, что в ООЗУ всегда будет существовать определенное количество свободных страниц. Таким образом,

подкачку данных с диска в ООЗУ в тех ситуациях, когда ЦУУП пытается обратиться к странице, откачанной на диск, можно производить без участия системного ПО, что позволяет избежать дополнительных накладных расходов.

Основной проблемой, связанной с откачкой страниц на диск, является проблема выбора *алгоритма откачки страниц*. Оптимальный алгоритм откачивает в первую очередь те страницы, которые максимально долго не будут использоваться в будущем (в общем случае, оптимальный алгоритм не может быть практически реализован из-за недоступности информации о том, как в будущем будет протекать процесс вычислений). Алгоритмы откачки, применяемые на практике, обычно основываются на том, что аппаратура выставляет для каждой страницы биты «R» (*referenced*) и «M» (*modified*). Бит «R» устанавливается при любом обращении к странице (по чтению или по записи), бит «M» – при записи в страницу. Этой информации достаточно для того, чтобы реализовать алгоритм, достаточно хорошо аппроксимирующий оптимальный алгоритм. Например, алгоритм «старения» страниц основан на периодическом опросе состояния битов «R» всех страниц с целью отследить те страницы, которые не использовались дольше всего. *Алгоритм подкачки страниц* в случае недоступности информации о том, как в будущем будет протекать процесс вычислений, сводится к подкачке страниц при обращении к ним. Если приблизительно известен момент времени в будущем, когда произойдет обращение к странице, то такая страница может быть подкачана заблаговременно.

Применительно к ВСАРР, стандартные алгоритмы откачки и подкачки могут быть определенным образом модифицированы, что повысит их эффективность. Во-первых, при откачке данных из АП в ООЗУ и при подкачке данных из ООЗУ в АП доступ к сегментам памяти ООЗУ, как правило, происходит по последовательным виртуальным адресам. С учетом этого обстоятельства, возникает возможность оптимизировать обмены между ООЗУ и дисками. Например, становится возможной опережающая подкачка страниц с диска в тех случаях, когда необходимо определенный сегмент, находящийся на диске, переместить в АП. Также, при откачке страниц на диск можно откачивать в первую очередь те страницы, которые находятся в «хвостах» сегментов (то есть, гарантированно будут востребованы позже всего). Во-вторых, задачи, обладающие высоким приоритетом, обычно должны откачиваться на диск лишь в редких случаях (например, если задача заблокирована ввиду ожидания внешнего события). Поэтому при выборе страниц, откачиваемых на диск, необходимо совместно анализировать временные метки последнего доступа к страницам и приоритеты задач, которым принадлежат страницы. Например, страницы высокоприоритетных задач могут быть «защищены» от откачки в течение определенного периода времени.

Таким образом, системное ПО выделяет задачам страницы в ООЗУ и откачивает страницы из ООЗУ на диск. ЦУУП осуществляет подкачку страниц с диска в ООЗУ. Чтобы обеспечить корректную совместную работу системного ПО и ЦУУП, необходимо предусмотреть *таблицу свободных страниц ООЗУ*,

хранящую физические адреса всех свободных страниц ООЗУ. Системное ПО должно забирать адреса из этой таблицы при выделении задачам новых страниц в ООЗУ и помещать адреса в эту таблицу при откачке страниц из ООЗУ на диск. ЦУУП должно забирать адреса из этой таблицы при подкачке страниц с диска в ООЗУ. Аналогично, необходимо предусмотреть *таблицу свободных страниц на диске*, хранящую свободные адреса страниц на диске. Системное ПО должно забирать адреса из этой таблицы при откачке страниц из ООЗУ на диск, а ЦУУП должно помещать адреса в эту таблицу при подкачке страниц с диска в ООЗУ. Разумеется, все три таблицы – страничного преобразования, свободных страниц в ООЗУ и свободных страниц на диске – должны изменяться согласованно, под контролем ЦУУП.

В третьей главе диссертационной работы определена структура программного комплекса системного сопровождения вычислений для ВСАРР, рассмотрена реализация компонентов программного комплекса.

Чтобы обеспечить гибкость и расширяемость программного комплекса, при его разработке использовался компонентный подход. Необходимо было четко разделить функциональные обязанности между компонентами и минимизировать межкомпонентные взаимодействия. На основе анализа требований к программному комплексу, были выделены следующие основные компоненты.

Клиент – компонента, выполняющаяся на персональном компьютере пользователя, предоставляющая программный интерфейс доступа к ВСАРР. Посредством этого интерфейса любые программы, выполняющиеся на персональном компьютере пользователя, могут взаимодействовать с основной частью программного комплекса, которая выполняется на хост-машине ВСАРР. Взаимодействие с программным комплексом значительно упрощается, поскольку клиент берет на себя такие функции, как организация удаленного доступа к ВСАРР, буферизация обмена данными с ВСАРР и так далее.

Административная подсистема – компонента, которая выполняет идентификацию и аутентификацию пользователей, осуществляющих доступ к ВСАРР. Для этого используются встроенные механизмы операционной системы хост-машины ВСАРР. Когда пользователь пытается подключиться к ВСАРР, административная подсистема определяет, какими правами доступа обладает данный пользователь, и сообщает эту информацию остальным компонентам программного комплекса.

Подсистема регистрации потоковых программ – компонента, поддерживающая совместное использование файлов различного типа (в первую очередь, совместное использование прикладных потоковых программ). Эта компонента позволяет пользователям помещать на хост-машину различные файлы, делая их доступными для считывания другими пользователями. Например, пользователь может зарегистрировать свою потоковую программу под определенным уникальным именем, используя которое, другие пользователи смогут запускать эту программу.

База данных – компонента, осуществляющая хранение различной информации – списка пользователей ВСАРР, списка зарегистрированных потоковых программ, различных административных настроек и так далее.

Процесс-представитель – компонента, которая осуществляет поддержку выполнения потоковых программ и поддерживает обмен данными между потоковыми программами, выполняющимися на потоковой вычислительной машине, и программами, выполняющимися на персональном компьютере пользователя. Процесс-представитель поддерживает окружение потоковой программы – создает начальное окружение и соответствующим образом модифицирует окружение, когда потоковая программа выполняет определенные системные вызовы. Например, процесс-представитель хранит список дескрипторов открытых потоковой программой файлов. Когда потоковая программа выполняет системный вызов доступа к файлу с определенным дескриптором, процесс-представитель непосредственно выполняет запрошенную операцию. Также, процесс-представитель поддерживает преобразование данных из вида, специфичного для потоковой программы (токены определенного формата) в универсальный формат (массивы, списки) и обратно (правила преобразования описываются программистом в самой потоковой программе). Благодаря этому, потоковая программа и программа, выполняющаяся на персональном компьютере пользователя, могут обмениваться данными в удобном для себя формате.

Расширение процесса-представителя – компонента, подключаемая к процессу-представителю с целью расширения его возможностей. Может поддерживать нестандартные форматы обмена данными, нестандартные системные вызовы и так далее.

Супервизор – ключевая компонента программного комплекса, непосредственно отвечающая за прохождение задач на потоковой вычислительной машине. Основными функциями супервизора являются:

- Взаимодействие с потоковой вычислительной машиной – прием и выдача данных и различной управляющей информации.
- Управление задачами, выполняющимися на потоковой вычислительной машине – в том числе, планирование задач.
- Управление памятью потоковой вычислительной машины – в том числе, совместная с ЦУУП организация виртуальной памяти.

В целом, функционирование программного комплекса выглядит следующим образом. Пользователь выполняет свою работу при помощи определенной программы, запущенной на его персональном компьютере. В определенный момент времени, эта программа обращается к клиенту с запросом на запуск потоковой программы. На основании этого запроса, клиент формирует заявку на выполнение потоковой программы и передает ее процессу-представителю. Процесс-представитель создает начальное окружение потоковой программы, генерирует образ потоковой программы и передает его супервизору. Супервизор загружает полученный образ в потоковую вычислительную машину. В процессе своего выполнения, потоковая программа может пересылать токены на хост-машину. Супервизор передает эти токены

процессу-представителю, который определяет тип полученного токена. Токены системных вызовов собираются в специальном буфере и затем обрабатываются. Токены-данные преобразуются в универсальный формат представления данных и отсылаются клиенту.

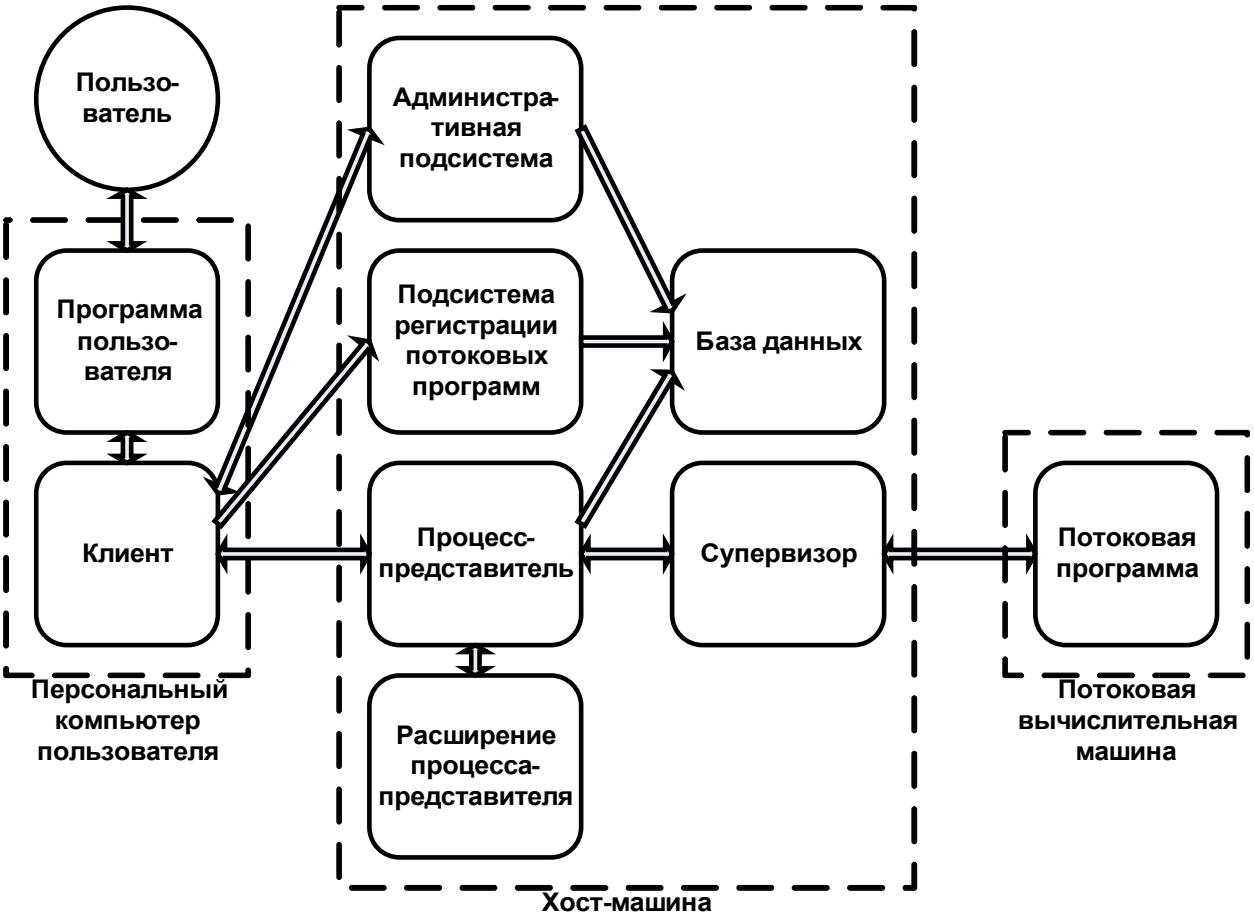

Рисунок 3. Структура программного комплекса

При разработке структуры программного комплекса предполагалось, что основным способом программного доступа к ВСАРР будет доступ через интерфейс клиента, который, как позднее было уточнено, является COM-интерфейсом. Такой интерфейс вполне удовлетворяет требованиям универсальности и надстраиваемости, так как эти требования изначально заложены в COM-технологию. Вместе с тем, COM-интерфейсы не всегда отвечают требованию удобства использования, поскольку не все языки и среды программирования имеют развитую поддержку технологии COM. Поэтому для определенных языков целесообразно было разработать обертки над интерфейсом клиента. Например, такая обертка была разработана для языка Фортран. Эта обертка предоставляет возможность просто и компактно записать процедуру взаимодействия с ВСАРР с использованием языка Фортран.

В четвертой главе диссертационной работы приведены результаты исследования выполнения различных программ на инструментальном испытательном комплексе ВСАРР с использованием программного комплекса системного сопровождения вычислений для ВСАРР.

Для отработки различных научно-технических решений и проверки работоспособности архитектуры ВСАРР в отделе «Проблем построения информационно-вычислительных систем высокого параллелизма» Института проблем информатики Российской академии наук был создан инструментальный испытательный комплекс ВСАРР. Он включает в себя программную имитационную модель ВСАРР, программную блочно-регистровую модель ВСАРР, выполненный на ПЛИС фирмы Altera макет ВСАРР.

С целью проверки работоспособности разработанного программного комплекса на инструментальном испытательном комплексе ВСАРР были пропущены различные классы задач. При этом основной интерес представляло исследование работы программного комплекса в многозадачном режиме. Для проведения исследований использовались задачи, характеризующиеся различными требованиями к доступному объему ассоциативной памяти. Исследование результатов экспериментов показало, что многозадачный режим работы ВСАРР в ряде случаев позволяет повысить загрузку вычислительной системы. Это объясняется тем, что если каждая из задач по отдельности не загружает вычислительную систему полностью, то параллельное выполнение нескольких задач позволяет добиться полной загрузки.

В работе приводятся результаты параллельного выполнения трех задач – обращение матрицы, пузырьковая сортировка и поиск простых чисел – на конфигурации с 64 ИУ и 64 МАП с различными размерностями задач.

Таблица 1. Загрузка системы в многозадачном режиме работы

Номер эксперимента	Размерность задач	Время работы, тактов	Загрузка ИУ
1	10, 200, 1000	13196	89
2	20, 300, 2000	35088	94
3	30, 400, 3000	71384	95
4	40, 500, 4000	121968	96
5	50, 600, 5000	190370	97

В данной серии экспериментов использование многозадачного режима обеспечивает заметный прирост производительности по сравнению с монозадачным режимом. Ниже приводится соответствующий график загрузки исполнительных устройств.

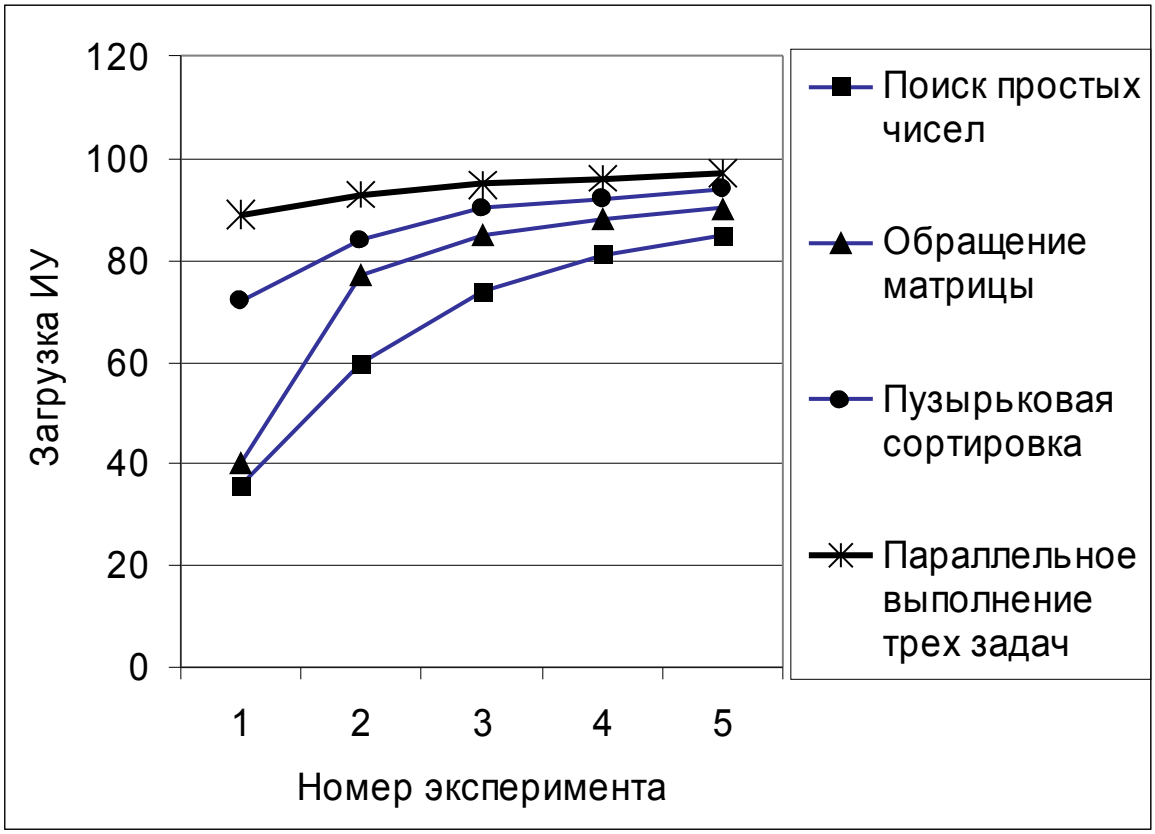

Рисунок 4. Сравнение загрузки системы в монозадачном и многозадачном режиме работы

В конце главы приводится формула расчета коэффициента эффективности разработанного программного комплекса. Данный коэффициент получен из отношения чистого времени выполнения задачи к величине накладных расходов, добавляемых программным комплексом к чистому времени выполнения задачи:

$$K_{эфф} = \max\left(\frac{T_{узл}}{p \times N_{ИУ}}, \frac{T_{соп}}{i \times N_{МАП}}, \frac{T_{ком}}{c}\right) \times \frac{\min\left(\Lambda_{ЦУУП}, \Lambda_{ХОСТ}\right)}{\alpha V_{пр} + \beta V_{дн} + V_{упр}}$$, где

$T_{узл}$ – суммарное время выполнения программ узлов (сек);

p – коэффициент многонитевой обработки пар в ИУ;

$N_{ИУ}$ – количество ИУ;

$T_{соп}$ – суммарное время сопоставления токенов (сек);

i – коэффициент интенсивности откачки-подкачки;

$N_{МАП}$ – количество МАП;

$T_{ком}$ – суммарное время коммутации токенов и пар (сек);

c – коэффициент топологии коммутационной среды токенов и пар;

$\Lambda_{ЦУУП}$ – скорость обработки информации ЦУУП (байт/сек);

$\Lambda_{ХОСТ}$ – скорость обработки информации хост-машиной (байт/сек);

$V_{пр}$ – объем загрузочного модуля программы (байт);

α – коэффициент обработки загрузочного модуля программным комплексом;

$V_{дн}$ – объем начальных данных и результатов вычислений (байт);

β – коэффициент обработки начальных данных и результатов вычислений программным комплексом;

$V_{упр}$ – суммарный объем управляющей информации, передаваемой между хост-машиной и ЦУУП (байт).

Проведенные исследования показали, что разработанный программный комплекс является работоспособным, и подтвердили правильность положений, сформулированных во второй главе диссертационной работы. В ряде экспериментов использование программного комплекса позволило получить прирост производительности ВСАРР в среднем до 20%.

В заключении обобщены основные результаты проведенных исследований и разработок, сформулированы выводы по всей работе.

Основные результаты диссертационной работы

1) Проведено исследование ранее разработанных потоковых архитектур и моделей вычислений, определены их основные недостатки.
2) Предложены принципы функционирования программного комплекса системного сопровождения вычислений для ВСАРР, в том числе, принципы планирования задач в многозадачном режиме работы ВСАРР и принципы организации программной поддержки виртуальной памяти ВСАРР, учитывающие аппаратные механизмы функционирования многозадачного режима работы и виртуальной памяти ВСАРР.
3) Предложена структура программного комплекса системного сопровождения вычислений для ВСАРР, позволяющая организовать многозадачный многопользовательский режим работы ВСАРР.
4) Созданы программные интерфейсы доступа к ВСАРР, позволяющие обеспечить удаленный доступ к ВСАРР и упростить интеграцию ВСАРР в традиционную вычислительную среду.
5) Реализован программный комплекс системного сопровождения вычислений для ВСАРР, создана интегрированная среда разработки для ВСАРР.
6) Проведено исследование выполнения различных программ на инструментальном испытательном комплексе ВСАРР с использованием программного комплекса системного сопровождения вычислений для ВСАРР, подтверждена работоспособность программного комплекса и получены оценки его эффективности.

Список работ, опубликованных по теме диссертации

1. Ширай А.Е, Петрищев Д.В. Проблемы построения системного программного обеспечения вычислительных систем с автоматическим распределением ресурсов // Сборник «Системы и средства информатики» 2004. Специальный выпуск: «Методы и средства разработки информационно-вычислительных систем и сетей». Москва: Изд-во ИПИ РАН, 2004. – стр. 213-220 (автору диссертации принадлежит описание структуры системного программного обеспечения вычислительных систем с автоматическим распределением ресурсов).
2. Петрищев Д.В., Ширай А.Е. Программные аспекты интеграции вычислительной системы с нетрадиционной архитектурой в традиционную вычислительную среду // Журнал «Искусственный интеллект» 3’2004. Донецк: Изд-во ИПИИ, 2004. – стр. 229-239 (автору диссертации принадлежит описание функций системного программного обеспечения вычислительной системы с нетрадиционной архитектурой).
3. Петрищев Д.В. Реализация сервера вычислений на базе вычислительной системы с автоматическим распределением ресурсов // Материалы Международной научной конференции «Искусственный интеллект. Интеллектуальные и многопроцессорные системы – 2004». Т.1. Таганрог: Изд-во ТРТУ, 2004. – стр. 111-114.
4. Петрищев Д.В. Системная поддержка выполнения потоковых программ на сервере приложений на базе системы с автоматическим распределением ресурсов // Материалы Международной научно-технической конференции «Интеллектуальные и многопроцессорные системы – 2005». Т.1. Таганрог: Изд-во ТРТУ, 2005. – стр. 328-331.
5. Петрищев Д.В. Возможные подходы к интеграции вычислительной системы с автоматическим распределением ресурсов в традиционную вычислительную среду // Материалы девятого научно-практического семинара «Новые информационные технологии в автоматизированных системах». Москва: Изд-во МГИЭМ, 2006. – стр. 283-288.
6. Петрищев Д.В. Проблемы управления виртуальной памятью в вычислительных машинах с архитектурой потока данных // Научно-технический журнал «Информационные технологии моделирования и управления», 2006, №9(34). Воронеж: Изд-во «Научная книга», 2006. – стр. 1179-1182.
7. Петрищев Д.В. Управление задачами и памятью в вычислительной системе с автоматическим распределением ресурсов // Журнал «Системы управления и информационные технологии», 2006, №4.2(26). Воронеж: Изд-во «Научная книга», 2006. – стр. 261-266.

Перечень сокращений

COM – Component Object Model (компонентная объектная модель).

АП – ассоциативная память.

АТЗ – аппаратная таблица задач.

ВСАРР – вычислительная система с автоматическим распределением ресурсов.

ИУ – исполнительное устройство.

МАП – модуль ассоциативной памяти.

ООЗУ – общее оперативное запоминающее устройство.

ПЛИС – программируемая логическая интегральная схема.

ПО – программное обеспечение.

ЦУУП – центральное устройство управления памятью.