На правах рукописи

САХАРОВ Илья Евгеньевич

УПРЕЖДАЮЩЕЕ КЭШИРОВАНИЕ В ПОДСИСТЕМЕ ВНЕШНЕЙ ПАМЯТИ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Специальность: 05.13.18 – Математическое моделирование, численные методы и комплексы программ

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Иваново, 2009

Работа выполнена в Костромском государственном технологическом университете.

Научный руководитель: *доктор технических наук, профессор* Шведенко Владимир Николаевич

Официальные оппоненты: *доктор технических наук, профессор* Пантелеев Евгений Рафаилович

кандидат технических наук, доцент Жирков Владислав Федорович

Ведущая организация: Московский государственный технический университет им. Н.Э. Баумана

Защита состоится " 25 " сентября 2009 года в 11 часов на заседании диссертационного совета Д 212.064.03 в по адресу: 153003, г. Иваново, ул. Рабфаковская, 34, ауд. Б-237.

С диссертацией можно ознакомиться в библиотеке ИГЭУ, с авторефератом можно ознакомиться на сайте ИГЭУ www.ispu.ru

Автореферат разослан « __» _____ 2009 года.

Ученый секретарь
диссертационного совета
кандидат технических наук, доцент Шульпин А.А.

Введение

Актуальность диссертационного исследования. Одной из современных тенденций развития высокопроизводительных систем является интеграция различных ресурсов в единую сетевую среду распределенных вычислений (ССРВ). Данная среда позволяет использовать разнородные средства как единый вычислительный ресурс. ССРВ состоит из множества территориально размещенных вычислительных систем, соединенных различными каналами связи, характеристики которых могут динамически изменяться в широких пределах и представляет собой GRID-систему.

Приложения (задачи) пользователей выполняются на одной или нескольких вычислительных системах, входящих в состав ССРВ. Из большого территориального расположения компонент ССРВ возникает задержка передачи данных по каналам связи и задержка обращения к дисковым накопителям.

Существует большой класс фундаментальных научных и инженерных задач с широкой областью применения, эффективное решение которых возможно только с использованием мощных вычислительных ресурсов. Для приложений, реализующих подобные задачи, требуются данные, значительно превышающие объем оперативной памяти или требуемые данные имеют огромный объем и расположены территориально удаленно от вычислителей. Примерами таких задач являются предсказания погоды, климата и глобальных изменений в атмосфере, построение полупроводниковых приборов, структурная биология, генетика человека, квантовая хромодинамика, астрономия, транспортные задачи, гидро- и газодинамика, управляемый термоядерный синтез, распознавание изображений, распознавание и синтез речи, математика, поиск различных чисел.

Разница между временем вычисления одной процессорной операции и временем доступа к единице дисковой памяти постоянно увеличивается. Эта тенденция служит причиной того, что приложения должны осуществлять дополнительную выборку данных из дисковой памяти, до момента обращение к этим данным. Осуществление предвыборки данных является хорошо известной техникой скрытия дисковых задержек, и потенциально может увеличить производительность вычислений. Вместо выборки данных с диска по запросу используется механизм упреждающего кэширования, включающий в себя выборку данных с дисков в предвидении будущей попытки обращения к этим данным. Поэтому необходимость ускорения доступа к распределенным данным обуславливает актуальность диссертационного исследования.

Объект исследования – подсистема внешней памяти высокопроизводительных распределенных вычислительных систем.

Предмет исследования – модели, методы и алгоритмы упреждающего кэширования.

Цель диссертационного исследования – повышение производительности исполнения приложений с помощью метода упреждающего кэширова-

ния. Для достижения цели диссертационного исследования необходимо решить следующие **задачи диссертационного исследования**:

1. Разработать имитационную модель упреждающего кэширования.
2. На основе имитационной модели разработать новый метод упреждающего кэширования для разнообразных приложений с интенсивным обменом данными, который давал бы возможность осуществлять предвыборку требуемых данных.
3. Реализовать алгоритм упреждающего кэширования, который автоматически обеспечивает функционирование приложений с применением упреждающего кэширования.
4. Разработать организацию упреждающего кэширования для эффективной работы подсистемы ввода/вывода ССРВ.

Методами исследования являются системный анализ, методы имитационного моделирования, аналитические методы разработки и оптимизация разрабатываемых алгоритмов и средств.

Научная новизна:

1. Разработана имитационная модель подсистемы внешней памяти с использованием упреждающего кэширования для вычислительного кластера. Модель позволяет заранее анализировать эффективность предлагаемой реализации метода упреждающего кэширования.
2. Разработан новый алгоритм автоматического формирования последовательности предвыборок, позволяющий в реальном масштабе времени определять точную последовательность будущих обращений к данным.
3. Разработан метод упреждающего кэширования для однопотоковых и многопотоковых приложений с интенсивным вводом/выводом.

Реализация результатов работы. Результаты, полученные в ходе проведения исследования, были экспериментально апробированы в Институте химической физики им. Н.Н.Семенова г. Москва, в учебном процессе Академии ФСБ РФ.

Апробация работы. Основные положения диссертационной работы изложены в докладах на следующих научно-практических конференциях: пятой Всероссийской научной конференции молодых ученых СПбГУ 2008, научно-технической и информационное обеспечение деятельности спецслужб» 2-3 февраля 2006 года ИКСИ АФСБ РФ, второй Всероссийской научной конференции факультета ВМК МГУ им. М.В. Ломоносова.

Публикации. По теме диссертации опубликовано 8 статей.

Структура и объем работы. Диссертационная работа состоит из введения, четырех глав, выводов, заключения и списка библиографии на 137 страницах, содержит 27 рисунка, 11 таблиц, список библиографии из 79 наименований.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введение обоснована актуальность выбранной темы, определены цели и задачи исследования.

В первой главе «Обзор современных технологий повышения производительности распределенных вычислений» описываются проблемы работы с данными, сопровождающие выполнение различных приложений с интенсивным вводом/выводом, а также рассматриваются существующие алгоритмы формирования предвыборок. Использование файловой предвыборки является сложно реализуемым методом из-за ряда недостатков.

Упреждающее кэширование увеличивает производительность выполнения приложений за счет скрытия доступа к дисковой памяти. Если приложению требуются данные, предвыборка которых еще не завершена, то приложение будет ожидать, лишь оставшееся время. Если же требуемые данные уже были выбраны, т.е. находятся в оперативной памяти (другими словами было осуществлено кэширование этих данных), то обращение к дискам будет полностью скрыто от приложения.

Все преимущества упреждающего кэширования построены на использовании дополнительных ресурсов. С одной стороны, упреждающее кэширование может увеличить производительность выполнения приложений, а с другой стороны, объема свободных ресурсов может не хватить для исполнения некоторых приложений.

Упреждающее кэширование основывается на алгоритме предсказания данных. Для того, чтобы получить увеличение эффективности выполнения приложения, необходимо получить информацию о данных к которым будет происходить обращение в будущем. Все подходы условно можно разделить на три категории: методы, основные на примерах, динамические методы, основанные на истории обращений и методы статического анализа исполняемого приложения.

Предлагается использовать новый четвертый способ упреждающего кэширования на основе разделения потоков приложения. Суть метода заключается в параллельном запуске нового упреждающего и вычислительного потоков. Предполагается, что аппаратные ресурсы позволяют без потерь производительности исполнять одновременно несколько потоков. Способ разделения потоков напоминает технику спекулятивного исполнения ветвей программы в микропроцессорах. Используя способ разделения потоков, мы пытаемся, некоторым образом, повторить подход спекулятивного исполнения ветвей программы и применить его для приложений с интенсивным обменом ввода/вывода.

Во второй главе «Разработка имитационной модели и исследование эффективности применения упреждающего кэширования для ССРВ» описывается имитационная модель упреждающего кэширования для ССРВ. Также дается представление об алгоритме моделирования, приводиться схема имитационной модели в терминах алгоритма моделирования и средства моделирования и описывается программная реализация модели. Дается полное представление системной модели обращения к данным, а также математической оценки эффективности упреждающего кэширования. Исследуется эффективность применения упреждающего кэширования для ССРВ. Описы-

вается механизм функционирования упреждающего кэширования в ССРВ, реализованный в имитационной модели. Время выполнения приложения T, равняется

$$T = N_{I/O}(\overline{T_{CPU}} + \overline{T_{I/O}}) \quad (1)$$

где $N_{I/O}$ - количество операций ввода-вывода, $\overline{T_{CPU}}$ - среднее время исполнения процессором приложения, между двумя последовательными запросами к данным, $\overline{T_{I/O}}$ - среднее время на выполнение одного запроса на ввод/вывод.

Время выполнения запросов выборки данных зависит от того, где располагаются эти данные. Пусть T_{HIT} - время выборки одного блока данных из кэш-памяти. Задержка выборки блока данных с диска равняется T_{disk}. Время передачи запроса по сети и время передачи блока данных равняется $T_{Network}$, а время обработки запроса удаленным компьютером или серверов T_{server}. Для выборки буфера и обслуживания этого запроса требуется процессорное время T_{driver} - время обслуживание запроса драйвером устройства. Итак, время выполнения запроса, если блок данных располагается не в кэш-памяти, в общем случае, будет равняться:

$$T_{MISS} = T_{HIT} + T_{Network} + T_{server} + T_{disk} + T_{driver} \quad (2)$$

Коэффициент попадания в кэш-память зависит от среднего времени доступа. Максимальный выигрыш от применения упреждающего кэширования для одной сформированной последовательности обращений к файлам не будет превышать $T_{Network} + T_{disk} + T_{server}$. Обозначим через B_x - временное преимущество от применения упреждающего кэширования, тогда

$$B_x = (T_{disk} + T_{network} + T_{server}) - x \cdot (\overline{T_{CPU}} + T_{HIT} + T_{driver}) \quad (3)$$

$$P = x = (T_{disk} + T_{network} + T_{server}) / T_{HIT}), \text{ при } B_x = 0 \quad (4)$$

P – горизонт предвыборки, определяет количество запросов после которых, предвыборка является не эффективной.

На основе полученных оценок производиться анализ эффективности упреждающего кэширования, а горизонт предвыборки используется для генерации последовательности предвыборок и правильного расчета размера очереди упреждающих вызовов.

При исследовании эффективности упреждающего кэширования посредством имитационного моделирования определяющее значение имеет адекватность используемой модели параллельных программ. Параллельная программа представляет собой совокупность одинаковых программ-ветвей, каждая из которых выполняется на одном вычислительном модуле ВМ. Программа-ветвь делится на строки, где каждая строка соответствует одной из операции ввода-вывода с файлом. Для выполнения одной строки программы и перехода к следующей необходимо, чтобы требуемые ей блоки данных

располагались либо в локальной кэш-памяти, либо в кэш-памяти на управляющей машине. Частота обращений к файлам задается с помощью таблицы Stat (обозначение, используемое в коде имитационной модели), в строках которой задается последовательность обращения к файлам.

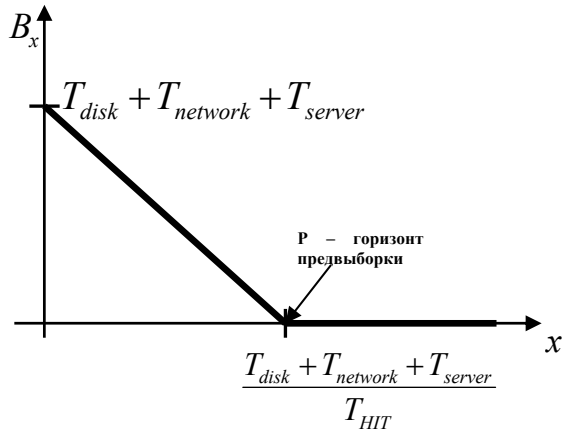

Рисунок 1 - Функция выигрыша применения упреждающего кэширования

На основе системной модели обращения к данным была создана имитационная модель модуля упреждающего кэширования на языке GPSS. Модуль упреждающего кэширования представляет собой многоканальную систему массового обслуживания с ожиданием. Модель поделена на сегменты. Изначальным транзактом является, так называемый, транзакт-паспорт задания, от него создаются копии, а сам он уничтожается. Число копий равно количеству узлов, на которых запускается задача. Каждый из созданных транзактов-паспортов занимает свой узел и остается в нем до момента полного выполнения задания на данном узле. Имитационная модель состоит из 12 сегментов. Структурная схема имитационной модели изображена на рис.2.

Для экспериментов были разработаны две модели: модель выполнения задания на вычислительном кластере с применением упреждающего кэширования и модель выполнения задания на вычислительном кластере без него. Успех применения упреждающего кэширования можно измерить путем сравнения модельного времени выполнения программы с применением упреждающего кэширования и модельного времени выполнения этой же программы без применения него.

Имитационное моделирование показало, что применение механизма упреждающего кэширования увеличивает эффективность распределенной обработки при определенных параметрах функционирования модели, что косвенно подтвердило теоретические выводы первой главы. Возможности оптимизации применённых алгоритмов далеко не исчерпаны.

Определяющее значение, для получения максимальной эффективности исполнения приложений, имеет выбор и реализация стратегии упреждения.

При выборе алгоритма предподкачки необходимо учитывать недостатки, связанные с введением упреждающего кэширования и стремиться их минимизировать. Одним из возможных способов оптимизации разработанного алгоритма является введение специализированного буфера, в который будут помещаться предподкачиваемые данные, и оставаться там до момента их первого использования.

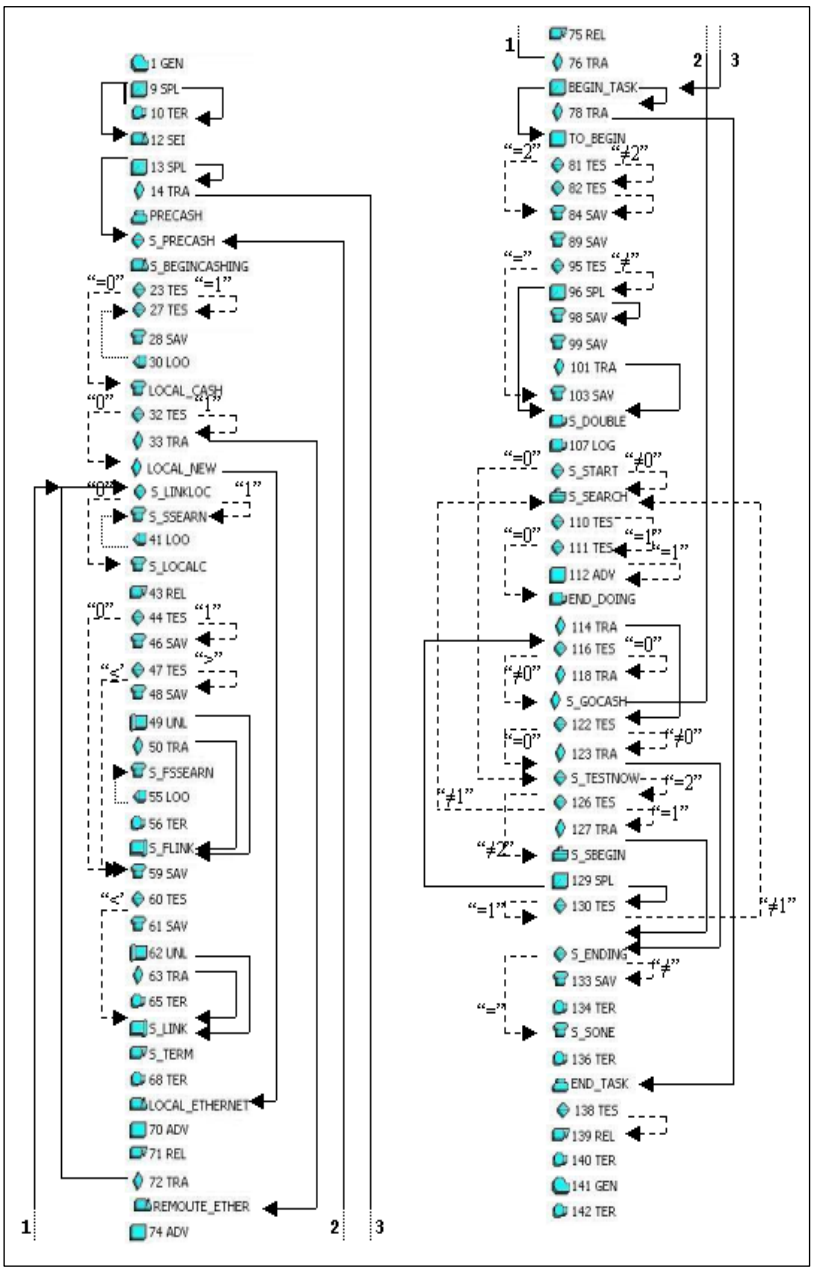

Рисунок 2 - Структурная схема имитационной модели

В третей главе «Программная реализация метода упреждающего кэширования» описывается предлагаемый метод упреждающего кэширования, преимущества и недостатки метода.

Упреждающее кэширование представляет собой процесс независимого *заблаговременного* копирования данных из одной памяти в другую в течение всего процесса выполнения приложения. В этом определении подчеркиваются три момента. Во-первых, упреждающее кэширования осуществляет копирование данных до реального момента обращения к этим данным. Во-вторых, упреждающее кэширование осуществляется непрерывно в течение всего времени выполнения приложения. В-третьих, упреждающее кэширования может выполняться *независимо* от процесса выполнения приложения. Упреждающее кэширование интегрирует в себе буферизацию, кэширование и упреждающее чтение, с возможностью независимого исполнения.

Для упреждающего кэширования обязательно должна быть определена стратегия упреждения, т.е. алгоритм по которому выбираются следующие кэшируемые блоки. Именно стратегия упреждения определяет эффективность метода упреждающего кэширования и является научной основой данного метода. Стратегия упреждения – алгоритм поиска блоков данных, которые будут затребованы приложением в ближайшем будущем. В зависимости от выбранного алгоритма определяется структура реализации метода упреждающего кэширования.

Все современные разработки (практические реализации) принято делить на три направления:

Предсказуемое упреждающее кэширование: Самые ранние разработки систем упреждающего кэширования основывались на методах, основанных на примерах и методах основанных на истории обращений.

Специальные приложения, контролирующие упреждающие выборки и кэширование: подобными системами занимался Patterson. Он модифицировал два unix приложения: make и grep, таким образом, что файловой системе передавалась информация о файлах, к которым собирается обратиться приложение заранее. Такой подход повысил эффективность исполнения приложения от 13 до 30%.

Специальное компилирование приложений: Основная идея этого направления заключается в исключении роли разработчика приложения для использования упреждающего кэширования. Chang и Gibson разработали средство SpecHint, которое специальным образом изменяло откомпилированные файлы, что позволяло использовать так называемое спекулятивное выполнение задачи в моменты блокирующих операций ввода/вывода.

Существующие методы доступа к данным, основанные на примерах, истории и статистическом анализе, не позволяют решить задачу эффективного доступа к данным. Предлагаемый метод упреждающего кэширования, основанный на разделении потоков, позволяет решить задачу эффективного доступа к данным для большого множества различный приложений с интенсивным обменом данными. Основная идея заключается в разложении приложения на два потока: вычислительный поток, содержащий неизмененный код оригинальной программы, включающий все вычислительные операции и все операции ввода/вывода и упреждающий поток, содержащий все оригинальные инструкции, которые имеют отношение к вводу/выводу. Упреждающий поток запускается одновременно с выполнением вычислительного кода программы. Упреждающий поток формирует именно те данные, которые будут затребованы в вычислительном потоке. Существуют ситуации, когда упреждающий поток не может сформировать упреждающие выборки. Например, когда имя требуемого файла является входящей переменной или когда существуют внутри циклов зависимости по данным. В этих ситуациях оба эти потока необходимо синхронизировать. К сожалению, запуск процесса синхронизации потоков является частым для приложений с интенсивным вводом/выводом. Главная особенность упреждающего кэширования с применением двух потоков заключается в том, что исключается необходимость вручную изменять код программы. Упреждающая выборка для программы строится с использование анализатора кода.

Упреждающий поток исполняется быстрее вычислительного потока, поэтому данные, к которым, обращается вычислительный поток, всегда уже находятся в кэш-памяти, так как к ним уже было обращение при выполнении упреждающего потока. Данные помещаются в буферизированный файл упреждающего кэширования. Разработан транслятор, который осуществляет

разделение на два потока и создания между ними канала, через который осуществляется взаимодействие и синхронизация выполнения обоих потоков. Упреждающее кэширование основывается на четырех компонентах: транслятор исходного кода, библиотека упреждающего кэширования, загрузочном модуле и модуле упреждающего кэширования (рис. 3).

Транслятор кода создает упреждающий поток из приложения, путем выборки тех частей кода, которые относятся к вводу/выводу и урезанию всех вычислительных частей. Для функционирования транслятора кода необходимо наличие исходного кода приложения. Причем, все вызовы ввода/вывода заменяются вызовами из библиотеки упреждающего кэширования. Само исходное приложение формирует вычислительный поток. Имеется однозначное соответствие между упреждающими вызовами в упреждающем потоке и вызовами ввода/вывода из оригинального приложения.

Синхронизация позволяет избежать отставания упреждающего потока от вычислительного потока. Основная идея синхронизации строится на метках. Каждому упреждающему запросу ставиться некоторый индентификационный номер ID соответствующего упреждающего вызова, исполняющего этот запрос.

Рассмотрим основные операции упреждающего потока и их взаимодействия с вычислительным потоком, и связь с библиотекой упреждающего кэширования. В таблице 1 показано как строится вычислительный и упреждающий поток. Программный интерфейс упреждающего потока включает в себя 4 функции библиотеки:

1. create_prefetch_thread(prefetch_function): эта функция позволяет приложению создать упреждающий поток и выполнить prefetch_fuction.
2. prefetch_xxx(): это множество функций, которые заменяют в упреждающем потоке все стандартные функции ввода/вывода.
3. inform_open(file_pointer) и inform_close(file_pointer): Эти функции необходимы для того, чтобы информировать упреждающий поток об открытии или закрытии файлов вычислительным потоком.
4. synchronize(synchronization_point, type): эта функция синхронизации двух потоков.

Вычислительный поток и упреждающий поток функционируют независимо друг от друга до точки синхронизации. Все три следующих вида представляют собой точки синхронизации:

1. Открытие файла: Упреждающий поток должен ожидать пока вычислительный поток откроет файл. Потоки синхронизируются вызовом synchronize() с указанием номера синхронизационной точки (табл. 1).
2. Ввод информации пользователем: Упреждающий поток вынужден ожидать, пока вычислительный поток вычислит адрес на основе входной информации(stdin) или считает его из специального файла. Если адрес следующих данных зависит от данных файла, для которого уже осуществляется предвыборка, тогда в каждый из потоков вставляются

точки синхронизации. Если входные данные считываются из конфигурационных файлов, тогда также добавляются точки синхронизации.

3. Чтение после записи: Если программа содержит только вызовы чтения из одного файла и запись в разные файлы или операции чтения и записи не пересекаются для некоторого файла, тогда во всех этих случаях синхронизация не нужна, в остальных случаях синхронизация необходима.

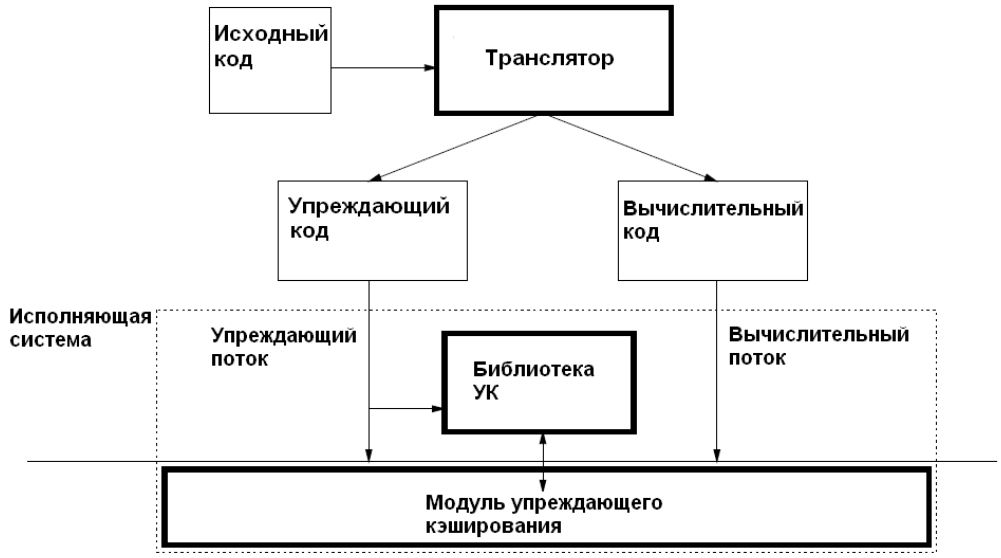

Рисунок 3 - Системная архитектура

Оба потока могут обмениваться данными через глобальные переменные, используемые в приложении. Кроме этого между потоками создается коммуникационный канал. Существует общий буфер для этих каналов, создается он в момент создания упреждающего потока. Обычно в качестве обменной информации выступают файловые дескрипторы и входные параметры. Существуют функции обеспечивающие обмен данными между потоками.

Транслятор основывается на двух алгоритмах: внутренний и внешний анализ зависимостей функции. Цель внутренннего анализа заключается в построение зависимостей внутри функции. В результате анализа определяются все переменные и выражения, относящиеся к дисковому вводу/выводу.

Этот алгоритм запускается итеративно для каждой процедуры до тех пор, пока не дойдет до конца приложения. Для создания правильного упреждающего потока, кроме внутреннего анализа функций необходимо произвести межфункциональный анализ зависимостей. Анализ построен на поиске информации о входных переменных функций, возвращаемых значениях и глобальных переменных. Анализатор кода включает упрощенную поддержку многопотоковых приложений, написанных под спецификации MPICH.

Все вызовы упреждения, которые использует упреждающий поток, содержатся в библиотеке упреждающих вызовов, размещаются и хранятся в памяти очередь команд на упреждение. Структура очереди команд на упреждение имеет следующий вид: {идентификатор файла, номер блока, идентификатор вызова}. Идентификатор файла и блока определяют реальный физи-

ческий блок данных, который необходимо предвыбрать. Идентификатор вызова определяет вызов, который инициировал запуск механизма упреждения для данного блока. Начальное значение идентификаторов вызовов начинается с 0. Файловая таблица для каждого файла дескриптора содержит смещение. Используя эту информацию всегда можно вычислить номер блока для вызовов prefetch_read и prefetch_lseek.

Таблица 1

Пример вычислительного и упреждающего потоков

	int fp;		
	Вычислительный поток(ВП)**		**Упреждающий поток(УП)**
	void main(void){		Void prefetch_fuction(void){
	int i; int data[100];		int I;
	/*создание упреждающего потока*/		
##C1	create_prefetch_thread(		
	(void*)prefetch_function);		
	/*открытие файла данных*/		
C2	fp=open("mydata.dat", O_RDONLY);		
	/*сигнал упреждающему потоку*/		
##C3	synchronize(1, signal);		/*ожидание открытия файла*/
		$$P1	synchronize(1, wait);
			/*указание об открытии файла*/
		$$P2	inform_open(fp);
C4	for(i=99; i>=0; i--){		/*предвыборка*/
	/*ввод-вывод*/	P3	for(i=99; i>=0; i--){
C5	lseek(fp, i*4, SEEK_SET);		
C6	read(fp, &data[99-i], 4);		/*только ввод-вывод*/
	/*вычисления*/	P4	prefetch_lseek(fp,i*4,SEEK_SET);
C7	data[99-i] = data[99-i]*i;	P5	prefetch_read(fp, 4);
	}		
	/*закрытие файла*/		}
C8	close(fp);		/*указание о закрытие файла*/
	}	$$P6	inform_close(fp);
			}

**Слева вычислительный поток, справа упреждающий поток. Внутри функции main строчки без ## являются оригинальным кодом приложения. В этот код были вставлены изменения по созданию упреждающего потока. Строчки без символа $$ были взяты с оригинального кода приложения. В упреждающем потоке не присутствуют вычисления.

В четвертой главе «Проверка соответствия разработанной имитационной модели и предлагаемого метода упреждающего кэширования» экспериментальное доказательство эффективности предлагаемого метода. Разработан вариант модуля упреждающего кэширования на основе разделения потоков под версию ядра Linux 2.4. Необходимо загрузить разработанные модификации ядра Linux и специальный драйвер устройства. Для проверки разработанного модуля упреждающего кэширования использовались следующие тестовые программы:

1. XDataSlice (версия 2.2) – средство визуализации трехмерных массивов. Позволяет динамически загружать большие массивы данных.
2. Приложение для расчета быстрого преобразования Фурье FFT (fast Fourier transform).
3. Agrep (версия 2.0.4) – программа поиска в текстовых файлах. Данная утилита осуществляет поиск текста по шаблонам среди большого количества текстовых файлов.

4. PostgreSQL (версия 7.0.3) – расширенная версия СУБД Postgres. Запускался процесс расчета индексов для четырех частей базы.
5. Mpich2 – реализация MPI. Использовалась программа перемножения двух матриц, файлы которых располагаются на множестве узлов ввода/вывода.

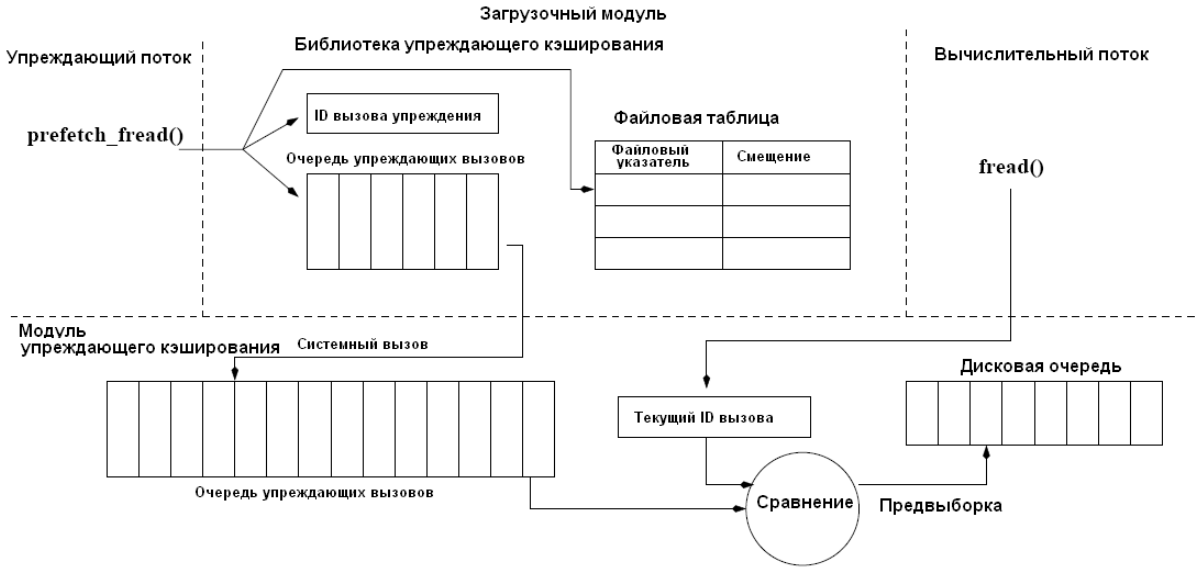

Рисунок 4 - Архитектура исполняющей системы

В работе метод проверялся на решении задач физики газовых лазеров. Основным препятствием к увеличению мощности лазеров с диффузионным охлаждением является теплоотвод, поскольку увеличение поступательной температуры в неравновесной области приводит к резкому увеличению скорости сброса энергии (тепловому взрыву) и, в итоге, к контракции разряда и прекращению генерации. Конвекция в системах с энерговыделением изучалась в плоском слое с постоянной мощностью источника, а также с учетом накачки энергии в колебательные степени свободы молекул. Рассматривались также задачи с сильным энерговыделением, встречающиеся в энергетике. Структура конвективных потоков зависит от геометрии объекта. Применительно к горизонтальному цилиндру, как правило, рассматривались либо задачи с коаксиальными цилиндрами, очень необходимые для техники, либо задачи теплопереноса при неравномерном нагреве стенок цилиндра.

Исследовалось влияние конвекции на теплоотвод в коаксиальном цилиндре с учетом объемного энерговыделения. Рассматривается жидкость в пространстве между двумя коаксиальными цилиндрами радиусов R_i и R_o. Сами цилиндры поддерживаются при температурах T_o и T_i . В системе происходит постоянное энерговыделение с объемной мощностью Q, не зависящей от координат среды. После обезразмеривания и перехода к новым переменным, исходная система сводилась к системе трех уравнений второго порядка:
На рис. 5 представлена поверхность $Ra_T(Ra,\sigma)$, которая отделяет точки, соответствующие двумерной конвекции, от трехмерных режимов. Приведенные в работе расчеты касались двумерной модели (R, θ) и не очень больших чисел Рэлея. Кроме того, достаточно просто учитывалось энерговыделение в

системе. Увеличение Ra_T приводит к изменению структуры конвективного течения, поэтому анализ конвективных потоков играет исключительно важную роль при расчете геометрии лазерной системы. На рис. 6 представлены линии тока и профиль температур в координатах радиус-угол для параметров: $\sigma = 2$, $r_i = 1$, $r_o = 2$, $Ra_T = 3 \cdot 10^4$, $Ra = 0$. Как видно из рисунка, наблюдается сильная неоднородность распределения температуры по углу. Максимальное значение температуры превышает соответствующее значение максимальной температуры в системе без конвекции. Для того, чтобы проследить эволюцию температур и линий тока с течением времени, необходимо вычислить соответствующие поля. Для вычисления каждого поля (в рамках реализации задачи это представляет собой матрицу некоторого размера) на каждом временном шаге необходимо решить систему трех уравнений (5-7).

$$\nabla^2 \psi = -\omega \tag{5}$$

$$\nabla^2 \omega = \frac{1}{\mathrm{Pr}}\left[\frac{\partial \omega}{\partial t} + u\frac{\partial \omega}{\partial r} + \frac{v}{r}\frac{\partial \omega}{\partial \theta}\right] + Ra_T\left[\sin\theta\frac{\partial \varphi}{\partial r} + \frac{1}{r}\cos\theta\frac{\partial \varphi}{\partial \theta}\right] \tag{6}$$

$$\nabla^2 \varphi = u\frac{\partial \varphi}{\partial r} + \frac{v}{r}\frac{\partial \varphi}{\partial \theta} - 1 + \frac{\partial \varphi}{\partial t} \tag{7}$$

Расчет всех полей для построения рисунка 6 занимал от 20 до 40 часов функционирования реализованной программы. При выполнении подобной задачи на вычислительном кластере с учетом упреждающего кэширования расчет полей составил от 16 до 32 часов. Для получения более точных оценок конвективного течения в системы коаксиальных цилиндров необходимо было использовать большие числа Рэлея и сложную модель энерговыделения в системе. Это привело к увеличению размеры матриц над которыми проводились измерения. Размер матриц достигали 14000 на 10000 элементов. Такие размеры матрицы необходимы для достижения требуемой точности. Каждый элемент представлял собой число типа float. Для расчета каждого элемента требуются еще три матрицы такого же размера, над которыми производились сложные вычисления по формулам (5-7). Для каждого шага по времени задавались граничные условия, которые хранились в отдельном файле.

Необходимо отметить, что приведенные расчеты касаются только двумерных массивов. Эту задачу необходимо было решать и для трехмерных массивов и для различных чисел Релея. Показано, что конвекция особенно сильно влияет на параметры системы именно в области теплового взрыва. Такой вывод указывает на необходимость учета конвективного теплоотвода при любых расчетах газовых лазеров и разрядов. Данная задача решалась на вычислительном кластере, состоящим из 12 вычислительных машин и одной управляющей машины. Характеристики вычислительных узлов: процессор Intel Pentium 3 800 Мгц с 256 Мбайт оперативной памяти. Использовалась файловая система PVFS.

В результате общее количество считанных блоков составило более 80 000 000 блоков по 4 Кбайта каждый. Общий объем обработанной информации составил порядка 305 Гбайт. Необходимо учитывать, что в дальнейшем придется решать данную задачу для трехмерного поля.

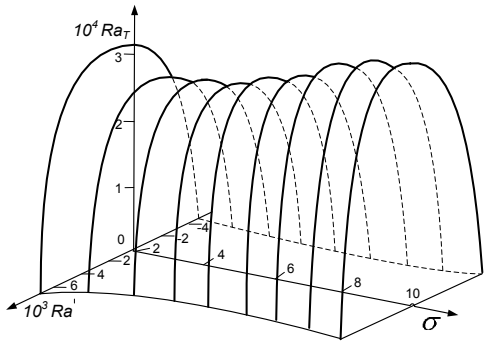

Рисунок 5– Критическая поверхность $Ra_T(Ra,\sigma)$

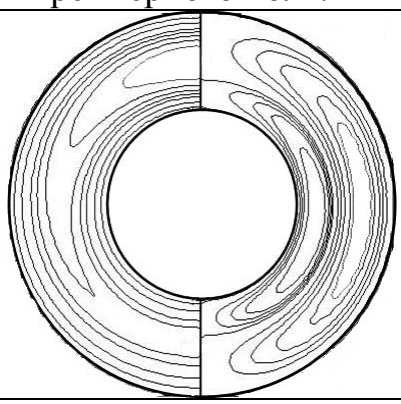

Рисунок 6 – Изотермы и линии тока при σ *=2,* $Ra_T = 3\cdot10^4$

Средние результаты функционирования системы упреждающего кэширования при расчете полей на каждом шаге по времени представлены в таблице. Общее количество шагов для получения конечного результата определяется требуемой точностью получения результатов. Прирост производительности варьируется в пределах от 15 до 50 %.

ВЫВОДЫ

Разработана имитационная модель функционирования ССРВ с применением упреждающего кэширования на языке GPSS. Модель позволила оценить потенциальную эффективность упреждающего кэширования для приложений с интенсивным обменом данными и дает возможность исследовать проектируемую или анализируемую систему упреждающего кэширования.

Созданная имитационная модель позволяет проверять различные алгоритмы упреждения данных, организацию подсистемы ввода-вывода с целью нахождения оптимальных параметров упреждающего кэширования для получения максимального ускорения исполнения приложений.

Разработан новый метод автоматической генерации точных и своевременных предвыборок без участия программиста. При разработке метода соблюдались следующие требования: метод реализован с минимальными аппаратными затратами, не оказывает влияние на системные процессы, работает на уровне приложений и не затрагивает организацию структур памяти. Предлагаемый метод упреждающего кэширования, основанный на разделении потоков, позволяет решить эту задачу для приложений с интенсивным обменом данными.

Создан модуль, который является независимым программным продуктом и который может применяться как для однопотоковых, так и для многопотоковых приложений. Для функционирования метода необходим исходный код приложения, что является недостатком модуля.

Проведенные эксперименты на тестовых приложениях доказали соответствие разработанной имитационной модели и показали, что предлагаемый метод позволяет уменьшить время исполнения приложений от 10 до 50%. Метод упреждающего кэширования эффективен для приложений, со сложным и интенсивным обменом данными.

Публикации по теме диссертации

По перечню рецензируемых изданий ВАК:

1. Киселев А.В., Корнеев В.В., Семенов Д.В., **Сахаров И.Е.** Управление метакомпьютерными системами [текст]. Открытые системы № 2 февраль-март 2005. С. 11-16.
2. **Сахаров И.Е.** Организация упреждающего кэширования для приложений, исполняющихся в ССРВ [текст]. Вестник ИГЭУ – апрель 2009.
3. **Сахаров И.Е.** Математическое представление метода упреждающего кэширования и оценка эффективности разработанного метода [текст]. Вестник ИГЭУ – апрель 2009.
4. **Сахаров И.Е.** Метод упреждающего кэширования для приложений, исполняющихся в ССРВ [текст]. Вестник ВНИИЖТ, 2/2009 С. 32-33.

Публикации в других изданиях:

5. **Сахаров И.Е.** Метод упреждающего кэширования на основе разделения потоков в высокопроизводительных распределенных вычислительных системах [текст]: монография. КГТУ, 2008. – 100 с.
6. **Сахаров И.Е.** Организация упреждающего кэширования в сетевой среде распределенных вычислений [текст]. Труды Пятой Всероссийской научной конференции молодых ученых СПбГУ 2008. С. 78-89.
7. **Сахаров И.Е.** Организация упреждающего кэширования в сетевой среде распределенных вычислений [текст]. Материалы шестой межведомственной конференции «Научно-технической и информационное обеспечение деятельности спецслужб» 2-3 февраля 2006 года ИКСИ АФСБ РФ» том 4. С. 139-142.
8. Киселев А.В., Корнеев В.В., Семенов Д.В., **Сахаров И.Е.** Организация упреждающего кэширования в сетевой среде распределенных вычислений [текст]. Труды Второй Всероссийской научной конференции М: Издательский отдел факультета ВМК МГУ им. М.В. Ломоносова. С. 152-158