МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра дискретной математики и информационных технологий

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ ИСПОЛЬЗОВАНИЯ ПАРАЛЛЕЛЬНЫХ ТЕХНОЛОГИЙ ПРИ ОБРАБОТКЕ БОЛЬШИХ ОБЪЁМОВ ДАННЫХ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 271 группы
направления 09.04.01 — Информатика и вычислительная техника
факультета КНиИТ
Бени-Лам Али Нури Шавкат

Научный руководитель
доцент, к. ф.-м. н. _____ А. Д. Панфёров

Заведующий кафедрой
доцент, к. ф.-м. н. _____ Л. Б. Тяпаев

Саратов 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Развитие вычислительной техники в первую очередь стимулировалось потребностями в решении вычислительно сложных задач и максимально быстрой обработке критически важной информации. Однако постепенно, с проникновением информационных технологий в самые разнообразные области человеческой деятельности, с формированием глобальной сетевой среды, появился огромный объём оцифрованных данных и эти данные сами стали предметом анализа с использованием мощных вычислительных систем.

Не смотря на продолжающийся прогресс в производительности компьютеров, обслуживающих конечных пользователей, объёмы доступной для анализа информации давно переросли их возможности. Это обусловило появления термина «большие данные», которым принято обозначать задачи по обработке данных, для решения которых недостаточно ресурсов одиночной вычислительной системы.

Работа с большими данными - область использования параллельных технологий. Их приходится привлекать на всех этапах: сбора, хранения и обработки. Масштаб используемого аппаратного параллелизма простирается от многоядерных мультипроцессоров и до территориально распределенных мультикомпьютерных систем.

Проблематика работы с большими массивами данных является одной из наиболее актуальных областей в современных информационных технологиях. Целью представляемой работы является системное изложение методов и подходов, используемых для анализа массивов данных, обзор программных инструментов используемых для этого и их демонстрация на примере обработки данных, получаемых при моделировании поведения физических систем.

Магистерская работа состоит из введения, пяти разделов, заключения, списка использованных источников и приложения. Общий объём работы - 57 страниц, из них 41 страница - основное содержание, включая 14 рисунков, список использованных источников включает 34 наименования.

1 Большие данные – определения и организация хранения

В первой части работы определяется используемая терминология и характерные технологические особенности проблематики работы с большими данными.

Эта область науки сформировалась недавно, в конце 2000-х годов. В настоящее время Большие данные наиболее активно генерируются Интернетом вещей. Наиболее популярное определение Big Data использует аббревиатуру VVV. Под ней подразумеваются Volume, Velocity и Variety (объём, скорость и многообразие) [1–3].

По имеющимся в свободном доступе данным в 2020 году мировой рынок технологий Big Data составлял 138,9 млрд долларов, к 2025 году прогнозируется рост до 229,4 млрд долларов. Это соответствует темпу роста в 10,6% в год, что много выше темпов роста самой мировой экономики. Уже сейчас без использования технологий работы с большими данными компании не выдерживают конкуренцию, так как не могут обеспечивать достаточный уровень клиентского сервиса.

Для хранения больших данных возможностей отдельных устройств и обычных ОС оказывается недостаточно. Большие данные обычно хранятся в распределенных файловых системах (distributed file systems - DFS) [4–6]. Стандартным свойством DFS является избыточность. Обеспечивается хранение нескольких экземпляров данных на разных серверах, что гарантирует их сохранение при сбоях. При этом дублирование данных может реализовываться на разных уровнях: файлов или отдельных блоков. Пользователю часто предоставляется возможность самому определять коэффициент избыточности. С помощью стандартного оборудования и открытых программных средств для управления DFS (например, Apache Hadoop), можно реализовать надежные хранилища данных с объёмом в несколько петабайт [7].

Аппаратные технологии, используемые для построения больших хранилищ данных, постоянно совершенствуются. Справедливо эмпирическое правило, согласно которому стоимость хранения обычно обратно пропорциональна скорости доступа к данным. Поэтому медленные и относительно устаревшие системы с ленточными накопителями до сих пор могут использоваться для хранения архивов. Обладая большими объёмами и низкой стоимостью, они хорошо подходят для хранения огромного количества редко запрашивае-

мой информации. При организации иерархических систем с использованием на разных уровнях различных аппаратных решений можно оптимально сочетать их преимущества и строить очень ёмкие хранилища с хорошим средним временем выполнения запросов.

В случае распределенного сетевого хранилища его пространственный масштаб может быть любым. От компактного размещения в одном ЦОДе (центре обработки данных), до глобальной системы, связанной каналами Интернет. В распределенной системе хранения данные могут отправляться по запросам клиентам или пересылаться между узлами хранилища. Операции чтения и записи клиентских данных могут производиться с использованием сразу нескольких узлов.

Предъявляемые к хранилищам данных требования зависят от области их использования. Так, для данных о финансовых транзакциях и торговых операциях характерен относительно короткий временной интервал их активного использования и затем они переходят в разряд архивных. В страховании, банковской деятельности цикл операции может занимать месяцы и годы и все это время соответствующие данные могут быть затребованы и должны быть быстро доступны.

В случае работы с действительно большими данными становится принципиально важно, что структурированные данные намного легче понять и обработать, в то время как многочисленные форматы неструктурированных данных создают большие проблему. Однако оба типа данных играют свою роль в эффективном анализе данных. Примеры трудно структурируемых данных включают аудио, видео, текстовые документы и презентации.

2 Использование параллелизма при обработке данных

Во второй части работы представлены базовые обоснования применимости аппаратного и программного распараллеливания при работе с данными и приведены примеры современных технологических решений.

Разработка программ для параллельных вычислительных систем требует соответствующей поддержки на уровне языков [8] и библиотек. Организация взаимодействия параллельных ветвей программы через передачу сообщений (MPI - Message Passing Interface), вероятно, является наиболее широко используемым вариантом параллельного программирования для мультикомпьютерных систем на сегодняшний день.

Возможны три способа реализации MPI и его взаимодействия с языками программирования:

— расширение стандартного языка последовательного программирования библиотечными функциями;
— расширение стандартного языка последовательного программирования специальными конструкциями;
— разработка нового языка.

На практике наиболее привычным и естественным является первый вариант. Он позволяет отталкиваться от знакомого и хорошо освоенного языка программирования. При этом подключение дополнительных библиотек также вполне привычная и прозрачная процедура.

С массовым переходом на мультикомпьютеры при решении вычислительно сложных задач появились новые проблемы, связанные с оптимизацией доступа к данным. В отличие от многопроцессорных ЭВМ с общей памятью, на системах с распределенной памятью необходимо произвести не только распределение вычислений, но и распределение данных, а также обеспечить на каждом процессоре доступ к удаленным данным, расположенным на других вычислительных узлах. Согласованное распределение вычислений и данных требует тщательного анализа всей программы, и ошибки могут привести к катастрофическому замедлению выполнения программы [9].

Параллельное программирование для мультипроцессоров проще за счет наличия общей памяти. В этом случае параллельная программа может представлять собой систему нитей (threads), взаимодействующих посредством общих переменных и примитивов синхронизации.

Первая попытка стандартизовать такую модель привела к появлению проекта языка PCF Fortran. Однако, этот проект [10] не привлек широкого внимания и, фактически, остался только на бумаге. Возможно, что причиной этого было снижение интереса к мультипроцессорам и всеобщее увлечение мультикомпьютерами и HPF. Однако, несколько лет спустя ситуация сильно изменилась. Успехи в развитии элементной базы сделали экономически выгодным создавать мультипроцессоры. Крупнейшие производители компьютеров и программного обеспечения объединили свои усилия и в октябре 1997 года выпустили описание языка OpenMP Fortran – расширение языка Фортран. Немного позже, в 1998 году, вышло аналогичное расширение для языка Си.

Ориентация на мультипроцессоры стала особенно актуальной после 2004 года, когда стало ясно, что дальнейший быстрый рост тактовых частот проблематичен. С этого момента началось быстрое развитие технологии многоядерных процессоров. Ресурс увеличения плотности активных элементов на чипе сохраняется до настоящего времени и именно это сделало многоядерные процессоры привлекательными для повышения производительности вычислений.

В настоящее время актуальны несколько подходов в распараллеливании на многоядерных процессорах:

— на уровне экземпляров виртуальных машин;
— на уровне процессов;
— на уровне потоков;
— на уровне инструкций.

Не смотря на рост производительности современных мультипроцессоров с общей памятью за счет увеличения количества ядер в них, при необходимости выхода за границы их возможностей, как и ранее, приходится использовать мультикомпьютеры. В этой области было сосредоточено много усилий разработчиков и они принесли результаты. Для распределенной обработки очень больших объёмов данных компанией Google была разработана модель вычислений MapReduce. Она была реализована в виде платформы (фреймворка) в первую очередь для обеспечения функционирования основного инструмента компании – системы индексации и поиска информации в глобальной сети Интернет. Поисковая система Google использует вычис-

лительные мощности более чем миллиона серверов ежедневно обрабатывая миллиарды пользовательских запросов и десятки петабайт данных.

Версия проекта с открытым исходным кодом имеет название Hadoop. Важным элементом проекта является распределенная файловая система HDFS (Hadoop Distributed File System). Она обеспечивает масштабирование системы хранения и её отказоустойчивость для гарантированной сохранности данных путем распределения нескольких экземпляров реплик данных на различные сервера. HDFS может использоваться в качестве распределенной файловой системы общего назначения или в качестве основы Hadoop системы для обработки данных с использованием процедур MapReduce.

Узлы хранения данных (DataNode, Node) хранят реплики данных, выполняют команды управляющего узла по их чтению и записи, отправке данных клиентам. Потоки данных между узлами хранения и клиентами передаются напрямую. NameNode только выдает необходимые команды. Наконец, клиенты системы. С учетом имеющихся прав доступа, они могут создавать, удалять, читать, записывать, переименовывать и перемещать файлы и каталоги.

Для обеспечения сохранности данных, как было отмечено выше, HDFS использует реплицирование данных в нескольких экземплярах. Распределенное хранение реализуется на уровне отдельных блоков. Пользователь имеет возможность явно задавать количество реплик своих данных (по умолчанию – 3) и размер используемых блоков (по умолчанию - 64 Мб).

Работа MapReduce состоит из двух этапов: распределение задачи по доступным вычислительным узлам (map - картирование) и сбор с последующей финальной обработкой полученных данных (reduce - редукция). Название возникло как слияние имен используемых при этом управляющих функций map и reduce.

Область применения этой технологии – системы из десятков, сотен и даже тысяч серверных узлов, хранящие и обслуживающие десятки петабайт данных.

3 Python – высокоуровневый язык для работы с данными

В третьей части работы представлен язык программирования Python и специализировавнные библиотеки, ориентированные на обработку данных. Они предоставляют готовые реализации многих необходимых алгоритмов, существенно упрощая процедуру написания сложных прикладных решений [11]. Наиболее популярны и востребованы: NumPy, SciPy, Pandas, StatsModels, Matplotlib, Seaborn, Plotly, Bokeh, Scikit-Learn, Keras. Представлены их краткие характеристики.

Отдельное внимание уделено инструментам для параллельной работы. На мощных современных мультипроцессорах количество вычислительных ядер может быть более ста. Такие вычислительные системы могут успешно работать с достаточно большими объёмами данных.

Для реализации аппаратного параллелизма необходимо обеспечить его эффективное использование на программном уровне. Особенностью Python является то, что это интерпретирующий язык программирования и для его интерпретатора синхронизация одновременной работы множества потоков в режиме реального параллелизма является непосильной задачей. По этой причине интерпретатором используется Python Global Interpreter Lock (GIL) - блокировка, позволяющая только одному потоку управлять интерпретатором Python. При запуске одного экземпляра интерпретатора вне зависимости от количества потоков в любой момент времени будет выполняться только один конкретный поток. Поэтому аппаратный параллелизм в Python приходится использовать с помощью создания дополнительных процессов. Инструментом для этого служит пакет ***multiprocessing***. Порождение новых процессов имеет смысл до тех пор, пока каждому из них может быть предоставлен для работы собственный процессор (ядро). Увеличение числа процессов сверх количества имеющихся процессоров только увеличит накладные расходы и не приведет к увеличению скорости работы программы.

Пакет ***multiprocessing*** предлагает набор методов для порождения процессов и взаимодействия с ними. Когда необходимо выполнять одинаковый набор операций с различными экземплярами выборки одновременно (параллелизм по данным) удобнее инструменты из класса ***Pool***. Методы этого класса ***apply(), map()*** и ***starmap()*** обеспечивают запуск любой функции в параллельном режиме.

4 Постановка задачи анализа данных

В четвертой части работы определены характеристики массива данных, предоставлявшихся для обработки, и решавшиеся задачи.

Исследование эффективности использования параллельных технологий при обработке больших объёмов данных проводилось применительно к задаче промежуточного анализа результатов научного моделирования.

В качестве примера тестовой задачи рассматриваются результаты моделирования действия на графен короткого лазерного импульса. Основные характеристики моделируемых процессов определяются через функцию распределения $f_1(p_1, p_2, t)$. Была поставлена задача реализовать ряд процедур поиска, сортировки и анализа данных из трехмерного массива результатов с индексами k, l, m, каждый элемент которого, в свою очередь, сформирован в виде одномерного массива вида

$$\{p_{1k}, p_{2l}, t_m, f_1(p_{1k}, p_{2l}, t_m), f_2(p_{1k}, p_{2l}, t_m), f_3(p_{1k}, p_{2l}, t_m)\}.$$

Первая процедура – поиск абсолютного максимума функции распределения $f_1(p_1k, p_2l, t_m)$.

Вторая процедура – построение двумерного среза для заданного значения t_m.

Третья процедура – построение двумерного массива максимальных значений f_1 за весь интервал наблюдения для каждой точки (p_{1k}, p_{2l}).

Четвертая процедура – построение двумерного массива средних значений f_1 за весь интервал наблюдения для каждой точки (p_{1k}, p_{2l}).

5 Программная реализация процедур анализа и результаты

Пятая часть работы демонстрирует особенности реализованного программного решения и полученные результаты.

Для программной реализации процедур анализа данных моделирования было разработано приложение на языке Python с использованием ряда описанных ранее библиотек. Базовые математические операции выполняются с использованием функций из библиотеки ***numpy***, для анализа данных применены возможности ***pandas***, распараллеливание процедур анализа реализовано с использованием ***multiprocessing*** и черновая отрисовка результатов выполняется библиотекой ***matplotlib.pyplot***.

Организация многопроцессного режима работы реализована методом ***pool.apply()*** класса ***pool*** модуля ***multiprocessing***. Для оптимального использования ресурсов системы перед запуском пула процессов пользователю выводится информация о количестве доступных процессоров и предоставляется возможность явно задать количество запускаемых процессов. Информация о количестве процессоров определяется функцией ***cpu.count()*** модуля ***multiprocessing***.

Отдельным предметом исследования была эффективность использования аппаратного параллелизма разработанным кодом. Для этого оценивалось время выполнения одинаковых тестовых заданий по формированию двумерных массивов максимальных и средних значений анализируемой функции при различном числе запускаемых одновременно процессов. Приведенные результаты получены при обработке трехмерного массива данных размером $64 \times 64 \times 57$:

Таблица 1 – Время выполнения тестового задания в зависимости от количества используемых процессов

Кол-во запущенных процессов	Время выполнения теста, сек	Ускорение	Эффективность
1	11.269	1.000	1.000
2	8.124	1.387	0.693
3	6.434	1.751	0.584
4	5.605	2.010	0.502
6	4.438	2.539	0.423

Можно сделать вывод, что удается обеспечить существенное ускорение

выполнения обработки данных. Максимальное полученное ускорение составило немного более 2.5 при использовании 6 процессов на 6-ядерном процессоре.

Результаты обработки по определению распределения максимальных и средних значений приведены на рисунках:

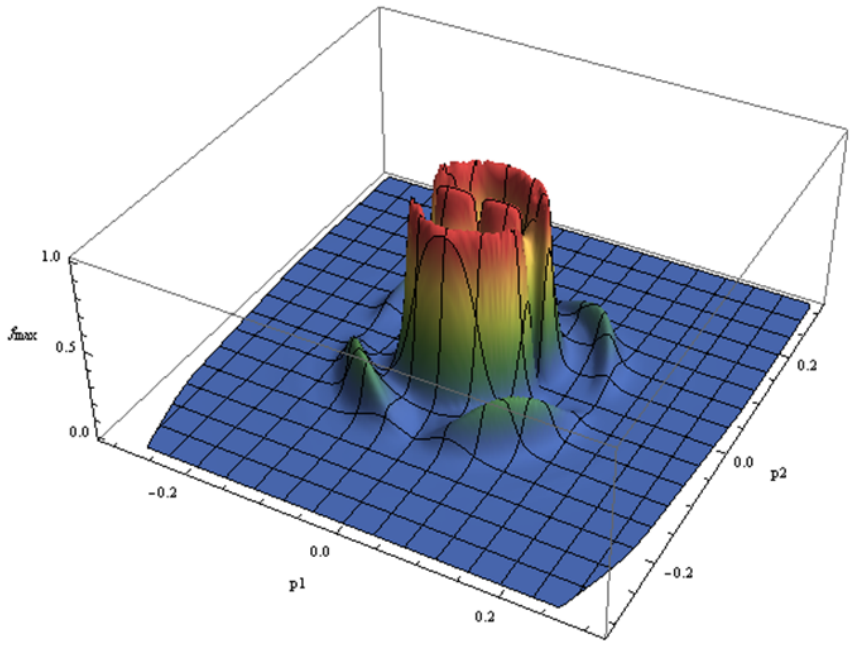

Рисунок 1 – Максимальные значения функции распределения $f_{max}(p_1, p_2)$

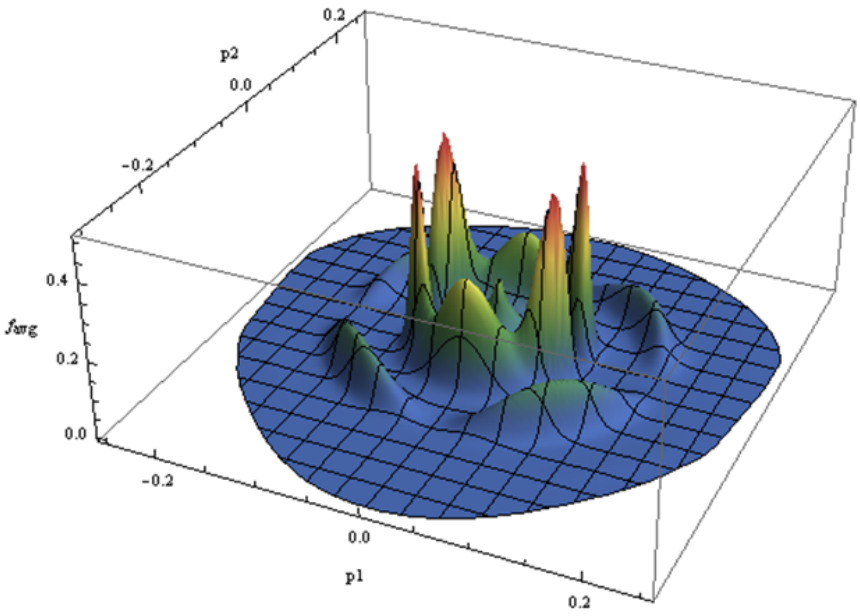

Рисунок 2 – Средние значения функции распределения $f_{avg}(p_1, p_2)$

ЗАКЛЮЧЕНИЕ

Поставленные в качестве цели выпускной квалификационной работы задачи были полностью выполнены. Проведен обзор актуального направления обработки больших массивов данных. Определена специфика и отличительные признаки проблематики BigData. При решении задач такого класса использование аппаратного параллелизма становится обязательным условием. Представлен обзор программных технологий для использования аппаратного параллелизма на различных уровнях. От многоядерных мультипроцессоров и до распределенных кластерных систем. В том числе OpenMP, MPI и Hadoop.

В качестве предмета детального изучения и демонстратора возможностей был выбран язык Python, поскольку в настоящее время он является стандартным инструментом обработки и анализа данных, как в научных проектах, так и в различных прикладных областях, включая бизнес-аналитику. Представлено описание особенностей этого языка и ряда специализированных библиотек, таких, как NumPy, SciPy, Pandas, StatsModels, Matplotlib.

Демонстрация полученных знаний и навыков работы с экосистемой Python выполнена на примере обработки научных данных – результатов численного моделирования поведения двумерного материала графена во внешнем электрическом поле. Был сформулирован ряд требования по обработке этих данных, предоставлявшихся в виде большого многомерного массива. Разработано и продемонстрировано программное решение, обеспечивающее предварительный парсинг данных и их обработку в параллельном режиме с использованием технологии генерации нескольких процессов. Полученные результаты представлены в работе. Разработанное программное решение будет использоваться для анализа новых данных, получаемых в вычислительных экспериментах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Unstructured Data: an overview of the Data of Big Data / Adanma Cecilia Eberendu // International Journal of Computer Trends and Technology. 2016. V.38, №1, P. 46-50.

2 (Big)Data in a virtualized world: volume, velocity, and variety in cloud Datacenters in Cloud Datacenters / Robert Birke, Mathias Bjorkqvist, Lydia Y. Chen, Evgenia Smirni, and Ton Engbersen // Proceedings of the 12th USENIX conference on File and Storage Technologies, FAST 2014, Canta Clara, CA, USA, Fabruary 17-20,2014, P. 177-189.

3 Big Data Multimedia Mining: Feature Extraction Facing Volume, Velocity, and Variety / Vedhas Pandit, Shahin Amiriparian, Maximilian Schmitt, Amr Mousa, Bjorn Schuller // In book: Big Data Analytics for Large Scale Multimedia Search - John Wiley & Sons Ltd, 2019, P 61-87.

4 Революция Big Data: Как извлечь необходимую информацию из «Больших Данных»: [Электронный ресурс] URL: http://statsoft.ru/products/Enterprise/Big-Data.php (дата обращения: 20.01.2022)

5 A Taxonomy and Survey on Distributed File Systems, Fourth International Conference on Networked Computing and Advanced Information Management / Tran Doan Thanh, Subaji Mohan, Eunmi Choi, SangBum Kim, Pilsung Kim: [Электронный ресурс] URL: https://ieeexplore.ieee.org/document/4623994 (дата обращения: 20.01.2022)

6 Performance modeling of a distributed file-system / Sandeep Kumar // arXiv:1908.10036

7 HDFS Architecture Guide: [Электронный ресурс] URL: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html (дата обращения: 20.01.2022)

8 A comparative study of parallel programming languages: the Salishan problems. / Feo J.T. – Elsevier, 2016. 396 p.

9 Document for a standard message-passing interface / Dongarra J. // Message Passing Interface Forum. – 1993.

10 Fortran DVM-язык разработки мобильных параллельных программ / Н.А.Коновалов, В.А.Крюков, П.Н.Михайлов, и др. // Программирование. 1995. №1. с. 49-54.

11 Python data science handbook: Essential tools for working with data / VanderPlas J. // – O'Reilly Media, Inc. 2017. 546 p.