Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

«Анализ методов распределенных вычислений в распределенных системах хранения информации»

Студент **ИУ7-75Б** _____ **Романов С. К.**

Руководитель НИР _____ **Бекасов Д. Е.**

Рекомендуемая руководителем НИР оценка _____

2023 г.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие термины с соответствующими определениями.

Test — TODO?

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие сокращения и обозначения.

TODO — Test?

ВВЕДЕНИЕ

Компьютерная индустрия изменила курс в 2005 году, когда Intel, последовав примеру IBM Power 4 [1] и процессору Niagara [2] от Sun Microsystems, объявили, что их высокопроизводительные микропроцессоры отныне будут опираться на несколько процессоров или ядер [3]. Новое в отрасли слово «многоядерный» отражает план удвоения количества стандартных ядер на матрицу с каждым поколением полупроводниковых процессов. Многоядерный процессор, очевидно, поможет многопрограммным рабочим нагрузкам, которые содержат набор независимых последовательных задач [4], но как отдельные задачи станут быстрее? Переход от последовательных вычислений к умеренно параллельным значительно усложняет разработку систем, не вознаграждая эти большие усилия значительно лучшим соотношением производительности к энергопотреблению [5].

Следовательно, многоядерные процессоры едва ли являются идеальным решением. Подкрасться к проблеме параллелизма с помощью многоядерных решений, скорее всего, не удастся, и отсюда вытекает нужда в конкретном решении для параллельного аппаратного и программного обеспечения. Однако главная гипотеза заключается не в том, что традиционные научные вычисления и строгие математические модели — это будущее параллельных вычислений; она заключается в том, что совокупность знаний, полученных при создании программ, которые хорошо работают на массово параллельных компьютерах, может оказаться полезной при распараллеливании будущих приложений [6]. Тем более, что многие приложения сегодня требуют обработку больших объемов данных, а не проведение сложных вычислений. Необработанная мощность процессора редко является ограничивающим фактором для подобного рода приложений — более серьезными проблемами обычно являются объем данных, их сложность и скорость, с которой они изменяются.

Цель работы — анализ методов распределенных вычислений в распределенных системах хранения информации.

Для достижения поставленной цели требуется решить следующие задачи:

- Провести обзор существующих систем распределенных вычислений;
- Провести анализ подходов к проектированию распределенных вычислений;
- Сформулировать критерии сравнения методов распределенных вычислений;
- Классифицировать существующие методы распределенных вычислений.

1 Аналитический раздел

В данном разделе будут рассмотрены основные понятия и термины, которые используются в данной работе, а также будут рассмотрены основные алгоритмы и существующие решения.

1.1 Проблематика

Различные сетевые приложения, такие как веб-серверы, поисковые системы, системы управления базами данных и т.д. стали неотъемлемой частью современного общества. Об этом свидетельствует рост количества пользователей и объемов данных, которые обрабатываются этими приложениями, а также общей долей рынка связанной с этими приложениями [7]. Масштаб современных сетевых приложений предоставляет конечному пользователю дополнительные вычислительные мощности для решения прикладных задач и распределяет общую нагрузку на сеть посредством распределения вычислений между узлами сети. Когда компьютеры работают вместе в рамках единой сети, мощность всех подключенных к сети компьютеров может использоваться для выполнения сложных задач. Подобного рода вычисления могут быть распределены на централизованные и распределенные.

Централизованное решение основано на том, что один узел назначается ответственным за все вычисления проводимые приложением, обрабатывает их локально, и данный узел является общим для всех пользователей системы. Следовательно, существует единая точка контроля и единая точка отказа. Возможно такого рода решение и является оптимальным для небольших задач, но при увеличении нагрузки на систему, возникает, например, необходимость в более отказоустойчивой системе, которая будет обеспечивать доступность данных на протяжении всего времени работы системы.

Мотивацией роста децентрализованных вычислений является доступность недорогих, высокопроизводительных компьютеров и сетевых инструментов. Такая система может обладать более высокой производительностью, чем один

конкретный суперкомпьютер. Целью таких систем является минимизация затрат на связь и вычисления. В распределенных системах этапы обработки приложения распределены между участвующими в ней узлами. Основным шагом во всех архитектурах распределенных вычислений является понятие связи между узлами системы.

Приложение, удовлетворяющему этим требованиям, обычно строится из стандартных «блоков», которые предоставлют необходимую функциональность. Так например, приложение может требовать реализации следующих функций:

- Хранение данных для дальнейшего к ним доступа (базы данных);
- Хранение результата дорогостоящей операции для ускорения чтения (кэширование);
- Возможность для пользователей искать данные по ключевому слову или фильтровать их различными способами (поисковые индексы);
- Отправление сообщений другому процессу, которые будут обрабатываться асинхронно (потоковая обработка);
- Периодическая обработка большого объема накопленных данных (пакетная обработка).

Распределенная система — это приложение, которое выполняет набор протоколов для координации действий нескольких процессов в сети таким образом, что все компоненты взаимодействуют друг с другом для выполнения одной задачи или небольшого набора взаимосвязанных задач [8]. Сотрудничающие компьютеры могут получать доступ как к удаленным, так и к локальным ресурсам данной распределенной системы. Существование нескольких автономных компьютеров в распределенной системе неочевидно для пользователя, т.е. пользователь не знает, что задания выполняются несколькими компьютерами в рамках единой системы.

Как и к любому другому ПО, к распределенным системам применимы часто рассматриваемые инженерные критерии [9], а именно:

1. `Reliability` — Надежность — Система должна продолжать корректно

работать (выдавать корректный результат на желаемом уровне производительности) даже перед лицом неблагоприятных факторов (аппаратных или программных сбоев и возможной человеческой ошибки);

2. `Scalability` — Масштабируемость — По мере роста системы (при увеличении объема данных, трафика или сложности) должны существовать разумные способы и инструменты управления этим ростом;
3. `Maintainability` — Сопровождаемость — Со временем, набор разработчиков, работающих над системой, может сильно изменяться, и все они должны быть в состоянии продуктивно работать над продуктом (проектирование и эксплуатация, как поддержание текущего поведения, так и адаптация системы к новым требованиям).

В дальнейшем описываемые алгоритмы будут расматриваться через призму этих критериев: как результирующие системы удовлетворяют этим критериям и какие компромиссы приходится делать при их реализации.

Также по мере разработки распределенных систем, существует набор различных вопросов, на которые необходимо дать ответ для корректной и эффективной работы системы как таковой. Например, как распределить задачи между узлами системы и как обеспечить надежность и целостность данных, как гарантировать доступность данных на всем протяжении работы системы и т.д. Одним из формальных ответов на эти вопросы служит формулировка `CAP`-теоремы — или теорема Брюэра — которая звучит следующим образом:

В любой распределенной системе можно обеспечить не более двух из трех свойств:

- **C**onsistency — Согласованность — каждый сервер возвращает верный ответ на каждый запрос, т.е. ответ, который является правильным в соответствии с требуемой спецификацией приложения;
- **A**vailability — Доступность — гарантируется, что на каждый запрос в конечном итоге будет получен некий ответ;
- **P**artition tolerance — Устойчивость к разделению — в произвольный мо-

мент времени сервера могут быть разделены на несколько групп, которые не будут взаимодействовать друг с другом. Иными словами, сообщения в такой системе могут задерживаться, а иногда и теряться.

К сожалению, такая формулировка вводит в заблуждение [10], поскольку в случае если система существует в сети, которая может терять произвольное количество сообщений, что является реальностью современных телекоммуникаций, то такая система не может быть одновременно доступной и согласованной, выбрать следует что-то одно [11]. Поэтому в дальнейшем, при рассмотрении распределенных систем, будут браться во внимание именно эти два критерия: доступность и согласованность, принимая устойчивость к разделению как данность. Это необходимое допущение позволяет лучше понять проблематику распределенных вычислений и рассматривать их в контексте реальных систем.

1.2 Основные алгоритмы

todo

1.2.1 MapReduce

todo

1.2.2 Graph traversals

todo

1.2.3 Finite State Machine

todo

1.2.4 Dynamic Programming

todo

1.2.5 Lambda architecture

todo

1.3 Существующие решения

todo

1.3.1 Hadoop

todo

1.3.2 Spark

todo

1.3.3 Flink

todo

1.3.4 Hive

todo

1.3.5 Databricks

todo

1.3.6 GraphX

todo

2 Конструкторский раздел

2.1 TODO

2.2 TODO

2.3 TODO

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. POWER4 system microarchitecture / J. Tendler [и др.] // IBM Journal of Research and Development. — 2002. — Янв. — Т. 46. — С. 5—25. — DOI: 10.1147/rd.461.0005.
2. Uddin I. Advances in computer architecture. — 2013. — arXiv: 1309.5459 [cs.AR].
3. Dual Core Era Begins, PC Makers Start Selling Intel-Based PCs [Электронный ресурс]. — (дата обращения: 22.10.2023). Режим доступа: https://www.intel.com/pressroom/archive/releases/2005/20050418comp.htm.
4. The Landscape of Parallel Computing Research: A View from Berkeley : тех. отч. / K. Asanović [и др.] ; EECS Department, University of California, Berkeley. — 12.2006. — UCB/EECS-2006—183. — URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html.
5. McKenney P. E. Is Parallel Programming Hard, And, If So, What Can You Do About It? (Release v2023.06.11a). — 2023. — arXiv: 1701.00854 [cs.DC].
6. Lynch N. Distributed Algorithms. — Morgan Kaufmann, 1996. — (The Morgan Kaufmann Data Manag). — ISBN 9781558603486. — URL: https://books.google.at/books?id=7C7oIV48RQQC.
7. Internet of Things (IoT) Market Size, 2023-2030 [Электронный ресурс]. — (дата обращения: 24.10.2023). Режим доступа: https://www.fortunebusinesscom/industry-reports/internet-of-things-iot-market-100307,
8. Thampi S. M. Introduction to Distributed Systems. — 2009. — arXiv: 0911.4395 [cs.DC].
9. Kleppmann M. Designing Data-intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. — O'Reilly Media, 2017. —

ISBN 9781449373320. — URL: https://books.google.at/books?id=BM7woQEACAAJ.

10. Brewer E. CAP Twelve years later: How the ”Rules”have Changed // Computer. — 2012. — Февр. — Т. 45. — С. 23—29. — DOI: 10.1109/MC.2012.37.

11. Henry Robinson: “CAP Confusion: Problems with ‘Partition Tolerance,’” [Электронный ресурс]. — (дата обращения: 28.10.2023). Режим доступа: https://web.archive.org/web/20120120140424/http://www.cloudera.com/blog/2010/04/cap-confusion-problems-with-partition-tolerance/.

ПРИЛОЖЕНИЕ А