УДК 004.032.26
DOI: 10.46548/21vek-2020-0950-0018

ИССЛЕДОВАНИЕ И АНАЛИЗ ПОДХОДОВ К ОРГАНИЗАЦИИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ

Синев Михаил Петрович, кандидат технических наук, доцент кафедры «Вычислительная техника»
Пензенский государственный университет
(440026, Россия, Пенза, улица Красная, 40, e-mail: mix.sinev@gmail.com)
Мартышкин Алексей Иванович, кандидат технических наук, доцент,
доцент кафедры «Вычислительные машины и системы»
Трокоз Дмитрий Анатольевич, кандидат технических наук, доцент,
доцент кафедры «Вычислительные машины и системы»
Пензенский государственный технологический университет
(440039, Россия, Пенза, проезд Байдукова/ул. Гагарина, д. 1а/11,
e-mails: alexey314@yandex.ru, dmitriy.trokoz@gmail.com)

Аннотация. Работа посвящена исследованию и анализу возможных подходов к организации параллельных вычислений в распределенных вычислительных системах. В ходе работы над статьей проанализированы основные подходы к организации параллельных вычислений на распределенных системах. Рассмотрены современные методы организации таких систем, их положительные и отрицательные стороны, а также теоретическая и реальная производительность. Проведены оценки метрик эффективности распределенных и параллельных систем, на примере закона Амадала. Приведено описание архитектуры вычислительного кластера, основанне на инфраструктуре облачного сервиса. Поведен ряд экспериментов, оценивающих эффективность построенной системы. Проведен ряд тестов, оценивающих производительность кластера для алгоритмов word-count и sort. Анализ полученных результатов показал, что spark может эффективно использовать распределенные вычисления, сравнимые с теоретической производительностью по закону Амдала от 90% вплоть до 97,5% доли задачи возможной для распараллеливания. В рамках анализа возможной оценки производительности не алгоритмической части, отвечающий за запись/чтение принято решение о невозможности объективной оценки данного компонента распределенного вычисления. Кроме того, определена ключевая метрика эффективности работы кластера, а именно общее время проведения распределенного вычисления.

Ключевые слова: большие данные, информационная безопасность, классификация, облачные сервисы, параллельные вычисления, производительность, распределенные системы, ускорение.

RESEARCH AND ANALYSIS OF THE APPROACHES TO ORGANIZING PARALLEL COMPUTING IN DISTRIBUTED SYSTEMS

Sinev Mihail Petrovich, candidate of technical Sciences,
associate Professor of sub-department «Computer engineering»
Penza State University
(440026, Russia, Penza, Krasnaya Street, 40, e-mail: mix.sinev@gmail.com)
Martyshkin Alexey Ivanovich, candidate of technical sciences, docent,
associate Professor of sub-department «Computers and systems»
Trokoz Dmitrii Anatolevich, candidate of technical sciences, docent,
associate Professor of sub-department «Computers and systems»
Penza state technological University
(440039, Russia, Penza, Baydukov Proyezd / Gagarin Street, 1a/11,
e-mails: alexey314@yandex.ru, dmitriy.trokoz@gmail.com

Abstract. The paper is devoted to the research and analysis of possible approaches to the organization of parallel computing in distributed computing systems. In the course of work on the article, the main approaches to organizing parallel computing on distributed systems are analyzed. Modern methods of organizing such systems, their positive and negative sides, as well as theoretical and real performance are considered. From evaluation metric of the effectiveness of distributed and parallel systems, on the example of the law of Amadal. The architecture of a computing cluster based on the cloud service infrastructure is described. A number of experiments were performed to evaluate the effectiveness of the constructed system. A number of tests were performed to evaluate cluster performance for the wordcount and sort algorithms. Analysis of the results showed that spark can effectively use distributed computing, comparable to the theoretical performance of Amdahl's law from 90% to 97.5% of the share of the problem possible for parallelization. As part of the analysis of the possible performance evaluation of the non-algorithmic part responsible for writing/reading, it was decided that it is impossible to objectively evaluate this component of distributed computing. In addition, a key metric for cluster performance has been defined,namely, the total time for distributed computing.

Keywords: big data, information security, classification, cloud services, parallel computing, performance, distributed systems, acceleration.

Введение. Сегодня необходимостью является возможность обработки и хранения больших данных. Возможность персистентного хранения данных предоставляются современными СУБД, использующими различные модели и подходы к хранению огромного объема информации, именуемыми нереляционными или «*NoSQL*» СУБД. Но даже среди таких СУБД не все могут хранить данные на одной физической машине, отсюда выходит так называемая проблема вертикального масштабирования. Для обработки таких данных нужны соответствующие способы и подходы обработки таких данных, и конечно специализированное программное обеспечение. Эти подходы подразумевают собой использование параллельных вычислений, на нескольких вычислительных системах, в том числе и в вычислительных облаках. В результате этого решения возник и соответствующий метод по обработке таких массивов данных – распределенные вычисления. Такой подход к работе с данными приносит не только возможность работать с большим объемом данных, но и значительно сократить время вычислений, что, в свою очередь, позволяет оперировать большими объемами данных, а использование облачных сред вычисления удешевляет стоимость таких вычислений, что улучшает экономическую составляющую таких систем. Представленная тема является актуальной по нескольким причинам, среди которых следующие: Тема обработки и вычислений больших данных является актуальной, количество данных в мире, растет с каждым годом и из всех этих данных можно извлечь информацию, которая может иметь тысячи применений в различных областях деятельности: от медицины до распознавания изображений; Экономические причины. Об эффективности использования облака в IT сказано большое количество раз [1]. Но использование облака для организации параллельных вычислений является крайне эффективным. Это обусловлено следующей причиной: сам характер описанной работы подразумевает кратковременные, непостоянные и высоконагруженные вычисления, что отлично укладывается в рамки вычислительного облака, потребитель использует только те ресурсы, которые ему необходимы в конкретные моменты для вычислений, таким образом, сокращается простаивание вычислительны мощностей, за счет чего повышается общая эффективность системы, в том числе и экономическая. Так же сюда относится и использование человеческих ресурсов, всей системой может управлять только разработчик, по причине отсутствия физических ресурсов. Экономию времени стоит отнести в отдельную категорию, потому что в некоторых задачах временные ресурсы имеют большее значение, чем экономические. В основном это касается параллельных вычислений, увеличивая количество узлов вычислений можно многократно сократить время на вычисление задачи.

Степень научной разработанности данной темы можно оценить как довольно слабую. Связано это в первую очередь с тем, что технологии, используемые при организации таких распределенных вычислений, появились относительно недавно, например, первый релиз фреймворка *Spark* произошел 30 мая 2014 года [2]. Данный факт влияет на малое количество научных публикаций как по фундаментальным исследованиям в данной области, так и по теме прикладного использования конкретных технологий. Среди трудов, на которых опиралась данная работа стоит выделить [3, 4].

Целью работы является анализ основных подходов к организации параллельных вычислений на больших массивах, данных с использованием облачных сервисов, а так-же использование выбранного для решения вычислительных задач. Для достижения поставленной цели необходимо решить следующие задачи: анализ современных способов организации параллельных вычислений; проведение анализа технических средств, используемых для организации таких вычислительных систем; построение архитектуры такой системы, и практическое применение построенной системы; анализ данных, полученных в результате проведения вычислений на построенной системе, при использовании различных параметров системы.

Область исследования заключается в анализе подходов к организации параллельных вычислений на примере распределенных систем.

Объектом исследования статьи является организации эффективного способа проведения параллельных вычислений в экосистеме *Hadoop* с использованием облачных сервисов.

Предметом исследования этой работы являются проверка возможности и эффективность использования экосистемы *Hadoop* и фреймворка *Spark* для организации параллельных вычислений в облачных сервиса, исследуются вопросы экономической эффективности, вопросы автоматизации процессов размещения, а так же законы, модели и метрики, по которым можно судить об эффективности или неэффективности производительности проводимых вычислений. Рассматривается вопрос целесообразности использования и особенности такого подхода для задач по обработке большого количества данных.

Научная новизна и теоретическая значимость работы обусловленs тем, что параллельные вычисления являются быстроразвивающейся темой в современном мире разработки и организации вычислений. А в сочетании с облачными вычислениями эта тема постоянно развивается, появляются новые технологии и способы организации такого рода вычислений. Выбор архитектуры построения параллельных вычислений, основанных на использовании облака, является значимой теоретической задачей. Именно поэтому, работа, содержащая анализ элементов таких систем вычислений, а также способы организации таких вычислений является слабо проанализированной и разработанной. Доказательством научной новизны является увеличивающий интерес к данной теме на профильных конференциях, например, крупнейшей

конференции посвященной *Hadoop – Hadoop Summit (DataWorksSummit)* [5].

Практическая значимость этой работы состоит в демонстрации практических возможностей параллельных вычислений на основе конкретного облачного сервиса и конкретной вычислительной задачи. Демонстрация различных переменных такой системы предоставит возможность провести анализ зависимости эффективности системы от различных параметров такой системы. На основе результатов данной работы можно сделать выбор о возможности использования для конкретной задачи и об эффективности использования облачных сервисов для распределенных вычислений на платформе *Hadoop*.

Обзор текущего состояния дел по данной теме. Прежде чем преступить к рассмотрению данной темы, стоит точно сформулировать определения тех сущностей, анализ которых будет проведен в данной работе. Четкой истории появления термина «Облачные вычисления» не существует и скорее всего появление этого термина было связано с использованием обозначений, используемых в схемах при проектировании и описании сетей, в которых «облако» описывало интернет или некий кластер серверов. Данная модель подразумевает под собой 5 основных характеристик: Сервис по требованию; Широкий сетевой доступ; Объединение ресурсов («*pooling*»); Быстрая эластичность; Измеряемость предоставленного сервиса.

Существует три типа облачных сервисов, классифицируемых по модели сервиса: Программный продукт как сервис («*SaaS*»), возможность использовать потребителем ПО через различные «тонкие» клиенты, которые предоставляются поставщиком услуг, на основе облачной инфраструктуры; Платформа как сервис («*PaaS*»), возможность размещать приобретенные или разработанные потребителем программные продукты, на поддерживаемой поставщиком платформе (языке, библиотеке, сервисе и т.п.); Инфраструктура как сервис («*IaaS*»), возможность использования базовых элементов инфраструктуры, предоставленных поставщиком облачного сервиса. Стоит отметить, что все перечисленные типы облачным сервисов, могут быть использованы при организации параллельных вычислений, некоторые из облачных провайдеров обеспечивают готовые решения по проведению параллельных вычислений (*PaaS*), однако финансовая эффективность таких решений гораздо ниже подходов по сравнению с использованием только инфраструктуры облачного провайдера.

«Облачные сервисы» имеют значительные плюсы по сравнению с традиционными вариантами архитектуры, применяемой в параллельных вычислениях. В статье [6] авторы выделяют следующие плюсы, по сравнению с традиционным «дата-центром»: возможность получения бесконечных ресурсов по требованию; отсутствие необходимости авансовых обязательств для облачного сервиса; возможность оплачивать вычислительные ресурсы в короткие сроки; экономия средств, ввиду большего масштаба «дата-центров»; большая утилизация вычислительных ресурсов, путем суммирования нагрузки от многих клиентов (при традиционном подходе это также возможно, однако при больших объемах ресурсов, что превращает его в некое корпоративное облако); улучшенная эксплуатация и утилизация, за счет использования виртуализации. Среди перечисленных характеристик, стоит особенно отметить возможность получения большого количества дополнительных вычислительных ресурсов, и их экономическую эффективность. Эти пункты являются важными, поскольку такой тип параллельных вычислений подразумевает под собой высоконагруженные и, относительно, непродолжительные вычисления, что в свою очередь накладывает определенные ограничения на инфраструктуру, используемую для этих вычислений.

В работе [7] авторы выделяют угрозы информационной безопасности для облачной системы, среди которых технические средства обработки информации, программное обеспечение. Авторы статьи [8] выделяют схожие угрозы для потребителей облачных сервисов, а именно: небезопасные интерфейсы и *API*, вредоносные инсайдеры, ошибки и проблемы технологий общего доступа, потеря или утечка данных, похищение аккаунта или сервиса, высокий контроль поставщика и низкий потребителя. Обобщая перечисленное, можно подтвердить, что угроза несанкционированного доступа к данным или их потеря, является основной проблемой систем, использующих облачные вычислений. Необходимость соблюдения правил и рекомендаций при работе с такими системами является необходимостью, а для некоторого рода задач, связанных, например, с не обезличенными приватными данными пользователей, и вовсе не имеет возможности для работы.

Под термином параллельные вычисления подразумевается такой тип вычислений, в котором множество действий происходит одновременно [9], который базируется на том, что более сложные задачи, могут быть поделены на простые и решены одновременно. Параллелизм существует на нескольких уровнях: на уровне данных, уровне инструкций, на уровне потоков и параллелизм на уровне процессов [10], однако, если речь идет о распределенных вычислительных системах, то следует выявить параллелизм на уровне целых задач. Сегодня параллельные вычисления распространились на всех уровнях, и стали доминирующей компьютерной архитектурой. Термин параллельных вычислений имеет широкое понятие и включает в себя множество, более специализированных терминов, и их можно разделить на следующие классы: многопоточные вычисления, симметричное мультипроцессирование; распределенные вычисления, которые также можно разделить на: кластерные вычисления; вычисления в массово-параллельной архитектуре; грид-вычисления.

Также можно выделить специализированные параллельные вычисления, в которые входит: *GPGPU*, *ASIC*, векторные вычисления и т.д., но они имеют узкую специализацию.

Материалы и результаты исследования. Задачи, по проведению вычислений можно разделить на крупные классы, где критерием различия будет являться эффективность масштабирования алгоритмов при распараллеливании. На эффективность таких вычислений, в том числе и распределенных оказывают влияние следующие поззиции [11]: доля параллельных вычислений к доле не параллельных вычислений в обще задаче; число и размер обменов данными между независимыми вычислительным узлами; сложность вычислений на каждом узле; число потребляемой памяти для вычислений; частота прерываний и возобновлений вычислений, связанных с вопросами синхронизации, отказами в работе, работами по управлению.

Использование облачных сервисов для организации параллельных вычислений, подразумевает распределенные вычисления, поскольку вычислительная система содержит множество элементов, выполняющих часть общей задачи. Любая распределенная система может одну из представленных архитектур:

• клиент-серверная архитектура – толстый клиент обращается к серверу за данными;

• трех уровневая архитектура – тонкий клиент обращается серверу, а сервер обращается к базе данных;

• многоуровневая архитектура – полученный запрос от клиента может передаваться большое количество раз;

• *peer-to-peer* архитектура – вся работа равномерна распределена между вычислительными элементами, именно эта архитектура используется при организации вычислений, в том числе при использовании облачных сервисов;

В системах распределенных вычислений каждая вычислительная сущность называется нодом («node») или пиром («peer»).

Бурый А. С. в своей работе [12] выделяет следующие преимущества у распределенных систем на основе облачных технологий: доступность, низкая стоимость, гибкость, надежность, безопасность, большие вычислительные мощности, и следующие недостатки: зависимость от среды передачи данных (интернет), ограничение используемого поставщиком ПО, конфиденциальность, надежность, безопасность (при слабой организации процесс работы системы).

Результаты исследования. Любая параллельная вычислительная система создается для ускорения решения той или иной задачи. Параллельные системы успешно справляются с этой задачей. Для измерения производительности параллельных вычислений используются специальные метрики, например ускорение выполнения работы в параллельной системе, сравнительно с последвательным вычислением («*speedup*»), но это понятие достаточно широко, поскольку понятия последовательного и параллельного вычислений разнятся, поэтому существует несколько метрик, таких как относительное ускорение, реальное ускорение, абсолютное ускорение, асимптотическое реальное ускорение и асимптотическое относительное ускорение [13].

$$Speedup_{parallel}(f,n) = \frac{1}{(1-f)+\frac{f}{n}} \quad (1)$$

Приведенное выражение описывает ускорение вычислений в параллельной системе [14]. На рисунке 1 приведены графики, показывающие теоретические ограничения для параллельных вычислений на основе закона Амдала.

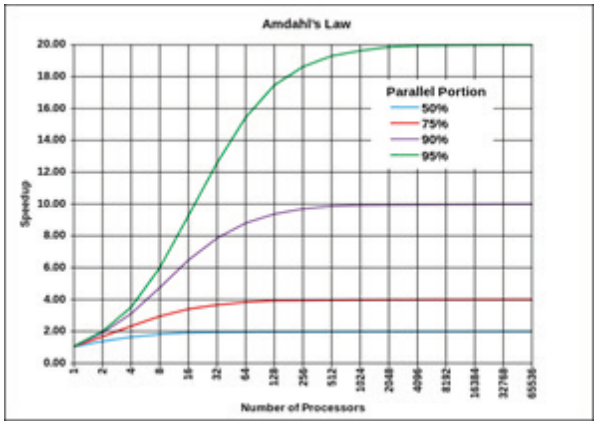

Рисунок 1 – Теоретические ограничения для параллельных вычислений на основе закона Амдала

Известен закон Густафсона-Барсиса, который оценивает максимальное ускорение выполнения параллельной программ, в зависимости от количества одновременно выполняемых потоков вычислений и доли последовательных расчетов, и представляет из себя следующее:

$$S_p = g + (1-g)p = p + (1-p)g \quad (2)$$

где g – доля последовательных расчетов в программе, p – количество процессоров.

Перечисленные характеристики и законы относятся к параллельным вычислениям, и хоть распределенные вычисления так же используют множество процессоров для проведения параллельных вычислений, данные законы не могут предоставить точной модели для оценки производительности распределенных вычислений, особенно при их использовании в «облаке». Данные законы не оценивают возможность использования изначально распределенных данных, за счет распределенных СБУД, а также необходимость пересылки данных между вычислительными узлами. В целом данные законы могут быть использованы при работе с распределенными вычислениями со значительными ограничениями, поскольку они не предусматривали горизонтальной масштабируемости распределенных систем.

Основным способом организации параллельных вычисления для организации вычислений на распре-

деленных частях сетей является модель *MapReduce*, представленная компанией *Google* [15]. Согласно, «*MapReduce Design Patterns*» *Map Reduce* – это вычислительная парадигма для обработки данных, которая происходит на сотнях компьютеров [16, 17]. Данный метод состоит из 3 основных стадий: *Map* – каждый из узлов системы применяет функцию *Map* на часть локальных данных, *shuffle* – данные из каждого узла перемещаются в зависимости от полученного на выходе функции *map*, *reduce* – на каждом узле полученные данные обрабатываются. Схема работы представлена на рисунке 2 [12]. Несколько модифицированная *MapReduce* используется в *Hadoop*.

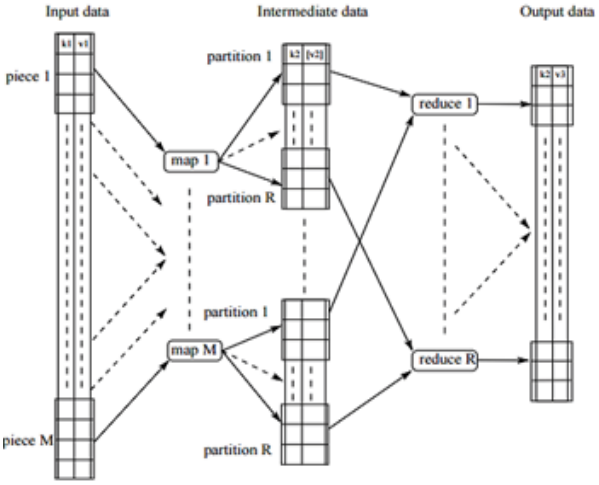

Рисунок 2 – Схема работы MapReduce

Согласно материалам статьи [18] простого автоматизирования процессов конфигурации, размещения и резервирования не достаточно, необходимы специальные оптимизации данных процессов, для достижения максимальной эффективности. Для выполнения параллельных распределенных вычислений применяются различные программные продукты. Данная область не богата на альтернативы и используемое ПО скорее решает более специфические задачи, такие как решение задач графов. Поэтому выбор используемого ПО должен быть ориентирован область применения и конкретной задачей. Не менее важным фактором выбора является модель проведения вычислений, и объем данных. Так, например, среди методов вычислений можно выбрать встроенные средства какой-либо *NoSQL* СУБД (*Aggregation Framework* для *MongoDB*), или использование статистических методов и специализированной системы хранений (например, *Amazon Redshift*). Однако перечисленные методы не являются абсолютными альтернативами готовым решениями, поскольку ограничены по объему обрабатываемых данных, либо требуют значительных усилий со стороны разработчик, в виде построения системы хранения, и написания системы обработки.

На текущий момент крупнейшей платформой для проведения параллельных вычислений является *Hadoop*, огромной платформой, состоящей из множества подсистем, модулей и элементов. Альтернативным решением можно считать *Spark*, но он использует часть инфраструктура проекта *Hadoop*. Между этими системами существуют отличия [19, 20]. На рисунке 3 приведено сравнение времени выполнения итеративных вычислений на *Hadoop* и *Spark*.

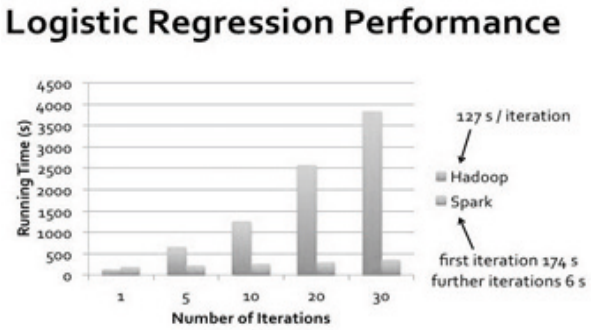

Рисунок 3 – Сравнение времени выполнения итеративных вычислений

Соответственно использование *Spark* несет свои ограничения, связанные с оперативной памятью, а также требует максимально возможное количество, что способствует росту цены используемых облачных сервисов.

Таким образом, если сравнивать *Spark* и *Hadoop* можно сделать вывод, что *Spark* позволяет максимально эффективно использовать оперативную память и процессор, что является основными пунктами стоимости вычислительной сущности, получаемой в облачном сервисе. А также он крайне эффективен при использовании его дополнительных возможностей, таких как машинное обучение с помощью встроенной библиотеки *Mlib*, *SparkR* библиотеки, позволяющий использовать язык *R* для доступа к данным.

«*Apache Hadoop*» – продукт с открытым исходным кодом, предназначенный для надежного, масштабируемого, распределенного вычисления. *Hadoop* спроектирован масштабируемым от 1 до 1000 компьютеров, имеющий локальные вычислительные ресурсы и хранилище данных [11]. *Hadoop* является инструментом с большим количеством всевозможных систем, подсистем и расширений. Одной из самых главных частей *Hadoop* является ее распределенная «файловая система» – *HDFS*. Однако назвать ее файловой системой нельзя, скорее она является распределенной базой данных. Отличает ее от обычной СУБД наличие следующий факторов [21]: объем хранимой информации; необходимость использования файлов, описывающих конкретную выборку из данных; отсутствие транзакций, нормализации данных, ограничений целостности; и др.

Однозначно, несмотря на отказ от большого количества функций, содержащихся в традиционных СУБД, *HDFS* достигает необходимой цели – линейной и не ограниченной масштабируемости. Помимо *HDFS* в *Hadoop* входят и другие модули: *Hadoop MapReduce*, программный компонент, непосредственно отвечающий за выполнение модели *MapReduce*; *YARN*, модуль, отвечающий за управление ресурсами

кластеров и планирование; *Hadoop Common*, набор различных утилит для поддержки остальных модулей

На рисунке 4 изображена схема работы модуля *MapReduce* для одного из элементов кластера. Он отвечает за непосредственную обработку данных, хранящихся в файловой системе *HDFS*, и позволяет разработчику перейти на более высокий уровень абстракции при работе с распределенными вычислениями путем автоматического распараллеливания программ и обеспечения отказоустойчивости системы.

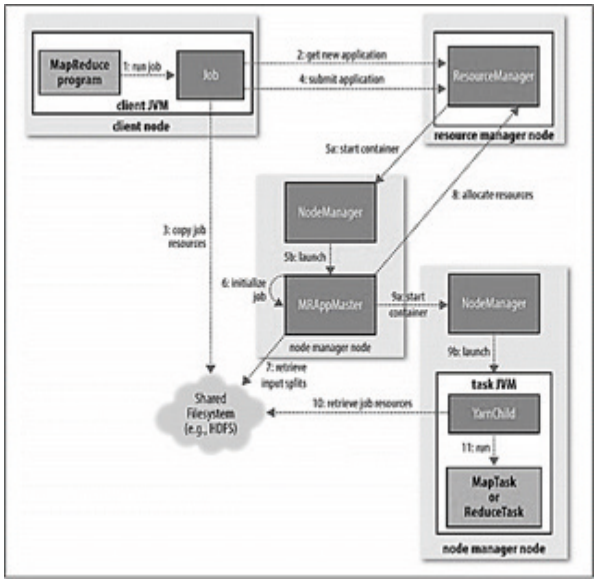

Рисунок 4 – Выполнение работы для одного элемента распределенной системы

На рисунке 5 изображена схема работа системы *YARN*, задача которой заключается в управлении несколькими фреймворками, в случае если *MapReduce* не может справится с определенного рода задачами, такими как вычисления в реальном времени, итеративные вычисления, вычисления графов и т.д. Так же эта система содержит *YARN Resource Manager,* который является главным процессом, планирующим вычисления и управляющим ресурсами для поставленной задачи.

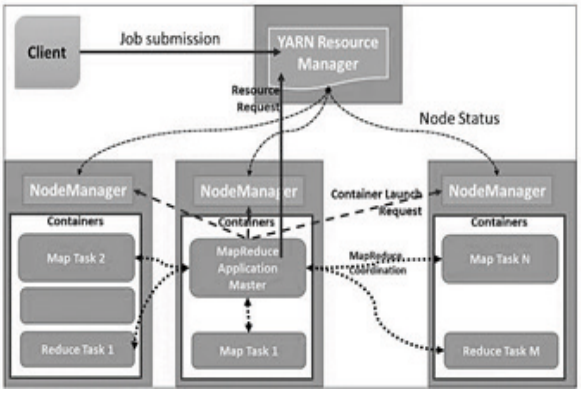

Рисунок 5 – Схема работы YARN

*YARN Node Manage*r представляет из себя процесс, находящийся на каждом ноде, управляющем локальными ресурсами.

Суть практического использования *Hadoop* заключается в: установке и конфигурации вычислительного кластера; загрузке входных данных в файловую систему *HDFS*; формировании приложения, выполняющегося на всех нодах распределенной системы; запуске всех модулей системы и ожидание процесса вычисления. *Hadoop* содержит несколько средств тестирования HDFS кластера для проверки нормальной работоспособности построенной системы. Для анализа данных производительности ввода\вывод используется *DFSIO*. Эта система использует *Map* задачи для записи и чтения файлов параллельно, а *Reduce* используется для суммирования и сбора данных о производительности. Также существует *TeraSort* тест, использующий набор данных объемом 1 терабайт, который необходимо отсортировать. Так проверяется работа всех компонентов системы [22]. Достижение максимальной производительности исследуемой может быть достигнуто с применением фреймворка *Spark*. Его производительность составляет двух - трехкратное уменьшение времени вычислений, вплоть до десятикратных ускорений для определенных задач. Данный факт отмечен в работе [23]. При сравнении *Spark* с фреймворками не находящимися в экосистеме hadoop, стоит отметить возможность добавлять вычислительные во время исполнения, возможность использовать распределенных файловую систему *HDFS*, с поддержкой управления отказами и менеджментом репликации, а также легким способом деплоймента инструментов для анализа и управления данными [24]. Из перечисленных пунктов особенно важными пунктами является управление отказами, поскольку облачные сервисы не могут гарантировать уровень стабильности самих вычислительных нодов, так и сетевых соединений между этими узлами.

Сам *Spark* является *in-memory* вычислительным фреймворком, за счет чего и достигается его высокая производительность [25]. По причине использования памяти возникает и соответствующая проблема, *OutOfMemory Exception*, проблема нехватки памяти при выполнении определенных задач, за исключением того *Spark* масштабируется линейно с количеством нод [26].

Построение вычислительной распределенной системы с использованием облачных сервисов. Для создания образов будут использоваться виртуальные образы, предоставляемые поставщиком облачных сервисов *Digital Ocean, Core OS*, следующих характеристик: *CPU:1; RAM: 1GB; SSD: 30 GB; 2TB: transfer*.

Для создания одного экземпляра (*single-node*) *Hadoop* необходим набор определенных средств для его работы, таких как установка *Java* 1.5 +, конфигурация *SSL*, добавление специализированного пользователя и т.д.

Hadoop не имеет единой глобальной точки хранения для каждого экземпляра *hadoop*, конфигурация определена набором конфигурационных файлов, которые являются общими для всех узлов распреде-

ленной вычислительной сети, для синхронизации конфигурации могут быть использованы различные утилиты, такие как *rsync, dsh u pdsh*. В набор конфигурационных файлов кластера входят следующие:

• *hadoop-env.sh* – переменные окружения среды *Hadoop*;

• *core-site.xml* – параметры конфигурации *hadoop core*;

• *hdfs-site.xml* – параметры конфигурации демонов *hdfs*;

• *mapred-site.xml* – парметры конфигурации демонов *MapReduce*;

• *masters* – список физических узлов на каждом из которых работает вторичный узел имен;

• *slaves* – список физических узлов на каждом из которых работает узел данных и трекер задач;

• *hadoop-metrics.properties* – свойства, управлявшие публикацией метрик;

• *log4j.properties* – свойства различных системных журналов, журнала аудита узла имен и журнала задач для дочернего процесса трекера задач.

Использование средств конфигурирования позволяет многократно сократить время на создание кластера *Hadoop*, как при использовании готовых образов, так и сконфигурированных вручную. Достоинства использования средств оркестрации при организации распределенных вычислений в облачном сервисе:

• легко расширяется и масштабируется;

• содержит множество дополнительных инструментов.

Недостатками данного средства оркестрации является то, что оно обладает меньшей гибкостью интеграции с дополнительными инструментами обработки больших данных за счет меньшей связности компонентов [27].

Несмотря на перечисленные недостатки, в архитектуре системы будет использоваться средство оркестрации *Ansible*, поскольку система позиционирует себя как быстро-развертывающаяся, что достижимо только при использовании такого средства автоматизации.

Архитектура динамической распределенной вычислительной системы для проведения вычислений в облачном сервисе. Для взаимодействия с облачным сервисом, управлением размещения контейнеров и конфигурации будет использовано известное средство оркестрации *Ansible*. Общая схема работы системы продемонстрирована на рисунке 6.

В общей архитектуре системы не указаны объекты, отвечающие за доставку, получение, хранение и отправку данных, по которым осуществляются вычисления. Связано это с тем, что решение данной задачи может быть выполнено различными способами в зависимости от объема данных и необходимой задачи. Например, при незначительном объеме данных (меньше 1 ГБ) данные могут быть загружены локально, на один из узлов, а далее *Spark* во время исполнения поделит данные между отдельными исполнителями и проведет решение. Если количество данных намного больше возможна организация на каждом вычислительном узле, отдельной ноды *HDFS*. Либо для каждого вычислительного узла, будет создан элемент кластера какого-либо *NoSQL* хранилища данных, например *Kafka* или *Cassandra*. Если же данные методы не позволяют использовать объем данных или ограничения облачного сервиса, то существует возможность использовать *Spark Streaming*, суть данного метода заключается в том, что данные поступают на кластер в виде множества потоков из указанного источника данных, такого как *Kafka, Flume, ZeroMQ* или не связанная с этим кластером *HDFS*.

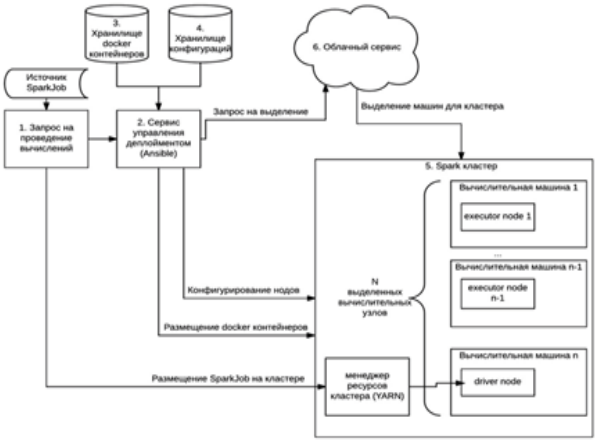

Рисунок 6 – Общая архитектура распределенной системы

Выполнение распределенных вычислений в распределенной системе. В качестве основного изменяемого параметра был выбрано количество *executor*, узлов, проводящих непосредственные вычисления распределенного алгоритма. Их количество будет являться степенью 2, для удобства сравнения с асимптотической функции, представленной, например, в законе Амдала. На один *executor* выделено 1 вычислительное ядро, так как такая конфигурация инстанса была предоставлена облачным сервисом. В ходе проведения эксперимента было выявлено, что объем памяти выделяемый для каждого *executor* не оказывает влияния на производительность, а следовательно, и на общее время выполнения вычислений. Связано это с тем, что общий объем тестовых данных, его отдельные партиции (*partition*) и промежуточные данные, генерируемые в ходе вычислений, могут целиком помещены в память, при больших объемах данных, данный параметр оказывает большее значение. Результаты экспериментов продемонстрированы в таблице 1, значениями является общее время выполнения отдельного алгоритма, без учета загрузки, выгрузки и очистки данных. Время указано в миллисекундах.

Таблица 1 – Результаты проведения экспериментов

	1	2	4	8	16
sort	91055	49750	27161	17306	8103
wordcount	126806	68133	38129	25501	14703

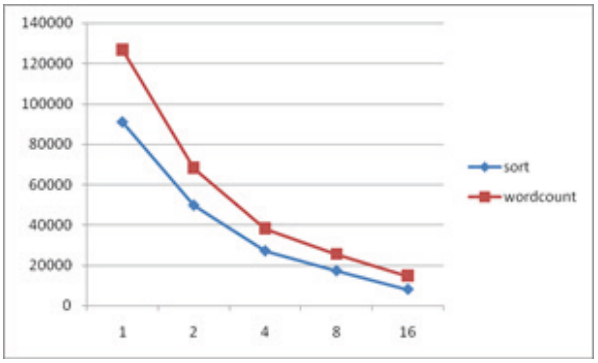

Рисунок 7 – Время выполнения вычисления для алгоритмов sort и wordcount

Результаты вычислений (рис. 7) свидетельствуют о значительном ускорении, близком к линейному, связано это с эффективностью парадигмы *map reduce*, не требующей значительных требований по вычислению не парализуемых вычислений. Стоит отметить, что в рассмотренной выборке не учитывается процесс получения данных в одном узле вычислений, что бы потребовало значительных временных ресурсов, которые требуются для загрузки и выполнения дополнительных вычислений. Основываясь на полученных данных, вычисляются значения ускорения для выполненных вычислений по выражению

$$S_p = \frac{1}{\alpha + \frac{1-\alpha}{p}} \qquad (3)$$

где α – доля времени вычисления, не являющаяся параллельными, а p – доля не параллельных вычислений.

В качестве целевой функции, по которой будут оцениваться результаты вычислений, выбраны зна-чения α, равные 0.025, 0.05, 0.1 и 0.75. Данные значения позволяют проанализировать эффективность построенной системы. На рисунке 8 продемонстрирован график роста ускорения производительности алгоритма *sort* с увеличением количества вычислительных узлов. Данные результаты указывают на то, что ускорение *sort* алгоритма в распределенной вычислительной системе сходна с параллельным вычислением с долей параллельных вычислений от 95 до 97.5 процентов.

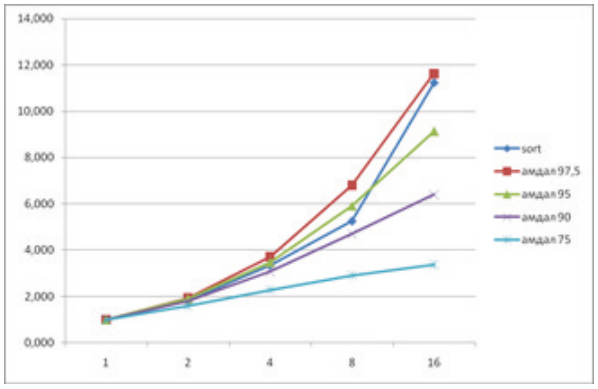

Рисунок 8 – Ускорение для алгоритма sort и закон Амдала для разной доли параллельных вычислений

На рисунке 9 продемонстрирован график роста ускорения производительности алгоритма *word-count* с увеличением количества вычислительных узлов. Данные результаты указывают на то, что ускорение *wordcount* алгоритма в распределенной вычислительной системе сходна с параллельным вычислением с долей параллельных вычислений от 90 до 95 процентов.

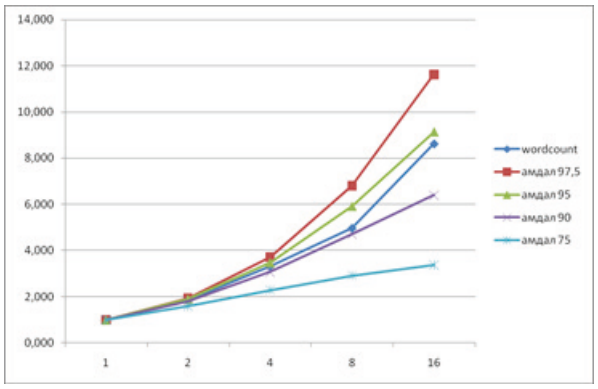

Рисунок 9 – Ускорение для алгоритма wordcount и закон Амдала для разной доли параллельных вычислений

Обобщая полученные результаты, приходим к выводу, что фреймворк *spark* эффективно использует парадигму параллельных вычислений и достигает практически линейной масштабируемости для количества вычислительных узлов меньше 16. Стоит отметить, что качественной характеристикой производительности вычислительного кластера может являться только общее время выполнения, особенно это касается комплексных задач, состоящих из меньших этапов вычислений, таких как функции *map*, *reduce, filter* и т.д. Поскольку временную величину трудно оценивать отдельно от контекста предыдущих выполнений определенного набора алгоритмов, стоит использовать регрессионное тестирование. Такая оценка позволяет оценивать качество проводимых операций по оптимизации и улучшения, как в программе содержащее выполняемые алгоритмы, так и в параметрах управляемых процессом вычислений.

Проведенные в работе эксперименты не указывают в себе этапы выполнения распределенного вычисления, которые включают в себя процессы по работе с операциями ввода/вывода и операции пересылки по сети. Данные операции трудно объективно оценить по ряду причин [28, 29].

Заключение. В статье проанализированы основные подходы к организации параллельных вычислений на распределенных системах. Рассмотрены современные методы организации таких систем, их положительные и отрицательные стороны, а также теоретическая и реальная производительность. Проведены оценки метрик эффективности распределенных и параллельных систем, на примере закона Амадала.

В работе описана архитектура вычислительного кластера, основанная на инфраструктуре облачного сервиса. Синтезированная архитектура нацелена на

высокую скорость развертывания сущностей кластера. В предлагаемой архитектуре используются средства контейнеризации для обеспечения стандартизированного подхода по управлению элементами кластера и независимой работы компонентов одного вычислительного узла и использования его не только как *spark executor*, но и узла *HDFS* или элемента распределенной СУБД. Такой подход позволяет использовать работу с данными, которые могут располагаться на развертываемом кластере, так и из другой системы при помощи стримнга.

В работе проведен ряд экспериментов, оценивающих эффективность построенной системы. Проведен ряд тестов, оценивающих производительность кластера для алгоритмов *wordcount* и *sort*. Анализ полученных результатов показал, что *spark* может эффективно использовать распределенные вычисления, сравнимые с теоретической производительностью по закону Амдала от 90% вплоть до 97,5% доли задачи возможной для распараллеливания. В рамках анализа возможной оценки производительности не алгоритмической части, отвечающий за запись/чтение принято решение о невозможности объективной оценки данного компонента распределенного вычисления. Кроме того, определена ключевая метрика эффективности работы кластера, а именно общее время проведения распределенного вычисления.

В ходе работы по тематике настоящей статьи выведен ряд сложностей и особенностей построения кластеров для проведения распределенных вычислений в облачной архитектуре, которые приведены далее. Классический кластер подразумевает высокое качество сети между вычислительными нодами, а именно высокая скорость передачи данных, отсутствие или крайне низкое количество потерь в канале, вследствие близкого физического расположения физически машин и общей инфраструктуры сети. При использовании облачной инфраструктуры публичных облаков для построения вычислительного кластера данные особенности накладывают ограничения на максимальную производительность кластера. Как было отмечено в статье, операции ввода/вывода менее эффективны, по сравнению с выделенным кластером, и это также является узким местом для рода задач, связанных с большим объемами записи в персистентную область памяти.

СПИСОК ЛИТЕРАТУРЫ:

1. J. Koomey 4 reasons why cloud computing is efficient [Электронный ресурс]. 2011. Режим доступа: http://www.reuters.com/article/2011/07/25/idUS59089929820110725 свободный. Яз. англ. (дата обращения: 14.05.2020).

2. Apache Spark [Электронный ресурс]. 2020. Режим доступа: https://en.wikipedia.org/wiki/Apache_Spark свободный. Яз. англ. (дата обращения: 14.05.2020).

3. H. Karau, R. Warren High Performance Spark Best Practices for Scaling and Optimizing Apache Spark [Текст] / H. Karau. — O'Reilly Media, 2016. — 358 с.

4. H. Karau, A. Konwinski, P. Wendell, M. Zaharia Learning Spark: Lightning-Fast Big Data Analysis 1st Edition [Текст] / H. Karau. — O'Reilly Media, 2015. — 276 с.

5. DataWorks Summit [Электронный ресурс]. 2012. — Режим доступа: https://hadoopsummit.czcams.com/ свободный. Яз. англ. (дата обращения: 14.05.2020).

6. M. Armbrust A view of cloud computing / M. Armbrust [и др.] // Magazin, Communications of the ACM. — 2010. —№53. - С. 50–58.

7. В.В. Богданов Актуальность обеспечения информационной безопасности в системах облачных вычислений, анализ источников угроз / В.В. Богданов, Ю.С. Новоселова // Математическое обоснование и теоретические аспекты информационной безопасности. — 2012. — №4. - С. 78–82.

8. Top Threats to Cloud Computing V1.0 [Электронный ресурс]. 2010. Режим доступа: URL: https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf свободный. Яз. англ. (дата обращения: 14.05.2020).

9. A. Gottlieb Highly parallel computing (2nd ed.) [Текст] / A. Gottlieb, G. S. Almasi. — Benjamin-Cummings Publishing Co. —1994. —с. 162.

10. S. Sahni Parallel Computing: Performance Metrics and Models / S. Sahni, V. Thanvantri [Электронный ресурс].1996. Режим доступа: http://ufdcimages.uflib.ufl.edu/UF/00/09/53/64/00001/1996206.pdf свободный. Яз. англ. (дата обращения: 14.05.2020).

11. What Is Apache Hadoop? [Электронный ресурс]. 2020. Режим доступа: URL: http://hadoop.apache.org/ свободный. Яз. англ. (дата обращения: 14.05.2020).

12. А. С. Бурый Тенденции развития распределенных информационных систем на основе облачных технологий / А. С. Бурый // Транспортное дело России. — 2013. —№6. - С. 70–72.

13. G. Fayez Algorithms and Parallel Computing [Текст] / G. Fayez. — Wiley, 2011. — 341 с.

14. M. D. Hill Amdahl's Law in the Multicore Era/ M. D. Hill, M. R. Marty // Computer. — 2008. —№41. - С. 33–38.

15. J. Dean, Sanjay Ghemawat MapReduce: Simplified Data Processing on Large Cluster / J. Dean, S. Ghemawat // OSDI'04: Sixth Symposium on Operating System Design and Implementation [Электронный ресурс]. 2004. Режим доступа: URL: http://research.google.com/archive/mapreduce.html свободный. Яз. англ. (дата обращения: 14.05.2020).

16. D. Miner MapReduce Design Patterns [Текст] / D. Miner, A. Shook. — O'Reilly Media, 2012. — 250 с.

17. Мартышкин А.И., Сальников И.И., Пащенко Д.В. Этапы сбора и представления больших данных для построения социального профиля человека // Известия Тульского государственного университета. Технические науки. – 2018. – № 9. – С. 617-628.

18. K. Kambatla, A. Pathak, H. Pucha Towards Optimizing Hadoop Provisioning in the Cloud [Электронный ресурс]. Режим доступа: https://www.usenix.org/legacy/events/hotcloud09/tech/full_papers/kambatla.pdf свободный. Яз. англ. (дата обращения: 14.05.2020).

19. Apache Spark vs Hadoop MapReduce [Электронный ресурс].2019. Режим доступа: http://www.edureka.co/blog/apache-spark-vs-hadoop-mapreduce (01.05.17)

20. Spark Framework [Электронный ресурс]. 2020. Режим доступа: http://spark.apache.org/ свободный. Яз. англ. (дата обращения: 14.05.2020).

21. What Is Apache Hadoop? [Электронный ресурс]. 2020. Режим доступа: http://hadoop.apache.org/ свободный. Яз. англ. (дата обращения: 14.05.2020).

22. T. White Hadoop: The Definitive Guide, 4th Edition [Текст] / T. White. — O'Reilly Media, 2015. — 756 с.

23. S. Perera Hadoop MapReduce Cookbook Second Edition [Текст] / S. Perera. — Packt Publishing Ltd, 2015. — 290 с.

24. L. Liu Performance comparison by running benchmarks on hadoop, spark, and hamr [Текст] — ProQuest, 2016. — 51 p.

25. J. L. Reyes-Ortiz Big Data Analytics in the Cloud: Spark on Hadoop vs MPI/OpenMP on Beowulf / J. L. Reyes-Ortiz, L. Oneto, D. Anguita // Procedia Computer Science. — 2015. —№53. - С. 121–130.

26. M. LI SparkBench: A Comprehensive Spark Benchmarking Suite Characterizing In-memory Data Analytics / M. LI, J. Tan, Y. Wang, L. Zhang, V. Salapura, A. Bivens [Электронный ресурс]. 2015. Режим доступа: https://research.spec.org/fileadmin/user_upload/documents/wg_bd/BD-20150401- spark_benchmark-v1.3-spec.pdf свободный. Яз. англ. (дата обращения: 14.05.2020).

27. A. Davidson Optimizing Shuffle Performance in Spark / A. Davidson, A. Or [Электронный ресурс]. — Режим доступа: https://pdfs.semanticscholar.org/d746/505bad055c-357fa50d394d15eb380a3f1ad3.pdf свободный. Яз. англ. (дата обращения: 14.05.2020).

28. О.Д. Борисенко, Создание виртуальных кластеров Apache Spark в облачных средах с использованием систем оркестрации. / Пастухов Р.К., Кузнецов С.Д. // Труды ИСП РАН. — 2016.— №28/6 - С. 111-120.

29. D. Ghoshal I/O Performance of Virtualized Cloud Environments / D. Ghoshal, R. S. Canon, L. Ramakrisnan // DataCloud-SC '11 Proceedings of the second international workshop on Data intensive computing in the clouds. — 2011.— №11 - С. 74-80.

Статья публикуется при поддержке гранта РФФИ «Конкурс на лучшие проекты фундаментальных научных исследований» (Грант № 19-07-00516 А).

Статья поступила в редакцию 05.05.2020
Статья принята к публикации 10.06.2020