Экспериментальная оценка временной эффективности обработки больших данных в заданных форматах хранения

В.А. Белов, Е.В. Никульчев

***Аннотация*— Одной из важнейших задач современной платформы обработки больших данных является задача выбора форматов хранения данных. Выбор форматов опирается на различные критерии эффективности, которые зависят от класса объектов и предъявляемых требований. Одними из наиболее важных являются временные затраты в различных операциях обработки больших данных. В статье исследуются пять наиболее популярных форматов для хранения больших данных (avro, CSV, JSON, ORC, parquet), предложен экспериментальный стенд для оценки временной эффективности, проведен сравнительный анализ экспериментальных оценок характеристик рассматриваемых форматов. Для эксперимента рассматривались основные операции обработки данных с использованием фреймворка Apache Spark. Алгоритм выбора формата разработан на основе метода анализа иерархий. В результате сформирована методика выбора формата из альтернатив на основе экспериментальных оценок параметров и метода анализа иерархий для задачи выбора эффективных по времени выполнения базовых операций форматов хранения больших данных в системе Apache Hadoop с использованием Apache Spark.**

***Ключевые слова*—большие данные, форматы хранения, временная эффективность, Apache Hadoop, Apache Spark.**

I. Введение

Развитие технологий обработки данными привело к росту инструментов для обработки больших данных [1]. Под большими данными понимаются данные, собираемые из различных источников и обладающие такими объемами, что их обработка традиционными методами становится очень сложной или невозможной [2, 3]. В то же время большинство исследователей согласны с тем, что под большими данными понимается не только их объем, но и их способность быть источником для новых знаний и идей [4].

Одним из направлений в сфере аналитической обработки данных стала разработка платформ для работы с большими данными [5]. Такие платформы предназначены для обработки и хранения собранных данных. В последние несколько лет инструментом для таких платформ стала система Hadoop [6], представляющая собой набор утилит [7] программного обеспечения, ядром которых служит распределенная файловая система, хранящая данные в определенных форматах, и обработчик данных, реализующий модель обработки MapReduce [8]. Однако из-за различных ограничений данной системы были разработаны новые реализации систем обработки больших данных (например, Hive [9], Impala [10], Apache Spark [11] и т. д.). С одной стороны, эти пакеты программ являются самостоятельными продуктами, а с другой – дополнениями для системы Apache Hadoop.

Фреймворки, такие как Apache Spark [12], позволяют работать с различными форматами файлов. При разработке системных архитектур на основе Apache Hadoop возникает задача выбора формата хранения данных.

Исследование направлено на разработку методики, позволяющей обоснованно выбрать формат данных, эффективных по времени обработки в заданных условиях. Для исследования были выбраны пять форматов: avro, CSV, JSON, ORC и parquet. Задачей исследования является изучение особенностей форматов файлов, используемых для хранения больших данных, а также проведение экспериментальной оценки временной эффективности форматов.

Проблема выбора компонентов программного обеспечения одна из важных задач разработки [13, 14]. В [15–19] представлены исследования форматов хранения больших данных, таких как avro, parquet, ORC и т. д. Эти исследования представляют результаты изучения различных форматов с точки зрения производительности или выбора формата для конкретной цели. Существуют работы по форматам файлов для хранения данных в веб-системах для конкретных задач [20]. В [21] рассматривается аналогичные задачи ресурсной эффективности, но исследованы только форматы хранения avro и parquet. Следует отметить, что хранение данных может осуществляться с использованием реляционных баз данных. Однако в последние годы стали популярными решения NoSQL [22], некоторые из которых поддерживают различные форматы хранения данных. Большинство исследований предоставляют результаты изучения каждого формата и рекомендации по выбору изучаемой проблемы.

Использование неэффективного формата для хранения больших данных может привести к различным ошибкам и затруднениям при работе с данными. Так, использование форматов, не поддерживающих сложные

Статья получена 05 июля 2021 г.
Работа выполнена в рамках Госзадния Министерства науки и Высшего образования Российской Федерации
В. А. Белов, аспирант, МИРЭА – Российский технологический университет; ведущий аналитик, РАО (belov_v.a@mail.ru)
Е. В. Никульчев, д.т.н, профессор, МИРЭА – Российский технологический университет; профессор РАО (nikulchev@mirea.ru).

структуры данных (например, массивы или даты), может привести к неправильным результирующим наборам при выборке данных с помощью инструментов, подобных языку SQL, в системах, таких как Hadoop. Кроме того, использование форматов, в которых не используются архивирование данных или метаданные, может привести к увеличению времени поиска необходимой информации в данных. Следовательно, для систем, где критично время обработки аналитических данных, может возникнуть вынужденная задержка.

В основе предлагаемой в статье методики лежит экспериментальная оценка форматов хранения данных и модель для выбора оптимального решения на основе метода анализа иерархий [23, 24]. Для выбора формата данных в статье решается задача построения системы оценивания, системы критериев и методов получения их достоверных численных значений на основе экспериментальных исследований, что наиболее адекватно отражает целесообразность использования форматов в заданных условиях.

II. Методы и эксперименты

Рассмотрены следующие форматы хранения данных: avro, CSV, JSON, ORC, parquet.

Avro является линейно-ориентированным форматом хранения данных. Основная особенность формата заключается в наличии схемы в формате JSON, что позволяет быстрее проводить операции чтения и интерпретации. Структура файла состоит из заголовка и блоков с данными. Формат поддерживает эволюцию схем данных, обрабатывая изменения схемы путем пропуска, добавления или модификации отдельных полей. Avro не является строго типизированным форматом: информация о типе каждого поля хранится в разделе метаданных вместе со схемой. Благодаря этому для чтения сериализованной информации не требуется предварительное знание схемы.

Comma-Separated Values представляет из себя текстовый формат, описывающий данные в табличном виде. Структура CSV файла представлена в виде строк, разделенных запятой. Предполагается наличие заголовка, однако это требование нестрогое. CSV файл не поддерживает разные типы и структуры данных – все данные представлены в виде строк.

JavaScript Object Notation является простым текстовым форматом, основанном на подмножестве языка программирования JavaScript. Наибольшее применение данный формат имеет в системах обмена данными, разработке API, удаленном вызове процедур. Однако с появлением NoSQL решений JSON формат приобрел популярность в хранении больших данных в документных базах данных. JSON поддерживает такие типы и структуры данных, как строка, число, логический тип, массивы, значение null, а также внутренние объекты.

Optimized Row Columnar – колоночно-ориентированный формат хранения больших данных в экосистеме Apache Hadoop. ORC оптимизирован для чтения потоков больших данных, включая интегрированную поддержку быстрого поиска нужных строк. Колоночное хранение данных позволяет читать, распаковывать и обрабатывать только те значения, которые необходимы для текущего запроса. Поскольку данные в ORC строго типизированы, поэтому при записи выбирается кодировка, наиболее подходящая для каждого типа данных, создавая внутренний индекс по мере записи файла [25].

Apache Parquet – это бинарный колоночно-ориентированный формат, позволяющий использовать преимущества сжатого представления информации. Паркет позволяет задавать схемы сжатия на уровне столбцов и добавлять новые кодировки по мере их появления. Parquet использует архитектуру, основанную на «уровнях определения» (definition levels) и «уровнях повторения» (repetition levels), что позволяет довольно эффективно кодировать данные, а информация о схеме выносится в отдельные метаданные. Формат Parquet явно отделяет метаданные от данных, что позволяет разбивать столбцы на несколько файлов, а также иметь один файл метаданных, ссылающийся на несколько файлов паркета.

Для оценки форматов хранения данных было проведено исследование, состоящее из трех частей:

1) сравнительный анализ основных характеристик форматов хранения данных;

2) экспериментальная оценка исследуемых форматов хранения данных;

3) анализ алгоритма обработки данных в Spark с использованием различных форматов.

Для проведения сравнительного анализа были выбраны следующие характеристики, описывающие особенности работы с каждым форматом хранения данных:

- платформозависимость;
- возможность изменения файла;
- возможность записи сложных структур;
- соблюдение требований ACID;
- тип формата (внутренняя структура файла);
- сжатие файла;
- наличие метаданных.

Результаты анализа представлены в табл. 1.

Вторым этапом исследования форматов хранения больших данных было проведение экспериментальной оценки этих форматов.

Экспериментальная оценка заключалась в симуляции обработки набора данных. Для проведения испытания был развернут экспериментальный стенд. В табл. 2 представлена конфигурация сформированного стенда.

Для проведения исследования был сгенерирован набор данных из 10 млн. записей. Описание сформированных данных представлено в табл. 3.

Таблица 1. Сравнительный анализ форматов хранения данных

Характеристика	**avro**	**CSV**	**JSON**	**ORC**	**parquet**
Платформонезависимость	да	да	да	нет	нет
Изменяемость	нет	да	да	нет	нет
Возможность записи сложных структур	да	нет	да	да	да
Соблюдение требований ACID	нет	нет	нет	да	нет
Тип формата	строчный	текстовый, строчный	текстовый, объектный	столбцовый	столбцовый
Сжатие	да	нет	нет	да	да
Наличие метаданных	нет	нет	нет	да	да

Таблица 2. Конфигурация экспериментального стенда

Элемент конфигурации	Характеристика
Процессор	Intel Core i7-8565U 1.8 GHz 4 ядра
Оперативная память	16 GB
Операционная система	Windows 10 64x
Платформа	Java Virtual Machine
Используемый язык программирования	Java v. 1.8
Используемый фреймворк	Apache Spark v. 2.4

Таблица 3. Описание сгенерированных данных

Название поля	Тип данных
name	строка
surname	строка
age	32-битное число
country	строка
balance	64-битное число
card number	строка
currency	строка
account open date	объект типа “Calendar”

На рис. 1 представлен пример записи в формате JSON, сформированной для проведенного эксперимента.


```
{
  "name": "Saffi",
  "surname": "Mruczek",
  "age": 34,
  "country": "Dominican Republic",
  "account_open_date": {
    "year": 2011,
    "month": 1,
    "dayOfMonth": 2,
    "hourOfDay": 18,
    "minute": 46,
    "second": 17
  },
  "balance": 943081,
  "card_number": "686888179934",
  "currency": "RUB"
}
```


Рис. 1. Пример сформированной записи

Рисунок 2 иллюстрирует схему проведения исследования. Файловая система хоста содержит файлы сгенерированного набора. На хосте установлена Java Virtual Machine, которая поддерживает работу исполнителя Spark-приложения (драйвера). При запуске драйвера формируется Spark Context, необходимый для дальнейшей работы приложения. После загрузки приложение поочередно считывает из файловой системы сгенерированные данные и замеряется время от начала чтения до окончания выполнения очередной операции. Поскольку особенностью Spark-приложения является наличие в нем ленивых вычислений [26], то есть любые трансформации не выполняются до тех пор, пока над этими трансформации не происходят действия, то моментом окончания операция считается считывание количества записей в результирующем наборе (операция count()), если иное не предусмотрено методом исследования. По окончании выполнения операции над всеми наборами данных приложение сохраняет результаты в текстовом файле.

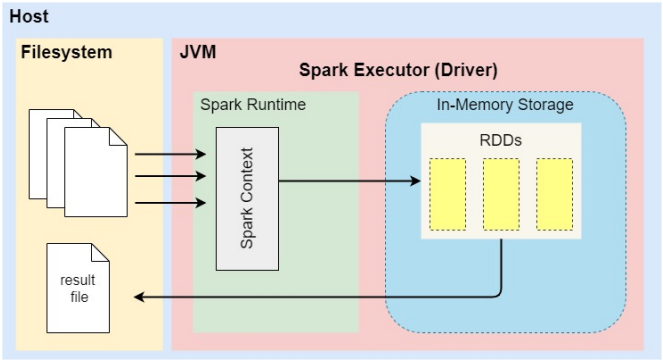

Рис. 2. Схема эксперимента

Для каждого формата данных было проведено исследование, состоящее из тестовых запусков Spark-приложения и выполнения одинакового набора операций. Алгоритмы обработки Spark исследованы в [27].

Чтение всех строк (C_1). Рассматривается время, затраченное на чтение данных в рассматриваемых форматах. Результаты, представленные на рис. 3, показали, что большинство форматов показало примерно одинаковый результат.

Фильтрация данных (C_2). Фильтрация данных является одной из наиболее часто используемых операций при обработке и анализе данных. Результаты проведенных исследований приведены на рис. 4. В этом случае JSON вновь показал наихудших результат. Следует отметить, что форматы AVRO и CSV отработали в два раза быстрее, а ORC и PARQUET более чем в 10 раз быстрее, что объясняется наличием метаданных, хранящих часть информации о содержащихся в файлах данных.

Поиск уникальных строк (C_3). Важной операцией при обработке и анализе данных является поиск уникальных объектов, известная на языке SQL как операция DISTINCT. В данном испытании, все форматы показали примерно равные результаты. Рис. 5 иллюстрирует результаты проведенного экспериментального оценивания.

Группировка (C_4). Группировка также является одной из наиболее часто используемых операций в анализе и обработке данных. Результаты оценки времени выполнения группировку по каждому формату представлено на рис. 6. Видно, что наилучший результат показали форматы ORC и PARQUET, что объясняется наличием метаданных по содержащимся в файлах данных. Наихудший результат показал формат JSON. Это объясняется тем, что приложению приходится каждый раз парсить очередной объект, содержащийся в файле.

Сортировка (C_5). Сортировка является наиболее сложной операцией и часто используемой операцией, поэтому результаты оценки временной эффективности имеют важное значение при обработке больших данных. На рис. 7 представлены оценки времени выполнения сортировки по строковому полю. Результаты показывают, что худшее время показал формат JSON, остальные форматы показали примерно одинаковое время.

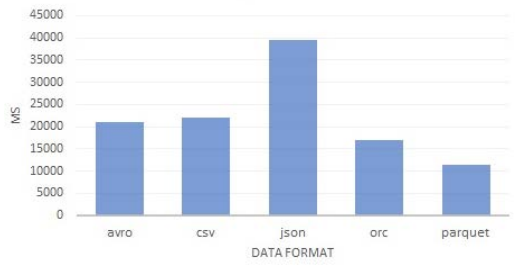

Рис 3. Время выполнения операции чтения всех строк, ms

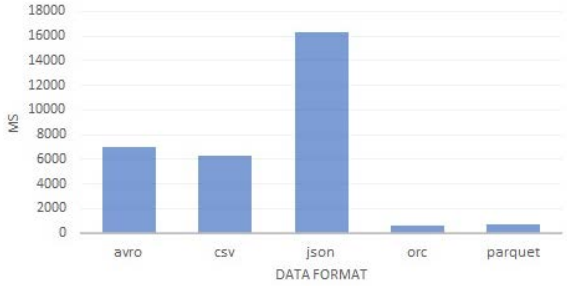

Рис. 4. Время выполнения операции фильтрации данных по двум параметрам, ms

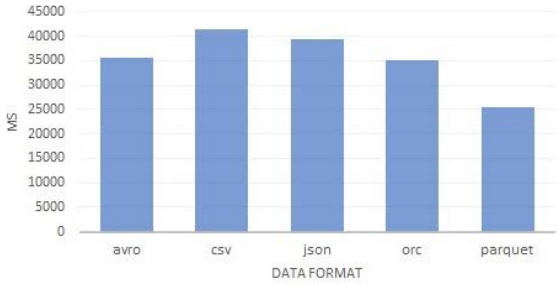

Рис. 5. Время выполнения операции поиска уникальных объектов, ms

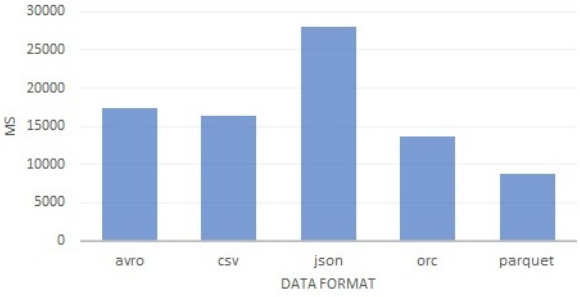

Рис. 6. Время выполнения операции сортировки, ms

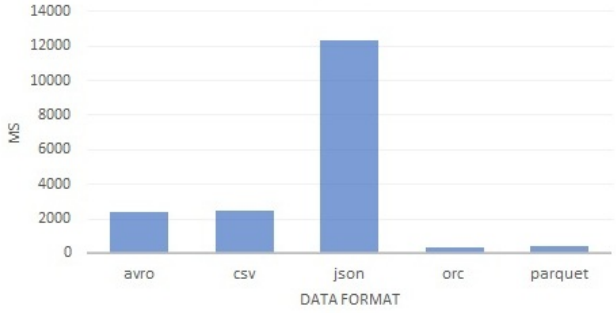

Рис 7. Время выполнения операции группировки, ms

Наихудший результат во всех тестах показал формат JSON. По результатам исследования следует вывод, что данный тип хорошо подходит для хранения одиночных объектов или данных, требующих периодического изменения. Также этот тип подходит для работы в платформонезависимых системах. Формат прост для чтения, отчего может быть использован специалистами, не связанными или опосредованно связанными с разработкой программного обеспечения.

Форматы AVRO и CSV показали средние результаты. Однако формат AVRO не пользуется такой популярностью в сфере работы с большими данными, как формат CSV. При близких результатах на испытаниях формат CSV, в отличии от формата AVRO, является более удобным в использовании, поскольку позволяет изменение и чтение содержащихся данных без использования специализированного программного обеспечения.

Наилучший результат показали два формата: ORC и PARQUET. Оба формата являются платформозависимыми – для работы с ними необходимо иметь кластер с развернутой на нем платформой Hadoop. Каждый из этих типов представляет из себя сжатый формат данных, содержащий в себе метаданные, хранящие статистики и информацию для быстрого доступа к данным. Однако оба формата уступают форматам типа JSON и CSV по возможности изменения. Формат ORC, в отличии от формата PARQUET, поддерживает технологию ACID за счет добавления файлов, содержащих информацию об обновлении или изменении данных, что влияет на его производительность в ряде испытаний.

III. Методика выбора эффективного решения

Алгоритм поиска оптимального решения состоит из двух этапов. На первом этапе проводятся парные сравнения каждого формата по критериям. Результатом первого этапа являются матрицы парных сравнений форматов по каждому критерию и матрица парного сравнения критериев. Второй этап представлен в виде решения задачи многокритериальной оптимизации с использованием метода анализа иерархий [23, 24].

Пусть $A_{(i)}$ – матрицы парных сравнений альтернатив по n критериям, а C – матрица парных сравнений самих критериев.

Проведем сравнение форматов хранения больших данных по каждой полученной оценке. Для этого введем несколько правил сравнения:

Поскольку результаты, полученные экспериментальным путем, обратно пропорциональны критериям успешности прохождения испытания, то степень предпочтения альтернативы i перед альтернативой j вычисляется по формуле

$$a_{ij} = a_i / a_j .$$

В экспериментальных результатах присутствует большой размах, способный повлиять на результат выбора. Если различие между максимальным и минимальным значением отличается более чем в 10 раз, то предлагается привести результаты к определенным диапазонам согласно следующему алгоритму:

- берем минимальное и максимальное значение из массива результатов по конкретному критерию;
- разделяем полученный диапазон на 5 отрезков;

– при попадании результата в диапазон с минимальными значениями присваиваем ему 1 балл, соответственно для формата с максимальным значением – 5 баллов.

По показателям, полученным экспериментальным путем, будем руководствоваться следующим правилам:

1. Чтение всех строк (C_1) является важным показателем, поскольку наиболее полно отражает скорость обработки данных с использованием конкретного формата хранения данных.

2. Фильтрация (C_2) и поиск уникальных значений (C_3) являются не менее важными характеристиками, однако данные функции опираются на вычитывание всех строк, важность которого определена в предыдущем пункте.

3. Группировка (C_4) являются следующими по важности показателями, поскольку интересна больше с точки зрения аналитики, чем программной инженерии.

4. Сортировка (C_5) является наименее важным из представленных критериев, поскольку чаще всего используется для наглядного представления данных.

Для оценки предпочтения того или иного показателя вводится следующая шкала:

– равно = 1;
– более (менее) важен = 2 (1/2);
– важнее (не важнее) = 4 (1/4).

Применяя во внимание все описанные выше правила, получаем матрицы парных сравнений по каждому критерию, приведенные в табл. 4.

Таблица 4. Матрицы парных сравнений для C_1, …, C_5

C_1	avro	csv	json	orc	parquet
avro	1	1	3/2	3/4	1/2
csv	1	1	3/2	3/4	1/2
json	2/3	2/3	1	1/2	1/3
orc	4/3	4/3	2	1	2/3
parquet	2	2	3	3/2	1

C_2	avro	csv	json	orc	parquet
avro	1	1	2	1/4	1/4
csv	1	1	1/2	1/4	1/4
json	1/2	1/2	1	1/4	1/4
orc	4	4	4	1	1
parquet	4	4	4	1	1

C_3	avro	csv	json	orc	parquet
avro	1	8/7	8/7	1	5/7
csv	7/8	1	1	7/8	5/8
json	7/8	1	1	7/8	5/8
orc	1	8/7	8/7	1	5/7
parquet	7/5	8/5	8/5	7/5	1

C_4	avro	csv	json	orc	parquet
avro	1	1/2	2	1/2	1/3
csv	2	1	3	4/5	3/5
json	1/2	1/3	1	1/4	1/5
orc	2	5/4	4	1	3/4
parquet	3	5/3	5	4/3	1

C_5	avro	csv	json	orc	parquet
avro	1	1	3/2	2/3	1/2
csv	1	1	3/2	2/3	1/2
json	2/3	2/3	1	1/2	1/3
orc	3/2	3/2	2	1	2/3
parquet	2	2	3	3/2	1

Матрица предпочтения критериев приведена в табл. 5.

Таблица 5. Матрица предпочтений

Критерии	C_1	C_2	C_3	C_4	C_5
C_1	1	1	1/2	1/2	2
C_2	1	1	1/2	1/2	2
C_3	2	2	1	1	4
C_4	2	2	1	1	4
C_5	1/2	1/2	1/4	1/4	1

В результате сравнения получены матрица предпочтения критериев и матрицы предпочтения форматов хранения больших данных по каждому критерию сравнения. Данные матрицы необходимы для дальнейшего вычисления вектора рейтинга альтернатив.

В качестве аппарата для выбора решения среди альтернатив был использован метод анализа иерархий Т.Л. Саати [23].

По приведенным матрицам приведем вычисление вектора рейтинга альтернатив.

1. По каждой матрице находим сумму элементов каждого столбца:

$$S_j = \sum_{i=1}^{n} a_{ij}.$$

2. Делим каждый элемент столбца на соответствующую сумму:

$$A_{ij} = \frac{a_{ij}}{S_j}.$$

3. Объединяем вектора весов, полученных из матриц по каждому критерию, в одну матрицу;

4. Умножаем полученную матрицу на столбец весов критериев.

Для рассматриваемых форматов получен следующий вектор:

$$w \approx (15 \;\; 16 \;\; 14 \;\; 26 \;\; 28)^{\mathrm{T}} \cdot 10^{-2}.$$

Согласно этому решению, рейтинг форматов выстраивается следующим образом:

$$parquet \cong orc \succ csv \cong avro \succ json$$

Как результат, получен рейтинг, показывающий приоритет каждого формата для решения конкретной задачи в рамках приоритетных правил критериев выбора.

IV. Заключение

В статье представлены методика выбора формата хранения данных на основе экспериментальной оценки временной эффективности операций обработки больших данных и метода анализа иерархий. В качестве примера были использованы форматы, поддерживаемые системой Apache Hadoop и фреймворком Apache Spark, как одним из самых популярных фреймворков для обработки больших данных.

Исследование можно использовать для построения систем обработки и хранения данных на основе платформы Apache Hadoop или аналогичных решений.

Полученное решение основано на результатах экспериментов в заданной инфраструктуре и не

отражает популярность или функциональность рассматриваемых форматов, а лишь отражает целесообразность использования форматов в рассматриваемых условиях.

Исследование состояло из двух частей: экспериментальных оценок и решение задачи выбора из заданных альтернатив. В экспериментальной части каждый формат оценивался на основе тестовых прогонов. Для эксперимента был развернут экспериментальный стенд с установленными на нем инструментами обработки больших данных. Целью было оценить временные характеристики операций обработки больших данных в разных форматах хранения с использованием фреймворка Apache Spark. Для выбора альтернатив использовался метода анализа иерархий, суть которого заключается в решении многокритериальной задачи принятия решения, результаты которой представлены в виде вектора степеней предпочтительности.

В статье рассмотрен пример присвоения рейтингов альтернативам. Алгоритм помогает найти эффективное решение для конкретных требований системы.

Разработанные решения могут быть полезны для практического использования. В любой компании при разработке или использовании программного обеспечения всегда есть выбор, какое решение из имеющихся использовать. Важным критерием выбора является обмен данными с другими компонентами, которые часто определяются форматами данных. В статье для рассмотренных вариантов использования больших данных был сделан выбор наилучшего решения. Однако методы, экспериментальная оценка, показатели качества, а также алгоритм оптимизации описаны достаточно подробно и могут использоваться для аналогичных задач, где важен выбор формата, а условия использования форматов и набор альтернатив может быть разным.

Следует заметить, что данное исследование не было направлено на изучение функциональных особенностей представленных форматов хранения данных. Основная цель исследования – сформировать методику выбора из представленных альтернатив на основе их экспериментальной оценки с использованием параметров временной эффективности, необходимых для решения конкретной задачи.

Кроме того, системы обработки больших данных представляют собой кластерные системы, которые позволяют обрабатывать больше данных, используя несколько узлов, объединенных в одной сети. В рамках данного исследования использовался один узел, результаты которого могут отличаться от кластеризации аналогичного набора данных.

Библиография

[1] D. Chong, H. Shi, “Big data analytics: A literature review,” J. Manag. Anal., vol. 2, p. 175–201, 2015.

[2] R. Moro Visconti, D. Morea, “Big Data for the Sustainability of Healthcare Project Financing,” Sustainability, vol. 11, p. 3748, 2019. doi:10.3390/su11133748.

[3] L. Ardito, V. Scuotto, M. Del Giudice, A. Messeni, “A bibliometric analysis of research on Big Data analytics for business and management,” Manag. Decis., vol. 57, p. 1993–2009, 2018. doi:10.1108/MD-07-2018-0754.

[4] F. Cappa, R. Oriani, E. Peruffo, I.P. McCarthy, “Big Data for Creating and Capturing Value in the Digitalized Environment: Unpacking the Effects of Volume, Variety and Veracity on Firm Performance,” Journal of Product Innovation Management, vol. 38, no. 1, p.. 49-67, 2021. https://doi.org/10.1111/jpim.12545.

[5] E. Nikulchev, D. Ilin, A. Silaeva, et al., “Digital Psychological Platform for Mass Web-Surveys,” Data, vol. 5, no. 4, p. 95. doi: 10.3390/data5040095

[6] I. Mavridis, H. Karatza, “Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark,” J. Syst. Softw., vol. 125, p. 133–151, 2017.

[7] S. Lee, J.Y. Jo, Y. Kim, “Survey of Data Locality in Apache Hadoop,” In 2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD), Honolulu, USA, 29-31 May 2019; pp. 46–53.

[8] K. Garg, D. Kaur, “Sentiment Analysis on Twitter Data using Apache Hadoop and Performance Evaluation on Hadoop MapReduce and Apache Spark,” In Proceedings on the International Conference on Artificial Intelligence (ICAI), Las Vegas, Nevada, USA, 29 July - 01 August 2019; pp. 233–238.

[9] Hive. 2020 Apache Hive Specification. Available online: https://cwiki.apache.org/confluence/display/HIVE.

[10] Impala. 2020 Apache Impala Specification. Available online: https://impala.apache.org/impala-docs.html.

[11] E. Nazari, M.H. Shahriari, H. Tabesh, “BigData Analysis in Healthcare: Apache Hadoop, Apache spark and Apache Flink,” Frontiers in Health Informatics, vol. 8, no. 1, p. 14, 2019.

[12] S. Salloum, R. Dautov, X. Chen, P.X. Peng, J.Z. Huang, ‘Big data analytics on Apache Spark,’ International Journal of Data Science and Analytics, vol. 1, no. 3, pp. 145-164, 2016.

[13] A. Gusev, D. Ilin, E. Nikulchev, “The Dataset of the Experimental Evaluation of Software Components for Application Design Selection Directed by the Artificial Bee Colony Algorithm,” Data, vol. 5, p. 59, 2020.

[14] A. Gusev, D. Ilin, P. Kolyasnikov, E. Nikulchev, “Effective Selection of Software Components Based on Experimental Evaluations of Quality of Operation,” Engineering Letters, vol. 28, no. 2, p. 420–427, 2020.

[15] A. Ramírez, J.A. Parejo, J.R. Romero, S. Segura, A. Ruiz-Cortés, “Evolutionary composition of QoS-aware web services: A many-objective perspective,” Expert Syst. Appl., vol. 72, p. 357–370, 2017,

[16] S. Gholamshahi, S.M.H. Hasheminejad, “Software component identification and selection: A research review,” Softw. Pract. Exp., vol. 49, p. 40–69, 2019.

[17] R.F. Munir, A. Abelló, O. Romero, M. Thiele, W. Lehner, “A cost-based storage format selector for materialized results in big data frameworks,” Distrib Parallel Databases, vol. 38, p. 335–364, 2020. doi:10.1007/s10619-019-07271-0.

[18] X. Wang, Z. Xie, “The Case For Alternative Web Archival Formats To Expedite The Data-To-Insight Cycle,”. In Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020, pp. 177-186, 2020.

[19] D. He, D. Wu, R. Huang, G. Marchionini, P. Hansen, S.J. Cunningham, ”ACM/IEEE Joint Conference on Digital Libraries 2020 in Wuhan virtually,” ACM Sigweb Newsl, vol. 1, p. 1–7, 2020.

[20] S. Ahmed, M.U. Ali, J. Ferzund, M.A. Sarwar, A. Rehman,; A. Mehmood, “Modern Data Formats for Big Bioinformatics Data Analytics,” Int. J. Adv. Comput. Sci. Appl., vol. 8, no. 4, p. 366-377, 2017, doi:10.14569/IJACSA.2017.080450.

[21] D. Plase, L. Niedrite, R. Taranovs, “A Comparison of HDFS Compact Data Formats: Avro Versus Parquet,” Moksl. Liet. Ateitis, vol. 9, p. 267–276, 2017.

[22] D. Ilin, E. Nikulchev, “Performance Analysis of Software with a Variant NoSQL Data Schemes,” In 2020 13th International Conference "Management of large-scale system development" (MLSD), p. 1-5, 2020. 10.1109/MLSD49919.2020.9247656

[23] Т. Л. Саати, “Об измерении неосязаемого. Подход к относительным измерениям на основе главного собственного вектора матрицы парных сравнений,” Cloud of science, vol. 2, no. 1, p. 5-39, 2015.

[24] Т. Л. Саати, “Относительное измерение и его обобщение в принятии решений. Почему парные сравнения являются ключевыми в математике для измерения неосязаемых факторов,” Cloud of science, vol. 3, no. 2, p. 171-262, 2016.

[25] S. Sakr, A. Liu, A.G. Fayoumi, “The family of mapreduce and large-scale data processing systems,” ACM Comput. Surv. (CSUR), vol. 46, p. 1–44, 2013.

[26] S. Chellappan, D. Ganesan, “Introduction to Apache Spark and Spark Core,” In Practical Apache Spark; Apress: Berkeley, CA, USA; pp. 79–113, 2018.

[27] V. Belov, A. Tatarintsev, E. Nikulchev, “Choosing a Data Storage Format in the Apache Hadoop System Based on Experimental

Evaluation Using Apache Spark," Symmetry, vol. 13, no. 2, p. 195, 2021. doi: 10.3390/sym13020195

Владимир Александрович Белов, аспирант кафедры Интеллектуальных систем информационной безопасности, МИРЭА – Российский технологический университет; главный специалист Дата-Центра, Российская академия образования.
e-mail: belov_v.a@mail.ru
scopus ID: authorId=57205752270
ORCID: 0000-0001-8769-2529
ResearcherID: AAT-3723-2021

Евгений Витальевич Никульчев, д.т.н, профессор, профессор кафедры Интеллектуальных систем информационной безопасности, МИРЭА – Российский технологический университет; профессор РАО, ведущий аналитик Дата-Центра, Российская академия образования, член международной ассоциации инженеров (International Association of Engineers, IAENG).
e-mail:nikulchev@mail.ru
e-library: authorid=396636
scopus ID:authorId=6504081534
ORCID: 0000-0003-1254-9132
ResearcherID: G-6557-2015

Experimental evaluation of the temporal efficiency of big data processing for specified storage formats

V.A. Belov, E.V. Nikulchev

***Abstract* — One of the most important tasks of a modern big data processing platform is the task of choosing data storage formats. The choice of formats is based on various performance criteria, which depend on the class of objects and the requirements. One of the most important criteria is the time spent in various big data processing operations. The paper studies the five most popular formats for storing big data (avro, CSV, JSON, ORC, parquet), proposes an experimental bench for assessing time efficiency, and conducts a comparative analysis of experimental estimates of the characteristics of the formats under consideration. For the experiment, the basic data processing operations were considered using the Apache Spark framework. The format selection algorithm is developed based on the hierarchy analysis method. As a result, a methodology was formed for choosing a format from alternatives based on experimental estimates of parameters and a methodology for analyzing hierarchies for the task of choosing time-efficient basic operations of storage formats for big data in the Apache Hadoop system using Apache Spark.**

***Key words*—bigdata, data storage formats, resource efficiency, Apache Hadoop, Apache Spark.**

REFERENCES

[1] D. Chong, H. Shi, “Big data analytics: A literature review,” J. Manag. Anal., vol. 2, p. 175–201, 2015.

[2] R. Moro Visconti, D. Morea, “Big Data for the Sustainability of Healthcare Project Financing,” Sustainability, vol. 11, p. 3748, 2019. doi:10.3390/su11133748.

[3] L. Ardito, V. Scuotto, M. Del Giudice, A. Messeni, “A bibliometric analysis of research on Big Data analytics for business and management,” Manag. Decis., vol. 57, p. 1993–2009, 2018. doi:10.1108/MD-07-2018-0754.

[4] F. Cappa, R. Oriani, E. Peruffo, I.P. McCarthy, “Big Data for Creating and Capturing Value in the Digitalized Environment: Unpacking the Effects of Volume, Variety and Veracity on Firm Performance,” Journal of Product Innovation Management, vol. 38, no. 1, p.. 49-67, 2021. https://doi.org/10.1111/jpim.12545.

[5] E. Nikulchev, D. Ilin, A. Silaeva, et al., “Digital Psychological Platform for Mass Web-Surveys,” Data, vol. 5, no. 4, p. 95. doi: 10.3390/data5040095

[6] I. Mavridis, H. Karatza, “Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark,” J. Syst. Softw., vol. 125, p. 133–151, 2017.

[7] S. Lee, J.Y. Jo, Y. Kim, “Survey of Data Locality in Apache Hadoop,” In 2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD), Honolulu, USA, 29-31 May 2019; pp. 46–53.

[8] K. Garg, D. Kaur, “Sentiment Analysis on Twitter Data using Apache Hadoop and Performance Evaluation on Hadoop MapReduce and Apache Spark,” In Proceedings on the International Conference on Artificial Intelligence (ICAI), Las Vegas, Nevada, USA, 29 July - 01 August 2019; pp. 233–238.

[9] Hive. 2020 Apache Hive Specification. Available online: https://cwiki.apache.org/confluence/display/HIVE.

[10] Impala. 2020 Apache Impala Specification. Available online: https://impala.apache.org/impala-docs.html.

[11] E. Nazari, M.H. Shahriari, H. Tabesh, “BigData Analysis in Healthcare: Apache Hadoop, Apache spark and Apache Flink,” Frontiers in Health Informatics, vol. 8, no. 1, p. 14, 2019.

[12] S. Salloum, R. Dautov, X. Chen, P.X. Peng, J.Z. Huang, ‘Big data analytics on Apache Spark,’ International Journal of Data Science and Analytics, vol. 1, no. 3, pp. 145-164, 2016.

[13] A. Gusev, D. Ilin, E. Nikulchev, “The Dataset of the Experimental Evaluation of Software Components for Application Design Selection Directed by the Artificial Bee Colony Algorithm,” Data, vol. 5, p. 59, 2020.

[14] A. Gusev, D. Ilin, P. Kolyasnikov, E. Nikulchev, “Effective Selection of Software Components Based on Experimental Evaluations of Quality of Operation,” Engineering Letters, vol. 28, no. 2, p. 420–427, 2020.

[15] A. Ramírez, J.A. Parejo, J.R. Romero, S. Segura, A. Ruiz-Cortés, “Evolutionary composition of QoS-aware web services: A many-objective perspective,” Expert Syst. Appl., vol. 72, p. 357–370, 2017,

[16] S. Gholamshahi, S.M.H. Hasheminejad, “Software component identification and selection: A research review,” Softw. Pract. Exp., vol. 49, p. 40–69, 2019.

[17] R.F. Munir, A. Abelló, O. Romero, M. Thiele, W. Lehner, “A cost-based storage format selector for materialized results in big data frameworks,” Distrib Parallel Databases, vol. 38, p. 335–364, 2020. doi:10.1007/s10619-019-07271-0.

[18] X. Wang, Z. Xie, “The Case For Alternative Web Archival Formats To Expedite The Data-To-Insight Cycle,”. In Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020, pp. 177-186, 2020.

[19] D. He, D. Wu, R. Huang, G. Marchionini, P. Hansen, S.J. Cunningham, ”ACM/IEEE Joint Conference on Digital Libraries 2020 in Wuhan virtually,” ACM Sigweb Newsl, vol. 1, p. 1–7, 2020.

[20] S. Ahmed, M.U. Ali, J. Ferzund, M.A. Sarwar, A. Rehman,; A. Mehmood, “Modern Data Formats for Big Bioinformatics Data Analytics,” Int. J. Adv. Comput. Sci. Appl., vol. 8, no. 4, p. 366-377, 2017, doi:10.14569/IJACSA.2017.080450.

[21] D. Plase, L. Niedrite, R. Taranovs, “A Comparison of HDFS Compact Data Formats: Avro Versus Parquet,” Moksl. Liet. Ateitis, vol. 9, p. 267–276, 2017.

[22] D. Ilin, E. Nikulchev, “Performance Analysis of Software with a Variant NoSQL Data Schemes,” In 2020 13th International Conference "Management of large-scale system development" (MLSD), p. 1-5, 2020. 10.1109/MLSD49919.2020.9247656

[23] T. L. Saaty, “Ob izmerenii neosyazaemogo. Podhod k otnositel'nym izmereniyam na osnove glavnogo sobstvennogo vektora matricy parnyh sravnenij,” Cloud of science, vol. 2, no. 1, p. 5-39, 2015.

[24] T. L. Saaty, “Otnositel'noe izmerenie i ego obobshchenie v prinyatii reshenij. Pochemu parnye sravneniya yavlyayutsya klyuchevymi v matematike dlya izmereniya neosyazaemyh faktorov,”Cloud of science, vol. 3, no. 2, p. 171-262, 2016.

[25] S. Sakr, A. Liu, A.G. Fayoumi, “The family of mapreduce and large-scale data processing systems,” ACM Comput. Surv. (CSUR), vol. 46, p. 1–44, 2013.

[26] S. Chellappan, D. Ganesan, “Introduction to Apache Spark and Spark Core,” In Practical Apache Spark; Apress: Berkeley, CA, USA; pp. 79–113, 2018.

[27] V. Belov, A. Tatarintsev, E. Nikulchev, “Choosing a Data Storage Format in the Apache Hadoop System Based on Experimental Evaluation Using Apache Spark,” Symmetry, vol. 13, no. 2, p. 195, 2021. doi: 10.3390/sym13020195

Vladimir Belov, graduate student, Department of Intelligent Information Security Systems, RTU MIREA; Data Center of Russian Academy of Education.
e-mail: belov_v.a@mail.ru
scopus ID: authorId=57205752270
ORCID: 0000-0001-8769-2529!
ResearcherID: AAT-3723-2021

Evgeny Nikulchev, Dr. Sci., professor, RTU MIREA; Data Center of Russian Academy of Education, member of International Association of Engineers.
e-mail:nikulchev@mail.ru
e-library: authorid=396636
scopus ID:authorId=6504081534
ORCID: 0000-0003-1254-9132
ResearcherID: G-6557-2015