ИНСТРУМЕНТЫ АНАЛИЗА ФОРМАТОВ ХРАНЕНИЯ БОЛЬШИХ ДАННЫХ ДЛЯ ПОСТРОЕНИЯ ОЗЕР ДАННЫХ

Белов В.А., Никульчев Е.В.

МИРЭА – Российский технологический университет, Россия, г. Москва, пр. Вернадского, д. 78

nikulchev@mail.ru, belov_v.a@mail.ru

Аннотация: Одним из современных средств построения аналитических платформы является концепция озер данных. Большинство реализаций данной концепции базируются на платформах, не имеющих установленного формата хранения, что приводит к необходимости решения задач выбора формата данных при проектировании озера данных. Проведен анализ основных форматов хранения больших данных в озерах данных. Представлены отличительные характеристики каждого формата, включая особенности внутренней структуру файлов, поддерживаемые типы данных, рекомендации по применению.

Ключевые слова: большие данные, озеро данных, форматы хранения данных.

Введение

Одна из важнейших задач любой системы обработки данных — это проблема хранения полученных данных. В традиционных подходах наиболее популярными инструментами для хранения данных было использование реляционных баз данных [1].

Рост объема данных и потребности потребителей систем обработки данных привели к появлению концепции больших данных [2]. Концепция больших данных основана на шести аспектах, таких как ценность, объем, скорость, разнообразие, достоверность и изменчивость [3]. Это означает, что под термином «большие данные» понимается не только объем этих данных, но и их способность выступать в качестве источника для генерации ценной информации и идей [3].

Новые концепции заменили традиционные формы хранения данных, среди которых стали популярными NoSQL решения [4] и так называемые озера данных [5]. Озеро данных — это масштабируемая система для хранения и анализа данных, хранимых в их оригинальном формате и используемых для извлечения знаний [6]. Озеро данных может быть спроектировано с нуля или разработано на основе существующих программных решений [7]. Многие реализации озер данных используют Apache Hadoop в качестве базовой платформы [5].

Для создания озер данных на основе экосистемы Apache Hadoop в качестве базовой файловой системы используется HDFS [8]. Эта файловая система дешевле в использовании, чем коммерческие базы данных. При использовании такого хранилища данных очень важно выбрать правильный формат файла хранения данных. Формат файла определяет, как информация будет храниться в HDFS. Необходимо учитывать, что Apache Hadoop и HDFS не имеют форматов файлов по умолчанию. Это определило появление и использование различных форматов хранения данных в HDFS.

Среди наиболее широко известных форматов, используемых в системе Hadoop, являются JSON [9], CSV [9], Apache Parquet [10], Apache Avro [11], Apache ORC [12]. Каждый из этих форматов файлов имеет свои особенности в файловой структуре. Кроме того, различия наблюдаются на уровне практического применения. Так, строковые форматы обеспечивают высокую скорость записи, но колоночные форматы лучше подходят для чтения данных.

Серьезной проблемой в производительности платформ для хранения и обработки данных является время поиска и записи информации, а также объем занятых данных. Управление обработкой и хранением больших объемов информации является сложным процессом.

В связи с этим при построении систем хранения больших данных возникает проблема выбора того или иного формата хранения данных. Для решения этой проблемы необходимо исходить из результатов оценки по нескольким критериям.

Цель данной статьи - описать основные форматы хранения больших данных, используемых при построении озер данных на базе Apache Hadoop, их особенности и возможности в приложении к таким задачам, как аналитика, потоковая загрузка, загрузка пачками данных и т. д. В статье описаны как известные, так и широко используемые форматы хранения больших данных, а также новые тенденции, которые сейчас набирают популярность.

1 Сравнительная характеристика наиболее популярных форматов хранения данных

Наиболее популярными форматами хранения данных в распределенной файловой системе HDFS являются JSON [9], CSV [9], Apache Parquet [10], Apache Avro [11], Apache ORC [12]. Рассмотрим их основные особенности, структуру хранения информации,

Apache Avro [11] является линейным форматом хранения данных, широко используемый для сериализации, что определяет его популярность в потоковых системах. Структура файла состоит из заголовка и блоков данных. В заголовке содержатся метаданные файлы, представленные в виде схемы данных и случайного 16-битного числа, маркирующего файл. Схема данных представлена в формате JSON, что облегчает чтение и интерпретацию данных программами.

Avro поддерживает эволюцию схемы, то есть возможность пропуска, добавления или изменения отдельных полей. Avro не является строго типизированным форматом: тип каждого поля хранится в разделе метаданных вместе со схемой. Это означает, что для чтения информации не требуется никаких предварительных знаний о схеме.

Apache Avro поддерживает следующие типы данных:

- примитивные (null, Boolean, int, long, float, double, string, bytes, fixed);
- сложные (union, record, enum, array, map);
- логические (decimal, date, time, timestamp, uuid).

JSON (JavaScript Object Notation) [9] является текстовым форматом. Структура файла представлена в виде пар ключ-значение, описывающих поля некоторого объекта. Формат часто используется в передаче данных посредствам сети, особенно в веб-сервисах на основе REST API. В последние годы JSON набрал популярность в документных NoSQL базах данных [13], таких как MongoDB [14]. Многие пакеты данных поддерживают сериализацию и десериализацию JSON, что определяет его популярность в потоковых системах.

JSON поддерживает следующие типы данных:

- примитивные (null, boolean, number, string);
- сложные (array, object).

CSV (comma-separated values) [9] является текстовым форматом, чаще используемый для хранения и передачи табличных данных между системами, использующих простой текст. Данные в файле представлены в виде строк, разделенных каким-либо символом (чаще всего запятой). В файле может находиться также заголовок, содержащий названия колонок. Несмотря на ограничения, CSV является популярным выбором для обмена данными, поскольку поддерживает широкий спектр деловых, потребительских и научных приложений [9].

Apache Parquet [10] является бинарным колоночно-ориентированным форматом хранения данных. Архитектура формата основана на «уровнях определения» (definition levels) и «уровнях повторения» (repetition levels). Важной частью данного формата является наличие метаданных, хранящих основную информацию о данных в файле, что способствует более быстрой фильтрации и агрегации данных в задачах анализа.

Структура файла представлена несколькими уровнями разбиения:

- row group – построчное разбиение данных на строки для более быстрого чтения при параллельной работе с использованием алгоритма MapReduce.
- column chunk – блок данных для колонки в группе строк. Данное разбиение предназначено для ускорения работы с жестким диском – в этом случае данные записываются не по строкам, а по колонкам;
- page – представляет из себя концептуально неделимую единицу, содержащую метаинформацию и закодированные данные.

Apache Parquet поддерживает следующие типы данных:

- примитивные (int32, int64, int96, float, double);
- сложные (byte array);
- логические (boolean).

ORC (Optimized Row Columnar) [12] – это колоночно-ориентированный формат хранения данных. Данный формат оптимизирован для чтения потоков больших данных.

Архитектурно данный формат схож форматом Apache Parquet. Структура формата разделена на метаданные и сами данные. Метаданные хранят статистическую и описательную информацию, индексы, информацию о разбиении данных. Сами данные разбиты на так называемые полосы (stripes). Каждая полоса является атомарной единицей для распределенной работы с данными. ORC поддерживает полный набор типов, включая сложные (структуры, списки, карты и объединения).

2 Особенности применения форматов хранения больших данных

Условно описанные форматы хранения данных можно разделит на группы, содержащие альтернативные форматы, в зависимости от задач, возложенных на данные форматы при их использовании в системах обработки больших данных.

По доступности к данным описанные форматы можно разделить на следующие группы:

- изменяемые (JSON, CSV);
- неизменяемые (Parquet, Avro, ORC).

По внутренней структуре файла выделяются следующие группы:

- текстовые (JSON, CSV);
- колоночные (Parquet, ORC);
- линейные (Avro).

Поскольку колоночные форматы обеспечивают быстрое чтение данных, то форматы можно разделить на две группы:

- форматы для потоковых данных (JSON, CSV, Avro);
- форматы для аналитических платформ (Parquet, ORC).

В [15-18] приводятся исследования, направленные на изучение альтернатив форматов хранения данных. В [15] сравниваются форматы Apache Parquet и Apache Avro с точки зрения производительности, однако в данном исследовании нет обоснования выбора именно такой альтернативы. Исследование исходит из экспериментальной оценки двух форматов при отсутствии конкретной задачи выбора альтернатив. Авторы [16] преследуют цель поиска альтернативы для формата WARC при разработке веб сервисов. В качестве альтернативы в этом исследовании выступают также Apache Parquet и Apache Avro. [17] предлагает обширное исследования различных форматов хранения данных для задачи аналитики в сфере биоинформатики. В данной статье представлена оценка всех описанных здесь форматов. Наиболее подходящим форматом выбраны Apache Parquet и ORC. Авторы также дают рекомендации по использованию того или иного формата. Конкретно, при выполнении нескольких запросов, рекомендуется использовать Apache Parquet, в то время как ORC не следует использовать [17]. Исследование [18] направлено на оценку форматов Avro и Parquet при выполнении запросов к данным. Результатами исследования являются рекомендации по использованию каждого формата для специфических задач.

При анализе работ по оценке форматов хранения данных выяснилось, что подобные исследования не рассматривают характеристические особенности формата, важные для его применения к тем или иным задачам.

3 Разработка инструментов анализа форматов хранения больших данных

Разработана методика [19] для анализа форматов хранения данных, основанная на сравнительном анализе, экспериментальной оценке и математической модели выбора альтернативы. Для экспериментальной оценки использовался фреймворк Apache Spark [20], являющийся одним из наиболее популярных инструментов для проведения анализа данных в системе Apache Hadoop [9].

Для экспериментальной оценки был подготовлен стенд и сгенерированы тестовые наборы данных, представляющие из себя одинаковую информацию, сохраненную в разных форматах. Были проведены экспериментальные запуски фреймворка, выполняющего разные операции над данными, с замером времени обработки каждого формата хранения данных.

Для сравнительной характеристики была разработана методика проставления оценки каждого формата согласно приведенной характеристики. В качестве математического аппарата для решения задачи многокритериальной оптимизации был выбран аппарат идемпотентной математики [21].

Заключение

Проведен анализ основных форматов хранения больших данных в озерах данных. Представлены отличительные характеристики каждого формата, включая особенности внутренней структуру файлов, поддерживаемые типы данных, рекомендации по применению. Выделены преимущества и недостатки каждого формата.

Проведен сравнительный анализ наиболее популярных форматов хранения больших данных. Выделены условные альтернативные группы форматов по практическому применению. Проанализированы исследования, направленные на выявление эффективности того или иного формата хранения данных применительно к задаче.

Приведен обзор исследования, направленного на разработку методики анализа форматов хранения данных. Методика основана на сравнительной характеристике, экспериментальной оценке и математической модели выбора среди альтернатив.

Литература

1. *Alasta, A.F., Enaba, M.A.* Data warehouse on Manpower Employment for Decision Support System // Int'l Journal of Computing, Communications & Instrumentation Engg. Vol. 1, 2014, №1 – P.48-53.
2. *Chong, D., Shi, H.* Big data analytics: A literature review // Journal of Management Analytics. Vol. 2. 2015, - P.175–201.
3. *Cappa, F., Oriani, R., Peruffo, E., McCarthy, I.P.* 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 2020, №1. - P.49-67.
4. *Moniruzzaman, A. B. M., Hossain, S. A.* NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison // International Journal of Database Theory and Application. Vol. 6. 2013, № 4. – P.1–13.
5. *Darmont, J., Favre, C., Loudcher, S., Nous, C.* Data Lakes for Digital Humanities // 2nd International Digital Tools & Uses Congress (DTUC 2020), Oct 2020, Hammamet, Tunisia. 2020, - P.38-41.
6. *Sawadogo, P.-N., Scholly, E., Favre, C., Ferey, E., Loudcher, S., Darmont, J.* Metadata Systems for Data Lakes: Models and Features // 1st International Workshop on BI and Big Data Applications, Vol. 1064. 2019, - P. 440–451.
7. Khine, P.P., Wang, Z.S. Data lake: a new ideology in big data era // ITM Web of Conferences 17:03025, Vol. 03025. 2018.
8. Apache. Hadoop official documentation 2020. Available online: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
9. *Daquino, M., Heibi, I., Peroni, S., Shotton, D.* Creating RESTful APIs over SPARQL endpoints using RAMOSE // International Journal. Vol.5. 2020. – P. 135-149
10. Apache. Parquet official documentation 2018. Available online: https://parquet.apache.org/documentation/latest/ (accessed on 11 January 2021).
11. Apache. Avro specification 2012. Available online: http://avro.apache.org/docs/current/spec.html (accessed on 11 January 2021).
12. ORC. ORC Specification 2020. Available online: https://orc.apache.org/specification/ORCv1/ (accessed on 11 January 2021)
13. *Truica, C.-O., Apostol, E.-S., Darmont, J., Pedersen, T.B.* The Forgotten Document-Oriented Database Management Systems: An Overview and Benchmark of Native XML DODBMSes in Comparison with JSON DODBMSes // Big Data Research. Vol. 25, 2021.
14. MongoDB, "Mongodb," http://www.mongodb.com/
15. *Munir, R.F.; Abelló, A.; Romero, O.; Thiele, M.; Lehner, W.* A cost-based storage format selector for materialized results in big data frameworks. *Distrib Parallel Databases* **2020**, *38*, 335–364, doi:10.1007/s10619-019-07271-0.
16. *Wang, X., Xie, Z.* The Case for Alternative Web Archival Formats to Expedite The Data-To-Insight Cycle // JCDL '20: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020. – P.177-186.
17. *Ahmed, S., Ali, M.U., Ferzund, J., Sarwar, M.A., Rehman, A., Mehmood, A.* Modern Data Formats for Big Bioinformatics Data Analytics // International Journal of Advanced Computer Science and Applications. Vol. 8. 2017, № 4. - P. 366-377.
18. *Plase, D., Niedrite, L., Taranovs, R.* A Comparison of HDFS Compact Data Formats: Avro Versus Parquet // Lietuvos ateitis. Vol. 9. 2017, № 3. – P.267-276.
19. *Belov, V., Tatarintsev, A., Nikulchev, E.* Choosing a Data Storage Format in the Apache Hadoop System Based on Experimental Evaluation Using Apache Spark // Symmetry. Vol. 13, 2021, № 2. – P. 95
20. *Chellappan, S., Ganesan, D.* Introduction to Apache Spark and Spark Core. In Practical Apache Spark // Apress. 2018, - P.79–113.
21. *Krivulin, N., Sergeev, S.* Tropical optimization techniques in multi-criteria decision making with Analytical Hierarchy Process // Proceedings of the 2017 European Modelling Symposium (EMS), Manchester, UK, 20–21 November 2017, - P.38–43.