АНАЛИЗ ЭФФЕКТИВНОСТИ МЕТОДОВ ОБРАБОТКИ БОЛЬШИХ МАССИВОВ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

И. А. Рыговский

Сибирский государственный университет телекоммуникаций и информатики,
630102, Новосибирск, Россия

УДК 519.25

Производится обзор проблемы обработки больших массивов данных методами интеллектуального анализа данных, и дается сравнительный анализ методов для решения различных типов задач на вычислительных системах с различной архитектурой. Основным критерием для сравнения выбрана эффективность решения поставленной задачи на сверхбольших массивах данных с использованием в перспективе вычислительных систем экзофлопсной производительности.

Ключевые слова: анализ данных, вычислительные системы, параллельные алгоритмы, распределенные файловые системы, большие массивы данных.

Overview of the problem of processing large data sets of data mining methods and provides a comparative analysis of methods for solving different types of tasks on computing systems with different architectures. The main criterion for comparing was chosen the effeciency of solution of the task on very large scale data sets using computing systems exaFLOPS performance in perspective.

Key words: data mining, computer systems, parallel algorithms.

Введение. В настоящее время непрерывно растут объемы данных, которые накапливаются различными организациями в процессе работы, и увеличивается число задач, которые требуют обработки больших массивов данных. Соответственно, также увеличиваются требования к вычислительным ресурсам. Управление бизнес-процессами в различных сферах бизнеса, в том числе электронного, немыслимо без процессов накопления, анализа, выявления определенных закономерностей и зависимостей в этом огромном количестве данных.

Существующие темпы роста данных привели к развитию таких подходов, как Data Mining [1] (также называемая Knowledge Discovery In Data — обнаружение знаний в данных или интеллектуальный анализ данных (далее ИАД)) и Big Data (серия подходов для анализа сверхбольших массивов данных на вычислительных системах [2]).

Для работы с большими массивами данных необходимо применять средства, которые можно использовать на вычислительных системах при помощи методов параллельного программирования. Научные школы и крупные компании в последние годы активно реагируют на эту проблему и создают продукты, потребность в которых быстро растет не только у крупных организаций, но в том числе и у среднего и малого бизнеса. Наиболее востребованными являются задачи анализа научных данных [3]. Распространение „облачных“ вычислений дало еще больший толчок этому направлению, поскольку позволило небольшим компаниям использовать ресурсы, которые им ранее не были доступны.

Задачей современных исследователей являются разработка, внедрение и применение методов Data Mining в науке, бизнесе, производстве и других областях жизни, которые активно развиваются и требуют необходимые инструменты для решения поставленных задач.

Наращивание мощностей вычислительных систем позволяет решить только часть проблем, возникающих при обработке данных. Часть задач позволяет естественно распараллелить вычисления без особого ущерба для производительности, другая (в частности, задачи ИАД) — требует адаптации алгоритмов под архитектуру вычислительной системы.

Эффективность масштабирования алгоритмов. Изучив различные задачи, которые ставятся перед исследователями, можно разделить их на крупные классы, где критерием различия будет являться эффективность масштабирования алгоритмов при распараллеливании. На эффективность масштабирования, как правило, влияет несколько параметров в совокупности [4]:

– доля параллельных вычислений в общем объеме вычислений — как правило, достигается параллелизмом по данным, когда независимые друг от друга данные могут обрабатываться на различных вычислительных узлах;

– число обменов между ветками — минимальное число обменов между вычислительными узлами, позволяет уменьшить задержку между этапами вычислений. Часть задач по сбору статистической информации в постоянном потоке данных может вообще не требовать обменов между вычислителями;

– сложность вычислений на каждой ветке — увеличение сложности позволяет увеличить долю параллельных вычислений в алгоритме, это косвенно влияет на масштабируемость алгоритма, как правило, на небольших выборках;

– число потребляемой памяти для вычислений — уменьшение потребляемой памяти дает бóльшую свободу в выборе архитектуры вычислительной системы, позволяя использовать общую оперативную память и кэш;

– частота прерываний и возобновлений вычислений — позволяет использовать крупномасштабные вычислительные системы, в которых всегда будут присутствовать отказы вычислительных узлов. Важно не терять вычисления, произведенные до отказа узла.

Часть задач анализа данных может естественным образом распараллеливаться за счет отсутствия каких-либо обменов между узлами вычислительной системы. Примером таких алгоритмов могут служить методы Монте-Карло [5], методы k-средних (k-means [6], fuzzy c-means, Густафсон-Кесселя), EM алгоритмы, алгоритмы, основанные на использовании нейронных сетей. В данном случае присутствуют параллельность по данным, вычисление независимых друг от друга задач, либо вычисление глобальной функции, минимизирующей число обменов сообщениями между отдельными ветками программы.

Более сложные алгоритмы, требующие анализа всего массива данных, постоянного общения между ветками, не позволяют без потерь во время решения задачи увеличивать количество данных и вычислителей. Примером таких задач могут служить иерархические алгоритмы, требующие на каждом этапе вычисления обмена между ветками алгоритма [7]. Значительное влияние также оказывает сложность вычислений на каждом этапе; естественно, при увеличении сложности увеличивается доля параллельных вычислений в программе. Как правило, при увеличении сложности вычисления метрик в анализе данных увеличивается и число данных, участвующих в вычислениях, и не удается без потерь качества решения задачи уменьшить объем передачи сообщений между вычислительными узлами [8].

Архитектура вычислительной системы для задач анализа данных. Существуют различные модели программирования, организации файловой системы вычислительных систем. Выбор правильной модели программирования и архитектуры напрямую зависит от используемых алгоритмов и наборов данных.

Для задач с большим числом обменов сообщениями между ветками параллельной программы наиболее эффективным будет использование систем с общей памятью. Общая память максимально уменьшает задержки между узлами, но накладывает ограничения на объеме обрабатываемого массива данных величиной памяти, задействованной в вычислениях в один момент времени. Для обмена сообщениями для таких систем существуют несколько различных моделей программирования, которые используют возможности мультипроцессоров OpenMP [9], Threads. Наиболее эффективно использовать такой подход в крупномасштабных вычислительных системах можно при использовании гибридных узлов с общей и распределенной памятью при помощи коммуникационной сети.

Системы с распределенной памятью также могут эффективно использовать на вычислительных узлах возможности оперативной памяти, при этом необходимо минимизировать обмен сообщениями между узлами. Также необходимо контролировать объемы данных на узлах, отказоустойчивость таких вычислений, так как при сбое данные, не выгружаемые в постоянное хранилище данных, будут потеряны. Стандарт MPI является наиболее распространенным для организации обменов сообщениями между узлами вычислительной системы [10].

Увеличивая объемы блоков данных для обработки одним вычислительным узлом, можно снизить относительную долю затрат на коммуникацию между узлами. Однако объем обрабатываемых данных или сложность вычисляемых метрик могут накладывать ограничения и требовать постоянного хранения всех промежуточных результатов в файловой системе даже при более низкой скорости доступа к данным [11]. В таких случаях все данные могут храниться в распределенной файловой системе (Distributed File System). При этом максимальная эффективность будет достигнута при минимальном количестве операций ввода-вывода. Аналогичным образом можно рассматривать работу распределенных реляционных баз данных, которые используют файловую систему максимально эффективно на любых объемах данных благодаря построению индексов, способу организации данных. Но это налагает значительные ограничения на настройку движения данных между узлами вычислительной системы. Если входные данные неструктурированы, использование реляционных баз данных бессмысленно, так как скорость доступа к данным будет значительно ниже, но это позволяет тратить минимальные ресурсы для подготовки данных к анализу и работать с непрерывным потоком данных.

Существующие решения. Существует несколько крупных направлений, которые занимаются решением задачи анализа больших массивов данных. Стандартные методы ИАД в коммерческих базах данных полезны, но „заточены“ на ограниченный круг задач: они соответствуют лишь малому числу из множества статистических библиотек, поставляемых в составе статистических пакетов, таких как R [12], SAS или Matlab. Кроме того, они обычно реализуются в виде „черного ящика“ — код компилируется в плагин сервера баз данных.

В отличие от этого, статистические пакеты типа R или Matlab являются гибкими средами программирования, в которых библиотечные подпрограммы могут расширяться и модифицироваться аналитиками. Такие решения частично предлагают некоторые компании в своей идеологии, где требуется, чтобы аналогичные возможности программирования

на основе расширяемых средств SQL и/или MapReduce [13] были привнесены в сценарии использования крупных данных. В этом контексте иногда могут быть полезны и процедуры ИАД в виде черных ящиков, но только в небольшом числе случаев.

В литературе описаны и другие попытки выполнения значительных научных вычислений на языке SQL, были созданы новые исследовательские системы управления научными данными [14].

Исходя из этого, возможно, один из лучших подходов может состоять в тесной интеграции статистических пакетов с массивной параллельной базой данных. К сожалению, для многих имеющихся в настоящее время статистических пакетов отсутствуют параллельные реализации какого-либо вида. Параллелизованные статистические библиотеки (например, посредством ScaLAPACK [15]) основываются на использовании протоколов обмена сообщениями между процессорами (на базе MPI) и не интегрируются естественным образом с параллелизмом по данным популярных решений для обработки больших объемов данных.

Все NoSQL-системы при всем их многообразии можно разделить на два больших класса. Во-первых, это различные виды NoSQL базы данных. Типичными представителями этого класса систем являются такие проекты, как MongoDB, Cassandra или HBase. Все они представляют собой разновидность баз данных, хранящих информацию в виде пар „ключ–значение“ и не имеющих жесткой схемы данных. Как правило, подобные продукты горизонтально масштабируются и имеют инструменты для организации отказоустойчивости кластера. Их удобно использовать в условиях постоянно изменяющейся неопределенной структуры данных.

Вторым большим классом NoSQL-систем являются проекты, обеспечивающие горизонтально масштабируемую платформу для хранения и параллельной обработки данных [16]. Они больше подходят для задач с запросами, связанными с сложными вычислениями. К таким системам относят Apache Hadoop, который состоит из двух основных компонент: распределенной кластерной системы Hadoop Distributed File System (HDFS) [17] и программного интерфейса Map/Reduce, где основным преимуществом является обработка неограниченного объема данных за счет хранения всех промежуточных результатов вычислений в файловой системе и выполнения кода программы на определенном наборе данных, находящемся на вычислительном узле, не передавая массивы данных по коммуникационной сети между хранилищами данных. Преимуществами данной системы могут служить простота программирования и наличие множества готовых решений. Однако ограничения централизованной системы управления и распределения задач не позволяют увеличивать бесконечно количество вычислительных узлов, которое ограничено несколькими тысячами.

Заключение. Таким образом, на основании рассмотрения основных подходов и инструментария обработки больших массивов данных, в частности инструментов их интеллектуального анализа, можно сделать вывод об отсутствии соответствующих универсальных средств параллельной обработки, что требует решения задачи их создания на базе последних достижений в технологиях программирования и математической теории.

Список литературы

1. Han, J. and M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, San. Francisco, 2000 [Электронный ресурс].
Режим доступа: http://media.wiley.com/product_data/excerpt/24/04712285/0471228524-1.pdf.

2. MIKE2.0, Big Data Definition [Электронный ресурс]. Режим доступа: http://mike2.openmethodology.org/wiki/Big_Data_Definition.

3. А. В. Карева, И. А. Рыговский. Система хранения и постобработки данных трафика научного учреждения // Труды конференции молодых ученых, Новосибирск. 2011. С. 114–121.

4. BaseGroup Labs — Масштабируемый алгоритм, Scalable Alghorithm. [Электронный ресурс]. Режим доступа: http://www.basegroup.ru/glossary/definitions/scalable_algorithm/.

5. Parallel computing and Monte Carlo algorithms by Jeffrey S. Rosenthal // Far East Journal of Theoretical Statistics. N 4. 2000. P. 207–236.

6. Weizhong Zhao, Huifang Ma and Qing He. Parallel K-Means Clustering Based on Map-Reduce, 2009.

7. Clark F. Olson, Parallel algorithms for Hierarchical clustering, 1995. [Электронный ресурс]. Режим доступа: http://www.cs.gsu.edu/~wkim/index_files/papers/ parallel_hierarchical.pdf.

8. Parallel Algorithms for Hierarchical Clustering and Cluster Validity, Xiaobo Li. [Электронный ресурс]. Режим доступа: http://www.cs.gsu.edu/~wkim/index_files/papers/parallelhierarchical.pdf.

9. OpenMP Consortium [Электронный ресурс]. Режим доступа: http://www.openmp.org.

10. MPI: A Message-Passing Interface Standard. [Электронный ресурс]. Режим доступа: http://www.mpi-forum.org/docs /docs.html.

11. Design an MPI-based parallel and distributed machine learning platform on large-scale HPC clusters, Zhi-Jie Yan Teng Gao Qiang Huo. [Электронный ресурс]. Режим доступа: http://www.ism.ac.jp/IWSML2012/r1.pdf.

12. The R Project for Statistical Computing. [Электронный ресурс]. Официальный сайт проекта R Project. Режим доступа: http://www.r-project.org.

13. Google's MapReduce Programming Model — Revisited, Ralf Lammel. [Электронный ресурс]. Режим доступа: http://userpages.uni-koblenz.de/~laemmel/MapReduce/paper.pdf.

14. M. Stonebraker et al. Requirements for science data bases and SciDB. In CIDR, 2009. [Электронный ресурс]. Режим доступа: http://www-db.cs.wisc.edu/cidr/cidr2009/Paper_26.pdf.

15. J. Choi et al. ScaLAPACK: a portable linear algebra library for distributed memory computers — design issues and performance. Computer Physics Communications, 97(1–2), 1996, High-Performance Computing in Science. [Электронный ресурс]. Режим доступа: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.7367.

16. NoSQL vs. Parallel DBMS for Large-scale Data Management, Kyu-Young Whang, 2011. [Электронный ресурс]. Режим доступа: http://www.cintec.cuhk.edu.hk/DASFAA2011/doc/KyuYoungWhang-Panel.pdf.

17. DFS Architecture Guide. [Электронный ресурс]. Режим доступа: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.

Рыговский Иван — аспирант СибГУТИ;
ivan.rigovsky@gmail.com

Дата поступления — 02.05.2014