УДК 004.421.2:519.16

А.С. Шоманов, М.Е. Мансурова, Е.Н. Амиргалиев, Б.А. Урмашев
Казахский Национальный университет им. аль-Фараби, г. Алматы

РЕШЕНИЕ РЕСУРСОЕМКИХ ЗАДАЧ НА ОСНОВЕ РАЗНОСТНЫХ СХЕМ С ПРИМЕНЕНИЕМ APACHE SPARK

В современном мире существует множество масштабных и ресурсоемких задач, требующих высокоэффективных и хорошо разработанных подходов к их решению. Важной задачей является создание эффективных механизмов и алгоритмов, которые обеспечивают производительность, ускорение, отказоустойчивость при решении различного класса задач на основе параллельных вычислений.

Apache Spark представляет собой быстрый механизм обработки данных в памяти с удобными API-интерфейсами разработки, позволяющий эффективно выполнять различные задачи, требующие быстрого итеративного доступа к наборам данных. Apache Spark успешно применяется для решения широкого спектра задач в науке, медицине, банковской сфере, государственном секторе [1]. Основной абстракцией Apache Spark является распределенная коллекция элементов, называемая Residient Distributed Dataset (RDD). В работе [2] авторы описали внутренний дизайн и свойства RDD и продемонстрировали способность выполнять вычисления в памяти на больших кластерах отказоустойчивым способом. В статье авторы также сообщили о значительном ускорении итеративных алгоритмов на графах и машинного обучения, используя Apache Spark по сравнению с Apache Hadoop. Также главной особенностью Apache Spark является способность производить кластерные вычисления в памяти, которые увеличивают скорость обработки приложения. Apache Spark предназначен для широкого спектра задач, таких как пакетные приложения, итерационные алгоритмы, интерактивные запросы и потоковая передача.

В параллельном программировании важной задачей является решение проблем, связанных с отказоустойчивостью выполняемого приложения. Для MPI (протокол передачи сообщений) характерны такие недостатки, как неактивное использование ресурсов центрального процессора при обмене данными между узлами, отсутствие надежности и потеря данных из-за сбоев в отдельных узлах кластера или нарушений в работе коммуникационной сети. В настоящее время не так много альтернатив для выбора, чтобы заменить или увеличить вычислительную парадигму MPI для крупномасштабных научных проблем с итерационными схемами, и исследования продолжаются с различными успехами в этой важной области.

Очевидное преимущество MPI над Apache Spark заключается в том, что MPI потенциально может быть расширен для широкого спектра приложений в HPC и по-прежнему будет достаточно быстрым. Однако основная проблема с MPI заключается в отсутствии встроенных механизмов отказоустойчивости. Отказы могут быть проблематичными для длительных заданий при присутствии большого количества вычислительных узлов. Существуют разные подходы к предотвращению сбоев в среде MPI, но они отличаются высокой трудоемкостью в их использовании и подвержены множеству ошибок [3]. Также в литературе приводятся различные способы улучшения Mapreduce вычислений на основе применения технологии MPI [4-6].

Описание подхода к решению сеточных задач на основе Apache Spark.

Для описания и тестирования данного подхода к решению задач на основе разностных схем нами была выбрана задача Дирихле для уравнения Пуассона. Формулировка данной

задачи для трехмерной вычислительной области D описывается следующим набором ограничений и уравнений:

$$D = \{(x, y, z): 0 \le x \le l_1, 0 \le y \le l_2, 0 \le z \le l_3\}, \quad (1)$$

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = f(x, y, z), \quad (2)$$

$$u_\Gamma = \varphi(x, y, z). \quad (3)$$

На основе дискретизации задачи (1)-(3) была получена следующая разностная схема:

$$u_{i,j,k}^{n+1} = \left(\frac{u_{i+1,j,k}^n + u_{i-1,j,k}^n}{\Delta x^2} + \frac{u_{i,j+1,k}^n + u_{i,j-1,k}^n}{\Delta y^2} + \frac{u_{i,j,k+1}^n + u_{i,j,k-1}^n}{\Delta z^2} - f_{i,j,k}\right) \Big/ \left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} + \frac{2}{\Delta z^2}\right). \quad (4)$$

Таким образом, задача состоит в написании параллельного алгоритма для получения решения задачи (1)-(3) на основе применения Apache Spark.

Алгоритм Apache Spark для решения задачи Дирихле для уравнения Пуассона использует объектно-ориентированное представление точек в вычислительной области. Класс Point используется для хранения одной точки в вычислительной сетке. Каждая точка состоит из следующих свойств: целочисленное значение partition_id, целое число x для координаты x, целое число y для координаты y, целое число z для координаты z, число с плавающей запятой для значения в определенной точке (x, y, z). Свойство partition_id отвечает за сохранение номера раздела точки, являющейся идентификатором подобласти.

Алгоритм состоит из нескольких шагов, каждый шаг выполняет определенные операции с RDD. Для алгоритма мы выбрали метод декомпозиции 1D. Метод 1D декомпозиции сообщает, что вычислительная область должна быть разделена согласно x-координате на смежные подобласти.

Алгоритм решения данной задачи может быть описан последовательностью из нескольких шагов, описанных ниже:

1. Сгенерировать начальную вычислительную область и сохранить значения в каждой точке в RDD <Point>.

2. Выполнить операцию map для отображения каждой записи RDD <Point> в PairRDD <key, Point>, где key представляет номер раздела, а Point - это значение в определенной точке вычислительной области.

3. Выполнить трансформацию groupBy для разделения точек на несколько разделов, где операции над каждым разделом выполнялись бы параллельно.

4. Выполнить трансформацию map для каждого раздела следующим образом:

4.1. Создать трехмерный массив для каждого раздела и отобразить значения точек этого раздела с соответствующими элементами этого массива.

4.2. Выполнить вычисления на разбиениях на основе разностной схемы уравнения Пуассона.

4.3. Отобразить обратно элементы 3-мерного массива в RDD <Point>.

4.4. Возвратить как результат список вычисленных значений точек, сохраненных в RDD <Point>.

5. Проверить условие завершения.

5.1. Если условие завершения выполнено, перейти к шагу 6.

5.2. Если условие завершения не выполняется, перейти к шагу 2.

6. Сохранить полученные значения точек в HDFS (распределенная файловая система Hadoop).

Особенностью данного алгоритма является то, что вычислительный процесс разбивается между отдельными процессами, которые в параллельном режиме производят вычис-

ления на определенной подобласти вычислительного региона. После каждой итерации алгоритма происходит обмен граничными точками отдельных подобластей. Схематическое описание алгоритма приведено на рис. 1.

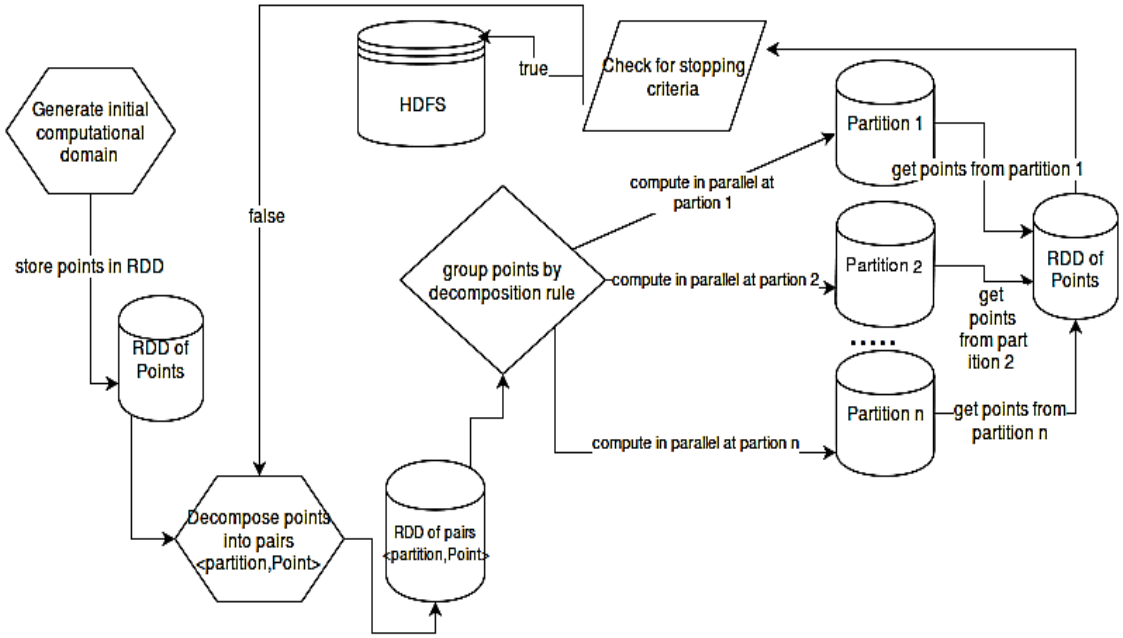

Рисунок 1 – Схема-описание алгоритма

Важно отметить, что перед выполнением любых действий или преобразований в PairRDD <key, Point> мы сделали дополнительное разбиение, чтобы указать, что каждый раздел должен быть расположен на отдельном узле кластера. Это необходимая оптимизация, чтобы избежать использования огромных сетевых ресурсов и, следовательно, очень низкой производительности программы.

Поскольку каждый раздел RDD определен для хранения только точек с одним определенным ключом, согласно алгоритму должен существовать механизм для обмена граничными точками между различными разделами. Основным узким местом в этом алгоритме являются дополнительные накладные расходы, связанные с выполнением этого обмена. Чтобы выполнить обменную операцию, мы должны применять преобразования отображения, группировки и разбиения на каждой итерации алгоритма. Преобразование отображения сформирует на основе элементов RDD<Point> RDD, состоящий из пар Point и номера раздела, где номер раздела будет представлять ключ. Номер раздела определяется координатами отдельных точек. После этого точки группируются путем применения преобразования groupByKey. И наконец, каждый раздел разделяется на отдельные разбиения, применяя преобразование partitionBy. Такие операции несколько ухудшают производительность программы.

Также важным моментом в применении данного подхода является тот факт, что все вычисления производятся в оперативной памяти. В случае с Apache Hadoop требуется после каждой итерации записывать данные в распределенную файловую систему, что приводит к значительным сетевым и вычислительным издержкам.

Нами было проведено тестирование данного подхода на кластере, состоящем из 10 вычислительных узлов, состоящих из 8 компьютеров с процессорами Core i-5 и оперативной памятью 16 Гб, а также 2 серверов с 4-ядерными Intel Xeon процессорами.

Рис. 2 описывает зависимость времени выполнения алгоритма, основанного на применении Apache Spark для различного количества вычислительных ядер и для различных размеров вычислительной области. Размер вычислительной области характеризуется количеством точек в кубе с количеством точек в каждом направлении равным указанной на графике величине. Таким образом, величине 720 будет соответствовать размерность равная 720×720×720 точек.

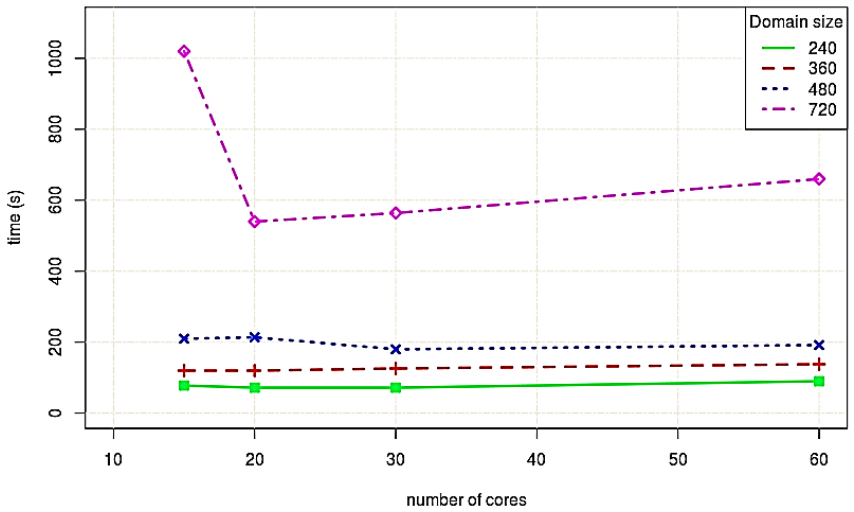

Рисунок 2 – График сравнения количества ядер/время выполнения для различных размерностей задачи

На рис. 3 приведен график времени выполнения алгоритма Apache Spark в сравнении с Apache Hadoop.

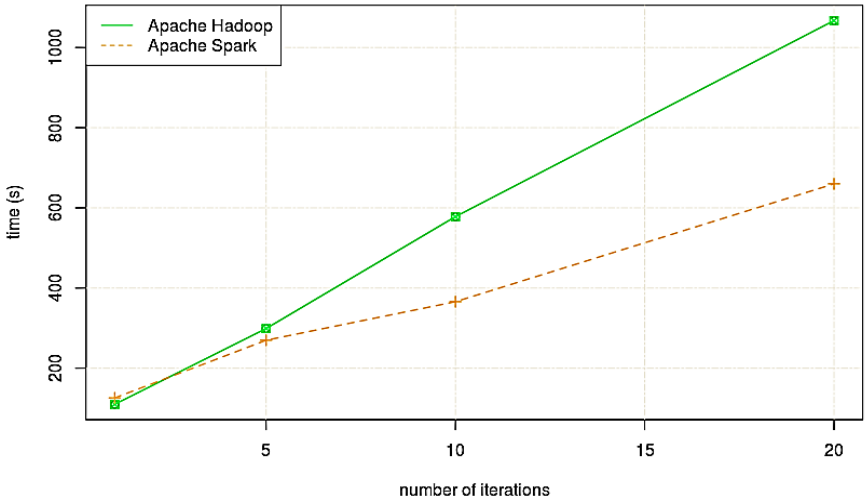

Рисунок 3 – График сравнения количества итераций/время выполнения для Apache Spark/Apache Hadoop

Таким образом, из графика времени выполнения алгоритма Apache Spark в сравнении с Apache Hadoop (рис. 3) видно, что для большего количества итераций Apache Spark показывает значительный выигрыш по времени выполнения по сравнению с Apache Hadoop.

Список литературы

1. Zaharia M., Chowdhury M., Franklin M.J., S. Shenker, I. Stoica. Spark: Cluster Computing with Working Sets. - HotCloud'10 Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, 2010. - 10-10.
2. Zaharia M., M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, I. Stoica. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. - NSDI'12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, 2012. - 2-2.
3. Gropp W., E. Lusk. Fault Tolerance in MPI Programs. - International Journal of High Performance Computing Applications, 2004. - 18(3), 363-372.
4. Lu X., F. Liang, B. Wang, L. Zha, Z. Xu. DataMPI: Extending MPI to Hadoop-like Big Data Computing. - Parallel and Distributed Processing Symposium, 2014 IEEE 28th International, 2014. - 829-838.
5. Lu X., F. Liang. Accelerating Iterative Big Data Computing Through MPI. - International Journal of computer science and technology, 2016. - 30 (2016), 283-294.
6. Lu X., B. Wang, L. Zha, Z. Xu. Can MPI benefit Hadoop and Mapreduce Applications?. - Proceedings of 40th International Conference on Parallel Processing Workshops, 2011. - 371-379.

Получено 15.03.2017

С ТОЧНОСТЬЮ ДО МЕТРА

Три спутника Европейского космического агентства, вооруженные радарами, проработали на орбите почти три года и передали на Землю более 500 терабайт данных о рельефе нашей планеты. В результате возникла объемная карта высот всей суши, промеренных радиолокацией с точностью до одного метра, хотя, по предварительным оценкам, точность должна была составлять до 10 метров. По карте можно, например, проследить за таянием ледников и другими изменениями суши за время работы спутников.

«Наука и жизнь» № 2, 2017