ПРИМЕНЕНИЕ ТЕХНОЛОГИИ APACHE BIG DATA В ЗАДАЧАХ КЛИМАТИЧЕСКОГО МОНИТОРИНГА

Золотов С.Ю.[(1)(2)], Турчановский И.Ю.[(2)]

[(1)] Институт мониторинга климатических и экологических систем СО РАН, г. Томск
[(2)] Институт вычислительных технологий СО РАН, г. Новосибирск

DOI: 10.25743/ICT.2019.43.75.011

Проведен эксперимент по апробации технологии Big Data в исследованиях климатических систем. В ходе эксперимента было реализовано четыре варианта решения тестовой задачи. Ускорение расчетов с помощью технологии Apache Big Data вполне достижимо и наиболее эффективный способ для этого найден в четвертом варианте решения тестовой задачи.

Задачи климатического мониторинга, технология Apache Big Data.

APACHE BIG DATA TECHNOLOGY APPLICATION IN THE CLIMATE MONITORING TASKS

Zolotov S.Yu., Turchanovsky I.Yu.

The experiment was conducted to test Big Data technology in the study of climate systems. Four options for solving the test problem were implemented during the experiment. Accelerating calculations using Apache Big Data technology is quite achievable and the most effective way to do this is found in the fourth solution to the test problem.

Climate monitoring tasks, Apache Big Data technology.

Введение

Исследования климатических систем, как правило, сопряжены с обработкой больших массивов данных, например, данных реанализов NCEP, JRA, ERA, а также данных, регулярно получаемых сетью метеорологических станций, данных космического мониторинга и т.д. Можно выделить две проблемы, связанные с обработкой больших массивов данных, лежащие в технологической области: большое время считывания многомерных данных и ограничение объема обрабатываемых данных в оперативной памяти ЭВМ.

С появлением технологий Big Data, развиваемых под эгидой фонда свободного программного обеспечения Apache, возникла возможность существенно ослабить имеющиеся технологические ограничения.

Был проведен эксперимент по апробации технологий Big Data в исследованиях климатических систем. В основные задачи этих исследований входят: мониторинг состояния климатической системы; изучение и анализ явлений и процессов в атмосфере, океане и на суше; мониторинг возможных физических и экологических изменений в окружающей среде в результате климатических изменений [1]. Для решения этих задач ученые часто используют архивы баз данных реанализов – это динамически согласованные поля климатических величин, характеризующих состояние атмосферы, суши и океана по всему земному шару. Основным достоинством глобальных баз данных реанализов является пространственно-временная непрерывность выходных данных.

В качестве тестовой была выбрана задача подсчета среднемесячных, среднегодовых и сезонных трендов температуры атмосферы Земли за период с 1960 по 2010 гг. по данным реанализов NCEP-NCAR [2] и JRA-55 [3]. По результатам теста необходимо ответить на вопрос: возникнет ли ускорение вычислений за счет распараллеливания на многопроцессорной ЭВМ или многомашинном кластере средствами технологии Apache Big Data? Основным критерием являлось общее время расчета, включая считывание данных и вычисление трендов. При возможности, отдельно оценивались вклад операций ввода-вывода и непосредственно вычислений.

Выбор тестовой задачи и технологии обусловлен требованиями, выдвинутыми в ИВТ СО РАН [4, 5], где в качестве центрального требования выступает возможность обработки сверхбольших массивов данных.

Технологии Apache Big Data

Ядром технологии Apache Big Data являются две технологические компоненты: Apache Hadoop [6] – для организации распределенных файловых хранилищ неограниченной ёмкости, и Apache Spark [7] – для организации параллельных вычислений на многомашинных кластерах. Данные компоненты взаимно дополняют друг друга.

Основная идея, реализуемая Spark – разделение данных на отдельные части (партиции) и обработка этих частей в памяти множества ЭВМ, объединенных сетью. Пересылка данных выполняется только при необходимости с автоматическим расчетом момента времени, когда будет произведен данный обмен. При использовании Spark в сопряжении с Hadoop вычисления производятся с учетом локальности данных. Распределенная файловая система Hadoop (HDFS) хранит файлы в виде блоков, распределенных между дисковыми подсистемами ЭВМ, объединенных в кластер. Принцип вычислений с учетом локальности предполагает, что программа (или фрагмент параллельной программы) передается и вычисляется на ЭВМ, на которой находятся необходимые блоки данных. Этот принцип позволяет минимизировать обмен данными в процессе счета. Результат расчета также может записываться в файлы HDFS. Можно представить процесс обработки данных в виде конвейера, когда они поэтапно трансформируются и выходные данные одного этапа являются входными для другого. Сохраняя промежуточные результаты, исследователь имеет возможность вернуться к ним без повторения всей предшествующей цепочки трансформаций и продолжить расчет иным методом или с другими параметрами.

Описание тестовой задачи

Было выбрано пять уровней по высоте и пять географических точек. Таким образом, результат представляет собой матрицу из 25 строк (5 уровней по 5 точек) и 17 столбцов (для всех месяцев, четырех сезонов и итоговый за год). В каждую ячейку таблицы заносится значение линейного тренда, рассчитанное за 51 год с 1960 по 2010 гг.

Данные реанализа NCEP-NCAR представляют собой совокупность файлов по каждой метеорологической величине отдельно. Один файл содержит значения величины за один календарный год с временным шагом 6 часов (1460 или 1464 отсчета) по всем уровням давления (17 уровней от 1000 до 10 гПа) на сетке 2,5° на 2,5° (73 значений широты и 144 значений долготы). Объем одного файла для температуры воздуха равен порядка 310 Мбайт.

Данные в файлах реанализа NCEP-NCAR находятся в формате HDF5 [8]. Одной из отличительной особенности этого формата является индексация данных в виде Б-дерева, что

позволяет получать к ним параллельный доступ.

В архиве JRA-55 в одном файле хранятся значения одной метеорологической величины за один временной срок. В файле имеются значения величины по всем уровням давления (37 уровней от 1000 до 1 гПа) на сетке 1,25° на 1,25° (145 значений широты и 288 значений долготы). Объем одного файла для температуры воздуха равен порядка 2,2 Мбайт, что составляет 3,2 Гбайт за один календарный год (1460–1464 файлов).

В архиве JRA-55 используется формат данных GRIB-1 [9], который не предназначен для организации параллельного доступа к данным. Дополнительно, в файле находятся массивы сжатых значений величины вместо их оригинальных значений.

Для проведения испытания использовалась вычислительная система на базе 4-х ядерного Intel Xeon, с частотой 2.4 ГГц, ОЗУ – 64 Гб. Система оснащена RAID-массивом и работает под управлением CentOS Linux 7.

Извлечение и трансформация данных реализованы на языке Scala, а непосредственно алгоритм расчета тренда был реализован на языке Java.

Результаты вычислений

В ходе эксперимента было реализовано четыре варианта решения тестовой задачи. Варианты отражают эволюцию программы расчета от полностью последовательной реализации с размещением оригинальных файлов моделей на локальной файловой системе Linux ext3 до полностью параллельной, с переходом на специальный формат хранения и размещением в распределенной файловой системе.

Первый вариант представляет наиболее простую реализацию, без организации параллелизма. В ходе повторных запусков программы ярко проявился эффект кэширования файлов в ОС Linux. Так как вычисления часто проводятся повторно на тех же входных данных при изменении каких-либо настроек программы, то было полезно оценить, какое ускорение можно получить за счет кэширования исходных данных.

Модель NCEP предоставляется в виде одного файла формата HDF5 ежегодно, в то время как JRA-55 предоставляется в виде 1460 файлов в формате GRIB за каждый год. Поэтому время считывания данных модели JRA-55 существенно выше, а ускорение за счет кэширования проявляется сильнее (примерно в десять раз для JRA-55 и 2,5 раза для NCEP). Для получения времени расчета без предварительного кэширования проводился сброс всех файловых кэшей узла с помощью специальных команд ОС Linux.

Второй вариант реализации предполагает параллельное считывание данных, агрегирование и последующий расчет трендов. Данные считываются из локальной файловой системы также, как и в первом варианте. Параллельные варианты реализации для NCEP и JRA-55 дают ускорение приблизительно в 2–4 раза, что соответствует линейному ускорению, пропорционально числу процессоров в системе. При этом время, затрачиваемое непосредственно на вычисления, остается стабильным и пренебрежительно малым по сравнению со временем считывания (на три порядка) и составляет примерно 0,05 с. Таким образом, в данной задаче время ввода/вывода преобладает над временем расчета. В этом варианте массивы обрабатываемых данных были организованы в ОЗУ в виде RDD (Resilient Distributed Datasets) – это один из первых способов представления наборов данных для параллельной обработки, появившийся в составе Apache Spark. Так как входные файлы хранились на локальной файловой системе, то число ядер ограничивалось максимальным числом ядер одного узла.

Дальнейшее ускорение расчета возможно за счет масштабирования, то есть, увеличения числа подзадач, параллельно считывающих и вычисляющих составные части результата, подвергаемые затем агрегации. Если увеличение числа ядер будет осуществляться за счет объединения множества ЭВМ в виде кластера, то и хранение данных тоже должно быть организовано в распределенном виде. Поэтому, третьим вариантом стал расчет тестовой задачи на двухузловом кластере с общим числом ядер 8.

Самым простым решением было разместить файлы архивов NCEP и JRA в исходном формате в хранилище Hadoop, объединяющем дисковые подсистемы двух вычислительных машин. Однако, на этапе адаптации программного кода выяснилось, что работа с архивом JRA не может быть организована в распределенном варианте, так как при обращении к файлам в формате GRIB создаются индексные файлы, записываемые в том же каталоге, что и файл архива. А так как библиотека GRIB не рассчитана на работу с Hadoop, а индексация производится автоматически, то расчет завершается с ошибкой. Дальнейшие усилия были сосредоточены на работе с архивом NCEP.

С архивом NCEP возникла другая сложность: прикладная библиотека рассчитана на работу с файлами в режиме прямого доступа к файлам оригинального формата, тогда как Hadoop обеспечивает только последовательный доступ. Обходным решением стала загрузка всего файла NCEP полностью в оперативную память рабочего процесса, тогда режим прямого доступа обеспечивался без обращения к файловой системе. Отрицательной стороной этого решения стало увеличение времени считывания всего файла (порядка 300 Мб), а также увеличение требований к оперативной памяти рабочего процесса, тогда как для алгоритма расчета требуется лишь небольшой сегмент файла. Тем не менее, применение распределенного хранилища и 8-ми ядер обеспечило ускорение расчета приблизительно в 2 раза по сравнению с 4-х ядерным вариантом и хранением в локальной файловой системе ext3, без предварительной буферизации. Повторный расчет (когда данные кэшированы в ОЗУ) показывает, что в распределенной файловой системе есть накладные расходы на ввод/вывод, а так как время вычислений на два порядка меньше времени ввода/вывода, то увеличение числа ядер до 8 не приводит к выигрышу в общем времени расчета. В задачах с более вычислительно емкой составляющей увеличение числа задействованных ядер должно приводить к ускорению расчета.

Результаты третьего варианта поставили вопрос о том, как избежать полной загрузки входного файла в ОЗУ, так как для более сложных расчетов может потребоваться считывание дополнительных файлов модели, например, с характеристиками ветра и, соответственно, ограничение на объем ОЗУ станет критическим.

Решение, предложенное в четвертом варианте, заключается в том, чтобы предварительно преобразовать исходный формат файлов HDF5 или GRIB к виду, когда считывание из HDFS происходит выборочно, на основе заданных параметров. В технологии Apache Big Data разработан мощный инструментарий, подходящий для решения такого сорта задач. Во-первых, это формат файлов Parquet, ориентированный на табличное представление данных. Во-вторых, это представление наборов данных Spark DataSet, являющееся развитием Spark RDD, и ориентированное на реляционное представление данных и соответствующий прикладной интерфейс, напоминающий работу с традиционными СУБД: Spark SQL.

В четвертом варианте сначала была выполнена программа, трансформирующая исход-

ный формат модели в табличное представление формата Parquet с записью в HDFS, а затем – программа, осуществляющая выборку данных и последующий расчет. Алгоритм расчета был кардинально переработан в части агрегации данных, а именно в вычислении средних значений для месяца, сезона и года, так как Spark SQL предоставляет встроенные агрегатные функции, а также функции сортировки и группировки данных. Программа четвертого варианта стала существенно короче, так как большая часть операций была совмещена с операциями выборки данных. Общее время расчета несколько сократилось по сравнению с третьим вариантом, как с кэшированием, так и без. При этом ускорение за счет кэширования составило 200% от общего времени расчета, в то время как в 1–3 вариантах оно составляло от 300% до 1000%. Это говорит о том, что механизм выборки Spark SQL не требует считывания большого объема данных в ОЗУ и кэширование обеспечивает меньшее ускорение. Также заметно, что время расчета для NCEP и JRA отличается незначительно так как формат и способ выборки для этих моделей идентичен, а отличие во времени обусловлено тем, что JRA предоставляет несколько более подробную пространственную сетку.

Заключение

Ускорение с помощью технологии Apache Big Data вполне достижимо и наиболее эффективный способ для этого найден в четвертом варианте решения тестовой задачи. Суть найденного решения сводится к преобразованию исходных массивов данных к формату, подходящему для хранения в распределенной файловой системе HDFS и применению технологии Spark SQL из стека Apache Big Data для параллельной обработки данных на вычислительных кластерах.

Следует отметить, что предварительно ожидалось более трудоемкое решение с точки зрения разработки исходного кода, в частности, что потребуется использование специально модифицированного вида RDD для работы с многомерными данными. Но обнаружилось, что стек Apache Big Data в настоящее время предоставляет мощные инструменты, открывающие хорошие перспективы по эффективному анализу больших массивов данных в исследованиях атмосферы и океана нашей планеты без излишнего усложнения программного кода.

ЛИТЕРАТУРА

[1] Оценочный доклад об изменениях климата и их последствиях на территории Российской Федерации. Том I. Изменения климата / М.: Росгидромет, 2008. 228 с.

[2] *Kalnay E., Kanamitsu M., Kistler R., et al.* The NCEP/NCAR 40-year reanalysis project // Bulletin of the American Meteorological Society. 1996. V. 77, No. 3. P. 437–471.

[3] *Kobayashi S., Ota Y., Harada Y., et al.* The JRA-55 reanalysis: General specifications and basic characteristics // Journal of the Meteorological Society of Japan. 2015. V. 93. No. 1. P. 5–48.

[4] *Юрченко А.В.* К концепции информационно-аналитической системы поддержки научных исследований, основанных на интенсивном использовании цифровых данных // Вычислительные технологии. 2017. Т. 22. № 4. С. 105–120.

[5] *Бойченко И.В., Турчановский И.Ю.* Построение сервиса данных в информационных системах научных исследований на основе парадигмы Big Data // Вестн. Новосиб. гос. ун-та. Серия: Информационные технологии. 2015. Т. 13. Вып. 2. С. 22–27.

[6] Apache Hadoop. **https://hadoop.apache.org** (дата обращения 25.09.2019).

[7] Apache Spark. **https://spark.apache.org** (дата обращения 25.09.2019).

[8] The HDF5 library & file format. **https://www.hdfgroup.org/solutions/hdf5/** (дата обращения

25.09.2019).

[9] A guide to the code form FM 92-IX Ext. GRIB. **https://www.wmo.int/pages/prog/www/WDM/Guides/Guide-binary-2.html** (дата обращения 25.09.2019).