УДК 519.6

М. В. Гладкий
Белорусский государственный технологический университет

МОДЕЛЬ СТОИМОСТИ ЗАТРАТ MAPREDUCE-ВЫЧИСЛЕНИЙ

Приведено описание математической модели стоимости затрат при выполнении MapReduce-заданий в распределенных средах. Выявлено, что общая стоимость всех затрат состоит из стоимости затрат на подготовку кластера к работе и запуск его задач, а также из стоимости ресурсов, необходимых на непосредственное выполнение этих задач.

Рассмотрены основные параметры, влияющие на скорость выполнения и количество потребляемых ресурсов при решении задач с использованием парадигмы MapReduce. Определено четыре класса параметров, участвующих в оптимизации MapReduce: поток данных, поля стоимости, статистика потока данных и статистика полей стоимости.

Установлен признак линейности модели стоимости затрат при решении задачи поиска частотности слов в массиве документов. Выявлено, что линейность сохраняется при решении задач, требующих последовательного применения нескольких MapReduce-моделей вычислений.

Ключевые слова: модель стоимости, MapReduce, распределенные вычисления, кластер.

M. V. Gladkiy
Belarusian State Technological University

THE COST MODEL OF MAPREDUCE COMPUTATIONS

There is a mathematical description of the cost model that executes the MapReduce jobs in a distributed environment. It is revealed that the total cost of all expenses consists of the cluster preparation work cost and tasks, start and also the cost of resources needed for direct execution of these tasks.

There are main parameters that affect the execution speed and the amount of consumed resources in solving problems using the MapReduce paradigm. Four classes of parameters which participate in MapReduce optimization are defined: dataflow field, cost field, dataflow statistic field, cost statistic field.

A sign of linearity of the cost model in solving the problem to find the words frequency in the array of documents was determined.

It was found that the linearity is maintained in solving problems that require consistent application of multiple MapReduce computation models.

Key words: cost model, MapReduce, distributed computing, cluster.

Введение. Технологии распределенной обработки больших объемов данных переживают пик своего развития. Потребность в их использовании непрерывно растет, что обусловлено активным внедрением вычислительных средств в производственный процесс. Все большее значение при разработке проектов с применением парадигмы MapReduce играют его стоимостные параметры и время реализации. Это способствует усилению значимости использования существующих стандартов и новых вычислительных технологий, позволяющих существенно сократить и удешевить этап проектирования.

Программный стек Apache Hadoop позволяет быстро реализовать распределенное приложение с требуемым функционалом, не задумываясь об архитектурных особенностях кластера, модели и концепции распределенных вычислений. Это приводит к тому, что программисты не знают или не понимают, какие параметры оказывают наибольшее влияние на скорость обработки информации в кластере, каким образом можно оптимизировать затраты и уменьшить время работы MapReduce-задач при выполнении операций в распределенных средах.

Основная часть. Затраты при использовании распределенных структур обработки данных можно разделить на следующие группы:

– затраты на покупку и обновление аппаратных частей кластеров и обслуживание внутренней инфраструктуры (серверы, сетевые хранилища, периферийные устройства);

– затраты на организацию глобальной или локальной сети между отдельными узлами кластера, а также обеспечение защищенного удаленного доступа к машинам кластера;

– стоимость разработки, сопровождения и лицензирования программного обеспечения (ПО), выполняющего определенные задачи обработки и анализа больших объемов данных в распределенных средах;

– затраты на управление и конфигурирование кластером.

Наиболее важными затратами, на которые следует обратить внимание, являются затраты на разработку и сопровождение программного обеспечения. При равных возможностях аппаратного обеспечения и количества узлов в кластере оптимизированное ПО, а также эффективные алгоритмы и методы обработки информации в распределенных средах позволяют сэкономить до 30% ресурсов кластера.

Инфраструктура Hadoop имеет встроенные механизмы анализа, настройки и оптимизации производительности. Все эти настройки основаны на создании специальных профилей выполнения. В зависимости от ситуации пользователь имеет возможность выбирать между определенными профилями. По умолчанию профилирование не используется при выполнении MapReduce-задач. Для настройки профилей в системе предусмотрен специальный класс *mapreduce.task.profile*. Программист может расширять или изменять поведение профилей, создавать новые профили, использующие определенные значения конфигурации, влияющие на логику работы всего кластера.

В основе создания конфигурационных файлов для эффективного профилирования MapReduce-задач лежит модель стоимости затрат вычислений. Модель включает в себя затраты на запуск заданий, вычисления, операции ввода/вывода на различных этапах работы фреймворка. Она параметризирована с размером входных данных решаемых задач и выбранными свойствами выполняемой программой MapReduce, а также свойствами и настройками кластера, на котором программа запущена. Это помогает программисту оценить поведение масштабирования, а также определить критерий для сравнения производительности различных программ MapReduce.

Выделяют четыре класса параметров конфигурации фреймворка MapReduce, влияющих на время выполнения заданий и количество использованных кластером ресурсов [1]:

1) поток данных – количество записей, протекающих через различные задачи, и этапы выполнения MapReduce-заданий. Примерами полей являются количество выходных строк, возвращаемых в результате работы Map-задачи, количество байт, прочитанных при работе с локальной файловой системой [2];

2) поля стоимости – время выполнения задачи или фаз определенной задачи в MapReduce. Примерами полей стоимости являются время, затраченное на настройку параметров кластера при запуске задачи, время, необходимое на выполнение операции Merge или Shuffle [2];

3) статистика потока данных – сбор статистических данных о потоке. Например, среднее число выходных записей Map-задачи на каждую входную запись, степень сжатия выходных данных Map-процедуры, количество оперативной памяти, необходимое в среднем на обработку одной Map-операции;

4) статистика полей стоимости – сбор статистической информации о времени выполнения MapReduce-задач. В данную группу попадают такие параметры, как затраты процессорного времени, необходимые на выполнение определенной операции, средняя стоимость операции ввода/вывода на 1 байт обработанной информации из DFS (распределенная файловая система).

Параметры конфигурации фреймворка влияют на количество задач, необходимых для решения MapReduce-задания. Для выполнения операции, в зависимости от ее типа, необходимо затратить определенное количество ресурсов: процессорное время, объем оперативной памяти, дисковое пространство и т. д. В результате общая стоимость затрат на вычисления в кластере состоит из стоимости затрат на подготовку кластера и запуск задания, а также из стоимости ресурсов, необходимых на непосредственное выполнение этой задачи:

$$C_O = C_{ST} + C_P, \qquad (1)$$

где C_{ST} – затраты на запуск задач; C_P – затраты на обработку данных.

Затраты на запуск задач. Стоимость запуска задания и всех задач, выполняемых в рамках этого задания, состоит из затрат, необходимых на конфигурацию задания MapReduce, включая время работы функции Splitter (функция вычисляет границы разделения входных файлов на блоки данных) и стоимость затрат на запуск всех операций Map и Reduce в рамках определенного задания:

$$C_{ST} = C_S + C_T, \qquad (2)$$

где C_S – затраты на конфигурацию кластера; C_T – затраты на запуск задач.

Стоимость затрат на запуск операций Map и Reduce прямо пропорциональна средней стоимости затрат, необходимых на подготовку к запуску одной задачи, общему количеству Map- и Reduce-задач и обратно пропорциональна количеству процессорных ядер в кластере [3]:

$$C_T = c_t \frac{N_m + N_r}{N_{cpu}}, \qquad (3)$$

где c_t – стоимость запуска Map- или Reduce-задачи; N_m – количество экземпляров Map-функции; N_r – количество Reduce-задач; N_{cpu} – количество процессорных ядер в кластере.

Количество Map-задач тесно связано с количеством блоков входных данных и находится как отношение общего размера входных данных и размера блока данных в операциях ввода/вывода. Данная величина используется внутри фреймворка для организации процессов ввода/вывода между локальной и распределенной файловыми системами:

$$N_m = \frac{S_i}{S_{ch}}, \quad (4)$$

где S_i – размер входных данных; S_{ch} – размер блока данных внутри фреймворка MapReduce.

При локальном запуске MapReduce-задания значением c_t можно пренебречь. При запуске заданий в режиме кластера время запуска одной задачи будет также мало, если размеры входных файлов меньше размера блока данных DFS. В противном случае функции Splitter необходимо время для разбиения входных данных на блоки определенной длины.

Затраты на обработку данных. Стоимость обработки, свертки, перемещения данных по сети в несколько раз превышает стоимость затрат на предварительную подготовку к запуску задания и его задач [1]. Она состоит из стоимости операции ввода/вывода, включая затраты на передачу информации между узлами кластера, и стоимости задач, необходимых для выполнения всех операций над данными в модели MapReduce [2]:

$$C_P = C_{IO} + C_{MR}, \quad (5)$$

где C_{IO} – стоимость операций ввода/вывода; C_{MR} – затраты на выполнение задач в модели MapReduce.

Стоимость операций чтения данных из постоянного запоминающего устройства (ПЗУ) включает в себя стоимость чтения 1 байта данных, передаваемого по сети между узлами кластера, а также стоимость чтения 1 байта данных из DFS. Данные операции выполняются на том же узле, что и Map-задачи:

$$C_R = R_{dfs} + R_{net}, \quad (6)$$

где R_{dfs} – стоимость чтения данных из DFS; R_{net} – стоимость передачи данных по сети.

Парадигму MapReduce обычно используют для обработки данных больших объемов. Однако нельзя просто умножить стоимость затрат на чтение 1 байта информации на общее количество считанных байт, поскольку фреймворк использует кластерную архитектуру. Таким образом, уравнение нахождения стоимости затрат на чтение данных имеет вид

$$R_{dfs} = r_{dfs} \frac{S_{mo}}{N_{cpu}}, \quad (7)$$

где r_{dfs} – стоимость чтения 1 байта данных из распределенной файловой системы; S_{mo} – общий размер выходных данных (байт) после выполнения всех Map-функций.

Стоимость считывания данных между узлами кластера зависит от размера выходных данных после выполнения всех Map-функций и числа эффективных Reduce-задач [1]:

$$R_{net} = r_{net} \frac{S_{mo}}{N_{Ref}}, \quad (8)$$

где r_{net} – стоимость передачи 1 байта данных по сети между узлами кластера; N_{Ref} – количество эффективных Reduce-задач.

Задача является неэффективной, если функция Reduce возвращает пустой массив данных. Система должна предугадывать такие исходы на ранних стадиях работы фреймворка MapReduce и освобождать ресурсы, необходимые для выполнения данных задач. Освободившиеся ресурсы должны быть использованы для более оптимального решения оставшихся подзадач [2].

Стоимость операций ввода во время выполнения задач состоит из стоимости операций ввода в локальную память узлов кластера, полученных после выполнения функции Combine, а также стоимости записи в DFS:

$$C_W = W_l + W_{dfs}, \quad (9)$$

где W_l – стоимость записи в локальную файловую систему; W_{dfs} – стоимость записи в распределенную файловую систему.

Стоимость затрат записи данных на локальный диск кластера зависит от общего размера выходных данных после выполнения всех Map-функций и функции Combine, а также количества процессорных ядер в кластере:

$$W_l = w_l \frac{S_{co}}{N_{cpu}}, \quad (10)$$

где w_l – стоимость записи 1 байта в локальную файловую систему; S_{co} – общий размер выходных данных (байт) после выполнения всех Map-функций и функции Combine.

Стоимость записи данных в распределенную файловую систему прямо пропорциональна стоимости записи 1 байта данных в DFS, общему размеру выходных данных задания и обратно пропорциональна количеству эффективных Reduce-задач [1]:

$$W_{dfs} = w_{dfs} \frac{S_o}{N_{Ref}}, \quad (11)$$

где w_{dfs} – стоимость записи в распределенную файловую систему; S_o – общий размер выходных данных.

Чтение и запись данных в DFS являются более дорогостоящими операциями по сравнению с аналогичными операциями в локальной файловой системе. В MapReduce-фреймворке существует специализированный инструмент Table Scan, основное назначение которого – считывание и запись данных в DFS при переключении между двумя Map- и Reduce-задачами. Для оптимизации Scan может помещать блоки данных в кэш. Для организации кэша используется оперативная память узла кластера. Отключение технологии кэширования блоков приводит к увеличению времени на выполнение операций ввода/вывода данных в несколько раз.

Существует способ уменьшения временных затрат на организацию операций ввода/вывода путем использования специального фреймворка Tez. Данный фреймворк был спроектирован и разработан для работы с большими данными (нескольких сотен гигабайт) в парадигме MapReduce. Tez предполагает, что все операторы должны работать в одном Tez-пространстве и, следовательно, исчезает необходимость затрачивать дополнительные ресурсы на написание промежуточного результирующего набора в DFS [1].

Стоимость выполнения задания в цепочке MapReduce равна сумме всех операций, используемых на каждой стадии работы этой цепочки. В зависимости от типа решаемой задачи могут применяться следующие стадии обработки данных: Map, Combine, Merge, Sort, Reduce [2].

Общая стоимость затрат на выполнение всех Map-задач прямо пропорциональна стоимости выполнения функции Map, размеру входных данных и обратно пропорциональна количеству процессорных ядер в кластере [3]:

$$M = m \frac{S_i}{N_{cpu}}, \quad (12)$$

где m – стоимость операции Map.

Функция Combine применяется в тех случаях, когда в результатах функции Map содержится значительное число повторяющихся значений промежуточного ключа, а определенная пользователем задача Reduce является коммутативной и ассоциативной. В таких случаях необходимо осуществить частичную агрегацию данных до их передачи по сети. Функция Combine реализуется на той же машине, что и задача Map [2]. Стоимость выполнения функции Combine прямо пропорциональна стоимости агрегации 1 байта информации, размеру выходных данных после выполнения Map-функций и обратно пропорциональна числу процессорных ядер в кластере:

$$C_b = c_b \frac{S_{mo}}{N_{cpu}}, \quad (13)$$

где c_b – стоимость операции Combine.

Операция сортировки применяется только в том случае, если разные атомарные обработчики возвращают наборы с одинаковыми ключами. Стоимость данной операции рассчитывается аналогично стоимости операции Combine и зависит от размера выходных данных функции Map [2]:

$$S_r = s_r \frac{S_{mo}}{N_{cpu}}, \quad (14)$$

где s_r – стоимость операции распределенной сортировки.

Если выходные данные функции Map не поместились целиком в оперативную память и были сброшены в дополнительное хранилище-кэш либо постоянную память, то к затратам на сортировку данных добавляются затраты на необходимые операции ввода/вывода.

Суммарная стоимость затрат, необходимых на выполнение операции слияния, зависит от стоимости слияния 1 байта данных, полученного из предварительно отсортированной информации, общего размер выходных данных после выполнения всех Map-функций, а также числа эффективных исходов:

$$M_e = m_e \frac{S_{mo}}{N_{Ref}}, \quad (15)$$

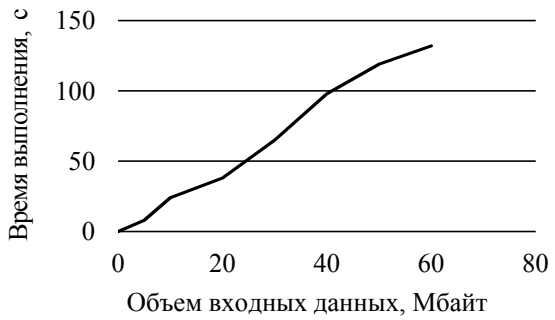
где m_e – стоимость затрат на выполнение задачи Merge.

Аналогичным образом рассчитывается стоимость затрат на выполнение операции свертки:

$$R = r \frac{S_{mo}}{N_{Ref}}, \quad (16)$$

где r – стоимость выполнения задачи Reduce.

Полученные уравнения стоимости затрат имеют линейную зависимость. Данный признак подтверждается практическими исследованиями при решении задачи подсчета слов в массиве документов. В рамках данных исследований определялась зависимость времени выполнения задания от объема информации, поступающей на вход экземплярам функции Map (рисунок).



Зависимость времени выполнения задания от объема входных данных

В результате линейного сглаживания экспериментальных данных получено следующее уравнение:

$$T = Vk_з + s, \quad (17)$$

где T – время выполнения задания; V – объем обработанных данных; $k_з$ – коэффициент стоимости затрат, зависящих от размера блока данных; s – временные затраты, не зависящие от размера блока данных.

В уравнении (17) значение коэффициента стоимости составляет 0,42, а значение временных затрат равно 1,25.

Заключение. В статье рассмотрена модель стоимости затрат MapReduce-вычислений в распределенных средах. Модель обладает линейностью и включает такие параметры, как стоимость затрат на 1 байт обработанной информации, общий размер входных и выходных данных, количество Map- и Reduce-задач, количество процессорных ядер в кластере и другие параметры.

Полученная модель стоимости может применяться в следующих случаях:

– как составная часть оценки эффективности реализации алгоритмов и профилей оптимизации, используемых в программном комплексе Apache Hadoop;

– для разработки функциональных моделей оптимального решения MapReduce-заданий;

– для определения загруженности процессора узла кластера на заданном временном отрезке времени.

При локальных запусках MapReduce-заданий модель стоимости затрат принимает упрощенный вид. В данном случае отсутствуют затраты на передачу данных между узлами кластера, уменьшается количество операций чтения/записи из локальной файловой системы в DFS и наоборот, исключаются затраты на конфигурацию узлов кластера.

Литература

1. Том У. Hadoop: Подробное руководство. СПб.: Питер, 2015. С. 291–293.

2. Гладкий М. В. Модель распределенных вычислений MapReduce // Труды БГТУ. 2014. № 6: Физ.-мат. науки и информатика. С. 121–123.

3. Натан М. Большие данные. Принципы и практика построения масштабируемых систем обработки данных в реальном времени. М.: Вильямс, 2015. С. 147–148.

References

1. Tom U. *Hadoop: Podrobnoye rukovodstvo* [Hadoop: The Definitive Guide]. St. Petersburg, Piter Publ., 2015, pp. 291–293.

2. Gladkiy M. V. The model of distributed computing MapReduce. *Trudy BGTU* [Proceedings of BSTU], 2016, no. 6: Physical-mathematical sciences and informatics, pp. 121–123 (In Russian).

3. Natan M. *Bol'shiye dannye. Printsipy i praktika postroeniya masshtabiruemykh sistem obrabotki dannykh v real'nom vremeni* [Big data. Principles and practice of building highly scalable data processing systems in real time]. Moscow, Vil'yams Publ., 2015, pp. 147–148.

Информация об авторе

Гладкий Максим Валерьевич – ассистент кафедры информационных систем и технологий. Белорусский государственный технологический университет (220006, г. Минск, ул. Свердлова, 13а, Республика Беларусь). E-mail: MaksHladki@gmail.com

Information about the author

Gladkiy Maksim Valer'yevich – assistant lecturer, the Department of Information Systems and Technologies. Belarusian State Technological University (13a, Sverdlova str., 220006, Minsk, Republic of Belarus). E-mail: MaksHladki@gmail.com

Поступила 12.12.2016