Реализация частотного анализа микроблога Twitter в гибридном облаке на базе Binder, платформы Everest и сервиса виртуальных рабочих столов Самарского университета

С.В. Востокин[1], И.В. Бобылева[1,2]

[1]Самарский национальный исследовательский университет им. академика С.П. Королева, Московское шоссе 34А, Самара, Россия, 443086
[2]Акционерное общество «Ракетно-космический центр «Прогресс», Земеца 18, Самара, Россия, 443009

Аннотация. В работе предлагается архитектура распределенного приложения обработки данных в гибридном облачном окружении. Производительность приложения исследована на примере распределенного алгоритма определения частоты использования слов в сообщениях англоязычного микроблога Twitter. Продемонстрирована возможность технически простой и эффективной с точки зрения производительности агрегации вычислительных ресурсов с использованием технологии многозадачных вычислений для обработки данных в гибридных облачных окружениях.

1. Введение

Прогресс в области технологий научных вычислений и обработки данных способствует интенсивному развитию инструментария разработки соответствующих приложений. Одним из популярных инструментов в данной области является интегрированная среда разработки JupyterLab [1]. Важная функциональная особенность среды разработки JupyterLab – возможность вычислений на специально сконфигурированном компьютере без графического интерфейса, а взаимодействие пользователя со средой JupyterLab может осуществляться на любом другом компьютере через сеть Интернет в веб-обозревателе. Облачные службы, в частности используемая в данном исследовании служба Binder [2], развивают данную возможность. Служба Binder позволяет автоматически подготовить требуемую конфигурацию приложения и среды JupyterLab в виде docker-образа [3] и запустить приложение на бесплатной виртуальной машине, например, в облаке Google Cloud. Таким образом, работа с приложением ведется полностью через веб-обозреватель и дополнительно исключается необходимость в специально сконфигурированном компьютере.

Данный метод работы с приложением является удобным для демонстрационных целей, обучения студентов, решения несложных в вычислительном плане задач, коллективной разработки. Однако при простоте реализации приложение оказывается ограниченным в ресурсах. Очевидно, что при развертывании в бесплатных окружениях пользователям приложения следует ожидать минимальной квоты на память и процессорное время, сокращения возможности сетевого взаимодействия, уменьшения прав в профиле на виртуальной машине.

Перечисленные ограничения затрудняют либо не позволяют организовывать высокопроизводительную обработку данных стандартным для JupyterLab способом.

В работе исследуется подход, при котором вместе с интерфейсом JupyterLab на виртуальную машину устанавливается только управляющая часть многозадачного приложения. Управляющая часть приложения, связываясь со вспомогательной облачной платформой Everest [4], задействует свою вычислительную часть, развернутую в другой облачной системе. Тем самым решается проблема получения необходимых процессорных ресурсов и памяти. Задачи исследования: (1) разработка и проверка в процессе развертывания описанной архитектуры распределенного приложения; (2) экспериментальная проверка эффективности вычислений по описанной схеме на примере актуальной задачи из области обработки данных.

2. Метод решения задачи частотного анализа в гибридном облачном окружении

В качестве модельной задачи для оценки эффективности предлагаемой архитектуры распределенного приложения использована задача вычисления частоты слов в текстовом массиве. Исходный текстовый массив для анализа брался из недельного дампа всех сообщений, передаваемых через сервис микроблогов Twitter. Данная задача была выбрана в качестве модельной, так как она имеет много практических приложений, например для анализа рынков [5]. С другой стороны, эта задача важна в нашем исследовании для оценки того, какие объемы данных можно обработать за приемлемое время, а также для оценки типичных времен обработки и пересылки по сети порций текстовых данных, что влияет на эффективность вычислений.

Специфика вычислений в гибридном облаке состоит в том, что компоненты приложения ограничены в возможностях взаимодействия: (1) они общаются не напрямую, а только через сервис-посредник; (2) связаны средними с точки зрения пропускной способности и/или латентности каналами связи. Эти особенности требуют организации обработки данных особым образом.

Одной из адекватных моделей организации вычислений в условиях перечисленных ограничений является модель вычислений, ориентированная на задачи [6]. Согласно данной модели вычисления понимаются как периодический запуск независимых (не взаимодействующих между собой) задач. Причиной пополнения множества запущенных в некоторый момент наблюдения задач может являться завершение очередной задачи.

Наша реализация алгоритма частотного анализа основана на запуске управляющей частью приложения задач двух типов. Задача первого типа получает дамп Twitter в виде JSON-файла, выделяет из него текстовые сообщения на английском языке, разбивает их на слова и формирует упорядоченный по алфавиту список слов с их локальной для данного дампа частотой. Задача второго типа, используя два файла со списком слов, первоначально построенных задачами первого типа, объединяет их в общий упорядоченный список слов и частот, а затем разбивает этот список пополам. Далее содержимое первого файла замещается «младшей» половиной, а содержимое второго файла — «старшей» половиной общего упорядоченного списка. Заметим, что обычный метод слияния файлов тоже решает проблему частотного анализа, но при этом постоянно увеличивается размер получающихся файлов. Разбиение необходимо, так как задачи должны оперировать не слишком большими файлами с целью облегчения передачи файлов для задач по сети.

Нетрудно заметить, что периодическое применение задачи второго типа к упорядоченным по отдельности файлам слов и частот приведет к упорядочению всего информационного массива. Например, если общее количество файлов есть N, то процедура

для всех i от 1 до N-1 выполнять
для всех j от 0 до j < i выполнять задачу 2-ого типа для файла j и файла i
конец цикла по i

приводит к общему упорядочиванию массива файлов. Содержание используемого в работе алгоритма управления задачами заключается в параллельном выполнении этой процедуры. В экспериментах мы применили небольшую оптимизацию, сокращающую диаметр графа зависимостей задач для большего ускорения вычислений. Распараллеливание описанной выше

процедуры для модели задач можно выполнить, как показано в работе [7]. Суть оптимизации выходит за рамки данной работы.

3. Описание процедуры развертывания и работы приложения частотного анализа

Особенностью предлагаемой архитектуры распределенного приложения является адаптация для вычислений в гибридном облачном окружении, построенном на основе ресурсов публичных бесплатных и академических облачных систем. Кроме этого приложение полностью развертывается и запускается через веб-обозреватель.

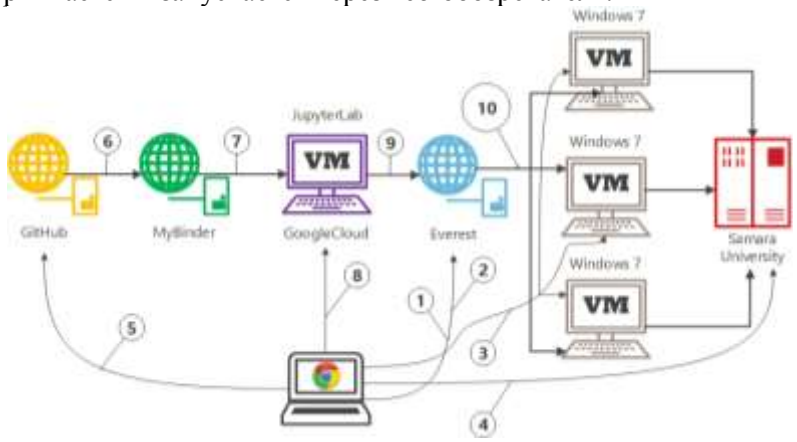

Рисунок 1. Схема развертывания и выполнения приложения.

Рассмотрим последовательность шагов развертывания и выполнения приложения, изображенную на рисунке 1, показывающую реализацию данной особенности его архитектуры.

Шаг 1. Регистрация вычислительных ресурсов приложения на платформе Everest. Получение токенов доступа для программ-агентов через веб-интерфейс платформы Everest.

Шаг 2. Установка компонентов приложения для первичной обработки блоков данных (задача первого типа из п.1) и объединения блоков данных (задача второго типа из п.2). Данная установка выполняется через веб-интерфейс платформы Everest.

Шаг 3. Запуск виртуальных машин под управлением Windows 7 в корпоративном облаке Самарского университета. Установка на них программ-агентов с использованием токенов доступа, полученных на шаге 1. Проверка активности программ-агентов через веб-интерфейс на платформе Everest.

Шаг 4. Загрузка набора данных в виде текстовых файлов Twitter в формате JSON на файловый сервер в корпоративном облаке Самарского университета. Эта загрузка может выполняться через одну из виртуальных машин, запущенных на шаге 3.

Шаг 5. Запуск управляющей части приложения – оркестратора приложения – из репозитория кода GitHub через веб-интерфейс.

Шаг 6. Автоматическое обращение к сервису Binder (после выполнения шага 5) для сборки docker-контейнера с оркестратором приложения, работающим в среде JupyterLab.

Шаг 7. Развертывание docker-контейнера (с шага 6) в облаке Google Cloud. Возвращение ссылки на веб-интерфейс оркестратора приложения в веб-терминал пользователя приложения.

Шаг 8. Запуск оркестратора приложения пользователем через веб-интерфейс, полученный на шаге 7. Начало автоматической обработки данных.

Шаг 9. Оркестратор приложения отправляет команды запуска очередных задач на сервер платформы Everest и опрашивает состояние запущенных ранее задач.

Шаг 10. Сервер платформы Everest распределяет задачи для выполнения на свободные виртуальные машины через программы-агенты ресурсов (установленные на шаге 3). Вычисления заканчиваются, когда запущены и выполнены N задач первого типа и N(N-1)/2 задач второго типа, где N – количество файлов в информационном массиве.

Заметим, что если выполняется серия экспериментов по определению частот слов, то настройка вычислительной части приложения (шаги 1-4) выполняется однократно. Для второго

и последующего запуска вручную выполняется только шаг 5 и шаг 8. Ограничением по способу развертывания вычислительной части приложения в тестируемой реализации (шаги 1-4) является доступ к разделяемой файловой системе из виртуальных машин, развернутых на шаге 3. Однако технически возможна передача файлов непосредственно из оркестратора приложения через сервер Everest или использование клиента распределенной файловой системы IPFS [8]. Эти способы в данном исследовании не рассматривались.

4. Результаты экспериментального исследования производительности

Целью экспериментов являлась проверка работоспособности схемы приложения в целом и подтверждение возможности эффективных вычислений по данной схеме. Данную схему уместно считать эффективной, если вычисления завершаются за разумное время и применение нескольких виртуальных машин в вычислительной части приложения приводит к ускорению вычислений.

Для экспериментов использовался фрагмент массива данных размером 5,88 ГБ, собранных в работе [5]. Массив состоял из 10 файлов. Размер входных JSON-файлов был в пределах от 524 МБ до 849 МБ. В результате обработки получался массив из 10 обычных текстовых файлов общим размером 1,83 МБ, обнаружено 148885 слов (включая словоформы и неологизмы). Размер результирующих файлов изменялся в пределах от 158 КБ до 223 КБ. Результирующие файлы состояли из записей вида <слово> <суммарное число повторений слова в 10 файлах> <CR>. Записи упорядочивались в сквозном порядке по всему набору из 10 файлов (слова на ‘a’ – в первом файле, слова на ‘z’ – в последнем файле).

Была проведена серия экспериментов, в которых изменялось (от 2 до 10) количество обрабатываемых файлов и количество виртуальных машин, на которых запускалась вычислительная часть приложения. Наблюдаемое время выполнения задач первого типа во всех экспериментах варьировалось примерно от 24 до 36 секунд, а время выполнения задач второго типа – примерно от 6 до 26 секунд. Максимальное ускорение относительно варианта с одной виртуальной машиной в вычислительной части приложения равное 3,6 раз было достигнуто при обработке 10 файлов на 10 виртуальных машинах. При этом время последовательной обработки массива данных составило 980 секунд (~16 минут); с использованием 10 виртуальных машин – 270 секунд (~4,5 минуты); абсолютное уменьшение времени на обработку – примерно 11,5 минут. Таким образом, исследуемую схему организации вычислений можно считать эффективной.

5. Заключение

В работе предложена архитектура распределенного приложения обработки данных для вычислений в гибридной облачной среде, построенной на базе комбинации бесплатных и академических облачных сервисов. В вычислительном эксперименте по определению частоты слов в массиве сообщений Twitter продемонстрирована возможность ускорения вычислений, что доказывает эффективность данной архитектуры.

Практическим преимуществом рассмотренной архитектуры приложения обработки данных является организация высокопроизводительных параллельных вычислений с использованием только веб-обозревателя, что удобно для демонстраций примеров, обучения и совместной разработки.

6. Литература

[1] JupyterLab: The next-generation Jupyter frontend [Electronic resource]. – Access mode: https://bids.berkeley.edu/events/jupyterlab-next-generation-jupyter-frontend (19.11.2019).

[2] Jupyter, P. Binder 2.0-Reproducible, interactive, sharable environments for science at scale / P. Jupyter, M. Bussonnier, J. Forde // Proceedings of the 17th Python in Science Conference, 2018. – P. 113-120.

[3] Merkel, D. Docker: lightweight linux containers for consistent development and deployment // Linux Journal. – 2014. – Vol. 239. – P. 2.

[4] Sukhoroslov, O. A web-based platform for publication and distributed execution of computing applications / O. Sukhoroslov, S. Volkov, A. Afanasiev // 14th International Symposium on Parallel and Distributed Computing, 2015. – P. 175-184.
[5] Воробьев, Д.А. Автоматизированная система прогнозирования поведения валютного рынка с применением анализа эмоциональной окраски сообщений в социальных сетях / Д.А. Воробьев, В.Г. Литвинов // Перспективные информационные технологии (ПИТ), 2018. – С. 416-419.
[6] Thoman, P. A taxonomy of task-based parallel programming technologies for high-performance computing / P. Thoman, K. Dichev, T. Heller // The Journal of Supercomputing. – 2018. – Vol. 74(4). – P. 1422-1434.
[7] Vostokin, S.V. Implementing computations with dynamic task dependencies in the desktop grid environment using Everest and Templet Web / S.V. Vostokin, O.V. Sukhoroslov, I.V. Bobyleva, S.N. Popov // CEUR Workshop Proceedings. – 2018. – Vol. 2267. – P. 271-275.
[8] Benet, J. Ipfs-content addressed, versioned, p2p file system // arXiv preprint arXiv: 1407.3561, 2014.

Implementation of frequency analysis of Twitter microblogging in a hybrid cloud based on the Binder, Everest platform and the Samara University virtual desktop service

S.V. Vostokin[1], I.V. Bobyleva[1,2]

[1]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086
[2]Joint Stock Company Space Rocket Center Progress, Zemetsa str. 18, Samara, Russia, 443009

Abstract. The paper proposes the architecture of a distributed data processing application in a hybrid cloud environment. The application was studied using a distributed algorithm for determining the frequency of words in messages of English-language microblogs published on Twitter. The possibility of aggregating computing resources using many-task computing technology for data processing in hybrid cloud environments is shown. This architecture has proven to be technically simple and efficient in terms of performance.