Смирнов А.Д., Хасанов Р.Х. Разработка распределенной системы на основе модели MAPREDUCE в кластере KUBERNETES // Международный журнал информационных технологий и энергоэффективности.– 2024. –
Т. 9 № 4(42) с. 21–26

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.75

РАЗРАБОТКА РАСПРЕДЕЛЕННОЙ СИСТЕМЫ НА ОСНОВЕ МОДЕЛИ MAPREDUCE В КЛАСТЕРЕ KUBERNETES

Смирнов А.Д., [1]**Хасанов Р.Х.**
ФГБОУ ВО "МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Н.Э. БАУМАНА (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)", Москва, Россия (105005, город Москва, 2-Я Бауманская ул, д. 5 стр. 1), e-mail: [1]xasanov_ramazan@bk.ru

В данной статье рассматривается процесс разработки распределенной системы на основе модели MapReduce в кластере Kubernetes. Приведено описание основных компонент и общей архитектуры системы оркестрации контейнерными приложениями Kubernetes. Приведено описание модели распределенных вычислений MapReduce, ее основные этапы работы и использование во фреймворке для разработки распределенных вычислительных систем Apache Hadoop. Представлен процесс интеграции системы Apache Hadoop в кластер Kubernetes, что обеспечивает высокую масштабируемость, эффективность использования ресурсов и оптимизированное управление для обработки больших объемов данных.

Ключевые слова: Распределенные системы, кластер Kubernetes, модель MapReduce, фреймворк Apache Hadoop, микросервис, оркестрация.

DEVELOPMENT OF DISTRIBUTED SYSTEM BASED ON MAPREDUCE MODEL IN KUBERNETES CLUSTER

Smirnov A.D., [1]Khasanov R.H.
BAUMAN MOSCOW STATE TECHNICAL UNIVERSITY (NATIONAL RESEARCH UNIVERSITY), Moscow, Russia (105005, Moscow, 2nd Baumanskaya str., 5/1), e-mail: [1]xasanov_ramazan@bk.ru

This paper discusses the process of developing a distributed system based on the MapReduce model in a Kubernetes cluster. The description of the main components and general architecture of the Kubernetes container application orchestration system is given. The description of the MapReduce distributed computing model, its main stages of operation and its use in Apache Hadoop distributed computing development framework is given. The process of integrating Apache Hadoop into a Kubernetes cluster is presented, which provides high scalability, resource efficiency and optimized management for processing large amounts of data.

Keywords: Distributed systems, Kubernetes cluster, MapReduce model, Apache Hadoop framework, microservice, orchestration.

Введение

В настоящее время многие задачи машинного обучения и интеллектуального анализа данных требуют большие наборы данных. Примерами такой обработки являются задачи обработки текстов, поиска в Интернете, онлайн-рекламы и обработки сигналов [1]. Соответственно при этом возникает необходимость в высоких вычислительных ресурсах для такого объема данных. Поэтому важным и актуальным направлением является разработка

распределенных вычислительных систем, позволяющих применять существующие методы машинного обучения и анализа данных на больших выборках. Один из самых известных и распространенных способов проектирования распределенных систем основан на модели MapReduce.

Важным моментом является то, что при проектировании информационной инфраструктуры современного предприятия первоочередной задачей является выбор стека технологий для реализации основных производственных ИТ-процессов предприятия. К таким первоочередным задачам можно отнести задачи хранения данных, мониторинга и оповещения, а также процессы непрерывной доставки и развертывания приложений. В последние годы все больше и больше компаний решают данные задачи с использованием микросервисной архитектуры. Эти задачи объединяет необходимость использования большого количества аппаратных ресурсов, собранных в кластер, для обеспечения связи между микросервисами, а также динамического выделения ресурсов. Наиболее подходящим для решения подобных задач решением является Kubernetes.

Использование кластера Kubernetes при проектировании распределенной системы позволяет получить крайне эффективное вычислительное средство для обработки больших объемов данных.

MapReduce

Модель распределенных вычислений MapReduce была разработана и представлена компанией Google в статье [2]. Данная архитектура позволяет выполнять параллельные задачи на множестве вычислительных машин на разных наборах данных.

Основными компонентами данной модели являются две описываемые разработчиком функции – Map и Reduce. Решение конкретной задачи сводится к последовательному выполнению MapReduce-задач, в свою очередь состоящих из двух этапов – выполнения функций Map и Reduce [3]. Схема работы модели MapReduce представлена на Рисунке 1.

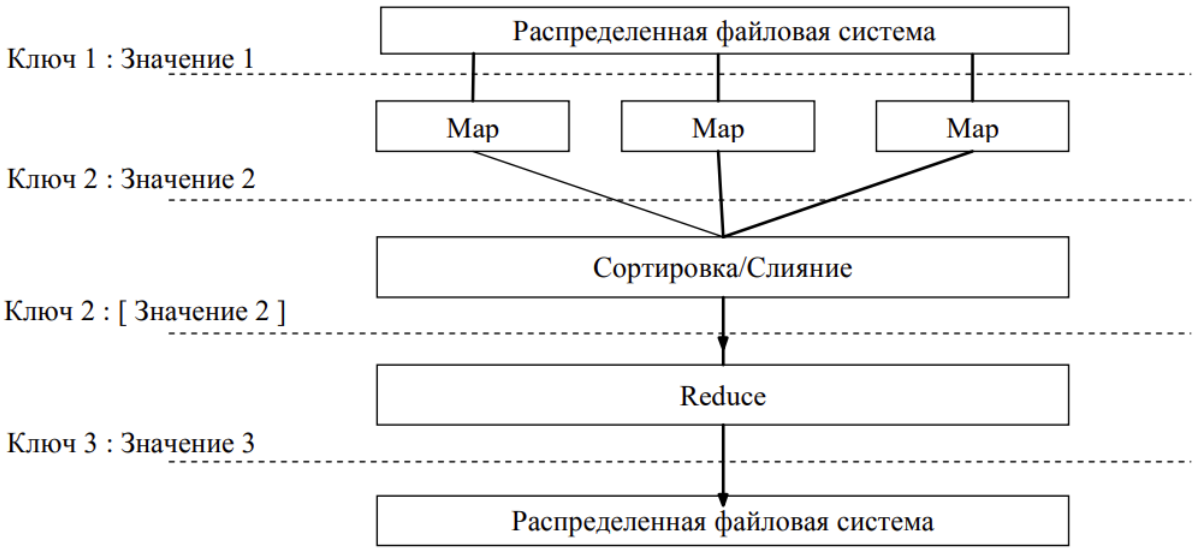

Рисунок 1 – Модель MapReduce [3]

Механизм MapReduce работает следующим образом: сначала функция Map принимает пару ключ-значение и возвращает массив промежуточных значений, также представляющих собой пары ключ-значение. Затем среда исполнения MapReduce запускает несколько инстансов функции Map одновременно на разных вычислительных узлах, автоматически распределяя данные между ними. После этого происходит барьер, где все задачи Map ожидают выполнения, и затем промежуточные данные из всех узлов Map группируются по ключу и передаются в функцию Reduce. Функция Reduce получает пару ключ-"массив значений" и выполняет агрегацию данных, в результате чего порождаются новые пары ключ-значение. Эти пары записываются в распределенную файловую систему и могут быть как окончательным результатом, так и исходными данными для следующей задачи MapReduce.

Самым распространенным способом реализации модели MapReduce является Apache Hadoop. Hadoop - это программный фреймворк для надежных, масштабируемых, параллельных и распределенных вычислений. Вместо того, чтобы полагаться на дорогостоящее оборудование и дорогостоящие системы для обработки и хранения данных, Apache Hadoop обеспечивает параллельную обработку больших данных на стандартном оборудовании. HDFS и стандарт программирования MapReduce позволяют выполнять аналитические задачи, требующие больших объемов данных, благодаря своей масштабируемой архитектуре и способности обрабатывать данные параллельным образом в многоузловых кластерах [4].

Kubernetes

Kubernetes является системой оркестрации контейнеров с открытым исходным кодом, позволяющей автоматически масштабировать и управлять ресурсами кластера [5]. Прототип Kubernetes начал разрабатываться командой инженеров компании Google в 2014 году [6], когда Google запустил проект под названием Borg. Это была система оркестрации контейнеров, предназначенная для управления и масштабирования микросервисов в облаке. Borg был построен на базе собственного механизма управления ресурсами, который не мог быть использован другими компаниями, что ограничивало его применение. В 2015 году команда Borg начала работать над проектом под названием Omega, целью которого было создание универсального механизма оркестрации ресурсов. В результате они разработали концепцию “Kubernetes”, которая стала основой для будущего проекта. Kubernetes был официально выпущен в 2016 году под руководством Cloud Native Computing Foundation (CNCF). CNCF – это некоммерческая организация, которая занимается поддержкой и продвижением проектов, связанных с облачными и контейнерными технологиями. После этого Kubernetes стал одним из самых популярных инструментов для работы с контейнерами и получил широкое распространение в индустрии.

Структура кластера Kubernetes

Кластер обычно состоит из нескольких узлов (node), которые по факту являются физическими или виртуальными вычислительными устройствами. В каждом кластере независимо от размера обязательно будет хотя бы один управляющий узел (master), а также произвольное количество рабочих узлов (worker node) [7].

Такое разделение используется для большей надежности всей системы, несмотря на возможность развертывания рабочей нагрузки на любом узле кластера крайне рекомендуется использовать только рабочие узлы. На Рисунке 2 изображен пример кластера, состоящего из одного управляющего узла и двух рабочих узлов.

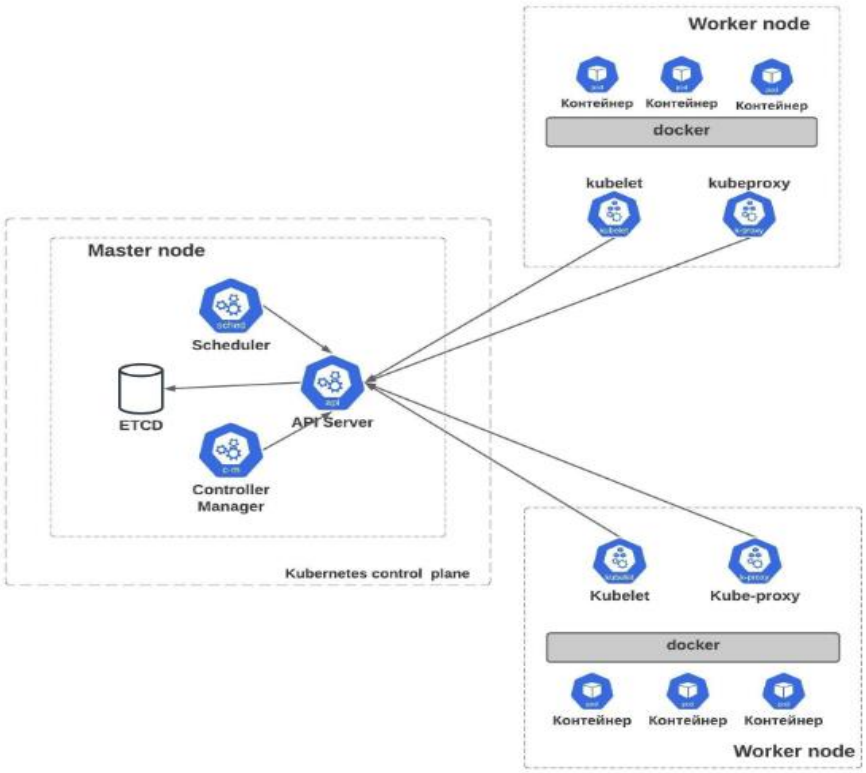

Рисунок 2 – Пример кластера Kubernetes

Как видно на Рисунке 2, на разных узлах используются разные компоненты, про каждый из них будет рассказано далее.

Управляющий узел состоит из:

- API сервер (kube-apiserver) – обрабатывает все запросы на манипуляции с ресурсами (создание, изменение, удаление);
- Контроллер менеджер – следит за состоянием ресурсов в кластере и применяют необходимые изменения. Например, если какой-то ресурс стал недоступен, контроллер может создать его заново;
- etcd – хранит состояния кластера (данные о ресурсах, их состоянии и т.д.).

Стоит отметить, что в крупных кластерах зачастую используется несколько управляющих узлов, которые распределяют административную нагрузку между собой, а также обеспечивают дополнительную отказоустойчивость.

Как было сказано выше, рабочие узлы используются для разворачивания рабочей нагрузки, такой как микросервисы, базы данных или простые контейнеры. Однако для работы в кластере необходимо иметь обязательный набор административных процессов, в основном для связи с управляющим узлом.

Рабочий узел состоит из:

- kubelet – это основной компонент рабочего узла, который отвечает за управление и запуск контейнеров. Он взаимодействует с API сервером управляющего узла и выполняет команды, которые поступают от контроллера;
- docker – это основная часть рабочего узла, которая отвечает за запуск и управление контейнерами. Он взаимодействует с демоном kubelet и выполняет его команды;
- kubeproxy – это прокси-сервер, который работает на управляющем узле Kubernetes и предоставляет сервисы для коммуникации между контейнерами в кластере. Он обеспечивает маршрутизацию запросов между контейнерами, скрывает информацию о внутренних адресах узлов и балансирует нагрузку на кластер.

Создание инфраструктуры для Hadoop на Kubernetes

Для развертывания компонентов Hadoop поверх Kubernetes необходимо решить несколько важных архитектурных задач [8]. В первую очередь нужно выбрать тип кластера Kubernetes – использовать облачное решение от какой-либо компании или же использовать собственные вычислительные мощности. Также нужно убедиться, что сетевая связь между узлами подходит под необходимые для Hadoop стандарты.

Вторым важным шагом является разбиение компонентов Hadoop на отдельные микросервисы, например HDFS, YARN, MapReduce. Стоит отметить, что для работы файловой системы HDFS необходимо использовать специальные типы объектов внутри Kubernetes – Persistent Volumes и Persistent Volumes Claims. Для связи микросервисов друг с другом нужно использовать объект типа Service, а также особый тип сервисов для связи со внешним миром – NodePort или LoadBalancer.

Третьим этапом является настройка мониторинга состояния кластера. Для этого лучше всего использовать связку инструментов Grafana и Prometheus, что позволит снимать необходимые метрики напрямую с кластера и предоставлять их в удобном формате.

Заключение

В данной статье описан способ разработки распределенной системы на основе модели MapReduce в кластере Kubernetes.

Дано описание использования Kubernetes в качестве инструмента при проектировании микросервисной архитектуры современного предприятия. Показаны основные ее компоненты и способы их взаимодействия между ними.

Представлено описание модели распределенных вычислений MapReduce и основные этапы работы ее в распределенной системе.

Дается подход для интеграции Apache Hadoop в кластер Kubernetes для обеспечения высокой масштабируемости и эффективности вычислений для больших наборов данных.

Изложенный выше подход для проектирования кластера Hadoop в информационной структуре предприятия на основе Kubernetes является одним из наиболее удобных для реализации, а также поддержки. Реализация Hadoop таким способом предусматривает высокую надежность и удобные инструменты для масштабирования кластера.

Список литературы

1. Дурнов Р. В. Модель распределенной вычислительной сети //Известия Тульского государственного университета. Технические науки. – 2022. – №. 9. – С. 151-153.
2. Dean, J. and Ghemawat, S. (2004). MapReduce : Simplifed Data Processing on Large Clusters. In OSDI' 04, San Francisco.
3. Митяков А. В., Татаринов Ю. С. Подходы к эффективному исполнению итеративных алгоритмов на модели MapReduce //Известия ЛЭТИ. – 2014. – №. 2. – С. 30-41.
4. Ghazi M. R., Gangodkar D. Hadoop, MapReduce and HDFS: a developers perspective //Procedia Computer Science. – 2015. – Т. 48. – С. 45-50.
5. Luksa M. Kubernetes in action. – Simon and Schuster, 2017. – С. 38-45.
6. The History of Kubernetes & the Community Behind It. URL: https://web.archive.org/web/20220227112321/https://kubernetes.io/blog/2018/07/20/the-history-of-kubernetes-the-community-behind-it/ (date of application: 03.03.2024).
7. Sayfan G. Mastering kubernetes. – Packt Publishing Ltd, 2017. – С. 5-10.
8. Zhang X. et al. Zeus: Improving resource efficiency via workload colocation for massive kubernetes clusters //IEEE Access. – 2021. – Т. 9. – С. 105192-105204.

References

1. Durnov R. V. Model of a distributed computing network //Proceedings of Tula State University. Technical sciences. - 2022. – No. 9. – pp. 151-153.
2. Dean, J. and Ghemawat, S. (2004). MapReduce : Simplifed Data Processing on Large Clusters. In OSDI' 04, San Francisco.
3. Mityakov A.V., Tatarinov Yu. S. Approaches to the effective execution of iterative algorithms based on the MapReduce model //News of LETI. – 2014. – №. 2. – pp. 30-41.
4. Ghazi M. R., Gangodkar D. Hadoop, MapReduce and HDFS: a developers perspective //Procedural Computer Science. – 2015. – Vol. 48. – pp. 45-50.
5. Luksa M. Kubernetes in action. – Simon and Schuster, 2017. – pp. 38-45.
6. The History of Kubernetes & the Community Behind It. URL: https://web.archive.org/web/20220227112321/https://kubernetes.io/blog/2018/07/20/the-history-of-kubernetes-the-community-behind-it / (date of application: 03.03.2024).
7. Sayfan G. Mastering kubernetes. – Packt Publishing Ltd, 2017. – pp. 5-10.
8. Zhang X. et al. Zeus: Improving resource efficiency via workload colocation for massive kubernetes clusters //IEEE Access. – 2021. – Vol. 9. – pp. 105192-105204.