УДК 004.35

Стрелец Андрей Иванович
Магистр кафедры «Компьютерные системы и технологии»,
Национальный исследовательский ядерный университет «МИФИ»
Россия, г. Москва
Орлов Александр Петрович
Магистр кафедры «Компьютерные системы и технологии»,
Национальный исследовательский ядерный университет «МИФИ»
Россия, г. Москва
Храпов Александр Сергеевич
Магистр кафедры «Компьютерные системы и технологии»,
Национальный исследовательский ядерный университет «МИФИ»
Атавина Анастасия Владиславовна
Магистр кафедры «Финансовый мониторинг»,
Национальный исследовательский ядерный университет «МИФИ»
DOI: 10.24411/2520-6990-2019-10422

ИСПОЛЬЗОВАНИЕ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПРЕДСКАЗАНИЯ УРОВНЯ ЗАГРУЗКИ РЕСУРСОВ СИСТЕМЫ

Strelets Andrey Ivanovich
Master degree Department of Computer Systems and Technologies,
National Research Nuclear University MEPhI
Moscow, Russia
Orlov Aleksandr Petrovich
Master degree Department of Computer Systems and Technologies,
National Research Nuclear University MEPhI
Moscow, Russia
Hrapov Aleksandr Sergeevich
Master degree Department of Computer Systems and Technologies,
National Research Nuclear University MEPhI
Atavina Anastasia Vladislavovna
Master degree Department of Financial Monitoring,
National Research Nuclear University MEPhI

USING THE ML METHODS FOR PREDICATING SYSTEM RESOURCES LOADING

Аннотация
В данной статье рассмотрены методы машинного обучения, предназначенные для предсказания уровня загрузки ресурсов системы. Рассмотренные находят широкое применение в области распределённых вычислений.
Abstract
This article is about methods for predicating system resources loading. These methods widely used in distributed computing

Ключевые слова: *машинное обучение, микросервисы, виртуализация, высокопроизводительные вычисления.*
Key words*: machine learning, microservices, virtualization, high performance computing.*

Введение

Все больше компаний обращаются к облачным технологиям. В облаках разворачиваются многомодульные системы, настройка которых, в основном, требует большого количества времени. Также разработчики могут столкнуться с проблемами отслеживания зависимостей, масштабирования приложения и обновлений. Большинство систем на основе контейнеризации чаще всего работают в кластере. Стандартные алгоритмы оркестрации Docker Swarm, Google Kubernetes не учитывают возможные изменения, которые могут происходить внутри контейнера, что может приводить к неравномерному распределению нагрузки по узлам. Для решения поставленных задач разрабатываются алгоритмы, работающие поверх стандартной реализации. Контейнеризация позволяет масштабировать систему с помощью добавления новых копий сервиса. Балансировщик нагрузки позволяет разделить запросы к разным копиям контейнера.

Архитектура сети

Подход к построению архитектуры сети сконцентрирован на распределение сервисов на узлах сети на основе алгоритмов ML после их запуска для уменьшения задержки, повышение пропускной способности системы в целом. Это является необходимостью из-за динамического поведения контейнера во время работы.

Архитектура разработанного решения состоит из следующих компонентов:

1) Менеджера запуска контейнеров на основе Docker Swarm;

2) Менеджера распределения контейнеров на основе ML;

3) Менеджера мониторинга узлов на основе сбора метрик и kpi узлов и контейнеров.

Сравнение виртуализации на гипервизорах и контейнерах

Рассмотрим два наиболее используемых подхода к виртуализации сетевых функций, которые, к тому же имеют реализации в виде свободного ПО.

Подход 1 — это виртуализация на основе развёртывания виртуальных машин с использованием гипервизора как управляющего элемента.

Подход 2 — это виртуализация на основе контейнеров. Гипервизор в данном случае отсутствует, и управление ресурсами осуществляет сама ОС.

Практическая часть данной статьи разрабатывалась с использованием второго подхода, но концептуально все решения применимы и для первого. Предлагаемая схема инфраструктуры абстрагирагирована от конкретного программного способа виртуализации сетевых функций.

Виртуализация с использованием гипервизоров.

Наиболее наглядное представление можно получить на примере схемы виртуализации, используемой в KVM (рисунок 1).

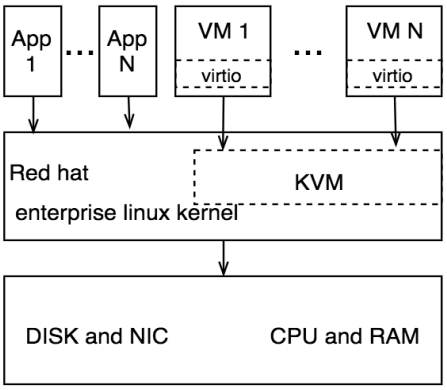

Рисунок 1. Схема виртуализации на гипервизорах

Оценки потерь производительности (overhead) контейнеризации приводятся в различных работах, и они находятся в районе 0.1-1% в зависимости от типа нагрузки и методики тестирования.

Контейнеризация: Docker, OpenVZ, LXC и т. д. Есть API, поэтому можно писать скрипты для развёртывания. Аппаратная поддержка: аналогично Linux. На гипервизорах: KVM, Xen и т. д.. Есть API, поэтому можно писать скрипты для развёртывания. Аппаратная поддержка: аналогично Linux, обязательна поддержка виртуализации на уровне аппаратного обеспечения. Предполагается выделение сервисов, занимающихся наблюдением и обслуживанием работы вычислительных узлов на отдельный узел. Предполагается выполнение двух основных задач данным узлом: мониторинг текущего состояния узлов, а также перераспределение инстансов по различным узлам в зависимости от текущей нагрузки.

Дополнительной решаемой задачей является визуализация различных текущих характеристик системы. Для более удобного наблюдения за текущим состоянием вычислительной инфраструктуры. Для получения характеристик с узлов используется Telegraf - универсальный агент для для сбора, обработки и агрегирования всех необходимых метрик, таких как нагрузка сети, памяти, процессора. Для более полного понимания состояния системы Telegraf агент расположен не только на каждом узле, но и в каждом instance.

Собранные метрики на каждом узле собираются системой мониторинга Prometheus, сохраняющей все собранные метрики в собственную локальную NoSQL базу данных, в которой каждая метрика имеет привязку к моменту времени, в который она была измерена, а также указание агента, с которого она была получена.

Сервис поставки метрик по заданному интервалу времени считывает метрики в агрегированном состоянии из Prometheus и персистит в Cassandra. Данное решение позволит упростить процесс анализа метрик в дальнейшем, за счет их предварительной обработки.

Из-за этого разница в производительности при использовании гипервизоров относительно обычной ОС достигает 5-15% в зависимости от типа используемого ПО и конфигурации системы. Есть несколько реализаций данного подхода: KVM (в т.ч. OVirt), Xen.

Заключение

Таким образом, в настоящей статье исследован новый способ оркестрации процессов управления контейнерами, основанный на анализе состояния узлов планировщиком, реализованным с использованием алгоритмов машинного обучения. Использование следующего планировщика позволяет повысить утилизацию и уменьшить время доступа к контейнерам. В качестве дальнейших исследований предлагается увеличить количество параметров, по которым будет определяться действия deploy manager.

Список литературы

1. S. M. Metev and V. P. Veiko, Laser Assisted Microtechnology, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

2. J. Breckling, Ed., The Analysis of Directional Time Series: Applications to Wind Speed and Direction, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.

3. S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, “A novel ultrathin elevated channel low-temperature poly-Si TFT,” IEEE Electron Device Lett., vol. 20, pp. 569–571, Nov. 1999.

4. O. Sallou and C. Monjeaud., Go-Docker, A batch scheduling system with Docker containers. IEEE International Conference of Cluster Computing, pp. 514-515, 2015