Добрый день, уважаемые члены аттестационной комиссии. Тема моей ВКР – **«Разработка системы автоматического развертывания микросервисных приложений в облачной инфраструктуре на основе алгоритма комбинаторной оптимизации».**

Слайд 2. Почему начал исследование. Актуальность. Новизна

С каждым годом на предприятиях внедряется все больше проектов автоматизации, которые требуют больше вычислительных мощностей.

Рассмотрим пирамиду автоматизации, представленную на слайде 2. Выделяют 2 уровня систем автоматизации – уровень автоматизированных систем управления предприятием и уровень автоматизированных систем управления технологическим процессом. И если на верхнем уровне применяются такие средства управления ресурсами, как контейнеризация и оркестрация, а системы располагаются в центрах обработки данных то на АСУ ТП такие инструменты не применяются из тех соображений, что

1. Цеховые серверные мощности ограничены.
2. Отдалены друг от друга, т.е. нет единого ЦОД

Поэтому возникает требование как можно более эффективно использовать данные ограниченные ресурсы.

Слайд 3

Исследование на данную тему является актуальным, т.к. на данный момент нет программных решений управления использованием ресурсов серверов на уровне АСУ ТП. Предприятие заинтересовано в данном исследовании, т.к. стремится максимально снизить издержки на покупку новых серверов.

Соответственно, требуется собственное решение.

Слайд 4.

Цель: минимизация количества задействованных виртуальных и физических серверов при распределении программных компонентов на серверной инфраструктуре.

Объект: инфраструктура развертывания программного обеспечения на физических и виртуальных серверах центра обработки данных АО «Северсталь-инфоком».

Предмет: методы оптимизации автоматического развертывания микросервисных приложений на серверах.

Слайд 5.

1. Проанализировать существующие решения по развертыванию ПО на серверных фермах. Провести патентный обзор.
2. Рассмотреть существующие алгоритмы оптимизации, разработать математическую модель системы, адаптировать алгоритмы и реализовать симулятор системы для выбора наиболее подходящего алгоритма оптимизации.
3. Реализовать автоматического развертывания ПО, провести экспериментальные исследования работы системы.
4. Выполнить технико-экономическое обоснование проекта.

Слайд 6.

Как уже говорилось ранее все больше применяется технология контейнеризации – на схеме показаны изолированные в контейнеры сервисы. Самым распространенным инструментом контейнеризации является Docker, а средством масштабирования и координации в условиях серверного кластера – Kubernetes.

Патентный обзор показал, что существуют решения для управления ресурсами сервера, но все они сводятся к использованию либо статических правил распределения ресурсов, либо не ставят цель минимизировать кол-во занятых серверов.

Слайд 8.

На слайде 8 представлена схема, на которой показаны инфраструктура до и после мероприятий по минимизации занимаемого пространства.

Слайд 9.

Задача минимизации количества используемых серверов в терминах комбинаторной оптимизации формулируется следующим образом: «имеется коллекция программных компонентов (далее будем называть их сервисами) и коллекция виртуальных машин (далее будем называть их серверами). Необходимо распределить все сервисы на минимальное количество серверов предполагая, что один сервис может располагаться только на одном сервере».

Слайд 10.

Согласно мат модели необходимо, чтобы кол-во серверов было наименьшим возможным, при условии соблюдения следующих ограничений

Мат модель:

где

Критерием оптимальности в данном случае будем считать число серверов . Решение считается тем более оптимальным, чем ниже значение .

Слайд 11.

На данном слайде приведено определение размера сервиса и вместимости сервера.

Слайд 12.

Вышеописанная задача сводится к задаче об упаковке в контейнеры, которая, в свою очередь, является NP-полной задачей. Трудоёмкость таких задач экспоненциально растёт с увеличением объема данных. Для NP-полных вычислительных задач не существует алгоритма решения, способного вернуть результат за полиномиальное время, поэтому рассмотрим ряд эвристических алгоритмов, позволяющих найти решение, находящееся близко к оптимальному. В качестве таких алгоритмов были выбраны 3 категории:

1. Простейшие алгоритмы NF, FF, BF, FFD, BFD. Они похожи, первые 3 это online алгоритмы, последние 2 нет. Был выбран последний, как алгоритм с полным доступом ко всем контейнерам и наиболее подходящий по смыслу к задаче.
2. Алгоритм имитации отжига. Алгоритм оптимизации, использующий упорядоченный случайный поиск на основе аналогии с процессом образования в веществе кристаллической структуры с минимальной энергией при охлаждении.
3. Генетический алгоритм. Цель генетического алгоритма при решении задачи оптимизации состоит в том, чтобы найти приближённое решение, близкое, но не гарантированно оптимальное решение.

Слайд 13.

Для того, чтобы выбрать наиболее подходящий для решения поставленной задачи алгоритм, был реализован симулятор инфраструктурной среды, состоящий из сервиса, возвращающего среду для оптимизации и сервиса, принимающего среду на вход и оптимизирующего ее с применением вышеперечисленных алгоритмов.

Слайд 14.

На слайде представлена блок схема алгоритма наилучший подходящий с упорядочиванием

Слайд 15.

На слайде представлена блок схема алгоритма имитации отжига

Слайд 16

И функция расчета энергии атома. Понижение температуры осуществляется по закону Больцмановского отжига

Слайд 17.

На слайде представлена блок схема генетического алгоритма

Слайд 18

И его функция приспособленности. Метод селекции – усечением, тип скрещивания – многоточечное, вероятность применения оператора мутации – 0,3

Слайд 19

Схема данного скрещивания показана на слайде.

Слайд 20

Сравнение производилось по выбранным экспертным путем критериям:

1. Время работы алгоритма
2. Вероятность ошибочного решения
3. Заполняемость серверов

Получены следующие усредненные данные по результатам 50 запусков на тестовой выборке.

По данным результатам можно видеть, что время работы алгоритма находится в пределах установленного (установлено 10 минут), вероятность ошибочного решения у генетического алгоритма одна из наименьших, а минимальное число использованных серверов также у генетического.

Слайд 21

Рассмотрим резулттаты заполянемости серверов на графиках. На слайде показан график зависимость процентного соотношения свободных серверов при увеличении устанавливаемых на них сервисов . Соответственно, для алгоритма «наилучший подходящий с упорядочиванием», алгоритм имитации отжига и генетического алгоритма.

Слайд 24

Результирующий график показывает, что при применении генетического алгоритма при увеличении кол-ва устанавливаемых сервисов инфраструктура содержит меньше заполненных серверов.

Слайд 25.

Подведем итоги:

В результате был выбран генетический алгоритм, т.к. его время работы не превышает порог, вероятность ошибки одна из наименьших и кол-во задействованных серверов при увеличении числа сервисов также наименьшее

Слайд 26.

На основе выбранного алгоритма реализовано клиент-серверное ПО, позволяющее в автоматическом режиме разворачивать программные компоненты на серверах и оптимизировать их загруженность. Использованные технологии: backend: .NET Core + EFCore. БД: postgres.

Frontend: wpf + prizm + catel + adonisUI.

Монолитный шлюз для API: Ocelot.

Брокер сообщений: RabbitMQ

Слайд 27

Как показали эксперименты по реструктуризации существующей инфраструктуры с применением разработанного программного обеспечения, возможно высвободить ресурсы на существующей инфраструктуре до 13%. Проект внедрения САУП рассчитан на 6 лет, в течение которых закупается по 8 новых серверов в год на сумму 1 128 000 рублей. При оптимизации существующей инфраструктуры мы освобождаем 2 физических сервера в первый год и по одному в последующие. Соответственно, экономия средств в первый год составит 517 100,6 рублей, а в последующие 274 143,2 рубля. Проект оправдывает затраченные на него средства менее, чем за полгода и позволяет сэкономить более 270 тысяч рублей в год.

Слайд 28

Итоги

На видео: Для развертывания приложений следует:

1. Создаем компоненты в модуле компонентов
2. В модуле хостов создаем целевую машину
3. Создаем среду развертывания, где указываем какие компоненты развернуть на каких серверах. Добавляем все сервисы только на первый сервер.
4. В модуле оптимизации сервисы распределены на сервера.
5. Разворачиваем данные компоненты на нашей инфраструктуре