

## Задание 2

### Введение

### Основная часть

1. Критерий оценки эффективности размещения машин на кластере
2. Описание алгоритма размещения машин в кластере

Задача называется *bin packing problem* - задача оптимизации, в которой предметы разного размера должны быть упакованы в конечное число корзин или контейнеров, каждый из которых имеет фиксированную заданную вместимость, таким образом, чтобы минимизирует количество используемых контейнеров.

Для решения может быть использован пакет *binpacking* или другая реализация решения. Простое решение может быть описано так:

- Получаем на вход характеристики ВМ
- Пытаемся вместить в свободные ресурсы на первом сервере
- Если есть доступные ресурсы, размещаем на этом сервере
- Если нет, пытаемся вместить в свободные ресурсы на втором сервере
- И так далее для каждой размещаемой машины

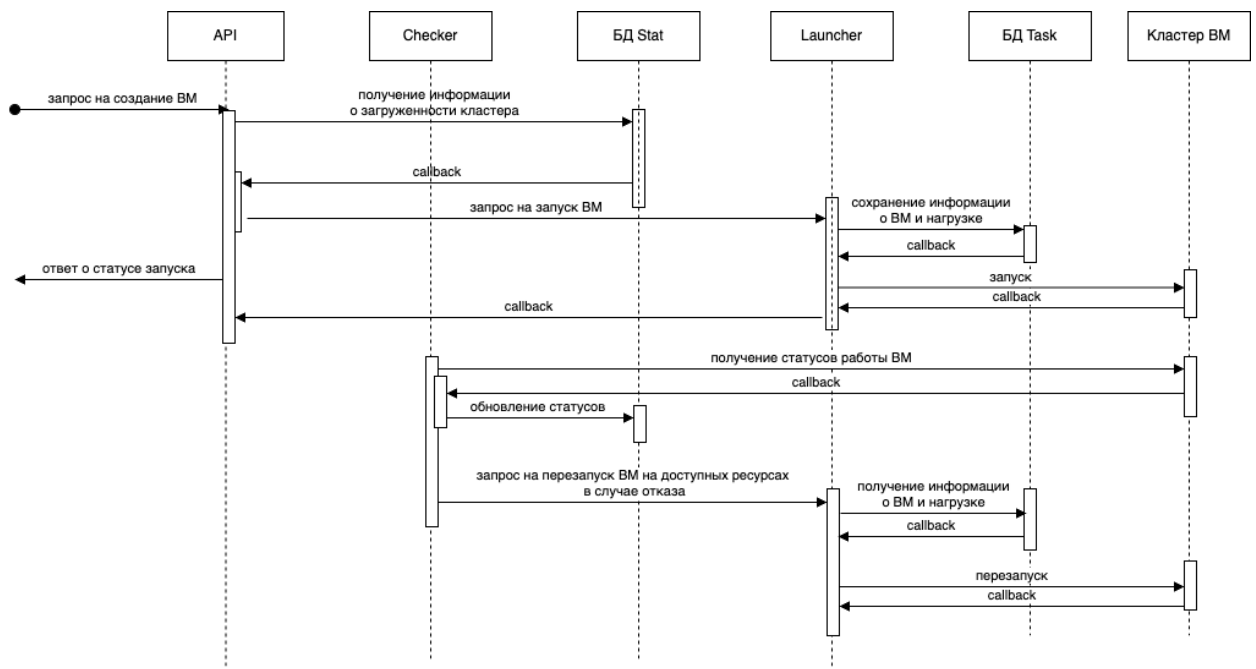
3. Модель процессов

UML-модель процессов в виде диаграммы последовательностей представлена на рисунке ниже. Предлагается отдельно хранить информацию о загрузженности кластере, статусе потребления ресурсов и работы ВМ (БД Stat) и заданиях на запуск ВМ и задании (полезной нагрузке) (БД Task).

Таким образом, при получении запроса API сможет оперативно получить актуальную информацию и принять решение о возможности запуска ВМ, а при отказе оборудования или падении скрипта полезной нагрузки будет возможен перезапуск на доступных ресурсах моментально или при освобождении

ресурсов. В качестве БД Stat предлагается использовать NoSQL БД (Mongo / Redis), так как запросы будут более простыми и легковесными, что обеспечит быстрое получение и обновление статусов VM и нагрузки. Для хранения информации о задании и характеристиках VM, которую необходимо запустить также может быть использована нереляционная БД, так как формат задания (команда на запуск, путь до скрипта и т.д.) может иметь разный формат, и будет удобнее хранить json с информацией о запуске, чем приводить к строгому типу данных.

Для повышения отказоустойчивости следует использовать средства резервирования и сторонние СХД, чтобы в случае падения VM результаты выполнения нагрузки не были утеряны.



#### 4. Прототип API

Исходный код размещён в публичном репозитории:

Реализованы схемы данных Server, VM, Replacement для хранения информации о сервере и его загрузке, виртуальной машине с запросом на ресурсы по памяти и задании и размещении VM на серверах.

Реализована валидация типов данных, уникальности переданного id для VM, значения статуса работы сервера (up or down) и значения size согласно

условию при помощи pydantic. API реализована с использованием фреймворка fastapi, swagger доступен по адресу <имя хоста>:9024/docs.

Реализован POST-запрос на создание VM с передачей параметров согласно условию.

Создан Dockerfile и docker-compose для развёртывания API в контейнере. В качестве БД для тестов используется SQLite, работа с БД реализована посредством sqlalchemy.

## 5. Параметры сбора статистики и оценки эффективности использования серверов в кластере, создание сценариев масштабирования (добавления и сокращения серверов) кластера

Для сбора статистики может быть использован как отдельный самописный сервис (на диаграмме - Checker), так и средства логирования (ELK-стек) и мониторинга (Prometheus). Можно использовать как минимум следующие метрики:

- Максимальное использование CPU за всё время работы на каждые VM, сервер, кластер
- Максимальное использование memory за всё время работы на каждые VM, сервер, кластер
- Количество одновременно запущенных VM в единицу времени
- Количество успешно завершённых заданий в единицу времени
- Количество неуспешно завершённых заданий в единицу времени
- Количество отказов кластера в единицу времени
- Количество случаев с превышением доступных ресурсов в единицу времени
- Время ответа API
- Время создания VM

Сценарием масштабирования предлагается горизонтальное масштабирование, так как по условию задания сервера имеют одинаковые характеристики. Таким образом, можно настроить реплицирование

подключения серверов в кластер при достижении нагрузки более 80% от доступной и отключения их при снижении нагрузки на кластер.

Также рекомендуется использовать геораспределённые ЦОД для избежания отказов при выходе из строя ЦОД. В этом случае кластер продолжит работать за счёт доступных ресурсов из другого ЦОД.

## **Заключение**