one-cloud
ОС уровня датацентра в ОК

Железо в Одноклассниках

ЦОД

Стоек

Серверов

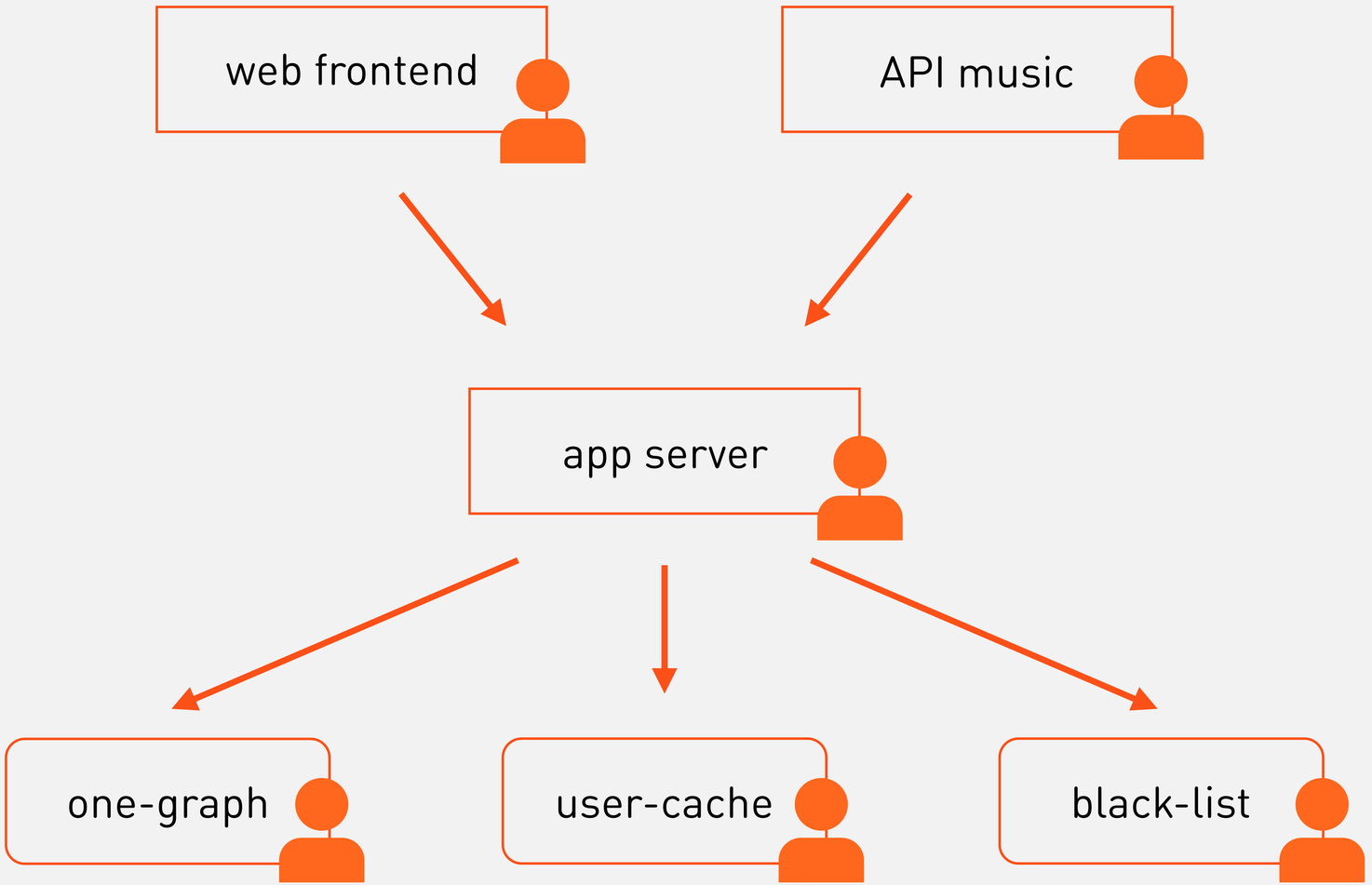
инженеры

админы

сетевики

разработчики
(функциональные команды)

Работает так



(микро) сервисы

Железо в Одноклассниках

- 1 сервер = 1 задача
  - Это просто:
    - В массовом управлении
    - В диагностике и мониторинге
- 1 сервис = X серверов
  - Просто распределяется ресурс 
- Специализированные конфигурации
  - Это эффективно

✗ Самое дорогое - это сервера

✓ Самое дорогое - место в ДЦ

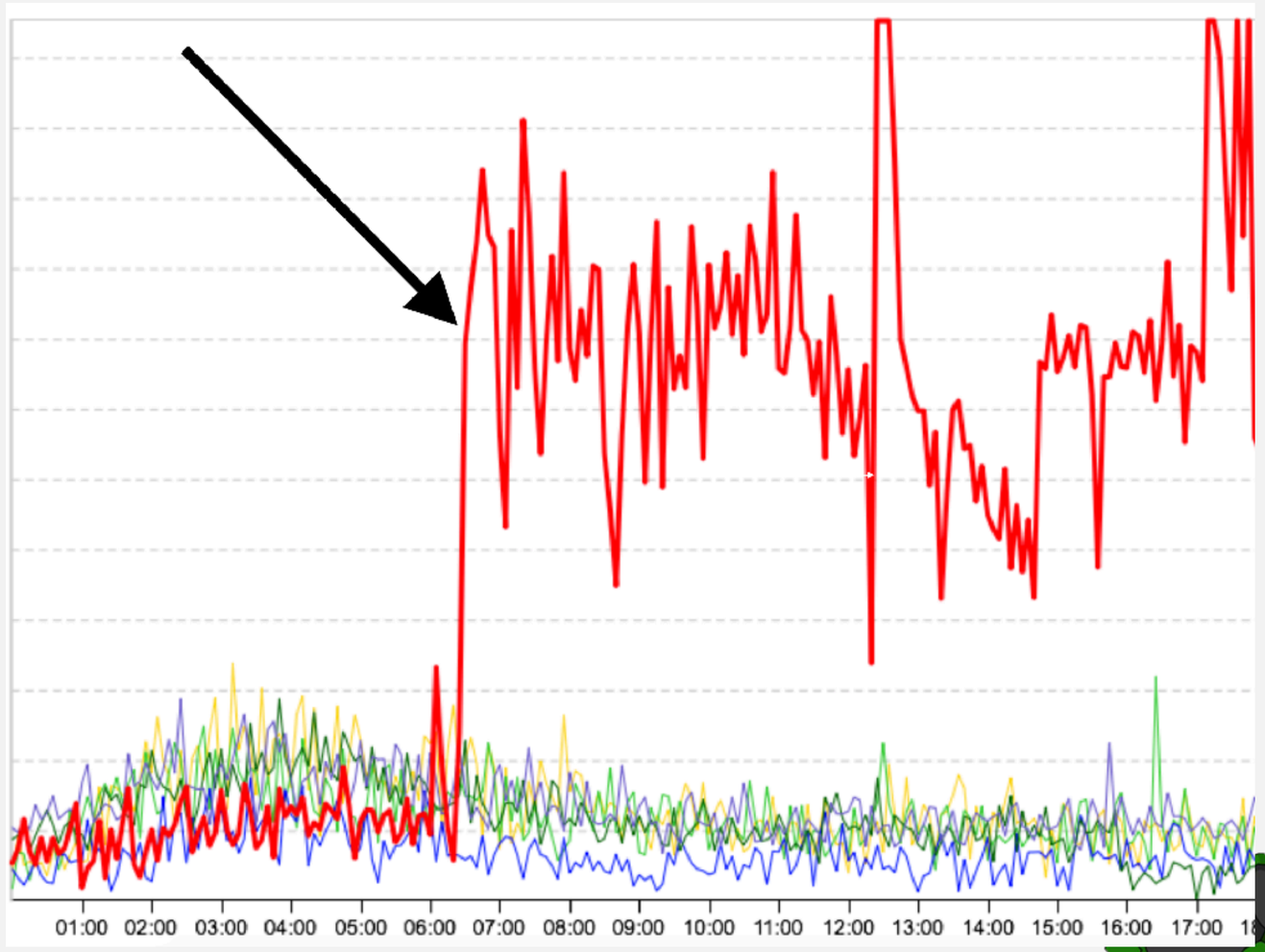
Утилизация стоек - 11 %

↓

Нужно повышать утилизацию

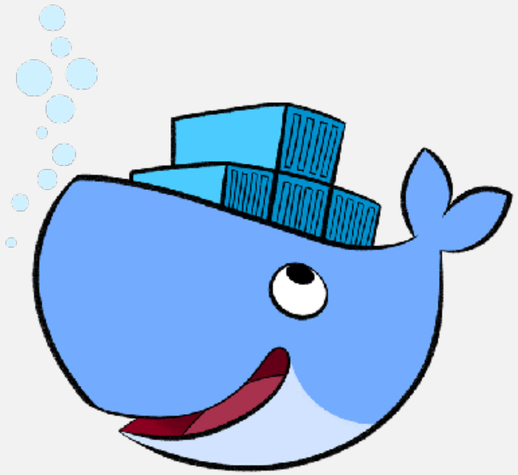
Повышаем %% по простому

- 1 сервер = X задач
- Все сложно:
  - Конфигурация
  - Диагностика
- Чьи ресурсы ?
- Нет изоляции



Часть проблем : Docker

- Изоляция
- Образы Файловой Системы
  - Многослойность
  - Таги
  - Регистр
- АПИ и унификация
  - автоматизируемость



Другая часть проблем

- Размещение контейнеров на сервера
  - Упаковка по ресурсам: ЦП, память, трафик
  - Может быть сложным: стойки
  - На 8,000 вручную - не вариант
- Выделение ресурсов на проект
  - Больше самостоятельности
  - Сохраняя контроль

Нужен управляющий слой

Три квадратика

one-cloud masters

one-cloud miniond

docker daemon

Linux Kernel

Распределение ресурсов

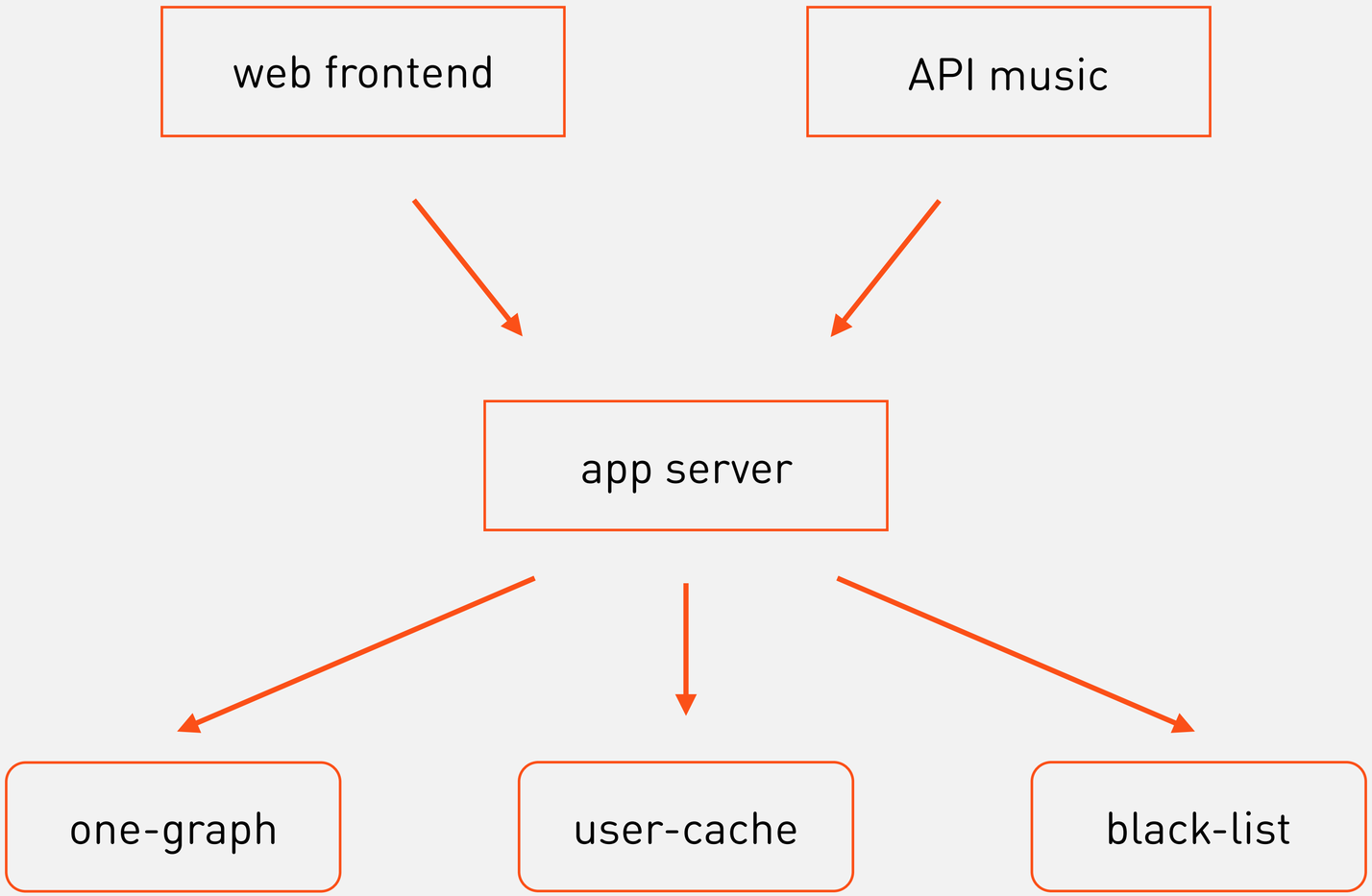
Распределение ресурсов

- Ресурс это
  - ЦПУ
  - Память
  - Трафик
  - Диски (место, тип, iops)
- Ресурсы конечны
  - Нужно квотировать **Q**

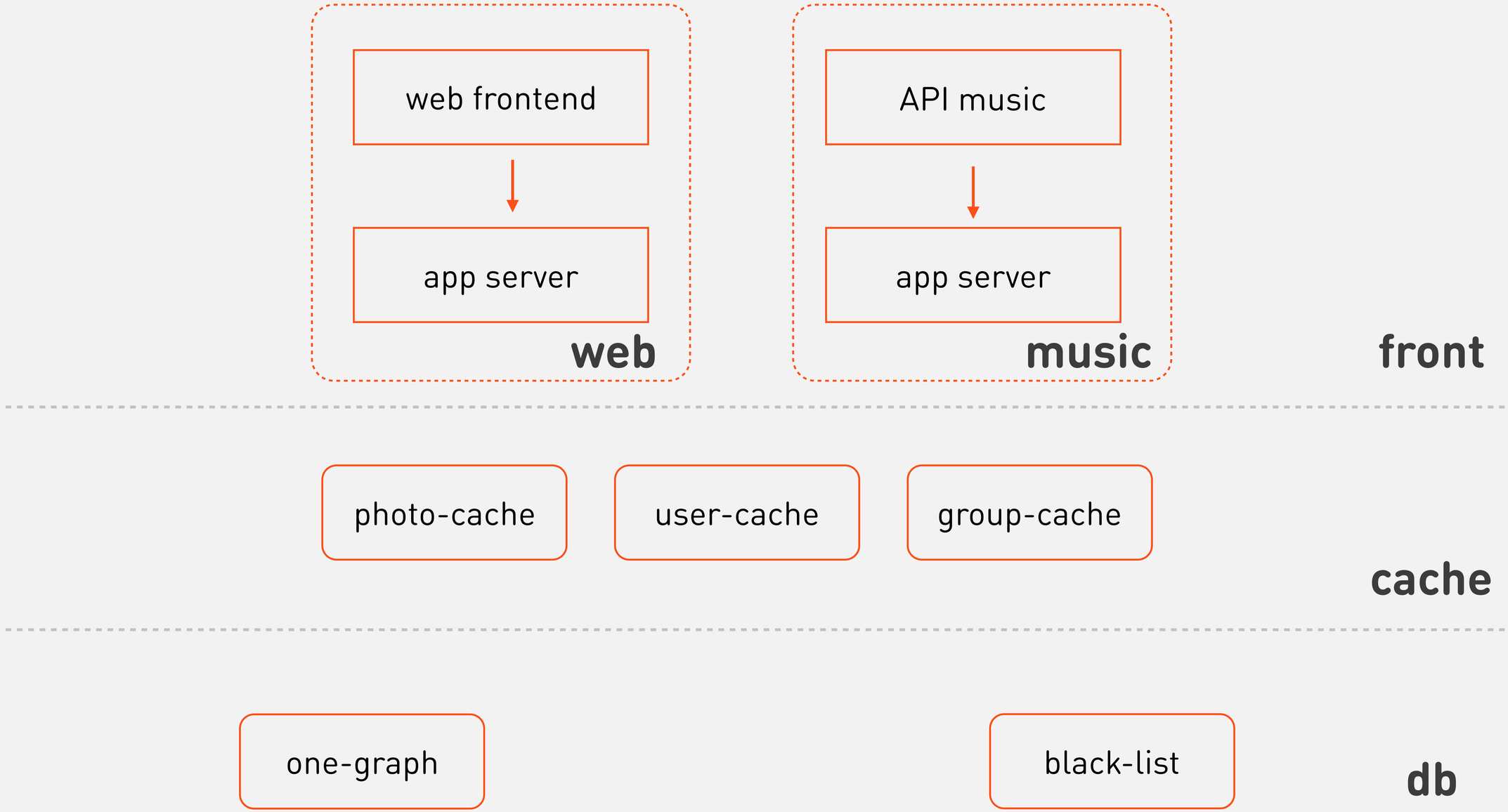
cpu = 1500
mem = 1.5 T
lan_in = 32 gbit
lan_out = 32 gbit
hdd = 20x15T

Но на что поставить квоту ?

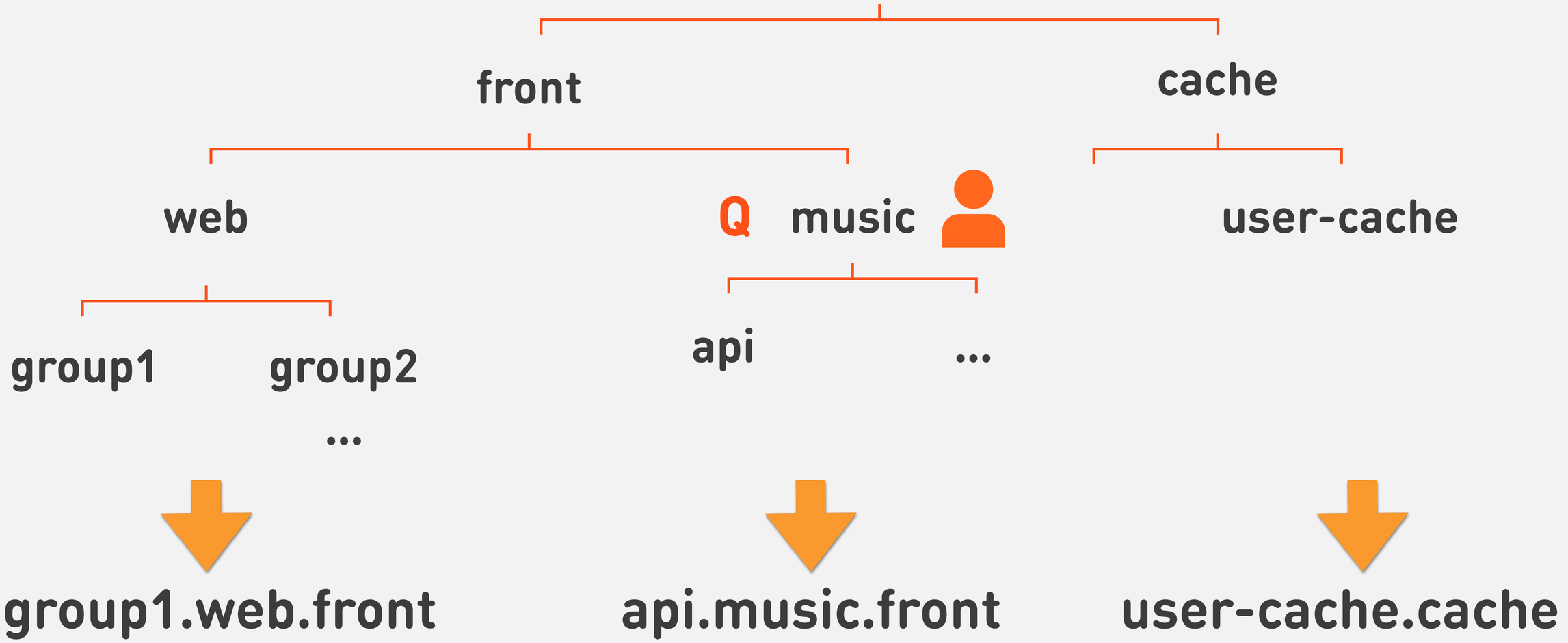
Работает так



(микро) сервисы


web frontend
app server
web
API music
app server
music
front
photo-cache
user-cache
group-cache
cache
one-graph
black-list
db

Иерархия.


front
cache
web
Q music
user-cache
group1
group2
...
api
...
group1.web.front
api.music.front
user-cache.cache

Иерархическая очередь

- Имя
- Квота на ресурсы
- Права пользователей
  - Отправка сервиса для **devops**
  - Административные права для **opsdev**
- Отправляем сервисы
  - Выполняются в пределах квоты

group1.web.front

Q (cpu = 1500 mem = 1.5 T)

Submit, Admin

Сервисы

- Сервис имеет:
  - Полное имя
  - Манифест
    (ресурсы, конфигурация, репликация, отказы)

ok-web.group1.web.front
ok-app.group1.web.front

- И экземпляры

1.ok-web.group1.web.front
2.ok-web.group1.web.front
...
42.ok-web.group1.web.front

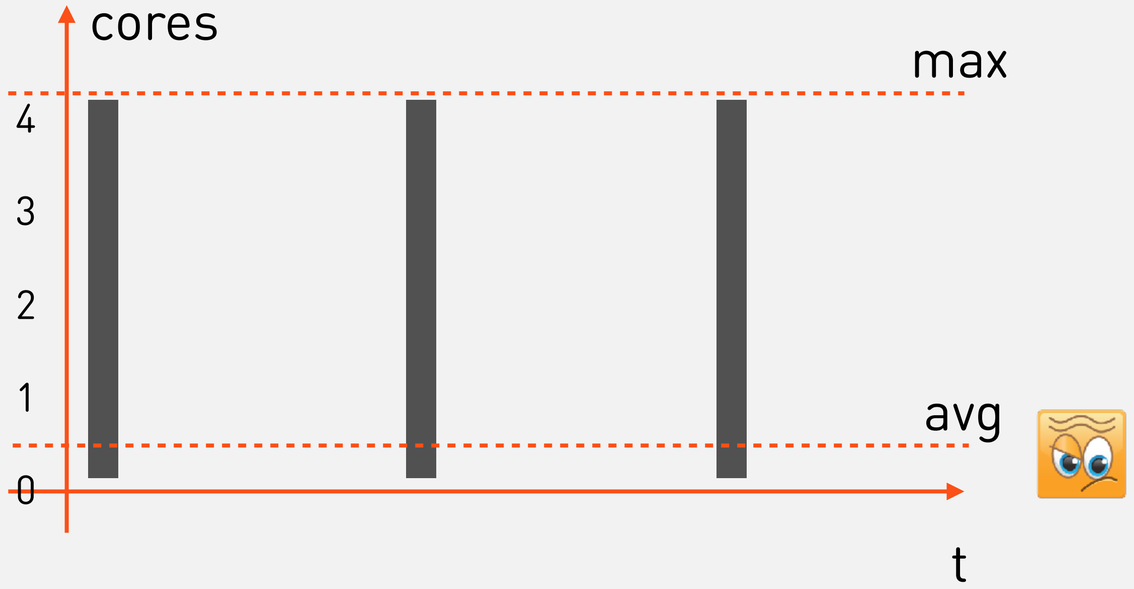
Классы изоляции задач

Классы задач в ОК

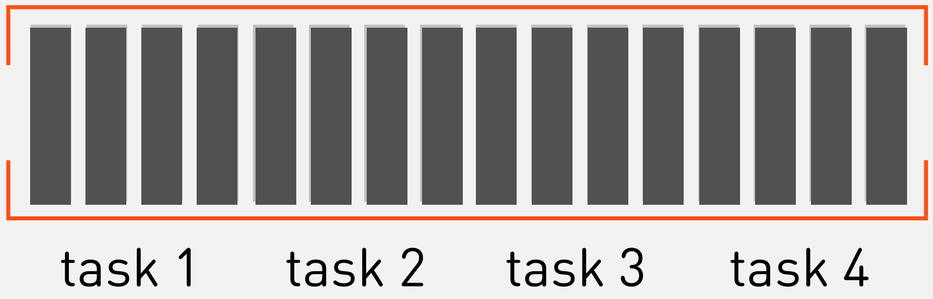
- С малой задержкой : prod
  - важна скорость ответа (latency)
- Расчетные : batch
  - важна пропускная способность (throughput)
  - Map Reduce, ML, DWH, etc.
- Фоновые : idle
  - тесты, пересчеты, конвертации

Классы задач в ОК : prod

- С малой задержкой
  - важна скорость ответа
    (latency)
  - размещение резервированием

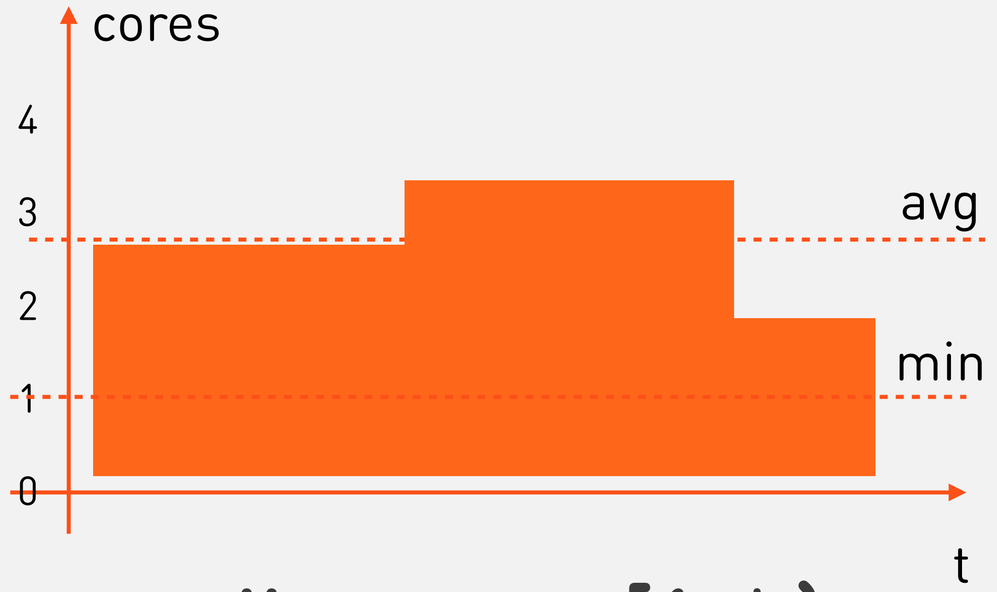


alloc: cpu = 4 (max)



Классы задач в OK : batch

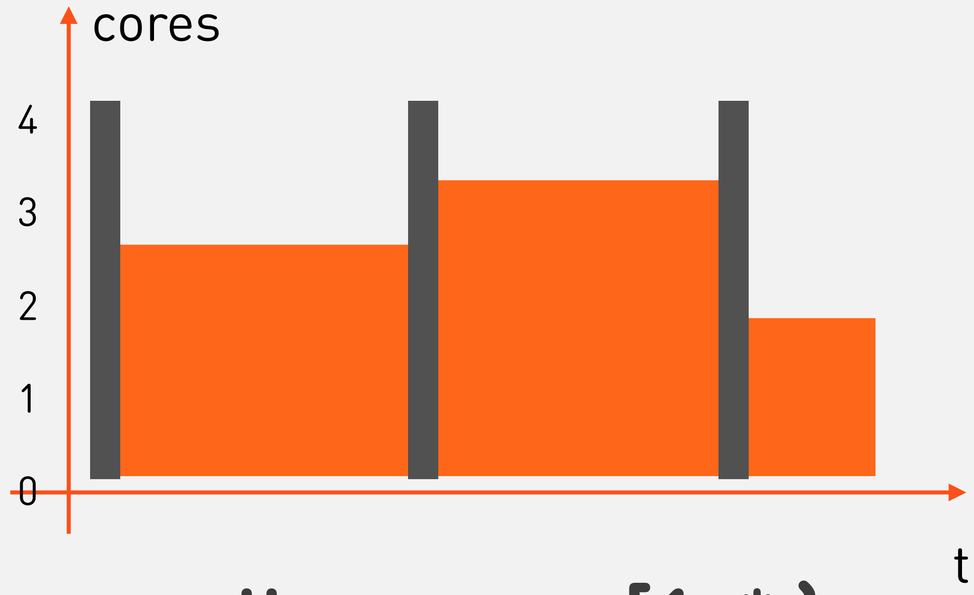
- Расчетные
  - важна пропускная способность (throughput)
  - Map Reduce, ML, DWH, etc.



alloc: cpu = [1, *)

Классы задач в ОК : prod + batch

- С малой задержкой
  - важна скорость ответа
  - размещение резервированием
- Расчетные
  - важна средняя скорость
  - MapRed, ML, etc.



alloc: cpu = [1, *)

Как это сделать в docker run

prod, cpu = 4

—cpuset = 1-4

—cpuquota = **4**00 000
—cpuperiod = **1**00 000

batch, cpu = [1, *)

?

Как это сделать в docker run

prod, cpu = 4

—cpuquota = **4**00 000
—cpuperiod = **1**00 000

batch, cpu = [1, *)

—cpushares = **1 024**

batch, cpu = [2, *)

—cpushares = **2 048**

Linux CPU scheduler policies

- SCHED_OTHER
  - обычный в Linux
- SCHED_BATCH
  - ресурсоемкий; штраф за активацию
- SCHED_IDLE
  - фоновый < nice -19

```
one.nio.os.Proc.sched_setscheduler( pid, Proc.SCHED_IDLE );
```

```
$ chrt -i 0 $pid
```

github.com/odnoklassniki/one-nio

*man sched_setscheduler

Как это сделать в docker run

prod, cpu = 4

—cpuquota = **4**00 000 —cpuperiod = **1**00 000 + SCHED_OTHER

batch, cpu = [1, *)

—cpushares = **1** 024 [—cap-add = SYS_NICE] + SCHED_BATCH

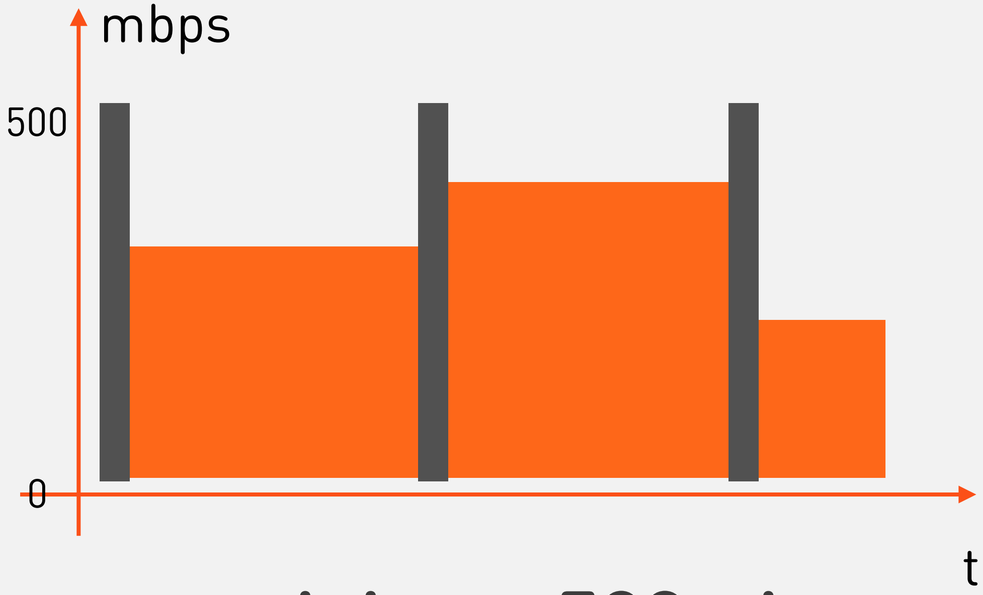
idle, cpu = [2, *)

—cpushares = **2** 048 [—cap-add = SYS_NICE] + SCHED_IDLE

Трафик

- prod приоритетнее batch
- batch > idle (по avg)

- на исходящий трафик
- и на входящий



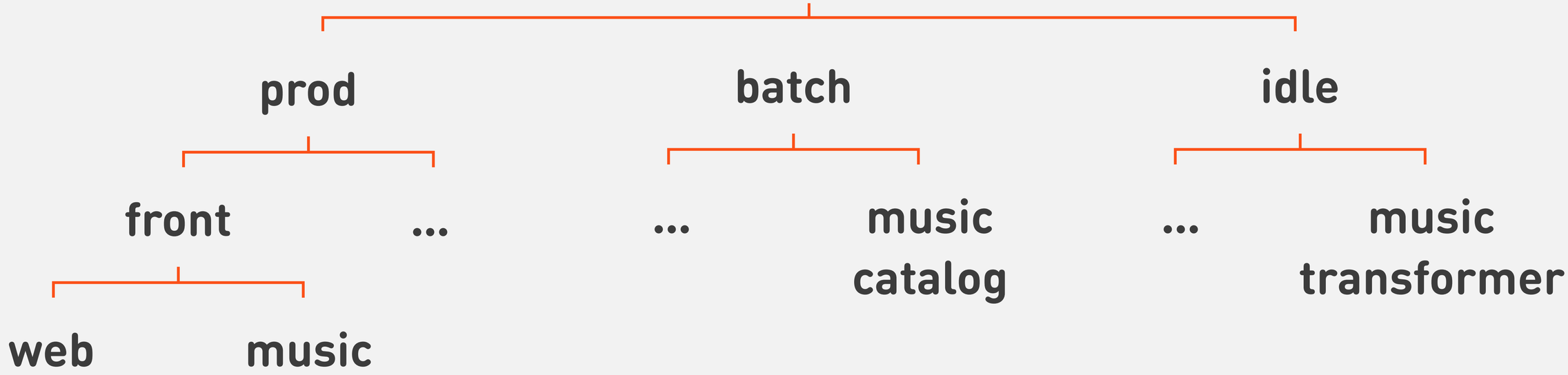
prod: lan = 500mbps

batch: lan = [100, *)

Как это сделать в docker run?...

Linux QoS

- Traffic Control (tc)
  - Hierarchical Fair Service Curve (hfsc)
  - 2 класса: prod; batch/idle
- modprobe ifb
  - для QoS входящего трафика
- регулируемая полоса для batch/idle
  - это пришлось дописать



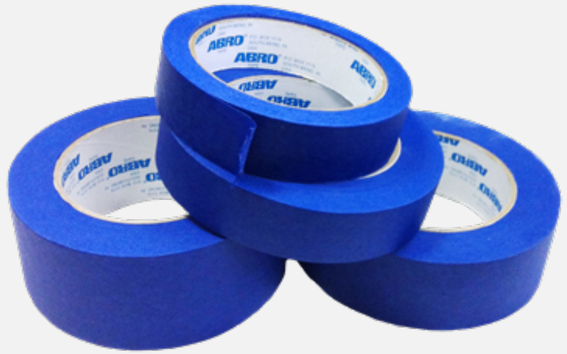
ok-web.group1.web.front.**prod**

catalog-manager.music.**batch**

transformer.music.**idle**

Полная изоляция невозможна

- CPU интенсивный batch
  - ~ нет влияния на prod
- Память
  - 😒 вымывается кеш CPU
  - ~ + 10% к задержке
- Трафик
  - 😒 внутренняя очередь сетевой карты
  - только TCP
  - ~ + 10 % к задержке



Отказоустойчивость

Отказ контейнера

- Изоляция
  - квота на ресурсы
  - нет влияния на других
- Политики рестарта
  - ALWAYS, ON_FAILURE
  - NONE

Отказ миньона

- Переносим!
- Нужен Service Discovery
  - (даже для ip-per-container)
  - Удобно
  - Много решений
  - +Баланировщик
- Нужен ли Service Discovery ? 
  - Балансировок уже много
  - Критическая система + Точка Отказа
  - Много переделывать. Очень много.

Жизнь (почти) без Service Discovery

- IP статичны
  - закрепляются при создании сервиса
  - max(replicas)
  - следуют за контейнером по сети
  - дублирующиеся IP - это ОК
- DNS
  - живые и мертвые IP (клиенты отфильтруют)
- Критичные сервисы - без DNS

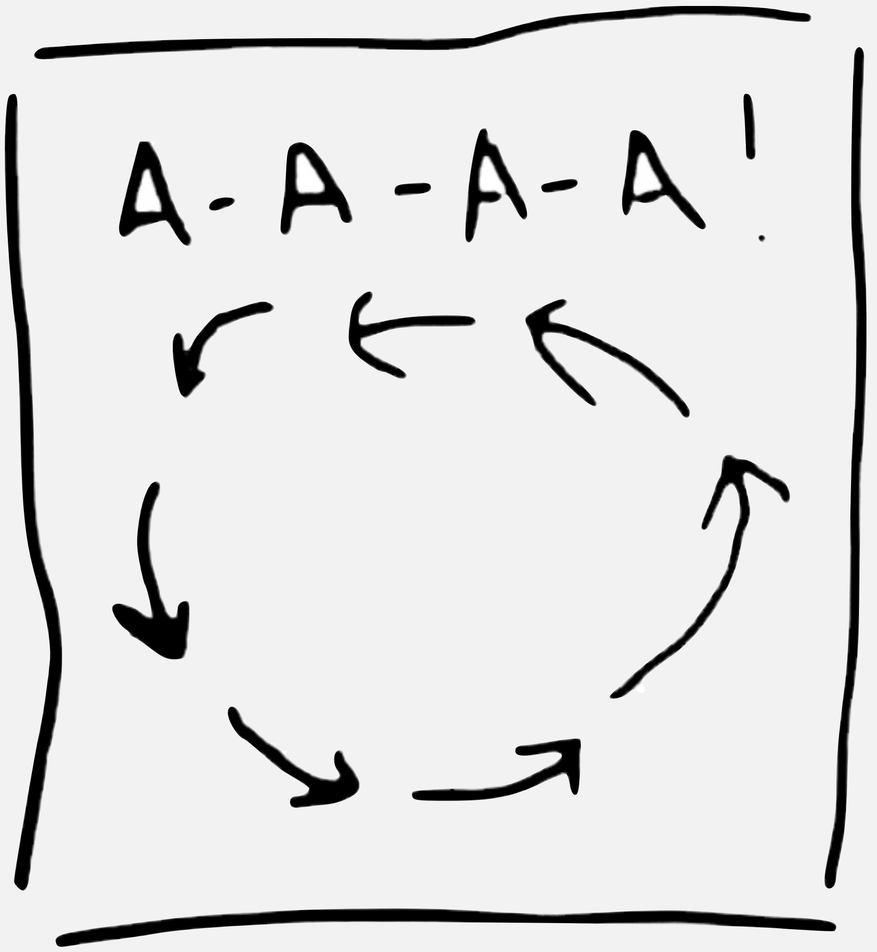
ok-web.group1.web.front.prod

1.1.1.1
1.1.1.2
...

Аварии

Авария - это

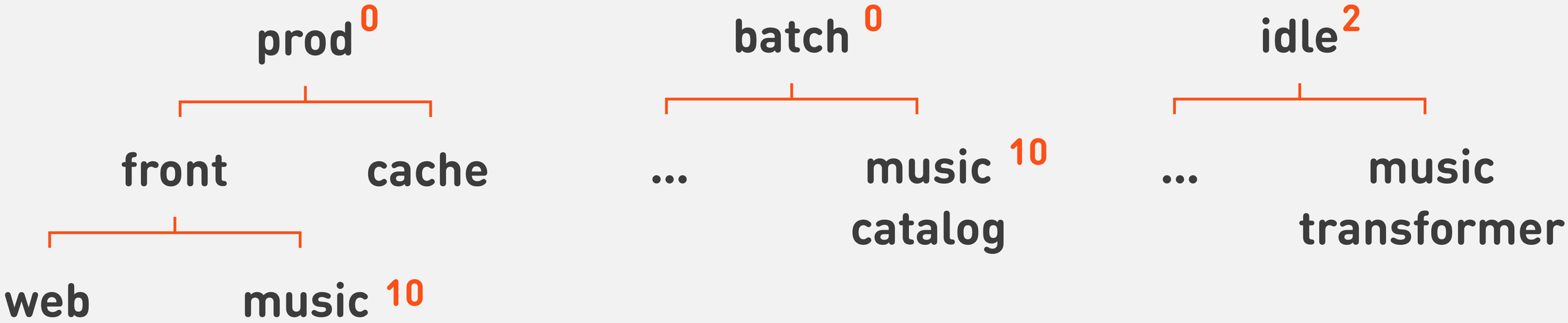
- Отказ множества машин
  - Массовые миграции контейнеров
  - Нехватка ресурсов
- Отказ управляющего слоя
- Взлет количества алертов
  - Тормоза/Шум в мониторинге



Массовые миграции

- Приоритет размещения
  - Выше приоритет - быстрее мигрирует
  - Применяется иерархически

Нехватка ресурсов



- Приоритет вытеснения задачи
  - Вытесняет (останавливает) задачу с миньона
  - Часть задач остается неразмещенными

Тотальное разрушение

Авария ДЦ целиком : причины

- Стихия

  https://habrahabr.ru/company/dataline/blog/333578/

- Человеческий фактор

- Баги

Это НЕ редкость !

Изолировать !

- 1 one-cloud = max(1 ДЦ)
  - Потеря облака = потеря 1 ДЦ
- Готовность приложений к потере ДЦ
  - Политика резервирования
  - Отказоустойчивые БД
  - Тестирование отказа/Учения
- 4 ДЦ = 4 one-cloud
  - Изолированы друг от друга

Проверка адеквантности оператора

```
oleg@[redacted]:~/cloud> ./mcc submit -r 1 -q user-pa
*** notice: using logged in cloud dc
New definition of service cache.user-part0.user.c
Input evaluated value for: 7-2
5
```

- При потенциально опасных командах
  - Массовый останов
- При “странных” командах
  - Уменьшение реплик, смена имени образа сервиса

Итоги

Три квадратика

one-cloud masters

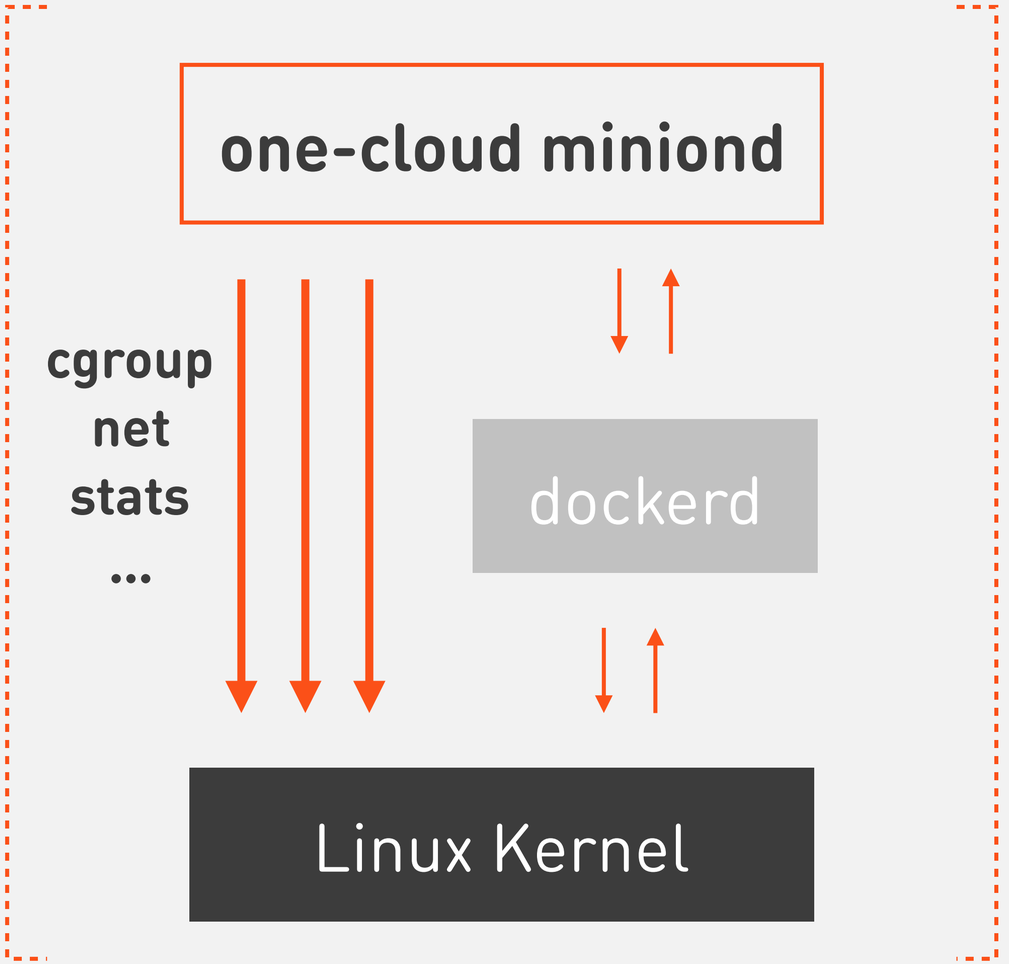
one-cloud miniond

docker daemon

Linux Kernel

Три костылика

one-cloud masters



docker, docker, docker, docker

- Минимум команд
  - pull, create, start, stop, rmi, inspect, events
- Свой регистри
  - --block-registry docker.io
  - —add-registry one-cloud.registry.odkl
- Centos Docker build

https://habrahabr.ru/post/332450/

https://www.infoq.com/presentations/spotify-event-delivery-system

45 плотно упакованных слайдов

- Иерархия сервисов, видимость
- prod + batch = плотная утилизация
  - docker —cpu* + sched_setscheduler/chrt
  - ~~cpuset~~
- Изоляция, отказы, изолента
- Аварии, а также
  - приоритеты и их польза в иерархии
  - ops неадекват страшнее стихии

одноклассники