one-cloud
DC/OS на Java и для Java

Железо в Одноклассниках

4

ЦОД

500

Стоек

8000

Серверов

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

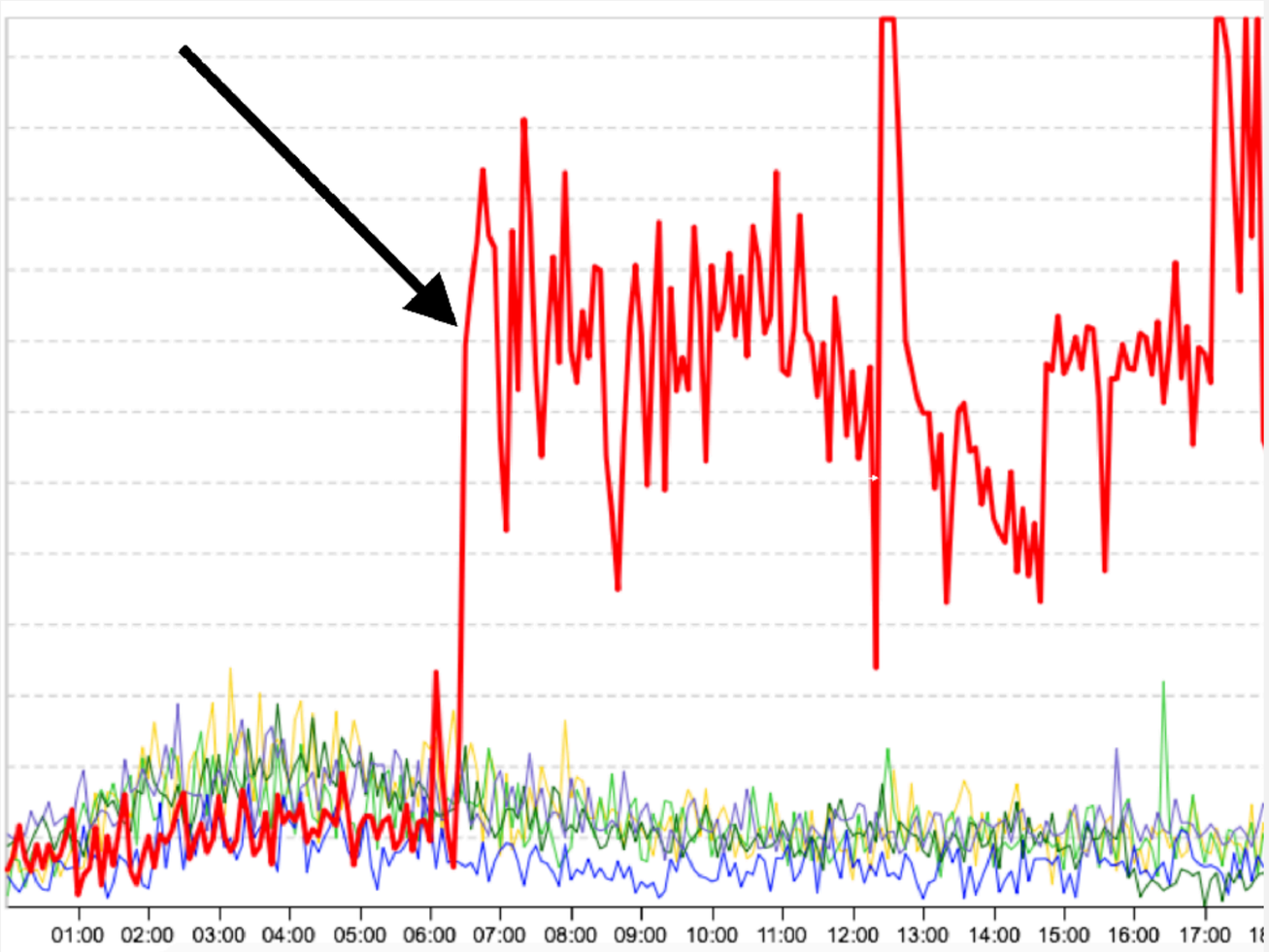
Утилизация стоек - 11 %

↓

Нужно повышать утилизацию

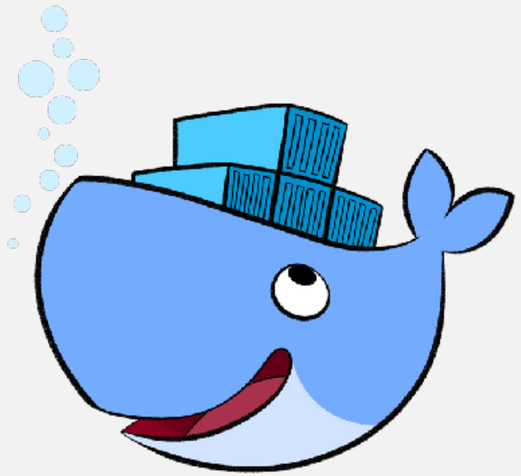
Повышаем %% по-простому

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



Контейнеризируем

- Конфигурация
  - Образы ФС
  - Многослойность, Таги, Реестр
- Изоляция
  - память, ЦПУ

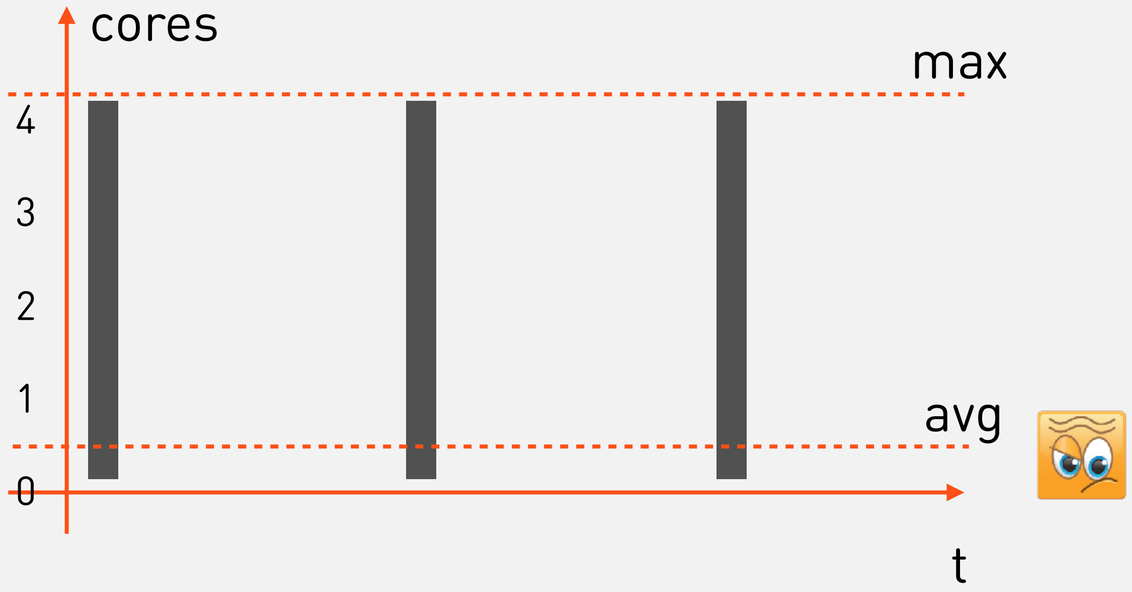


Классы задач в ОК

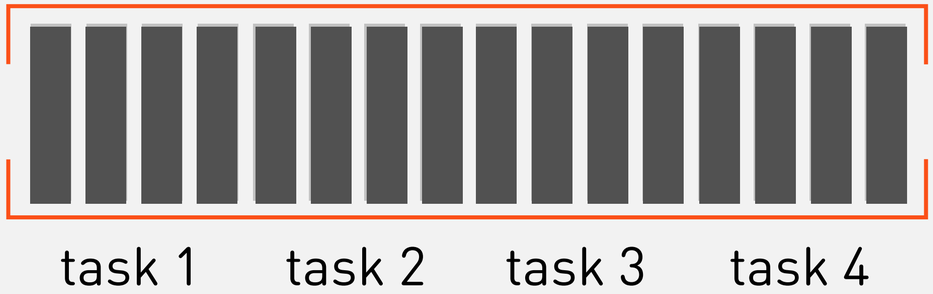
- С малой задержкой : prod
  - важна скорость ответа (latency)
  - web, api fronts, caches, storages
- Расчетные : batch
  - важна пропускная способность (throughput)
  - Map Reduce, ML, DWH, etc.
- Фоновые : idle
  - тесты, пересчеты, конвертации

Классы задач в ОК : prod

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

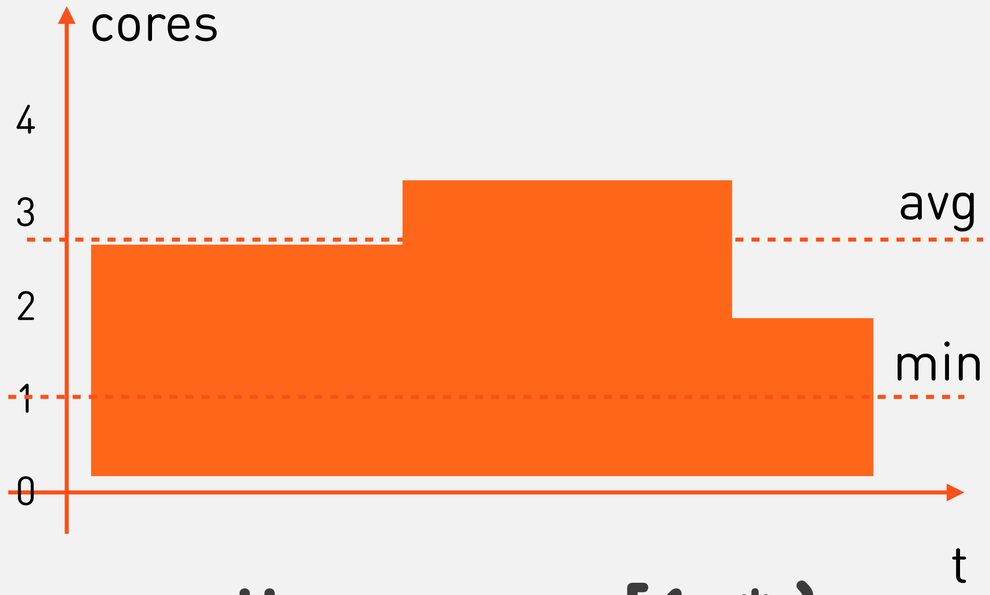


alloc: cpu = 4 (max)



Классы задач в ОК : batch

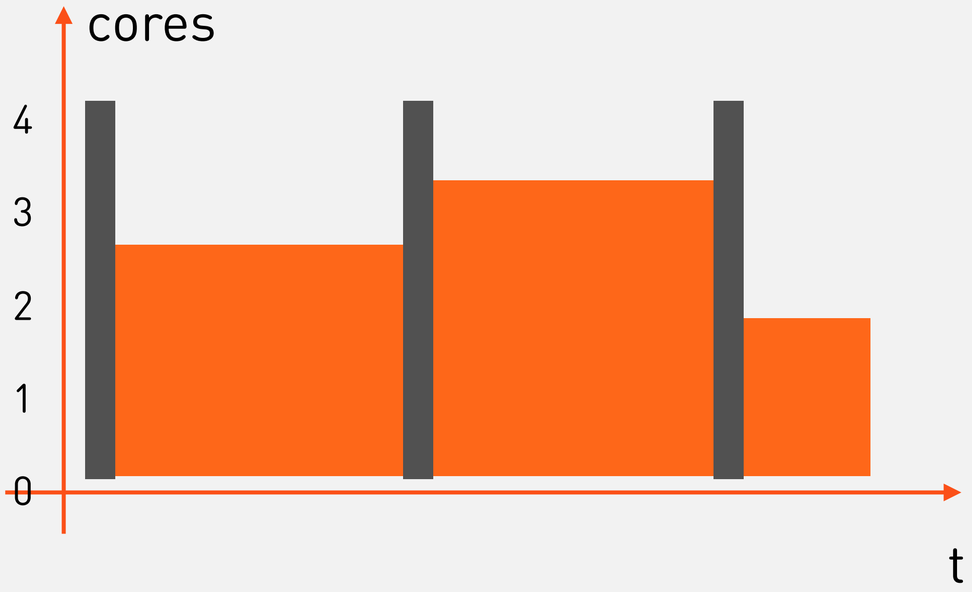
- Расчетные
  - важна пропускная способность (throughput)
  - Map Reduce, ML, DWH, etc.



alloc: cpu = [1, *)

prod + batch = love

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



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 );
```

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 + SCHED_BATCH

idle, cpu = [2, *)

—cpushares = **2** 048 + SCHED_IDLE

Проблема Управления

- Размещение
  - с учетом классов prod, batch, idle
  - Может быть сложным: стойки, залы
- Устойчивость к отказам и авариям
- Kubernetes, Mesos
  - из коробки не подошли
  - почему ?

    см. devoops.ru, highload.ru

Три квадратика

one-cloud masters

one-cloud miniond

docker daemon

Linux Kernel

Мастера one-cloud

one-cloud + C* в одной JVM

Отказоустойчивое хранилище
на локальных дисках

Много уже реализовано
Кластер, кворумы, детектор отказов,
резервное копирование,
восстановление, ...

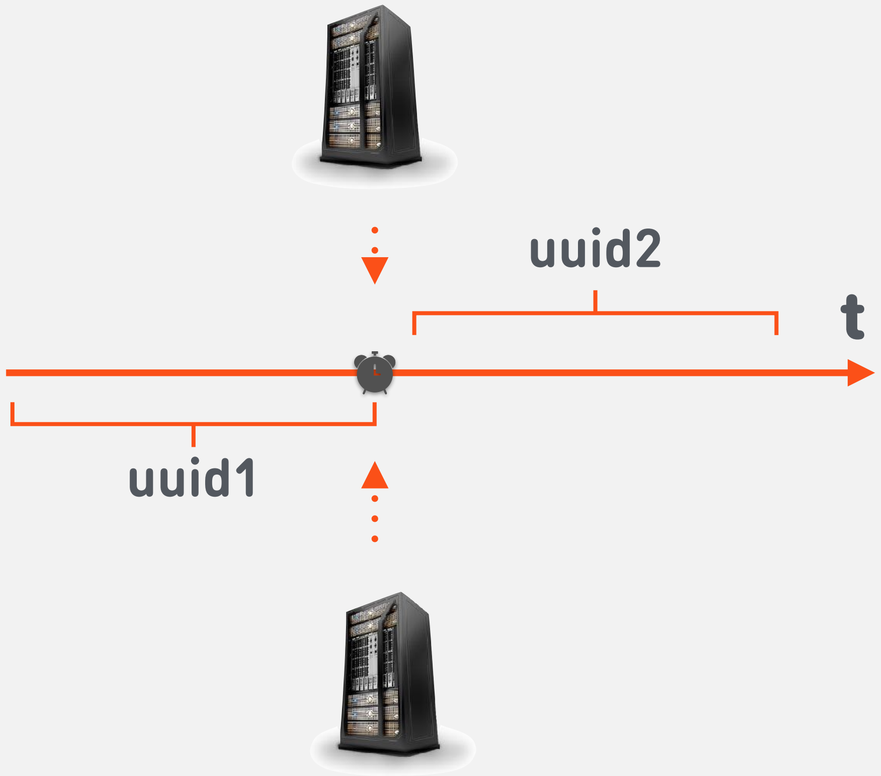
Состояние миньона

```
String name;
/** Hardware available for one-cloud **/
Resource hardware;
/** Service instance reports **/
TaskReport[] states;

/** This report built @ millis **/
long timestamp;

byte[] signature;
```

Детектор отказа миньона: Hashibara et al. http://fubica.lsd.ufcg.edu.br/hp/cursos/cfsc/papers/hayashibara04theaccrual.pdf

Выборы лидера

- Аренда на управление
  - выдается на равное время (1 мин)
- Выборы @ lease_expiration



Выборы лидера

- Аренда на управление
  - выдается на равное время (1 мин)
- Выборы @ lease_expiration
  - Cassandra Lightweight Transaction

    CAS over Paxos
  - UPDATE applied -> leader

```
CREATE TABLE leader (
    cloud text,
    leader_ip inet,
    lease_id timeuuid,
    lease_expiration timestamp,
    PRIMARY KEY(cloud)
)
```

```
UPDATE leader
    SET lease_id=:new , leader_ip=:newip
        lease_expiration=:timestamp
    WHERE cloud='23oblako'
    IF lease_id=:old
```

Выбираем лучшего лидера

- Мертвые не голосуют
- Живые смещают момент
- Приз
  - Текущему лидеру
- Штрафы
  - За мертвого миньона
  - За runtime exceptions

Дай порулить, папаша !

Размещение

- Сервис (манифест)
- Количество реплик
- Ресурсы реплики
  - ЦПУ
  - Память
  - Трафик
  - Диски (место, тип, iops)
- Ресурсы миньонов
  - ЦПУ, память, трафик, диски

user-cache

10 x

cpu = 4
mem = 26 G
lan_in = 450 mbit
lan_out = 450 mbit
hdd = 24 G

Размещение

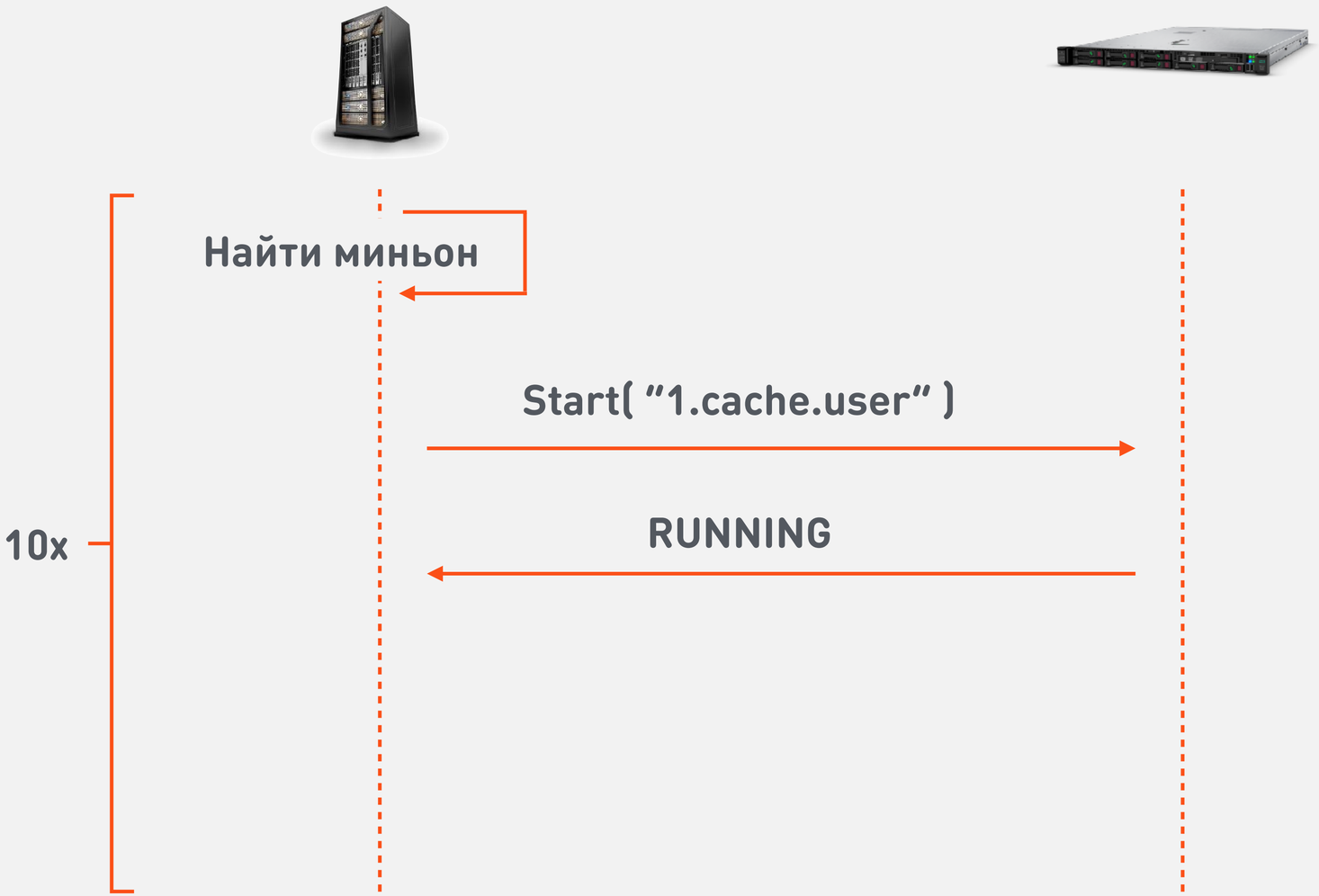
- Для каждой реплики:
  - Определить миньон
  - Запустить задачу
  - Убедиться что она работает

user-cache

10 x

cpu = 4
mem = 26 G
lan_in = 450 mbit
lan_out = 450 mbit
hdd = 24 G

Размещение: по-простому



Размещение: по-простому

10x

Найти миньон

Start(“1.cache.user”)

Pull Image

Start Container

Container Starting

RUNNING

Pull (доставка образа) = 5 сек

Старт ( ) = 1 сек

Старт контейнера ( ) = 10 - 300 сек

10x = 160 - 3000 сек

Размещение: ThreadPool

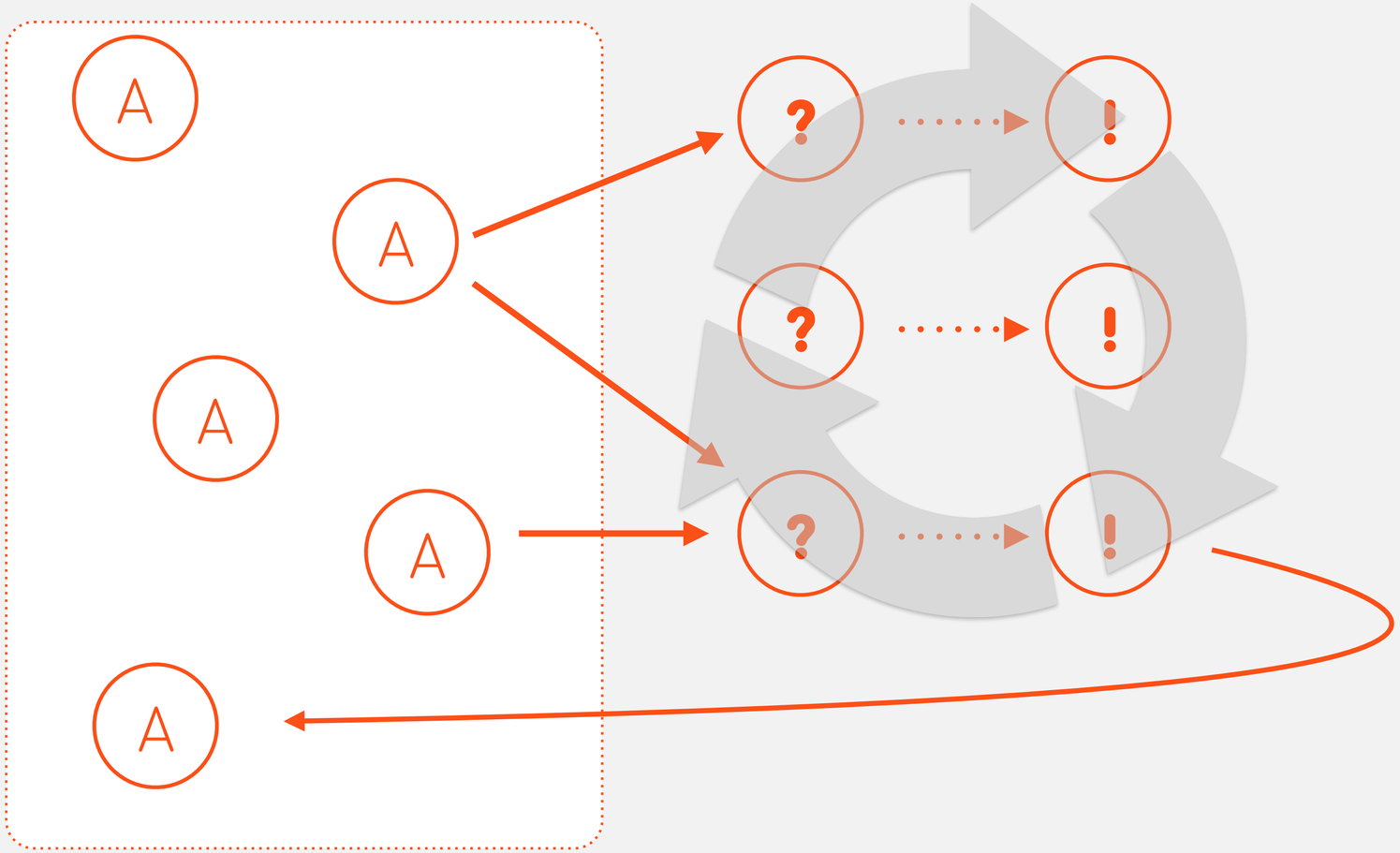
- Тысячи потоков
  - каждая реплика - свой поток
  - пул с миньонами общий
- Куча интересных проблем с конкарренси
  - синхронизации на миньонах, дедлоки и т.п.
- Сложно отработать все отказы
  - миньонов, rpc, контейнеров
  - происходят асинхронно

Программирование по Правилам

Цель вместо алгоритма

Правила

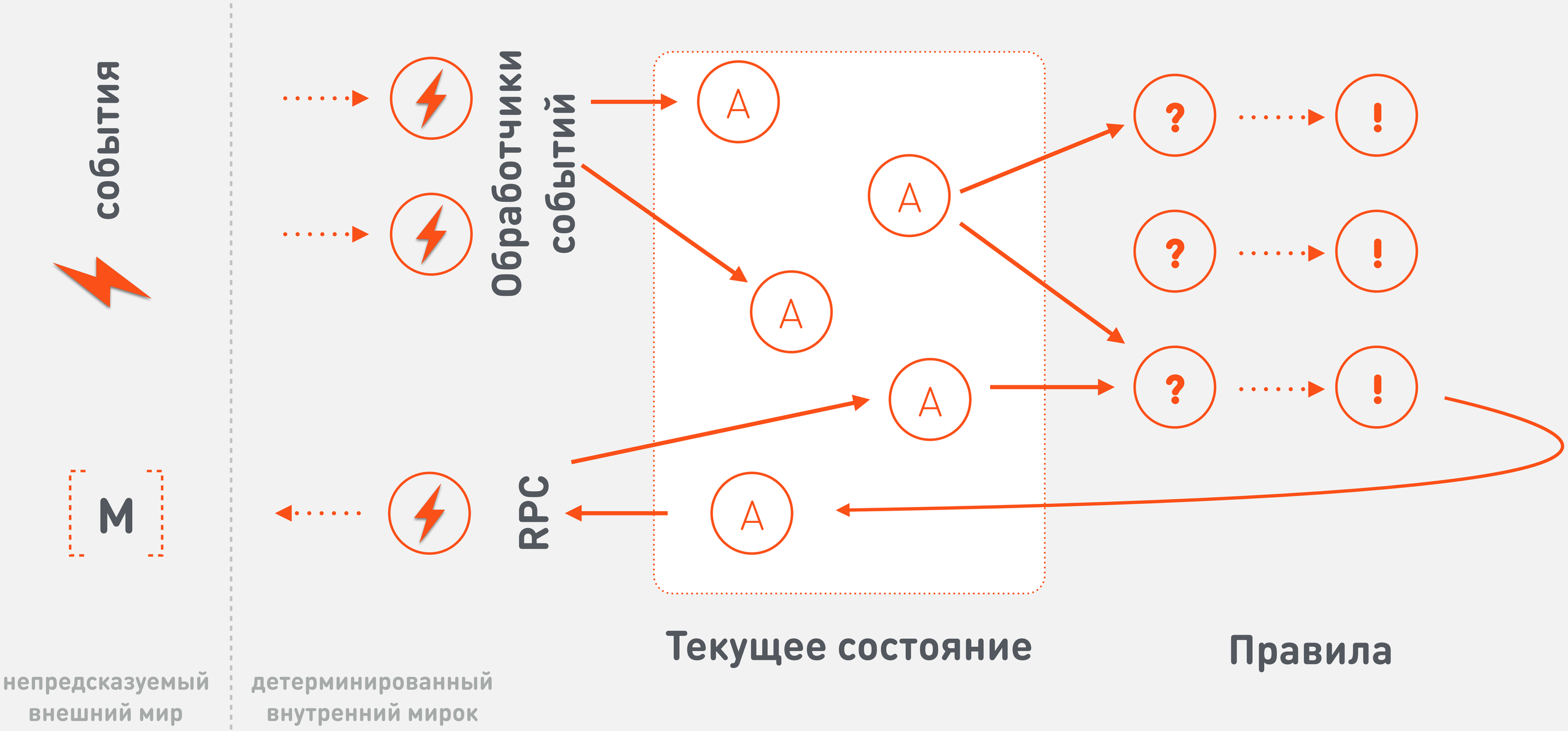
Неблокирующие,
Детерминированные



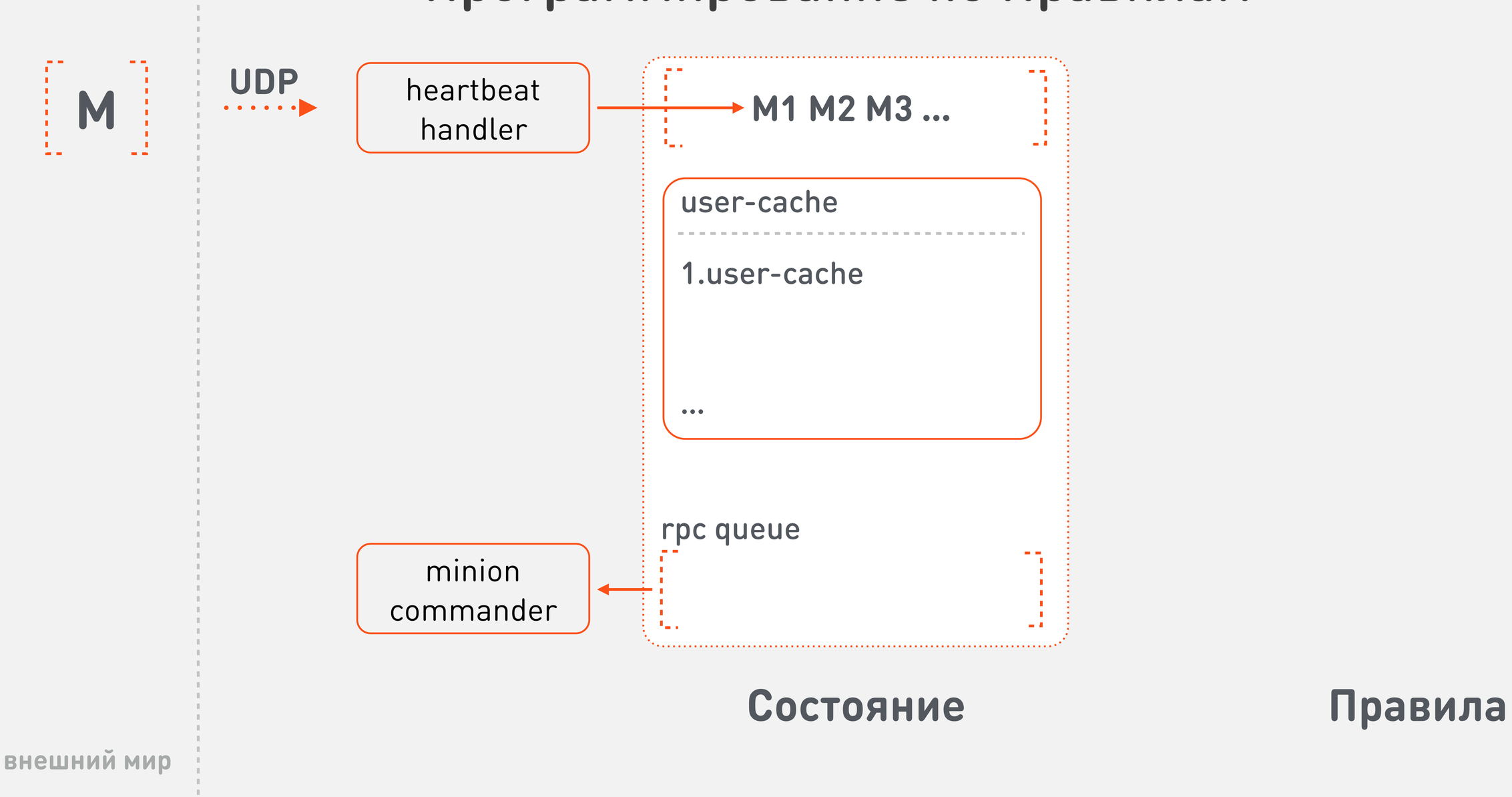
Состояние

Правила

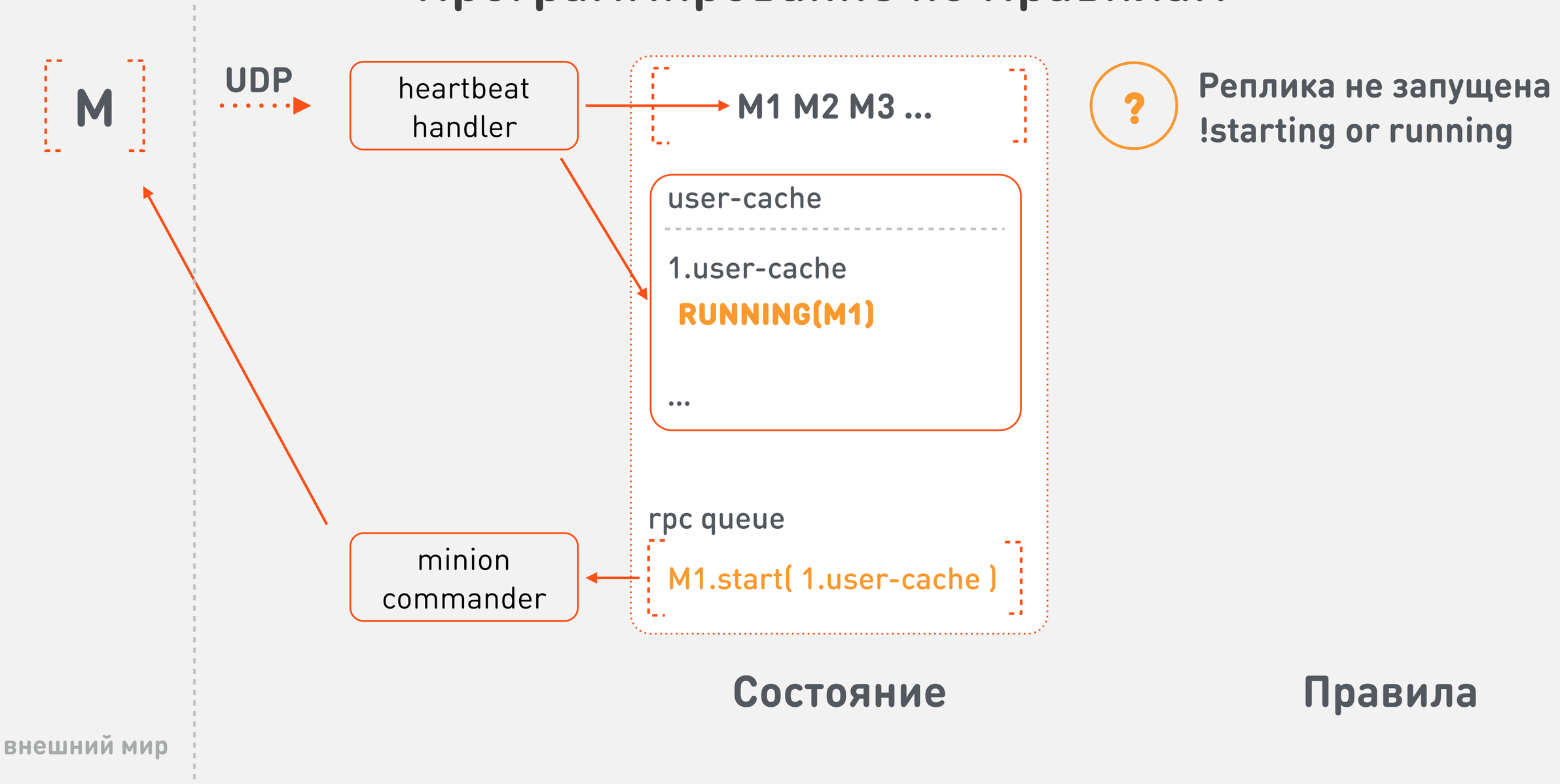
Программирование по Правилам



Программирование по Правилам



Программирование по Правилам



Программирование по Правилам

UDP

heartbeat handler

M1 M2 M3 ...

user-cache

1.user-cache

FAILED

...

rpc queue

M1.stop(1.user-cache)

minion commander

Состояние

Реплика не запущена
!starting or running

Старт дольше таймаута
STARTING > 5 min

Правила

внешний мир

Программирование по Правилам

M2

UDP

heartbeat handler

M1 M2 M3 ...

user-cache

1.user-cache

STARTING(M2)

...

rpc queue

M2.start(2.user-cache)

minion commander

Реплика не запущена
!starting or running

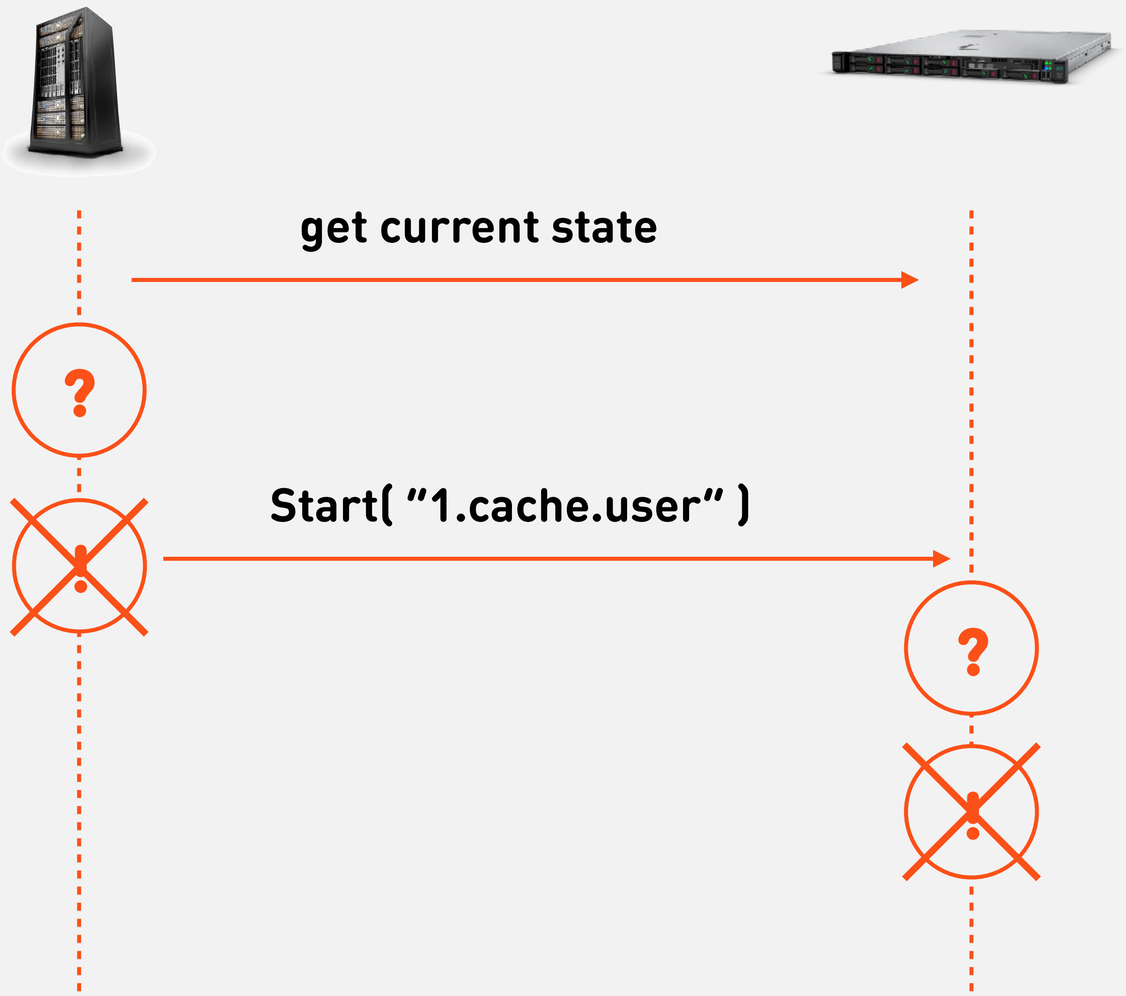
Старт таймаут
STARTING > 5 min

Состояние

Правила

внешний мир

Асинхронная команда



“Асинхронная” это:

Время исполнения негарантировано

Устаревшее состояние мастера

Практические занятия

Эксплуатация

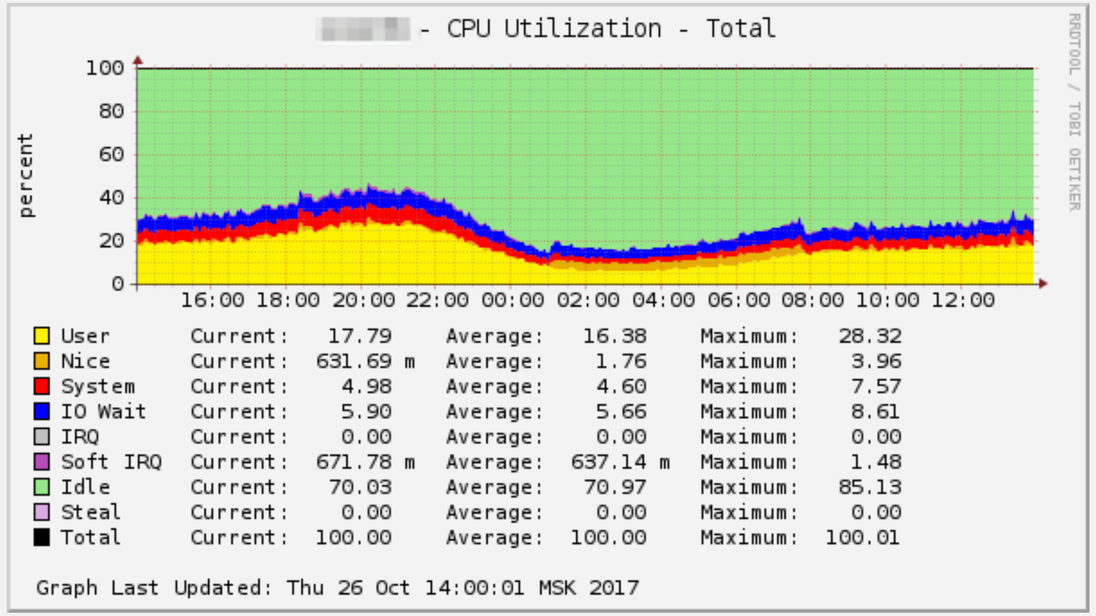
- 4 one-cloud - в каждом ДЦ свое
  - 15 000 ядер, 122T RAM = 6% инфраструктуры
  - 22% (-> 40%) утилизация стойкомест

    2x = 12% ok.ru

- Мигрируем HW -> one-cloud
  - 20% до конца года

Операционные метрики : HW

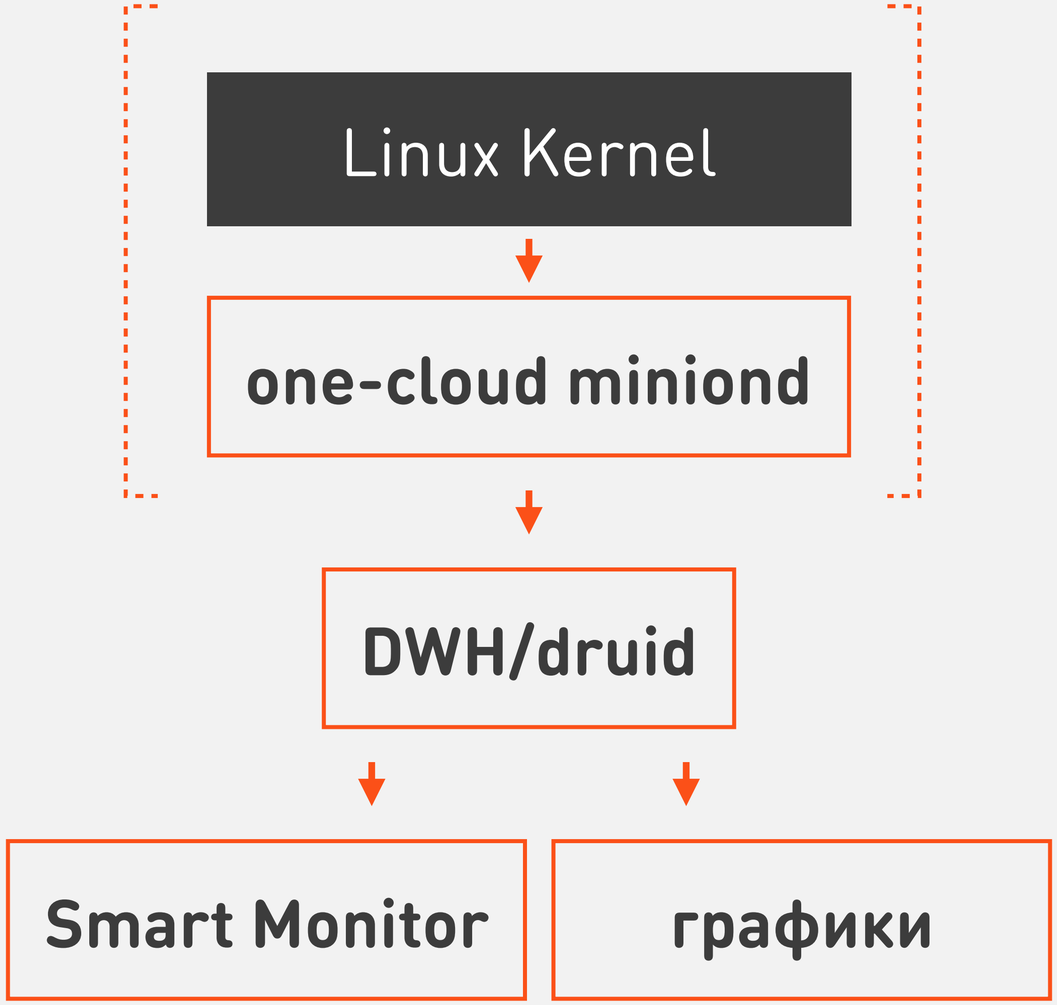
Cacti, Zabbix



нет поддержки динамики

плохая скалируемость

Операционные метрики : контейнеры



https://www.kernel.org/doc/Documentation/cgroup-v1

Дроссель - процессору !

Останов приложения

Виден недостаток ЦПУ

Нужно больше cpu !

```
cpu.stat: nr_periods 284612
          nr_throttled 36
          throttled_time 7939105856
```

—cpuquota = 400 000 —cpuperiod = 100 000

Runtime.availableProcessors()

JDK-6515172

Runtime.availableProcessors() ignores Linux taskset command

- Учитывает только cpuset
  - Java8, Java9, ...
- А нам нужно cpuquota !

Как это сделать в контейнере ?

Как это сделать в docker run

prod, cpu = 4

echo **0-3** >mycpuquota
docker run -v mycpuquota:/sys/devices/system/cpu/online ...

Работает в любой Java

и не только в Java

42 плотно упакованных слайда

- prod + batch = плотная утилизация
  - docker —cpu* + sched_setscheduler/chrt
  - ~~cpuset~~
- Программирование по Правилам
  - Повышает простоту и надежность распределеных систем управления
- Дроссель процессора - это замечательно
- Как заставить JVM видеть квоту ядер

одноклассники