ArchSummit全球架构师峰会深圳站2016

七牛自定义数据处理

自我介绍

- 袁晓沛
- 经历：盛大、七牛、EMC、七牛
- 领域：分布式存储，容器、微服务，大规模数据处理

大纲

- 业务、产品介绍
- 官方数据处理
  - 业务特点、挑战
  - 架构演化
  - 解决方案
- 自定义数据处理
  - 业务特点、挑战
  - 注册、开发、构建
  - 启动、升级、伸缩

业务定义

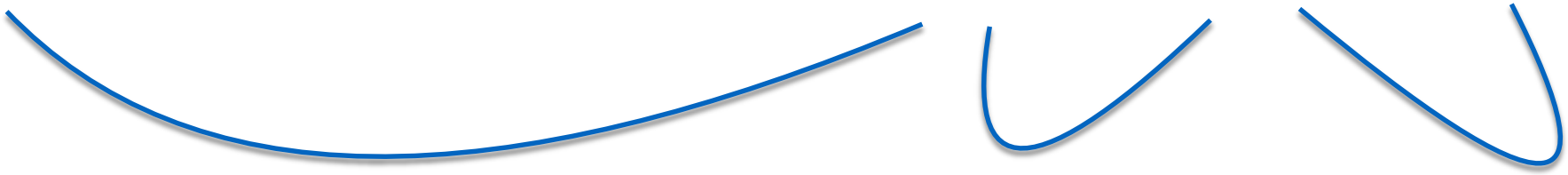
- 针对海量数据
- 提供零运维、高可用、高性能的数据处理服务
- 日处理数近百亿次
- 让用户轻松应对图片、音视频以及其他各类数据的实时、同步处理场景

处理方式

- 官方数据处理
  - 提供基础的数据处理服务，包括但不限于图片转码、水印、原图保护、防盗链等，及音视频的转码、切片和拼接等。
- 自定义数据处理
  - 允许用户构建、上传自定义的私有数据处理服务，并无缝对接存储在七牛的数据及其他数据处理服务。
- 第三方数据处理
  - 开放应用平台，提供大量功能丰富的第三方数据处理服务，如图片鉴黄、人脸识别、广告过滤、语言翻译、TTS等。

使用方式

www.clouddn.com/beauty.jpg?facecrop/200x200



图片URL

UFOP命令

请求参数

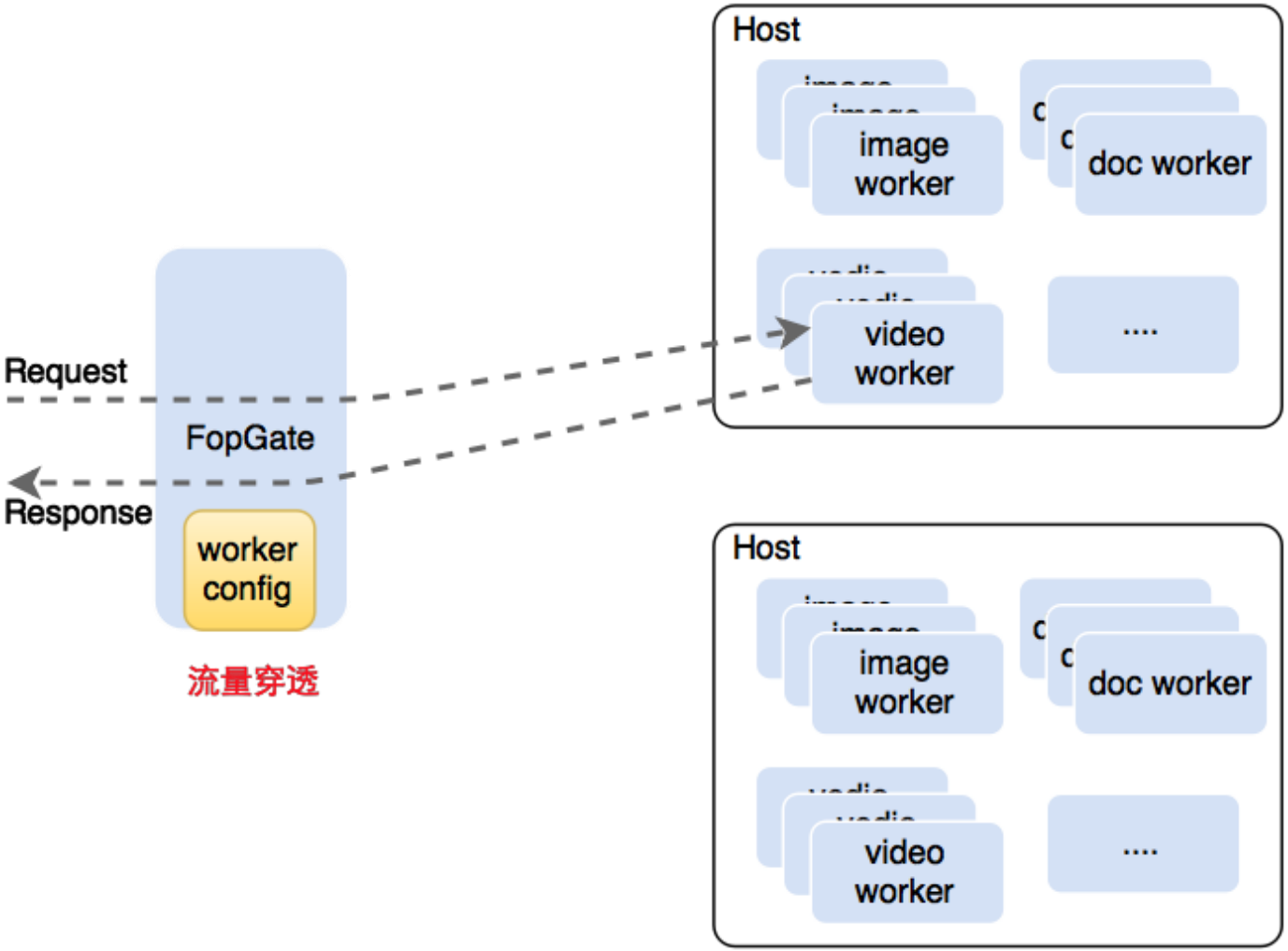
原图

结果

官方数据处理 挑战

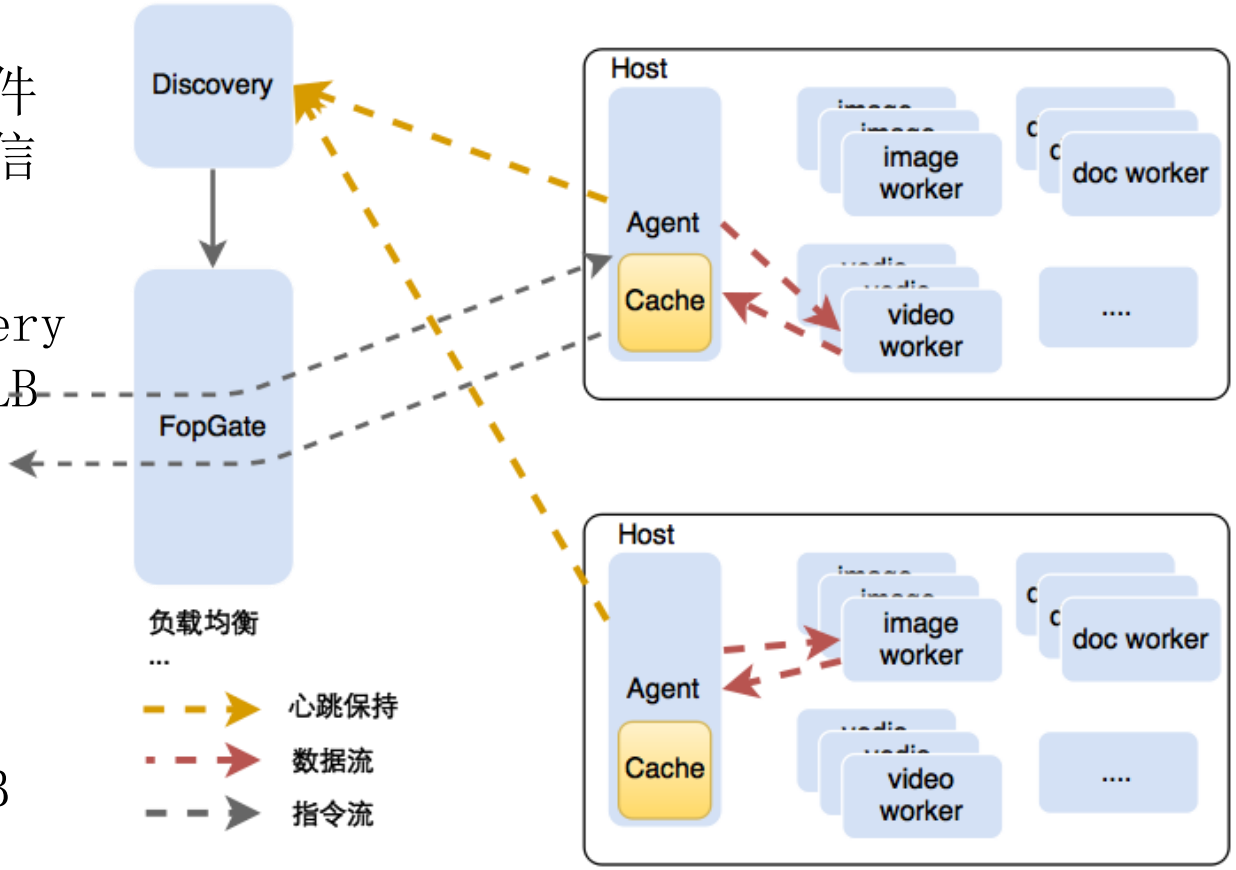
- 请求量非常大
- 突发流量频繁
- CPU密集型计算
- IO操作频繁

官方数据处理 - v1


Host
image worker
doc worker
video worker
....
Request
FopGate
Response
worker config
流量穿透
Host
image worker
doc worker
video worker
....

官方数据处理 - v2

- 增加 Discovery 组件，收集 Agent 上报信息
- FopGate 从 Discovery 获取集群信息，做 LB
- 增加业务 Agent
  - 上报后端信息
  - 上报保活信息
  - 单机内 worker LB



系统测量

- FopGate
  - 单机最大请求数、句柄数
  - 根据实际的业务量，确定机器数
- Image/Audio/Video Worker
  - 找到资源使用最佳范式
  - 根据最佳范式，合理分配资源、配置实例
- 意外发现
  - 大实例、高并发，不如多实例、限制并发
  - 操作系统对CPU调度，比进程好

增加队列

- 服务质量
  - 请求排队，不争抢资源
  - 保证运行速度最快

- 运营角度
  - 根据节点个数、队列长度，
  - 区分免费、付费客户
    - 免费用户，确保高可用
    - 付费用户，确保高质量

限流

- 为什么限流？
  - 大量长链接影响FopGate性能
  - 突发流量，导致队列过长

- 限流手段
  - 并发HTTP请求限制
  - 单用户请求数限制
  - 但Cmd数限制

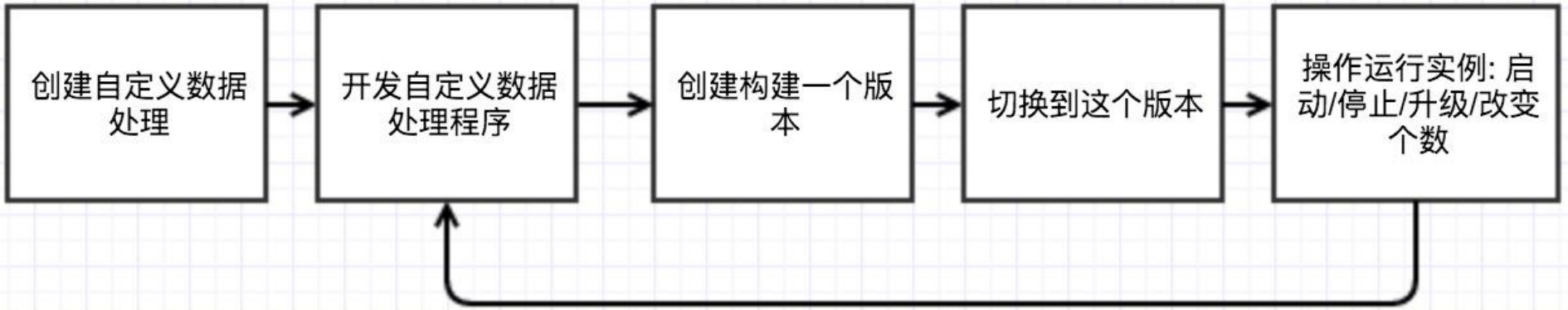
合理协调IO、CPU

- 为什么？
  - 下载、写盘、处理、写盘、返回
- 协调方式
  - 总原则：就近计算
  - FopAgent、Worker混布(1:N)
    - 缩减网络IO的路由次数
  - 挂载ramfs，将内存当磁盘使用
    - 跳过磁盘IO

自定义数据处理挑战

- 处理程序由客户提供
  - 安全性
  - 隔离性
- 业务规模不确定性
  - 可伸缩性

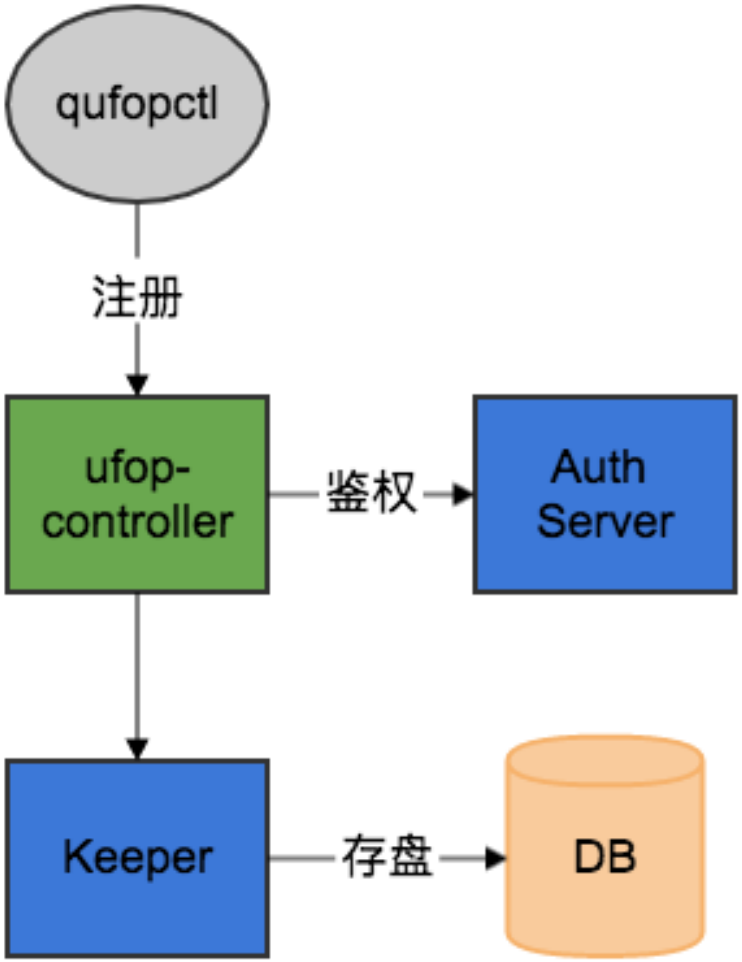
自定义数据处理 - 业务流程



注册

```
$ ./qufopctl reg ufop-demo -m 2
```

注册后端



```go
type ReqArgs struct {
    Cmd  string `json:"cmd"`
    Mode uint32 `json:"mode"`
    Src  struct {
        Url      string `json:"url"`
        Mimetype string `json:"mimetype"`
        Fsize    int32  `json:"fsize"`
        Bucket   string `json:"bucket"`
        Key      string `json:"key"`
    } `json: "src"`
}

func demoHandler(w http.ResponseWriter, req *http.Request) {
    body, _ := ioutil.ReadAll(req.Body)

    var args ReqArgs
    json.Unmarshal(body, &args)

    resp, _ := http.Get(args.Src.Url)
    defer resp.Body.Close()

    buf := make([]byte, 512)
    io.ReadFull(resp.Body, buf)
    contentType := http.DetectContentType(buf)
    lengthStr := strconv.Itoa(int(resp.ContentLength))
    w.Write([]byte("Hello World!\n"))
    w.Write([]byte("The file's mime type is: " + contentType))
    w.Write([]byte("The file's length is: " + lengthStr))
}

func main() {
    http.HandleFunc("/uop", demoHandler)
    err := http.ListenAndServe(":9100", nil)
    if err != nil {
        log.Fatal("Demo server failed to start:", err)
    }
}
```

ufop.yml

```yaml
image: ubuntu
build_script:
- chmod a+x ufop-bin
run: ./ufop-bin
```

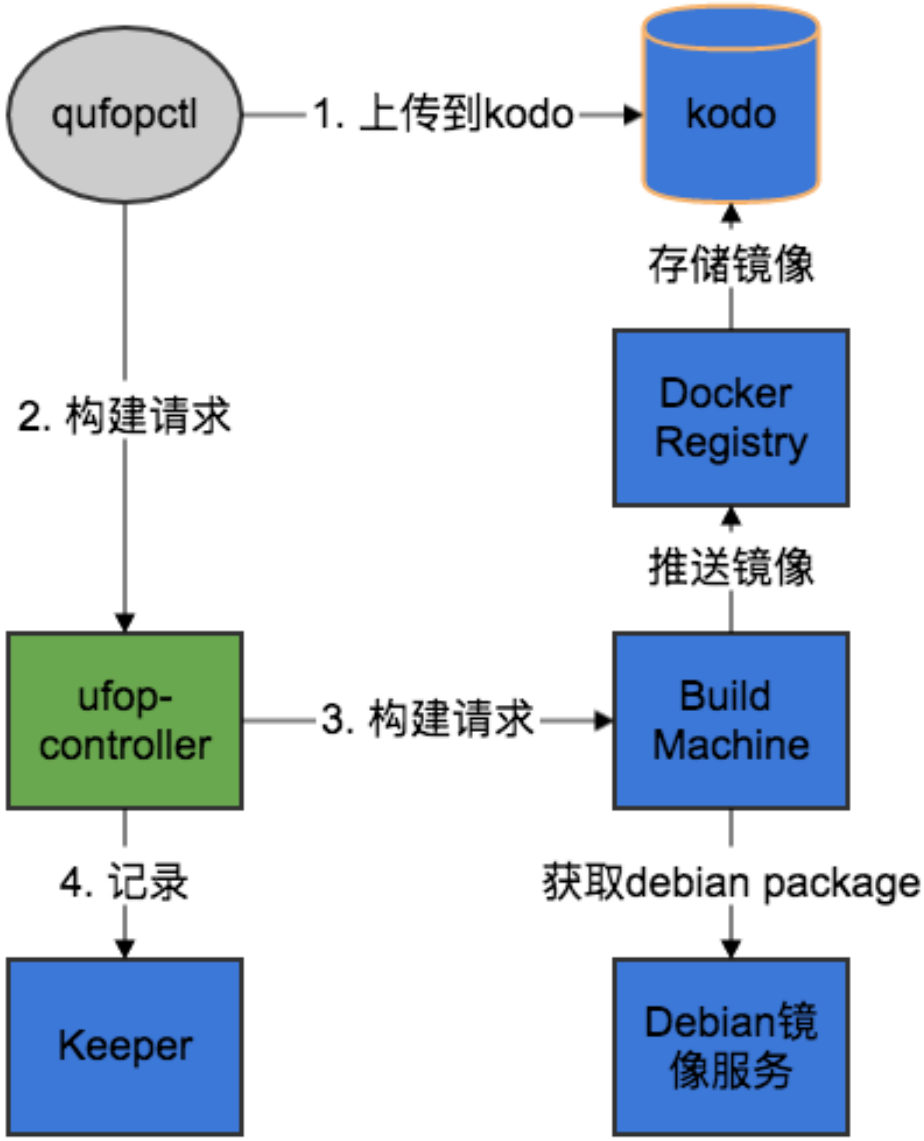
ufop.tar

```
berty@udev:~$ tree ufop-demo-folder/
ufop-demo-folder/
├── ufop-bin
└── ufop.yaml
```

构建

```
$./qufopctl build ufop-demo -d ./ufop-demo-folder
```

构建后端


qufopctl
1. 上传到kodo
kodo
存储镜像
Docker Registry
2. 构建请求
推送镜像
ufop-controller
3. 构建请求
Build Machine
4. 记录
获取debian package
Keeper
Debian镜像服务

使用Debian镜像服务

AppRox

- 经常下载超时
- 下载出错后，需要手动清除

Debian Pkg Mirror

- 首次全量下载
- 定时增量更新

避免Docker构建缓存

Wrong

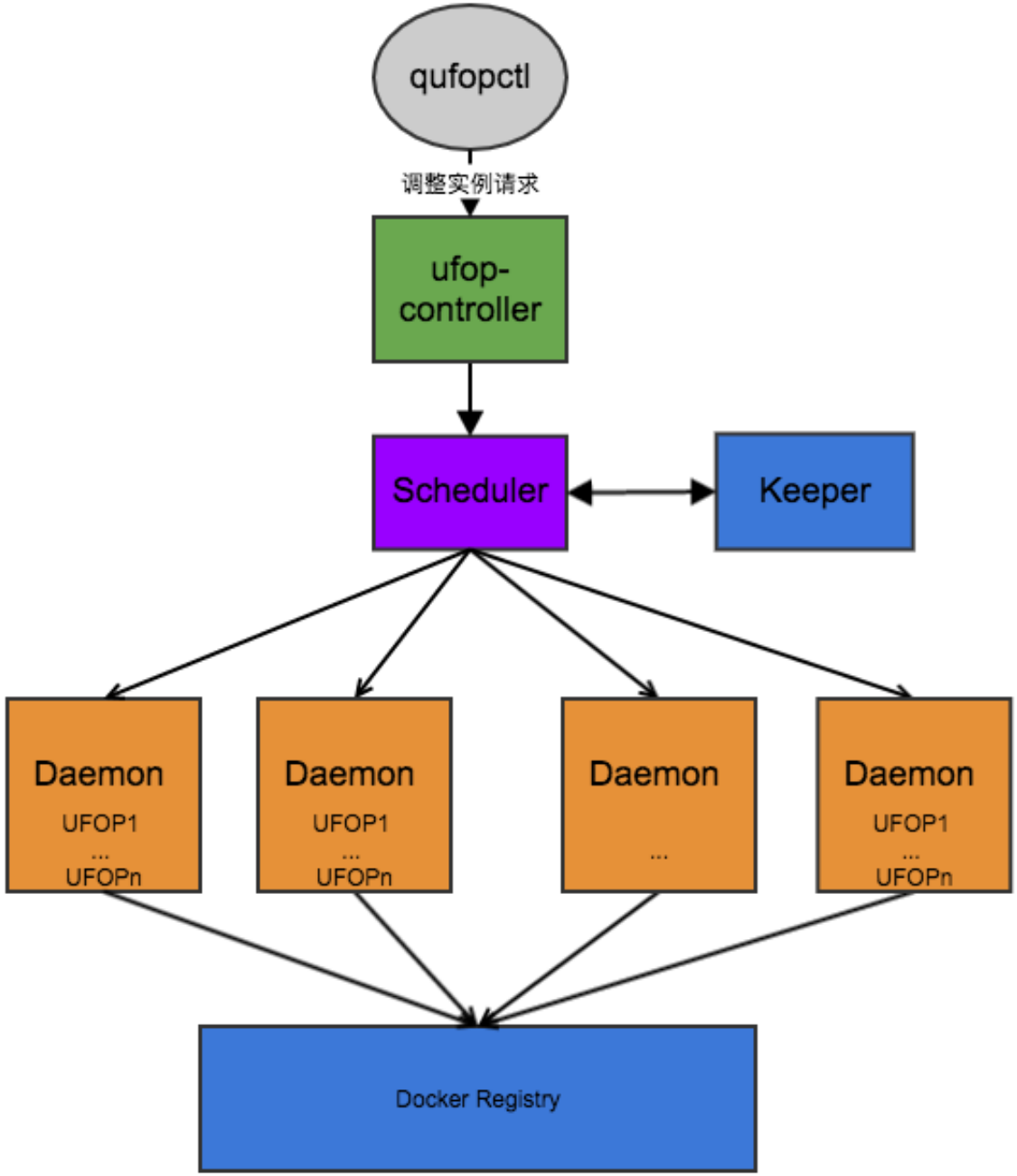
```
-RUN curl -o jdk.tar.gz https://dn-qcos.qbox.me/jdk-7u15-linux-x64.tar.gz

-RUN mkdir -p opt && tar -xf jdk.tar.gz -C /opt && rm -rf jdk.tar.gz
```

Correct

```
+RUN curl -o jdk.tar.gz https://dn-qcos.qbox.me/jdk-7u15-linux-x64.tar.gz \
+       && mkdir -p opt \
+       && tar -xf jdk.tar.gz -C /opt \
+       && rm -rf jdk.tar.gz
```

调整实例数

```
$./qufopctl resize ufop-demo -n 3
```


qufopctl
调整实例请求
ufop-
controller
Scheduler
Keeper
Daemon
UFOP1
...
UFOPn
Daemon
UFOP1
...
UFOPn
Daemon
...
Daemon
UFOP1
...
UFOPn
Docker Registry

升级实例

```
$./qufopctl upgrade ufop-demo -r 1:2
```

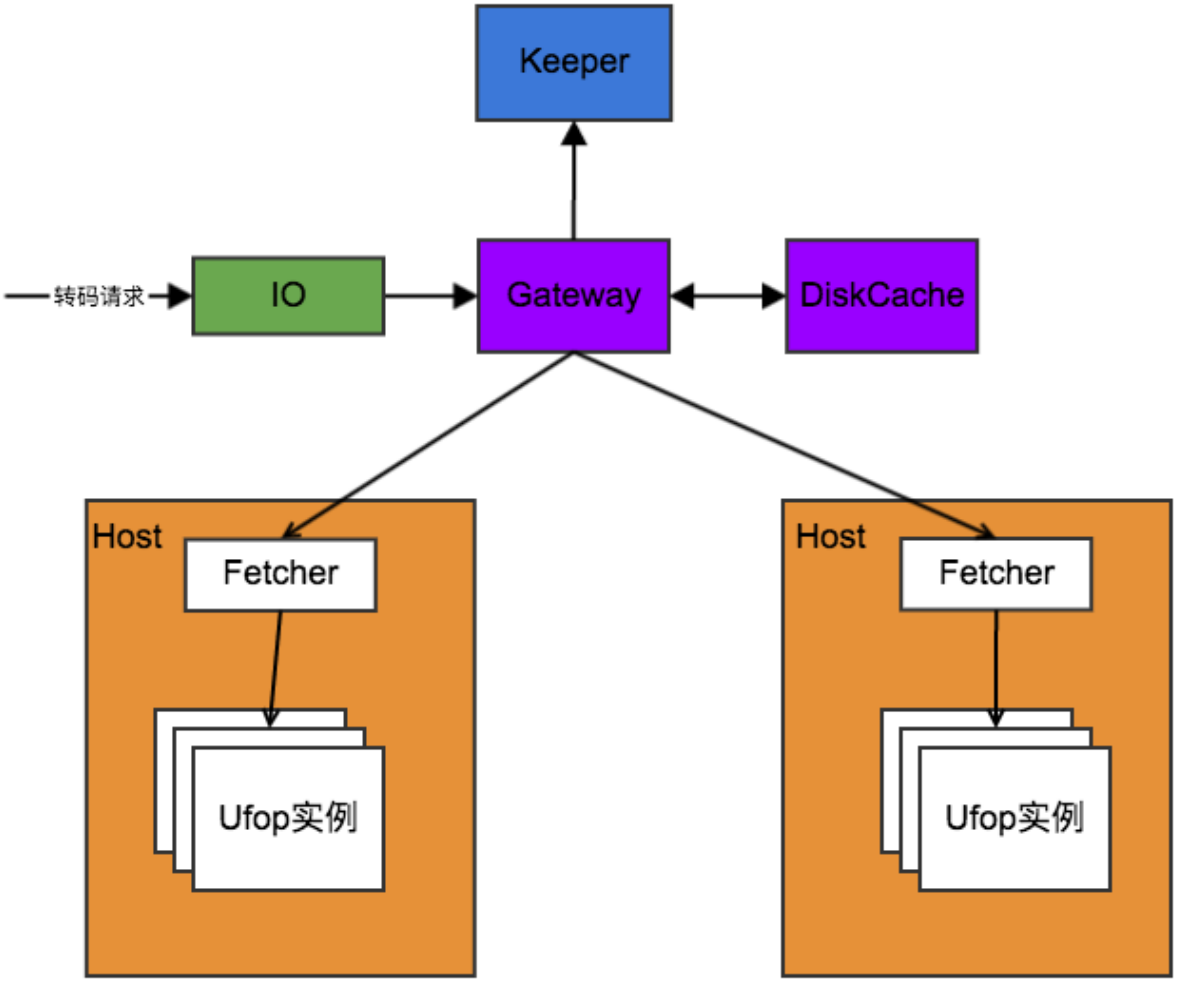
灰度升级阶段


1. 原始状态
实例1(v1)
实例2(v1)
实例3(v1)
2. 启动新实例
实例1(v1)
实例2(v1)
实例3(v1)
实例4(v2)
实例5(v2)
3. 禁掉老实例
实例1(v1)
实例2(v1)
实例3(v1)
实例4(v2)
实例5(v2)
3. 删除老实例
实例3(v1)
实例4(v2)
实例5(v2)

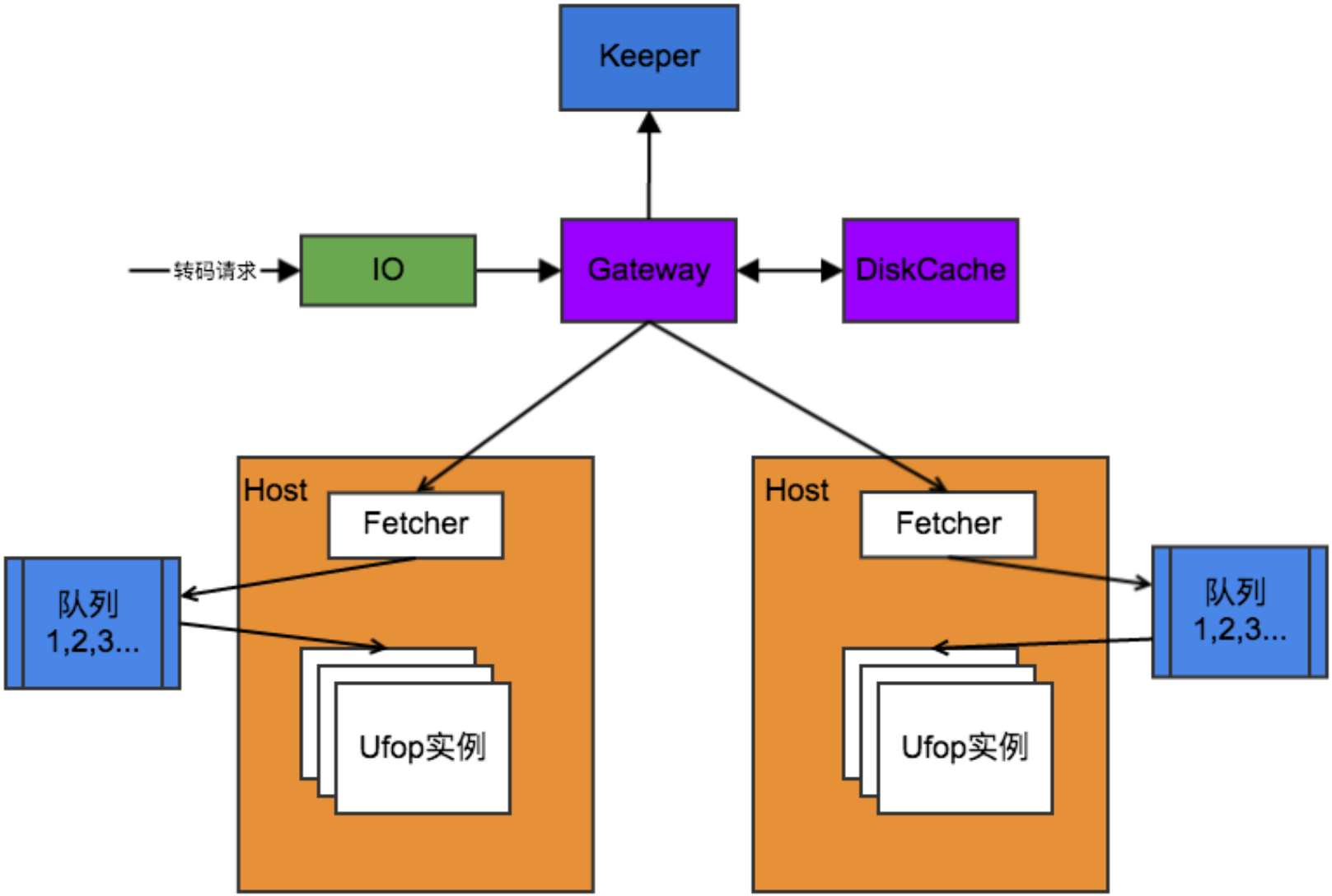
升级的细化

- 新实例WarmUp
  - 内存池、线程池、连接池初始化，初始请求太慢
  - 设定预热时间段，期间请求权重比正常小一点
- 老实例CoolDown
  - 老的请求正在处理，直接停掉影响可用性
  - 应用Docker StopWait
- 计算冗余
  - 预留足够的计算冗余
  - 升级步长 < 小于冗余实例数

数据流 - v1


Keeper
转码请求
IO
Gateway
DiskCache
Host
Fetcher
Ufop实例
Host
Fetcher
Ufop实例

数据流 - v2

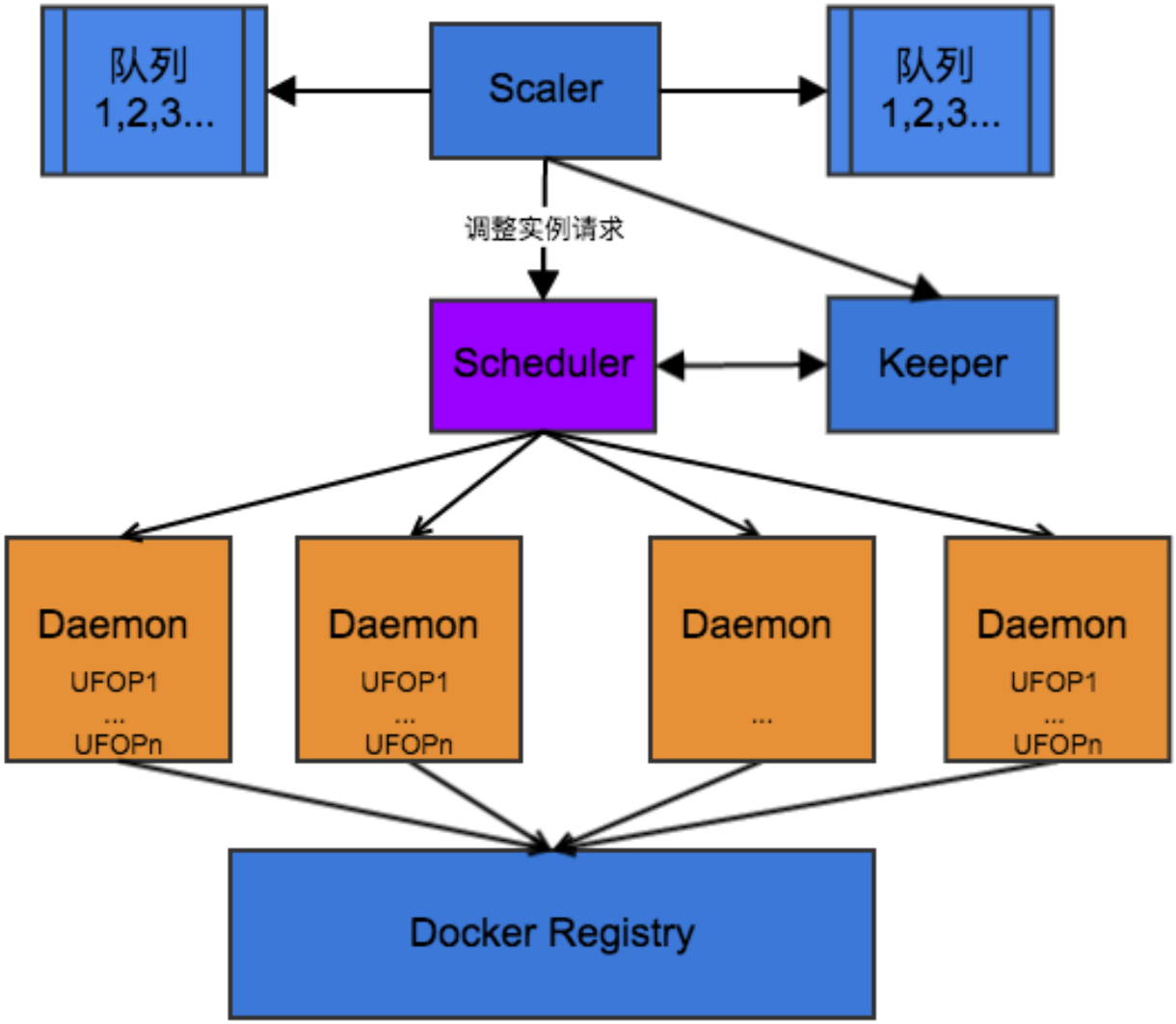

Keeper
转码请求
IO
Gateway
DiskCache
Host
Fetcher
队列
1,2,3...
Ufop实例
Host
Fetcher
队列
1,2,3...
Ufop实例

自动伸缩设置

- 用户配置
  - 默认实例数
  - 平均单实例待处理任务数
  - 是否自动伸缩

- 自动伸缩
  - 增大、或者缩小实例数，以保证：
  - 平均单实例待处理任务数

自动伸缩后端


队列
1,2,3...
Scaler
队列
1,2,3...
调整实例请求
Scheduler
Keeper
Daemon
UFOP1
...
UFOPn
Daemon
UFOP1
...
UFOPn
Daemon
...
Daemon
UFOP1
...
UFOPn
Docker Registry

解决方案

- 安全性
  - 借助iptables
- 隔离性
  - 借助容器的cgroup
- 可伸缩性
  - 实现容器调度系统，支持秒级伸缩
  - 暴露伸缩API，手动伸缩
  - 利用队列长度，自动伸缩

Thanks!