美团构建部署平台解析与Go 的实践

Plus系统分享

美团点评基础研发平台

CODE A BETTER LIFE

一 行 代 码　亿 万 生 活

分享目录

美团点评

1 系统介绍

2 任务调度

3 构建与缓存管理

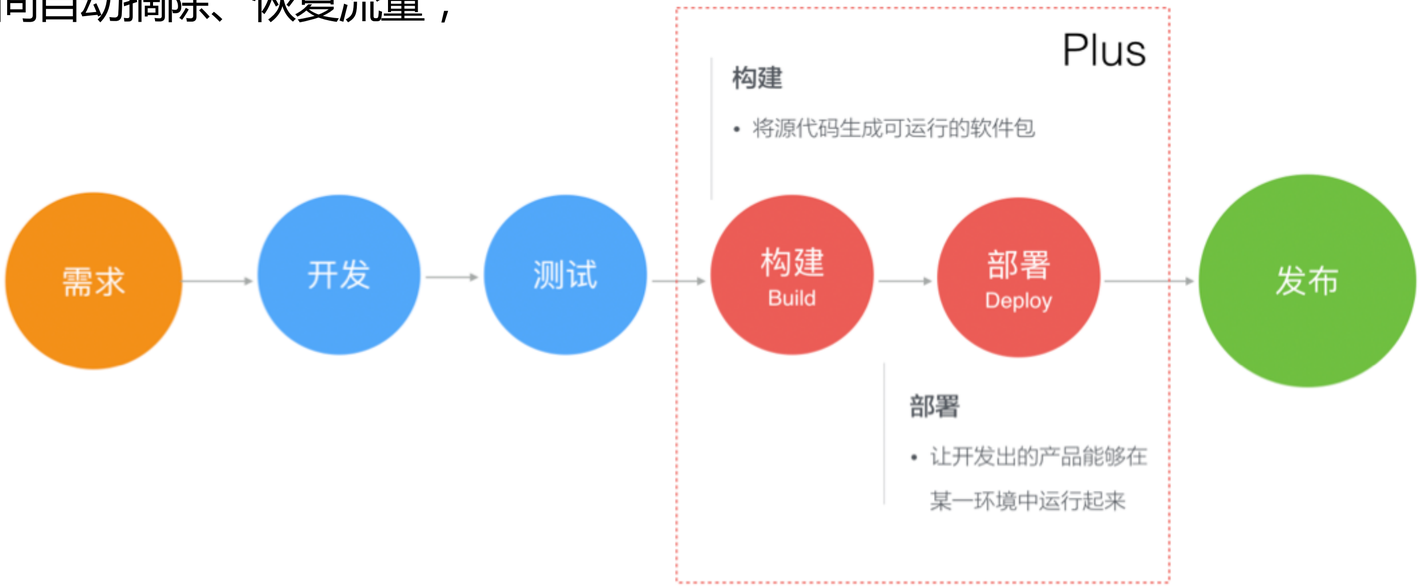
4 部署实现

5 总结

系统介绍

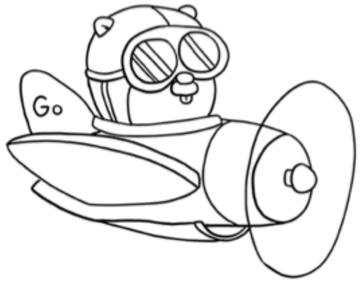
在 2016 年 Plus 启动，采用 go 语言开发

- 主流语言的应用构建部署；
- 通过描述文件定义构建过程与启动方式；
- 自动生成服务镜像；
- 提供了服务平滑发布的能力，在部署期间自动摘除、恢复流量；
- 支持规范化的线上部署流程。



Why go?

美团点评



静态语言

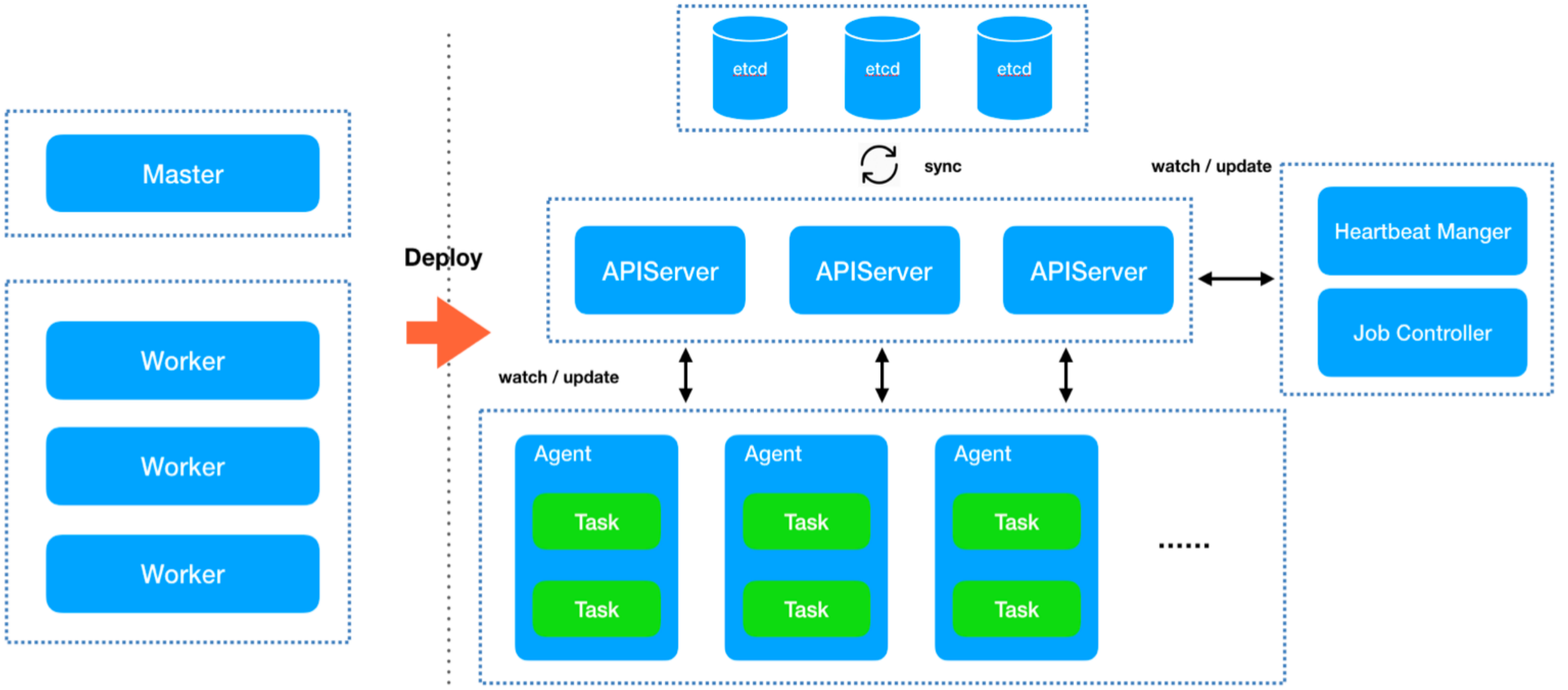
部署简单

并发友好

上手成本低

docker, kubernetes 等大型开源项目使用

系统架构


Master
Worker
Worker
Worker
Deploy
etcd
etcd
etcd
sync
watch / update
APIServer
APIServer
APIServer
Heartbeat Manger
Job Controller
watch / update
Agent
Task
Task
Agent
Task
Task
Agent
Task
Task
.....

分享目录

美团点评

1 系统介绍

2 任务调度

3 构建与缓存管理

4 部署实现

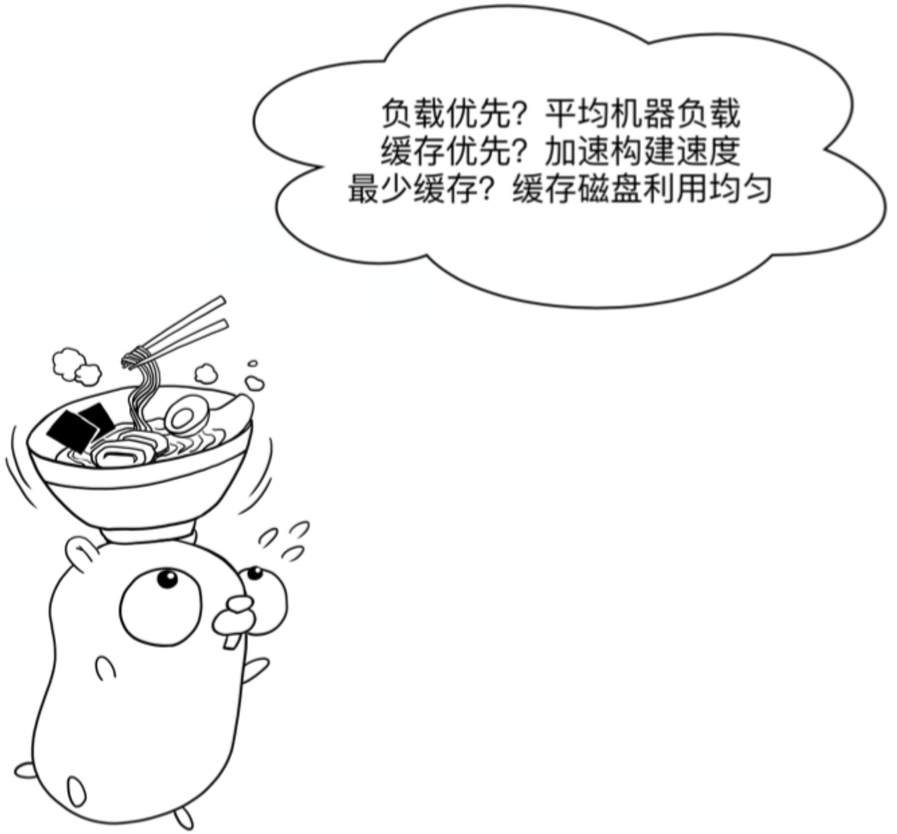
5 总结

系统调度

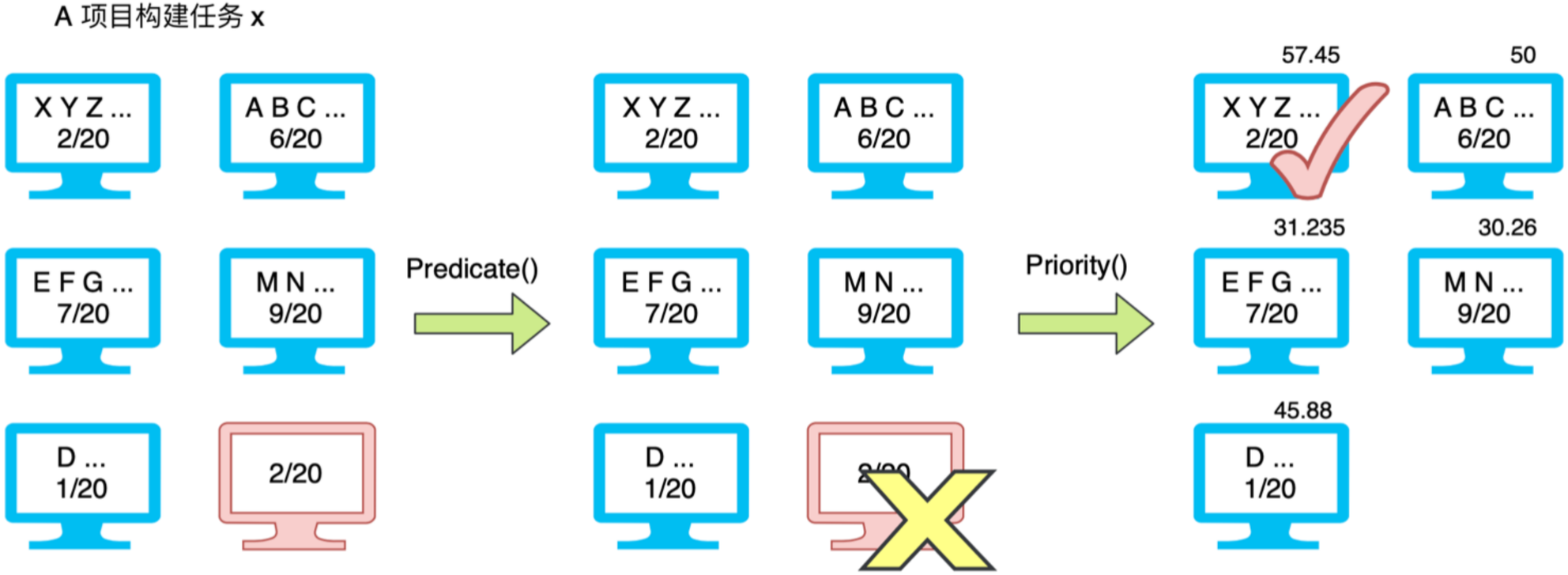
美团点评

如何选择最合适的 worker 来执行任务

1. worker 有不同种类：
    - 构建 worker：物理机，构建代码依赖 docker 创建隔离环境
    - 部署 worker：容器或者虚拟机，较轻量
    - 线下环境 worker
2. worker 负载情况不同
3. worker 对任务的缓存情况不同
4. worker 新增导致缓存不均匀



加权计算，选择最优


A 项目构建任务 x
X Y Z ...
2/20
A B C ...
6/20
E F G ...
7/20
M N ...
9/20
D ...
1/20
2/20
Predicate()
X Y Z ...
2/20
A B C ...
6/20
E F G ...
7/20
M N ...
9/20
D ...
1/20
Priority()
57.45
X Y Z ...
2/20
50
A B C ...
6/20
31.235
E F G ...
7/20
30.26
M N ...
9/20
45.88
D ...
1/20

分数计算

1. 数值转化

纬度	值	类型	转化	说明
负载 load	1/20	分数	小数 [0,1]	负载越小越好
存在对应项目缓存	true false	bool	0 或者1	有缓存优先
缓存总数	0 – 正无穷	int	int 数值	总缓存越小越好

2. 标准化：使得某个纬度下 worker 得分最终得分在 [0, 1] 之间分布

标准化分数 = (val - min) / (max - min)　（当值越小越优：标准化分数 = 1- (val - min) / (max - min) ）

3. 加权计算得分 = $\sum_i^n weight(i) * normalizedScore(i)$

workerid	cachedSum	标准化分数	存在项目 cache	标准化分数	负载	标准化分数	最终得分
worker1	100	1	false	0	2/20	0.67	35*0.67 +50*0 +34*1 = 57.45
worker2	2000	0	true	1	6/20	0.25	35*0.25 +50*1 +34*0 = 50
woker3	500	0.79	false	0	7/20	0.125	35*0.125 +50*0 +34*0.79 = 31.235
worker4	300	0.89	false	0	9/20	0	35*0 +50*0 +34*0.89 = 30.26
worker5	1300	0.32	false	0	1/20	1	35*1 +50*0 +34*0.32 = 45.88

Go benchmark test 调整权重

美团点评

根据历史任务执行情况

- mock cache 不均匀的 worker
- 模拟项目执行后 worker 记录缓存

观察执行 50000次任务之后，

- cache 分布是否均匀
- 发布过程中负载是否均匀
- cache 命中情况

纬度	权重
负载 load	35
存在对应项目缓存	50
缓存总数	34

```
DEBU[0040] job 49999 release release-5331 round 1 worker:worker-3 current score: 0.000000, 0.000000+1.000000*34.0=35.000000
DEBU[0040] job 49999 release release-5331 round 1 worker:worker-2 current score: 0.000000, 0.000000+1.000000*34.0=35.000000
DEBU[0040] job 49999 release release-5331 round 1 worker:worker-5 current score: 0.000000, 0.000000+1.000000*34.0=35.000000
DEBU[0040] job 49999 release release-5331 round 1 worker:worker-1 current score: 0.000000, 0.000000+1.000000*34.0=35.000000
DEBU[0040] job 49999 release release-5331 round 1 worker:worker-6 current score: 0.000000, 0.000000+1.000000*34.0=35.000000
DEBU[0040] job 49999 release release-5331 round 3 worker:worker-8 current score: 35.000000, 35.000000+0.000000*34.0=35.000000
DEBU[0040] job 49999 release release-5331 round 3 worker:worker-3 current score: 35.000000, 35.000000+0.000000*34.0=35.000000
DEBU[0040] job 49999 release release-5331 round 3 worker:worker-6 current score: 35.000000, 35.000000+0.000000*34.0=35.000000
DEBU[0040] job 49999 release release-5331 round 3 worker:worker-0 current score: 35.000000, 35.000000+1.000000*34.0=85.000000
DEBU[0040] job 49999 release release-5331 round 3 worker:worker-1 current score: 35.000000, 35.000000+0.000000*34.0=35.000000
DEBU[0040] job 49999 release release-5331 round 3 worker:worker-2 current score: 35.000000, 35.000000+0.000000*34.0=35.000000
DEBU[0040] job 49999 release release-5331 round 3 worker:worker-9 current score: 35.000000, 35.000000+0.000000*34.0=35.000000
DEBU[0040] job 49999 release release-5331 round 3 worker:worker-4 current score: 35.000000, 35.000000+0.000000*34.0=35.000000
DEBU[0040] job 49999 release release-5331 round 3 worker:worker-5 current score: 35.000000, 35.000000+0.000000*34.0=35.000000
DEBU[0040] job 49999 release release-5331 round 3 worker:worker-7 current score: 35.000000, 35.000000+0.000000*34.0=35.000000
DEBU[0040] job 49999, worker worker-0 get score: 85.000000
DEBU[0040] job 49999, worker worker-4 get score: 35.000000
DEBU[0040] job 49999, worker worker-7 get score: 35.000000
DEBU[0040] job 49999, worker worker-9 get score: 35.000000
DEBU[0040] job 49999, worker worker-3 get score: 35.000000
DEBU[0040] job 49999, worker worker-1 get score: 35.000000
DEBU[0040] job 49999, worker worker-8 get score: 35.000000
DEBU[0040] job 49999, worker worker-2 get score: 35.000000
DEBU[0040] job 49999, worker worker-5 get score: 35.000000
DEBU[0040] job 49999, worker worker-6 get score: 35.000000
INFO[0040] worker-3: cacheSum: 3254
INFO[0040] worker-7: cacheSum: 3251
INFO[0040] worker-9: cacheSum: 3250
INFO[0040] worker-4: cacheSum: 3251
INFO[0040] worker-5: cacheSum: 3252
INFO[0040] worker-6: cacheSum: 3250
INFO[0040] worker-8: cacheSum: 3250
INFO[0040] worker-0: cacheSum: 3251
INFO[0040] worker-1: cacheSum: 3250
INFO[0040] worker-2: cacheSum: 3251
goos: darwin
goarch: amd64
BenchmarkPriority-8            1        40208333054 ns/op
PASS
ok      _/Users/songcy/mtwork/plus/devtools/deploy/master/daemon        40.751s
```

分享目录

美团点评

1 系统介绍

2 任务调度

3 构建与缓存管理

4 部署实现

5 总结

构建与缓存管理

美团点评

描述文件

YAML格式

存放到代码仓库

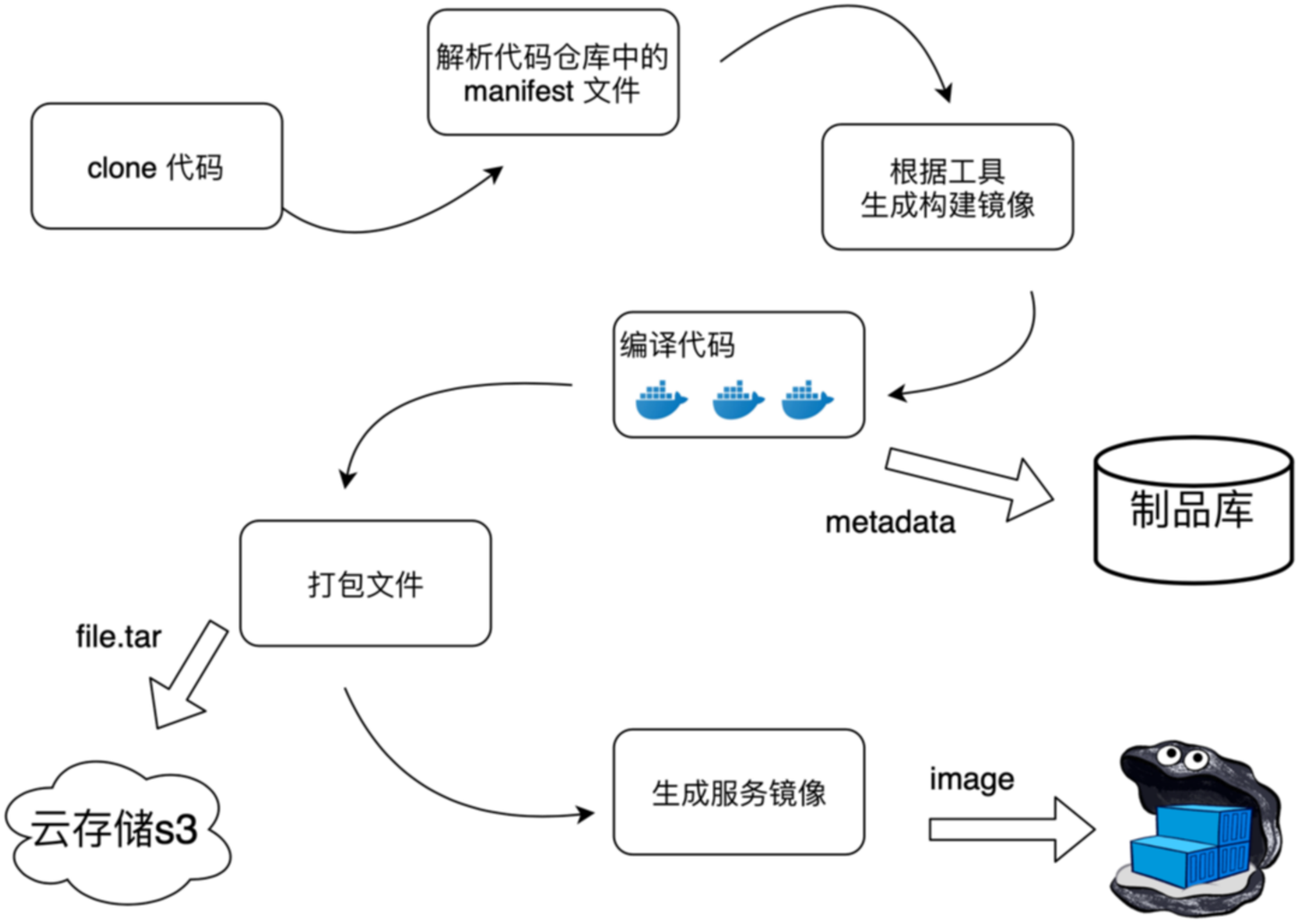
定义构建与部署过程

自动生成服务镜像

manifest.yaml

```yaml
version: v1
build:
    tools:
        oracle-jdk: 8
        maven: 3.3.3
    run:
        cmd:
            - mvn package
    target:
        distDir: target
        files:
            - ./
autodeploy:
    targetDir: /opt/meituan/apptest/
    tools:
        oracle-jdk: 8
    run: java -jar dependency/jetty-runner.jar helloworld.war
    check: sh check.sh
    checkRetry: 2
    checkInterval: 30s
```

构建过程


解析代码仓库中的
manifest 文件
clone 代码
根据工具
生成构建镜像
编译代码
metadata
制品库
打包文件
file.tar
云存储s3
生成服务镜像
image

缓存管理

美团点评

本地文件共享缓存

- 运行构建容器时通过 mount 方式挂载到编译容器中，比如代码仓库缓存与 maven 编译缓存

远程缓存

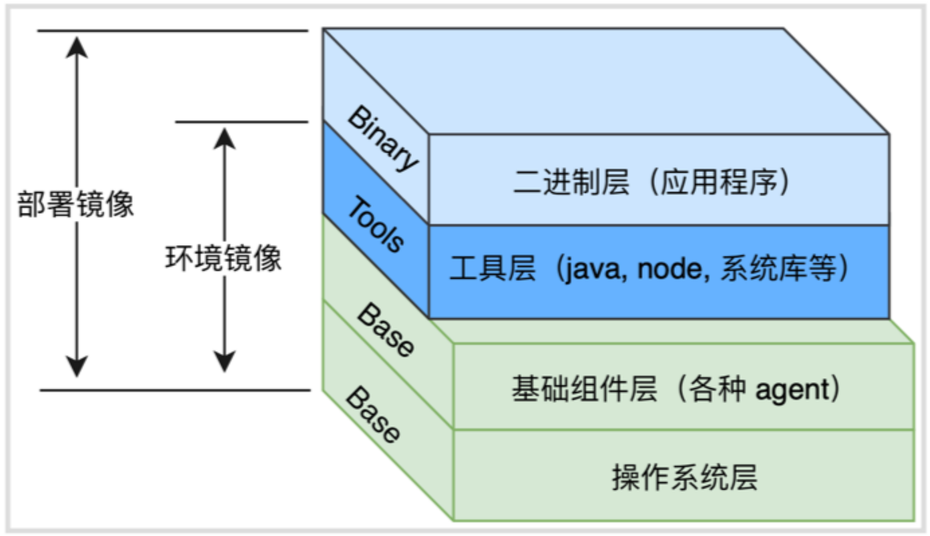
- manifest 文件描述缓存的目录，每次编译结束将目录内容上传到 S3，下次编译直接解压

```
build:
  cache:
    dirs:
      - /opt/gocache/go-build
```

适用于构建存在中间缓存文件的语言，比如 nodejs go 等

生成服务镜像

镜像的分层



环境镜像（base+tools镜像) 管理会遇到的问题

1. 构建工具环境镜像耗时
2. 基础镜像升级与工具升级

可能的解决思路

- 提前生成镜像

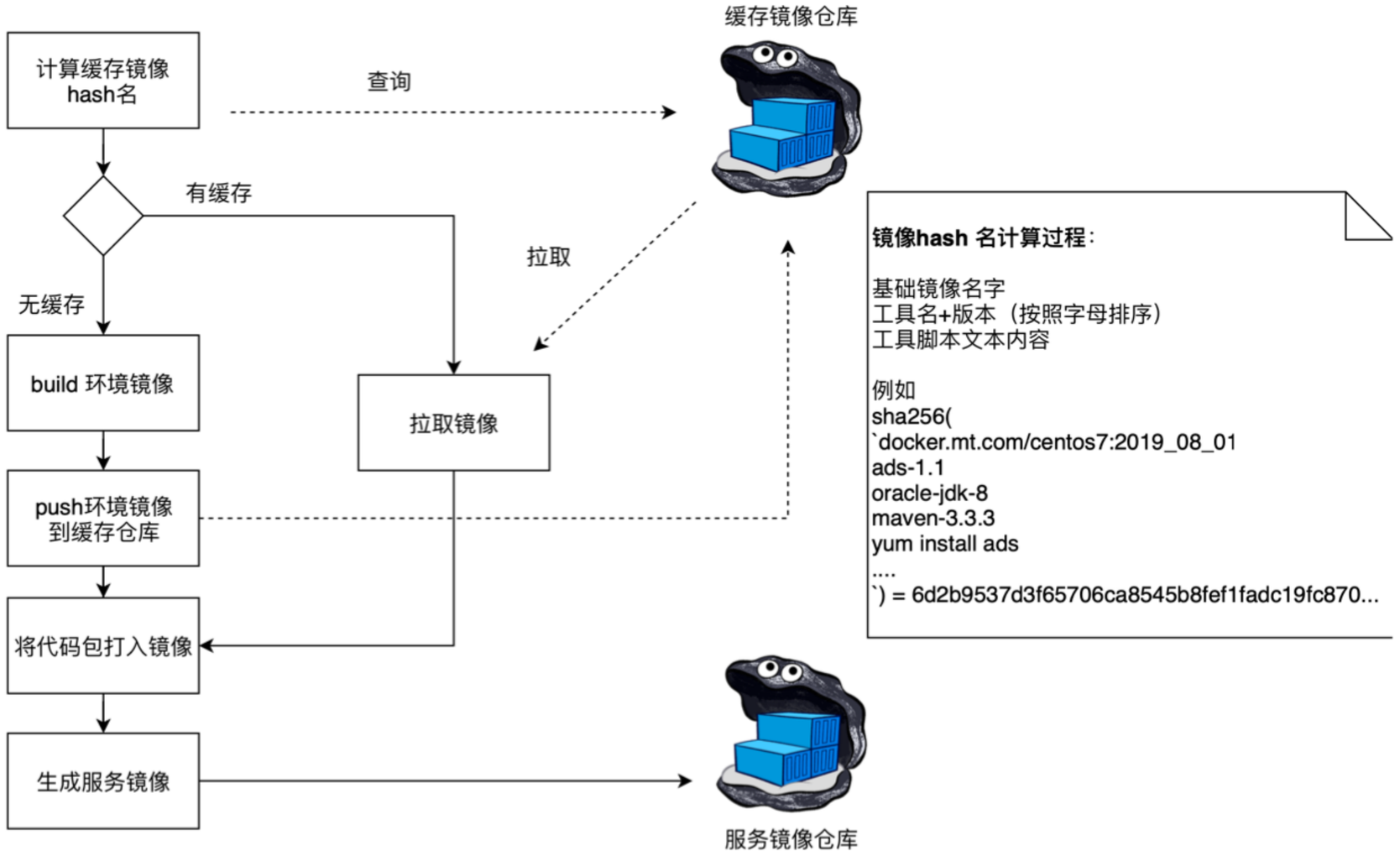
  基础os+工具组合多，升级基础镜像需要全部重新打包

- 使用大而全的环境镜像

  镜像过大，多版本工具管理成本高

- 每次重新生成耗时长

缓存环境镜像



e.g.

基础os: centos7

工具: maven3.3.3+jdk11

只要有一次构建成功下次都会直接使用cache 镜像，不会重新进行构建

无论工具/基础镜像升级对服务的构建时间影响很小

没有管理成本

分享目录

美团点评

1 系统介绍

2 任务调度

3 构建与缓存管理

4 部署实现

5 总结

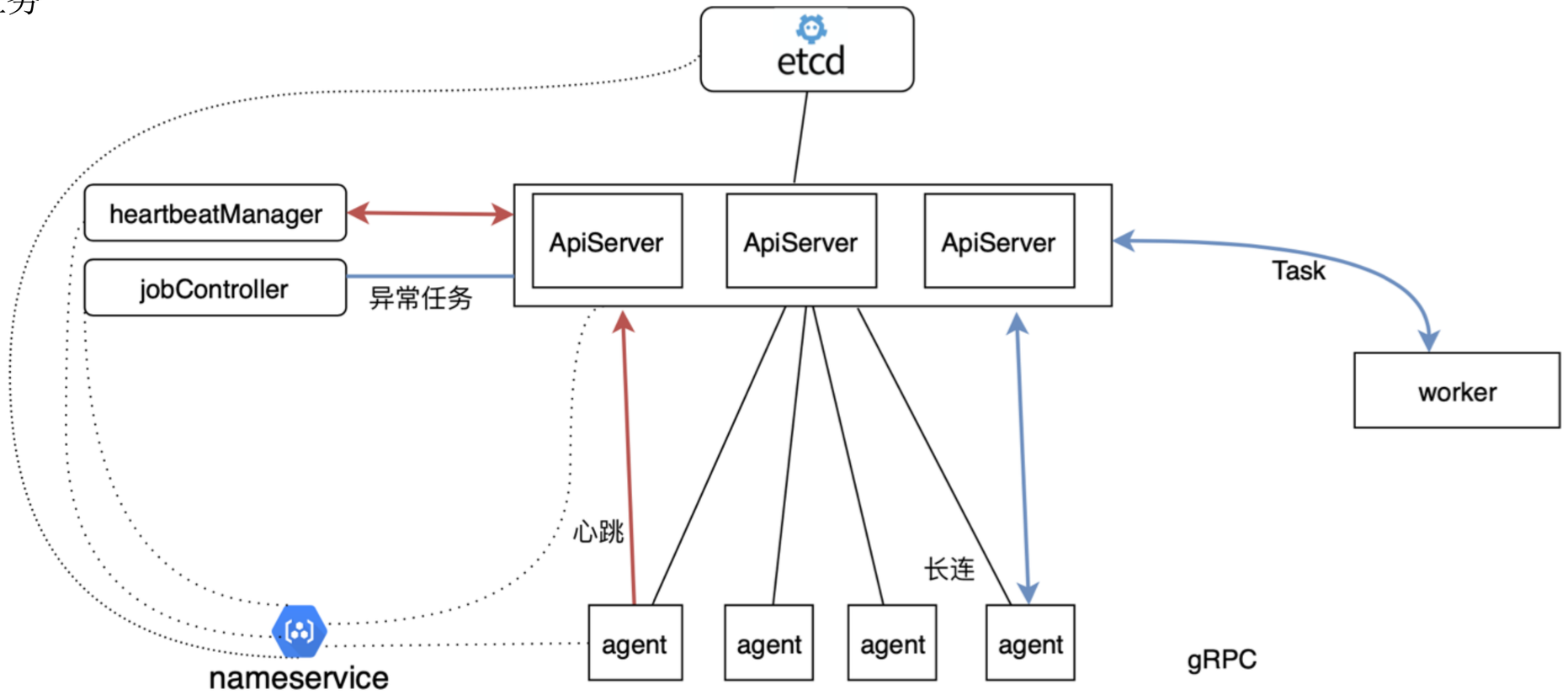
架构

heartbeatManager 处理心跳

jobController 设置异常任务

任务流向 ---

心跳流向 ---



并行度控制

发布20台机器，同时只有6台在更新

[redacted]	成功	2019-11-25 22:06:47	2019-11-25 22:18:53	登录
[redacted]	成功	2019-11-25 22:07:08	2019-11-25 22:16:24	登录
[redacted]	运行中	2019-11-25 22:13:34	-	登录
[redacted]	运行中	2019-11-25 22:15:35	-	登录
[redacted]	运行中	2019-11-25 22:15:56	-	登录
[redacted]	运行中	2019-11-25 22:16:03	-	登录
[redacted]	运行中	2019-11-25 22:16:24	-	登录
[redacted]	运行中	2019-11-25 22:18:53	-	登录
[redacted]	待调度	-	-	登录
[redacted]	待调度	-	-	登录
[redacted]	待调度	-	-	登录

使用channel控制部署并发度

美团点评

- 无锁实现
- 核心控制代码30几行
- 支持遇到一台机器部署失败后停止或继续的策略 stop_on_failure

```go
// Controller 用于控制机器按照并发度进行发布
type Controller struct {
    parallel      int
    stopOnFailure bool
    context       worker.Context

    // funnel 用于控制并发度，支持stop_on_failure 策略
    // 只有当任务部署失败的时候 funnel 会写入 true 值
    funnel chan bool
    err    error
    hosts  []*HostWrapper
}
```

```go
func (c *Controller) schedule() {
    defer close(c.funnel)
    index := 0
    finishCount := 0
    stopped := false
    // 首先开始发布 n 个机器（n为并发度），这些机器不会受到 stop_on_failure 策略影响
    for index = 0; index < c.parallel; index++ {
        go c.hosts[index].Run()
    }
    // funnel 总共会收到 n 个事件（n 为机器总数）
    for failed := range c.funnel {
        finishCount++
        if c.stopOnFailure && failed && !stopped {
            stopped = true
        }
        if index < len(c.hosts) {
            h := c.hosts[index]
            if stopped {
                go h.Cancel()
            } else {
                go h.Run()
            }
        }
        if finishCount == len(c.hosts) {
            break
        }
        index++
    }
}
```

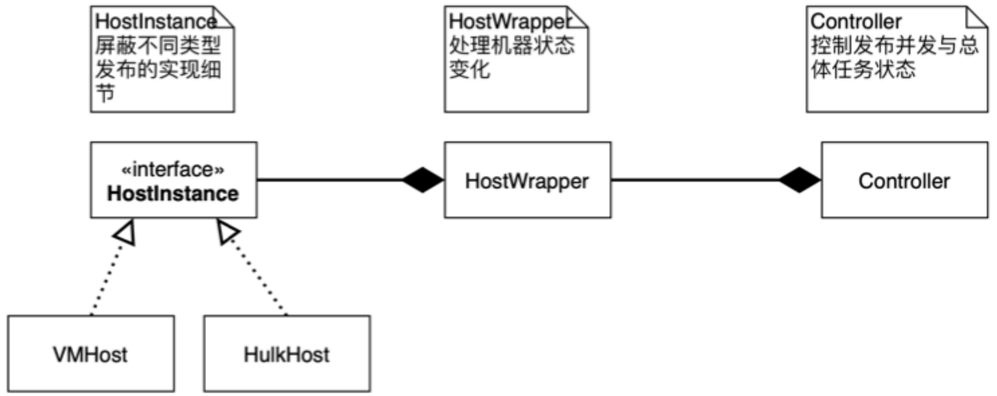
使用channel控制部署并发度

美团点评

```go
type HostInstance interface {
    Setup(common.Host, worker.Context)
    Run() error
}

type HostWrapper struct {
    Instance HostInstance
    common.Host

    hostDetail common.HostDeployDetail
    finishCh chan<- bool
    context  worker.Context
}
```



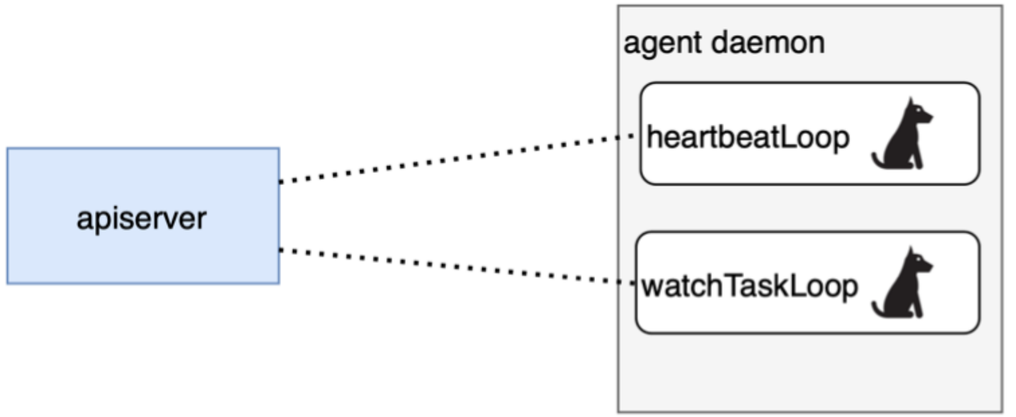

```go
func (wp *HostWrapper) Cancel() {
    wp.updateDetail(common.StatusRevoked, time.Now(), worker.ErrRevoked.Message)
    wp.finishCh <- false
}

func (wp *HostWrapper) Run() {
    var err error
    select {
    case <-wp.context.Done():
        wp.Cancel()
        return
    default:
    }
    wp.updateDetail(common.StatusRunning, time.Now(), worker.ErrRevoked.Message)
    err = wp.Instance.Run()
    if err != nil {
        log.Errorf("Host %s run error: %s", wp.Name, err)
        wp.updateDetail(common.StatusFailure, time.Now(), err.Error())
        wp.finishCh <- true
        return
    }
    wp.updateDetail(common.StatusSuccess, time.Now(), err.Error())
    wp.finishCh <- false
}
```

Agent 异常处理

agent 与 apiserver 之间长连接如何保证通常，常见问题

- 网络抖动
- 代码问题 （ context 未加超时 ）
- 连接卡住



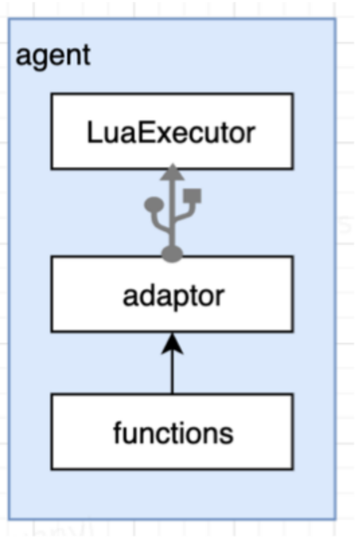
解决办法

- agent 到 apiserver 心跳 – agent 探活
- apiserver 到 agent 检测 – 长连接探活
- 重连机制 – 连接异常重连
- watchdog – 其他异常（代码实现bug，或者第三方库问题）

gopher-lua

Agent 节点数量多，更新周期长

- 新增部署流程
- bugfix



解决方案：

- 动态下发执行过程
- 使用Lua编写部署脚本
- 通过gopher-lua调用agent原有的go函数

https://github.com/yuin/gopher-lua

分享目录

美团点评

1 系统介绍

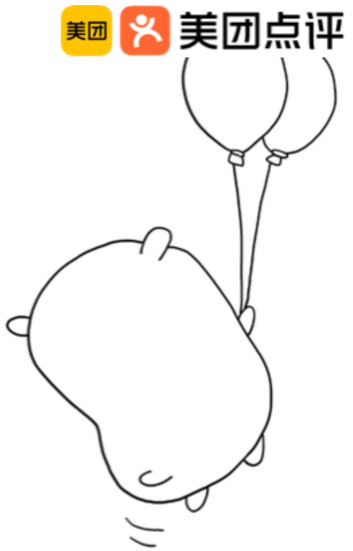
2 任务调度

3 构建与缓存管理

4 部署实现

5 总结

总结



使用go作为开发语言对项目的带来的收益

1. 维护与重构
2. 开发效率
3. 并发友好

反思

1. 注重模块分层
2. 提前划分目录结构 https://github.com/golang-standards/project-layout

Go在美团的使用情况

从16年来越来越多的项目使用 Go 开发，包括但不限于

- Plus构建部署平台
- 美团容器平台HULK
- 基础运维系统
- 美团云存储/计算
- 基础组件

更多技术干货
欢迎关注“美团技术团队”

招聘：基础技术部-Go高级研发工程师/技术专家
邮箱：songchuanyi@meituan.com

美团 美团点评

CODE A BETTER LIFE
一 行 代 码　亿 万 生 活