轻量分布式服务框架

Skynet 分析与应用

邢星（mikespook）
mikespook@gmail.com
http://mikespook.com
weibo & twitter @mikespook

云之神话

- 资源共享
- 按需分配
- 隔离抽象
- 服务式使用
- 自助化/自动化
- ……

- 软件即服务（SaaS）
- 平台即服务（PaaS）
- 基础设施即服务（IaaS）
- ……（XaaS）

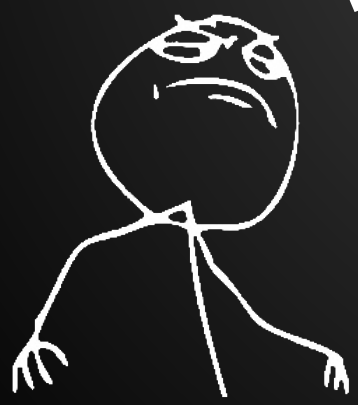
我们有什么？

- Google App Engine
- Google Compute Engine
- Amazon Web Service
- Heroku
- Openhift
- DNSPod
- 监控宝
- 七牛云存储
- ……

够了吗？

- 物理瓶颈——跨洲际、大延迟、不可控制
- 政策风险——被墙、支付困难、沟通不顺
- 历史负债——一个不可拆分的核心服务

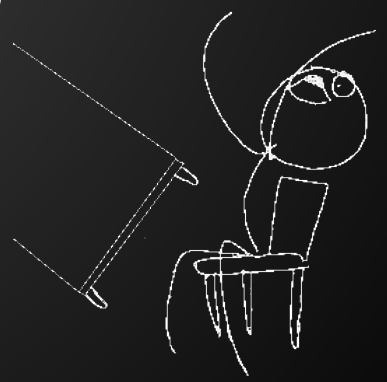
老板信任吗？



拍脑袋

- 租赁SaaS对外围功能进行支撑。
- 用 OpenStack 什么的，自建私有云……

对于核心业务……
这样有意义吗？



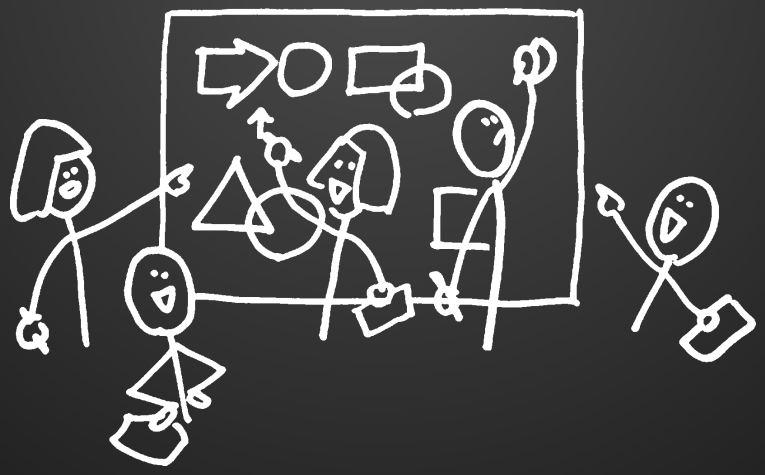
最终倒在了一个无法云化的核心业务系统上？

Brain Ketelson

- 一个巨大的代码单元实现的大应用，很难在不带来麻烦的前提下，对某些部分进行微小的修改。
- 更小的组件允许在不影响其他部分的前提下进行调整，也更容易通过增加服务器的方式增加更多的服务节点来提升系统容量。

新的问题

- 服务组件碎片化
- 服务总线成为瓶颈
- 管理单元带来的单点隐患



Skynet - 分布式服务框架

- 使用 Go 语言编写
- 用于在分布式应用中，同服务进行通讯
- 重度依赖 Doozer
- 无需管理节点，服务自我完备
- 开源，社区活跃

Doozer

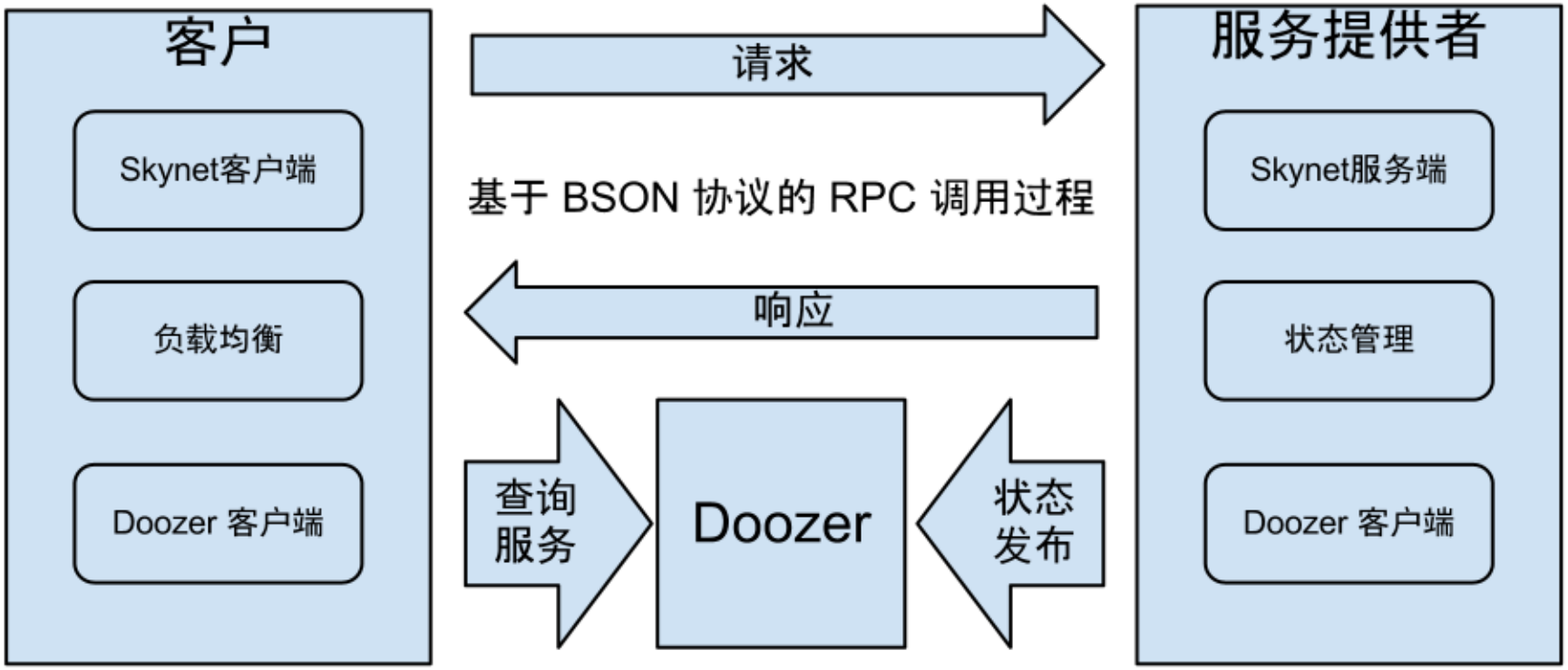
用于存储小量、极端重要的数据，保证了高可用和完全一致性。当数据变化时，它立刻通知接入的客户端(无缓存)。对于那些很少更新，但是希望更新发生时实时性高的客户端来说是非常理想的。

Doozer

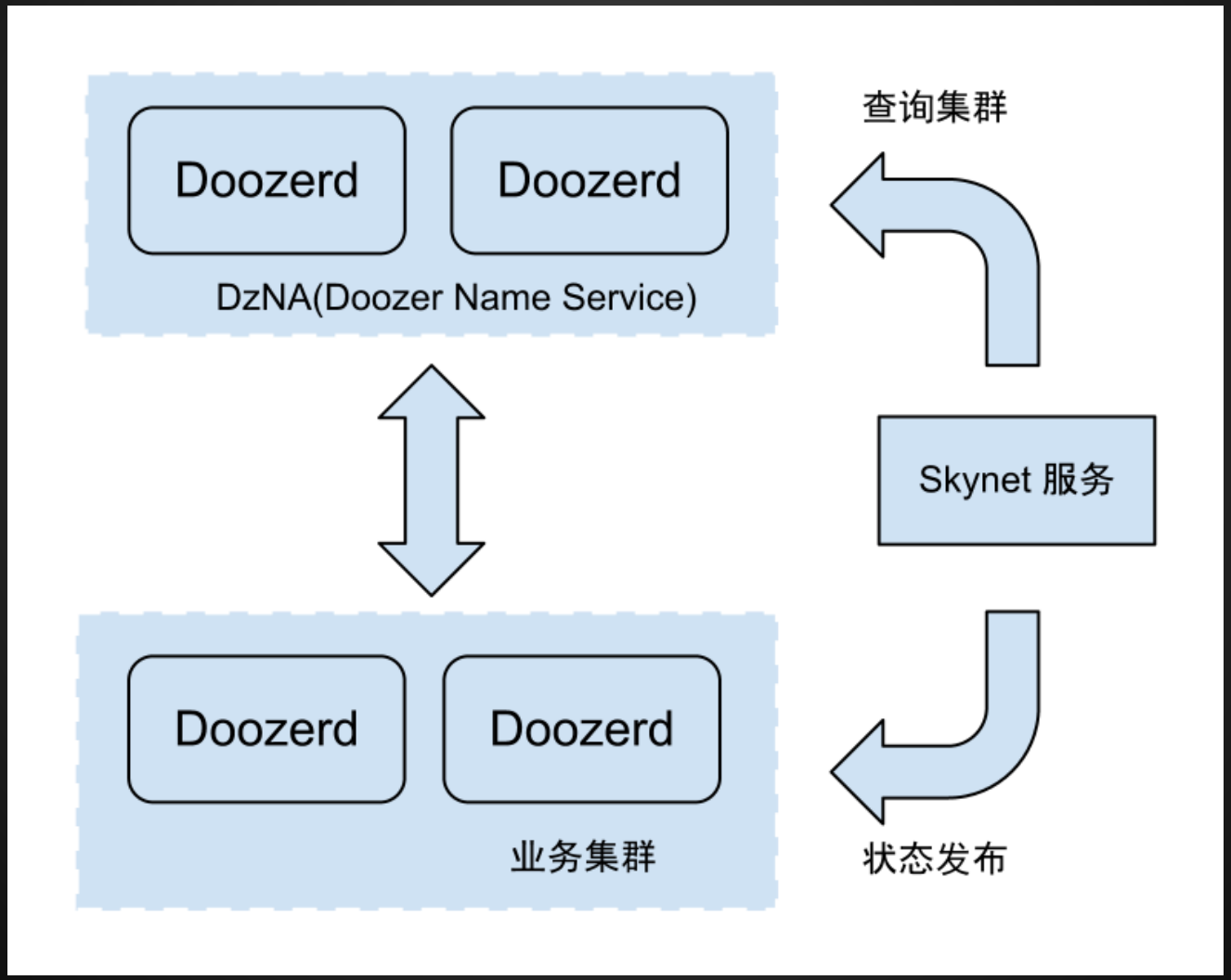
- 最初由 Heroku 的工程师开发并开源
- 轻量版 ZooKeeper
- 实现 Paxos 算法

“Skynet 服务都活在哪？”

Skynet 怎么使用 Doozer



整体架构


Doozerd
Doozerd
DzNA(Doozer Name Service)
查询集群
Skynet 服务
Doozerd
Doozerd
业务集群
状态发布

服务

- 接口

```
type ServiceDelegate interface {
    Started(s *Service)
    Stopped(s *Service)
    Registered(s *Service)
    Unregistered(s *Service)
}
```

- 调用原型

```
func (ri *skynet.RequestInfo,
    req interface{},
    resp interface{}) error
```

声明

- 请求和响应

```
type MsgRequest struct {
    Sender, Receiver, Data string
}
type MsgResponse struct {
    Code int
}
```

- 服务

```
type MsgService struct {
    // 消息发送服务所需数据
}
```

实现

```
func (s *MsgService) SendMsg(
    ri *skynet.RequestInfo,
    req *MsgRequest,
    resp *MsgResponse) error {
    // 发送消息的代码
    return err
}
```

主函数

```
func main(){
    s := &MsgService{}
    config, _ := skynet.GetServiceConfig()
    config.Name = "MsgService"
    config.Version = "1"
    config.Region = "Development"
    srv := service.CreateService(s, config)
    defer srv.Shutdown()
    waiter := srv.Start(true)
    waiter.Wait()
}
```

客户端

```
config, _ := skynet.GetClientConfig()
clt := client.NewClient(config)
req = &MsgRequest {Sender: "user1",
    Receiver: "user2", Data: "Blablabla..."}
resp = &MsgResponse{}
```

客户端

```
srv := clt.GetService(
    "MsgService", "1", "Development", "")
if err := srv.Send(nil,
    "SendMsg", req, resp);err != nil {
    fmt.Println(err)
} else {
    fmt.Println(resp.Code)
}
```

轻量分布式服务框架

- 需要分布式、高可用的协调服务
  - Zookeeper/Doozer/ESB
- 开放的通讯协议有助于跨语言的服务调用
  - Protobuf/BSON/JSON/XML/SOAP
- 民主制负载均衡消除了裁判服务的单点隐患
  - 服务端发布自身状态，客户端根据状态进行选择
  - 民主带来的风险
- 每个人都有自己的天堂
  - Doozer, Skynet
  - Zookeeper, Norbert
  - MQ/Redis/Memcache...

延伸阅读

1. 美国国家标准与技术研究院对云计算的定义
   - http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
2. 什么是 GAE
   - https://developers.google.com/academy/apis/cloud/appengine/intro/whatisgae
3. Norbert, LinkedIn 发布的基于 Zookeeper 的轻量分布式开源框架
   - http://data.linkedin.com/opensource/norbert
4. ZooKeeper 之道
   - https://cwiki.apache.org/confluence/display/ZOOKEEPER/Tao