服务治理

许式伟@七牛

2016-4-22

自我介绍

- 关于许式伟
  - 七牛云CEO
  - qlang 语言创始人
  - 前Go语言大中华区首席布道师
  - 前盛大祥云计划、盛大网盘发起人
  - 前金山实验室发起人
  - 前WPS Office首席架构师
- 关于七牛云
  - 立足富媒体(存储/数据处理/CDN/视频直播)，打造综合云服务平台
  - 致力于改善中国的创新环境，共同完成中国制造=>中国创造

关于qlang

- A script language cooperating with Go
- Go-alike syntax
- Calling Go functions without any wrapper
  - So, standard libraries implemented already by Go
- Support most of Go features, including
  - for range
  - string, slice, map, chan
  - goroutine, closure, defer
  - TODO: select
- Homepage: https://github.com/qiniu/qlang

```
import "fmt"
import "log"
import "strings"

import "qlang.io/qlang.v2/qlang"
import   _ "qlang.io/qlang/builtin"

var strings_Exports = map[string]interface{}{
   "replacer": strings.NewReplacer,
}

func main() {

   qlang.Import("strings", strings_Exports)
   lang, err := qlang.New(qlang.InsertSemis)
   if err != nil {
        log.Fatal(err)
   }

   err = lang.SafeEval(`x = strings.replacer("?", "!").replace("hello, world???")`)
   if err != nil {
        log.Fatal(err)
   }

   v, _ := lang.Var("x")
   fmt.Println(v) // Output: hello, world!!!
}
```

话题来由：服务治理=?

- 互联网的胜利
  - 从娱乐为主到百姓日常生活
  - 到治病救人（领域越来越严肃）
- 服务端程序员的特殊职责：on call
  - 服务光考虑完成功能是不行的
  - 要为跑得好做很多很多准备
- 服务治理
  - 我们试图用系统化的方法，达到
    - 让系统有更高的可用性（越来越好的稳定性）
    - 让每个服务只需要很傻就可以（业务开发更容易）

服务治理的问题域

- 服务发现与负载均衡
- 配置中心化及变更管理
- 服务监控与扩容缩容
- 授权与防攻击
  - 不能一台机器被攻下所有机器遭殃
- 服务自我保护(不能被某些客户拖垮)
- 服务降级(放弃一些不重要的特性)
- 上线验证与灰度发布
- 错误分析与调用链跟踪
- 机器负载与服务实例的调度
- ...

服务

- SOA (service-oriented architecture)
- Microservice architecture
  - http://martinfowler.com/articles/microservices.html
- API based architecture

微服务

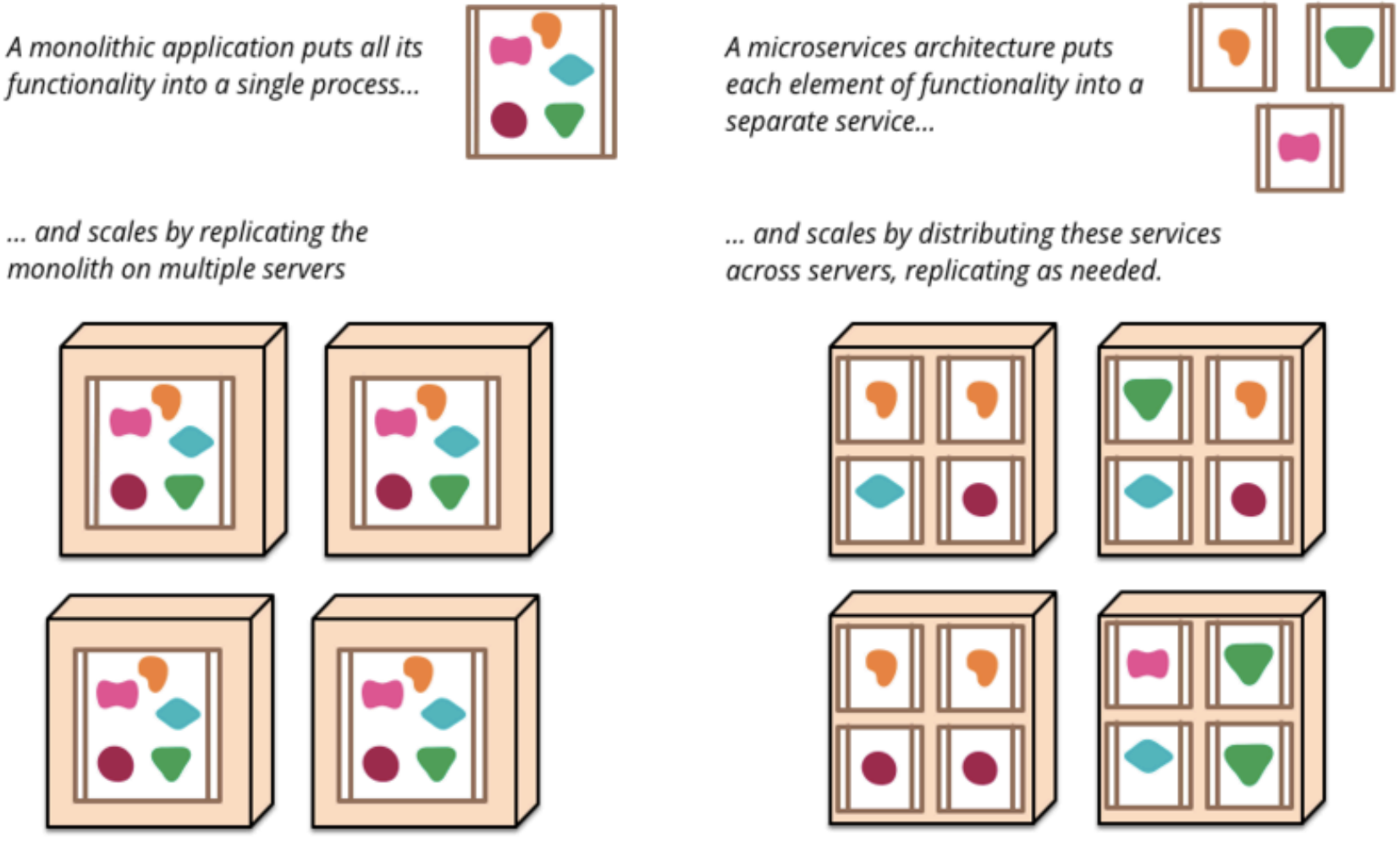


Figure 1: Monoliths and Microservices

传统组织

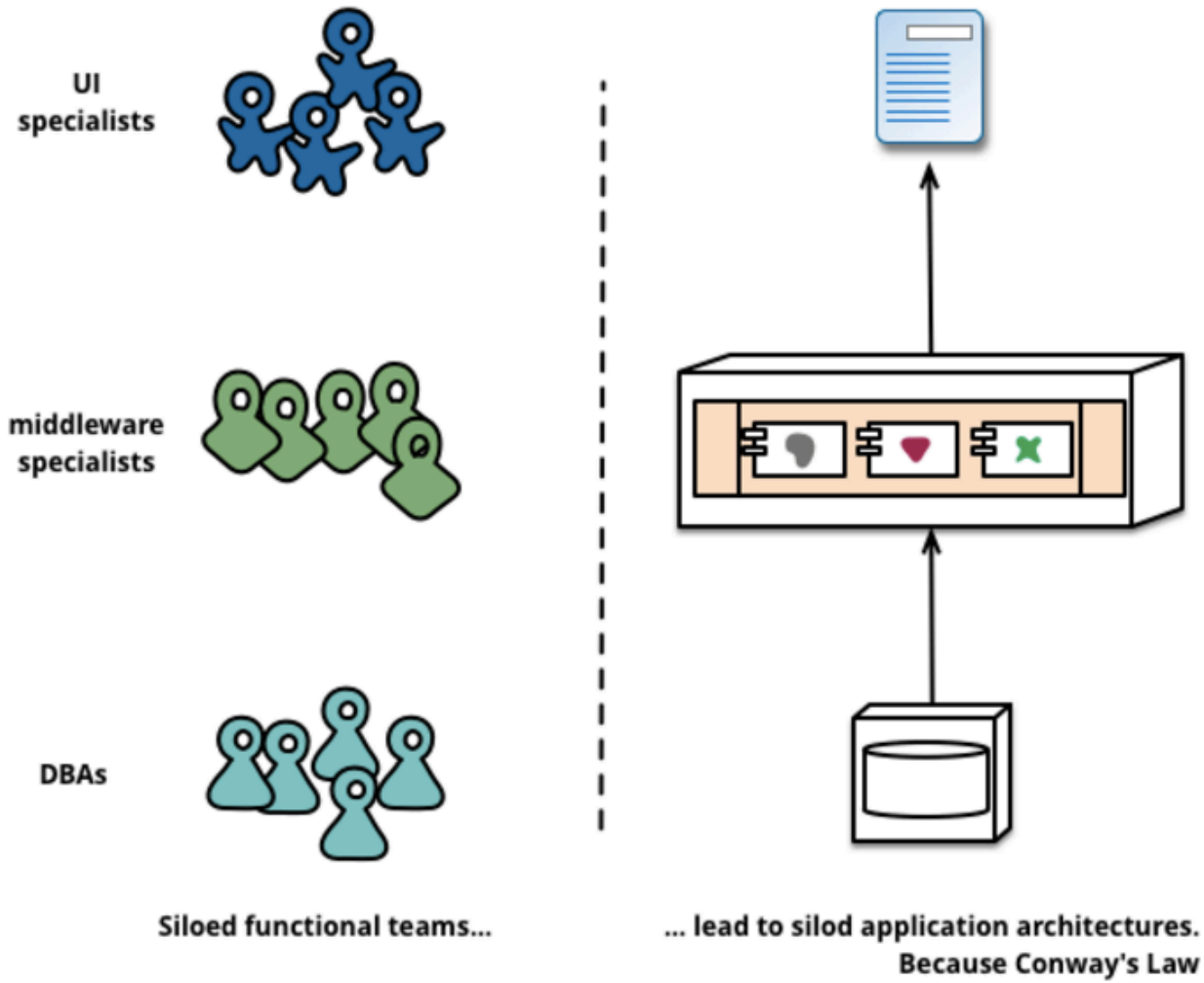


Figure 2: Conway's Law in action

微服务化组织

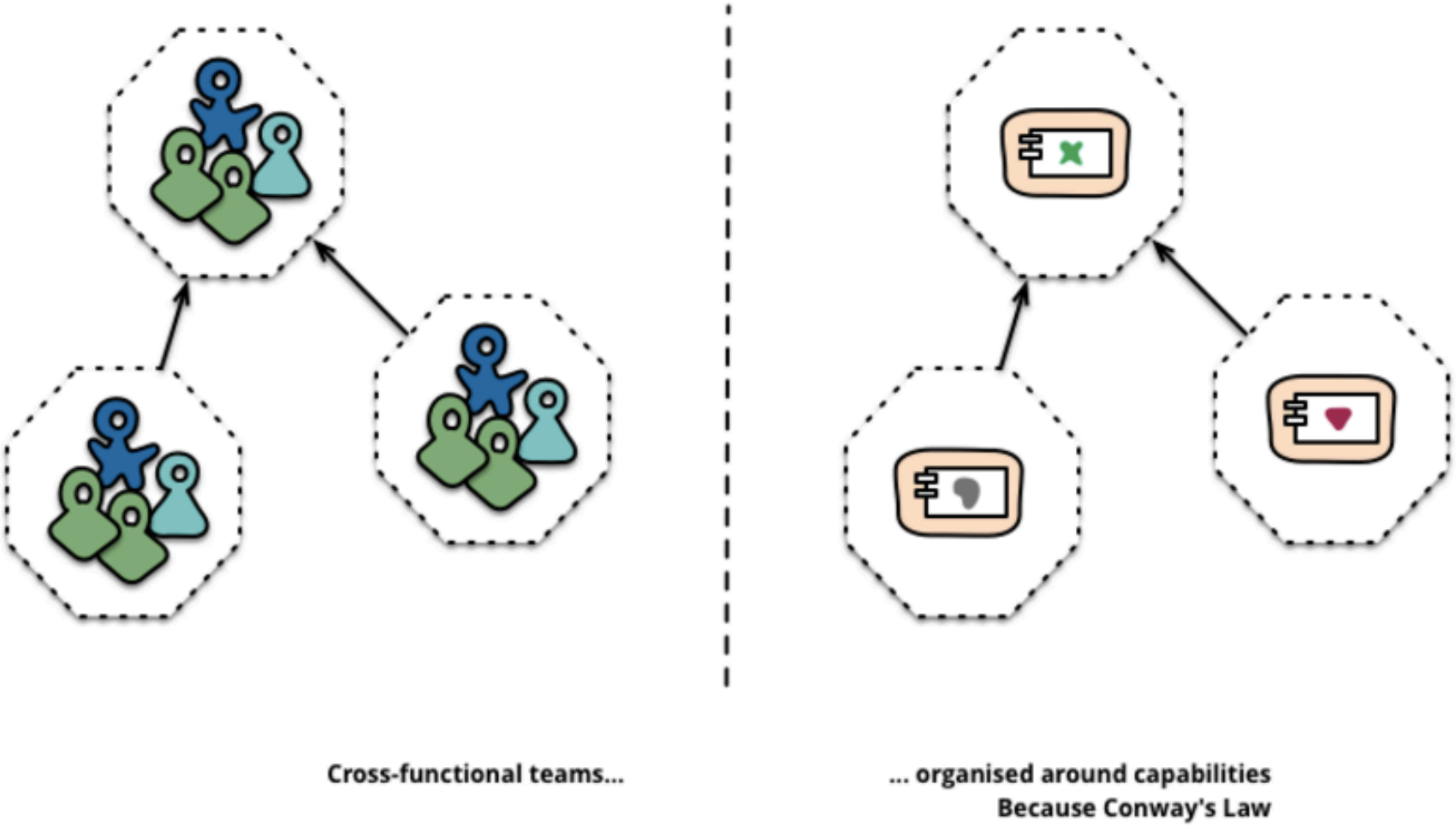
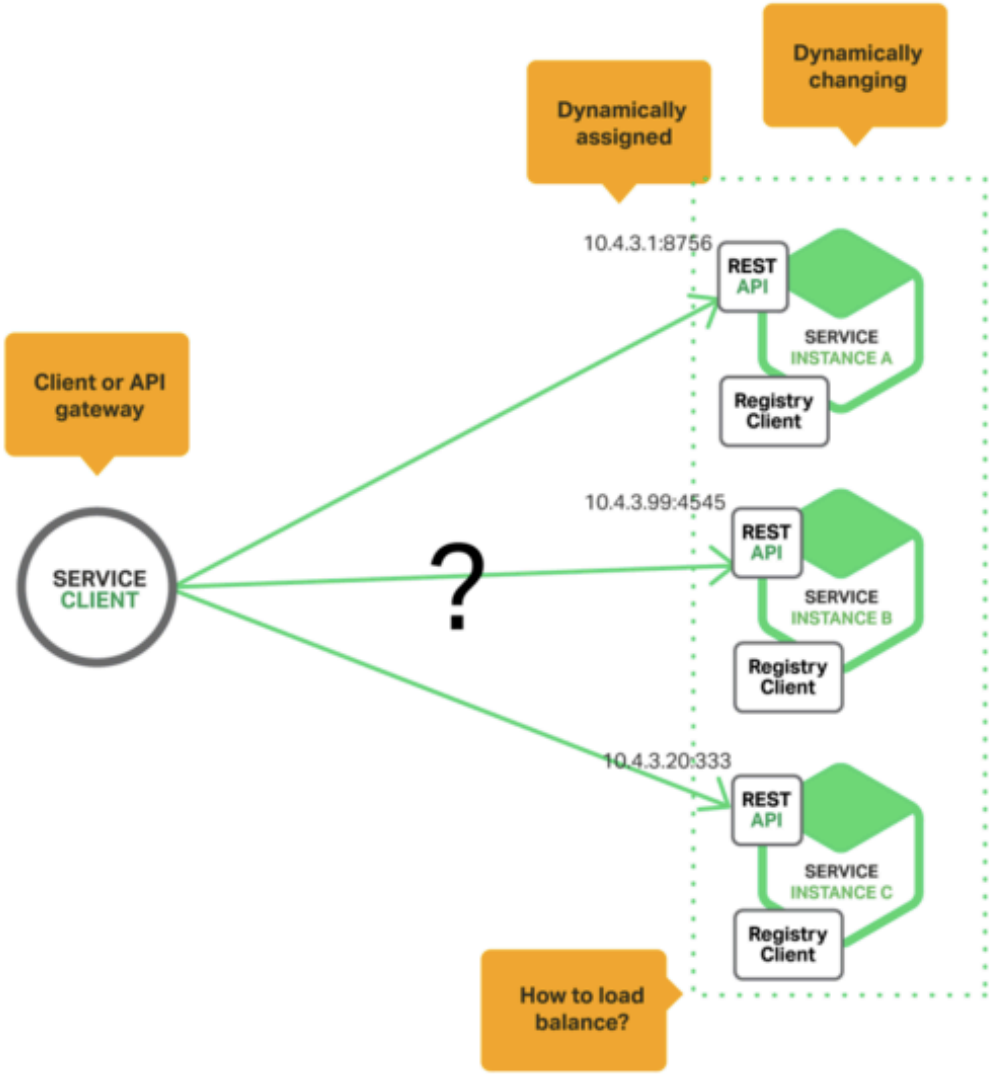


Figure 3: Service boundaries reinforced by team boundaries

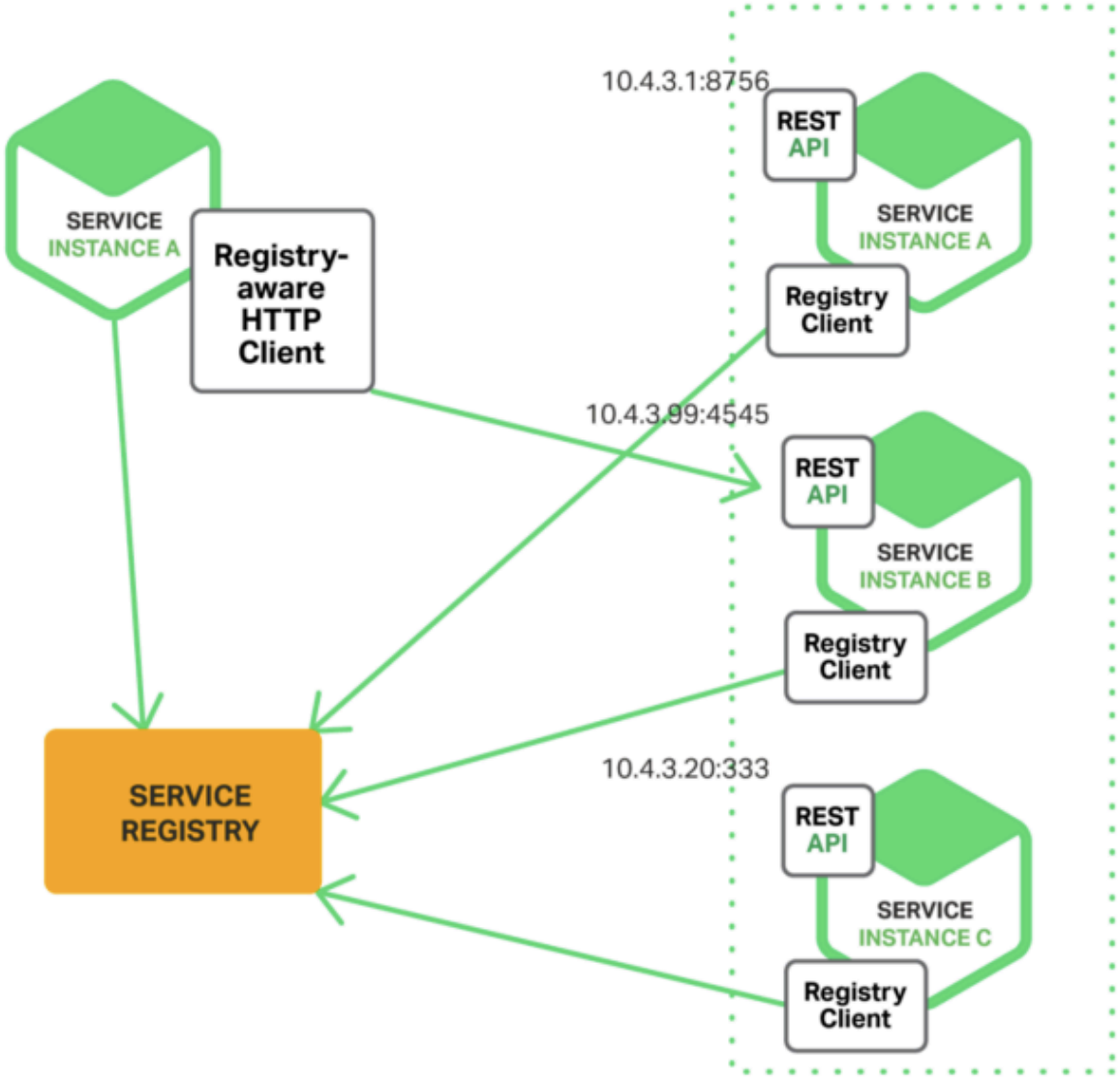
服务发现

- 为什么需要服务发现？
  - 服务的扩容缩容
  - 服务实例的调度
    - 服务迁移
- 我应该把请求发给谁？
  - 负载均衡

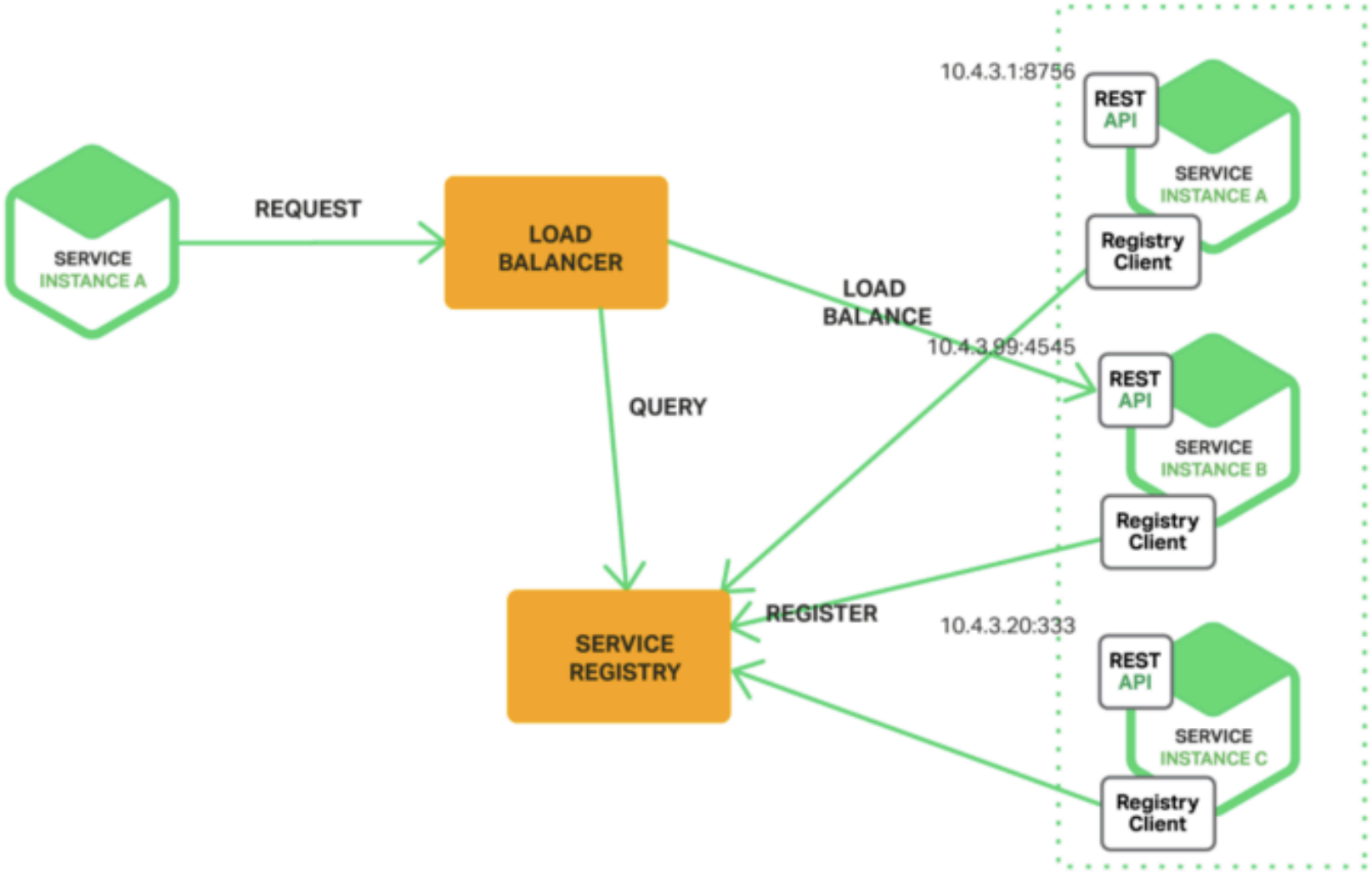
服务发现


Dynamically changing
Dynamically assigned
10.4.3.1:8756
REST API
SERVICE INSTANCE A
Registry Client
Client or API gateway
SERVICE CLIENT
?
10.4.3.99:4545
REST API
SERVICE INSTANCE B
Registry Client
10.4.3.20:333
REST API
SERVICE INSTANCE C
Registry Client
How to load balance?

服务发现：客户端(Smart Client)



服务发现：负载均衡器


SERVICE
INSTANCE A
REQUEST
LOAD
BALANCER
QUERY
LOAD
BALANCE
SERVICE
REGISTRY
REGISTER
10.4.3.1:8756
REST
API
SERVICE
INSTANCE A
Registry
Client
10.4.3.99:4545
REST
API
SERVICE
INSTANCE B
Registry
Client
10.4.3.20:333
REST
API
SERVICE
INSTANCE C
Registry
Client

Smart client vs. Load balancer

- Smart client
  - 更好的性能*
- Load balancer (API gateway)
  - 更好的治理能力

服务发现与负载均衡

- 建议方式
  - 用一个 Service IP 表征服务位置
  - 由 Load balancer (API gateway) 做负载均衡，而不是 Smart client
  - 尽可能不影响网络性能，所以 Load balancer 服务最好是分布式的(每台物理机有一个)
- 参考实现：Kubernetes

服务监控与扩容缩容

过载保护

- 资源
  - Socket connections
  - File handles
  - Network, Disk I/O, CPU, Memory
  - …
- 过载保护措施
  - 自动扩容
  - 主动拒绝一部分请求

自动扩容与过载保护

- 自动扩容是复杂的
  - 过载有可能是业务发展好，也可能是系统异常
  - 例子
    - 自动扩容的条件是：Socket connections > Limit
  - 问题在哪？
    - 有可能过载是由于后端的数据库慢了
    - 自动扩容可能反而会加大数据库压力，从而系统更加异常

自动扩容

- 建议方式
  - 要区分好请求与坏请求 (SLA)
    - code = 200, duration < T0
  - 好请求增加才自动扩容；坏请求增加应该报警
  - 无论如何，自动扩容要设置扩容上限
    - 比如 2*N (其中N为日常该服务的实例数)

过载保护：拒绝部分请求

- 系统异常情况下，拒绝部分请求是服务自我保护的重要机制
  - 什么情况下触发保护？
  - 应该丢弃哪些请求？

过载保护：拒绝部分请求

- 假设：N = 告警值
- 那么
  - 保护时机：资源到达 N*2 时进行过载保护
    - 要保证系统处于过载状态时告警仍然还在，而不是因为过载保护告警自动消失了
  - 拒绝哪些请求？
    - 最简单：拒绝新请求
    - 最直接：干掉最老请求，接受新请求
    - 最聪明：预测新请求是否是坏请求，如果是就干掉

服务降级

- 当过载保护机制失效时，考虑服务降级
  - 为重要请求独立保证资源
    - 例如：为保护读请求不受影响，独立分配只读的数据库作为资源
  - 主动放弃不重要的请求
    - 例如：所有写请求直接丢弃

拓扑发现及故障定位

- Automatically discover cluster topology
- Find root cause of failure
- Recover service as fast as possible

七牛求才

- 地点：北京/上海
  - 技术总监
  - 架构师
  - 资深研发
- 技术领域：
  - 服务治理
  - 富媒体/存储/直播
  - 容器/服务端中间件/大数据
- 邮件：hr@qiniu.com
- 微信：xushiwei (注明七牛求才)