

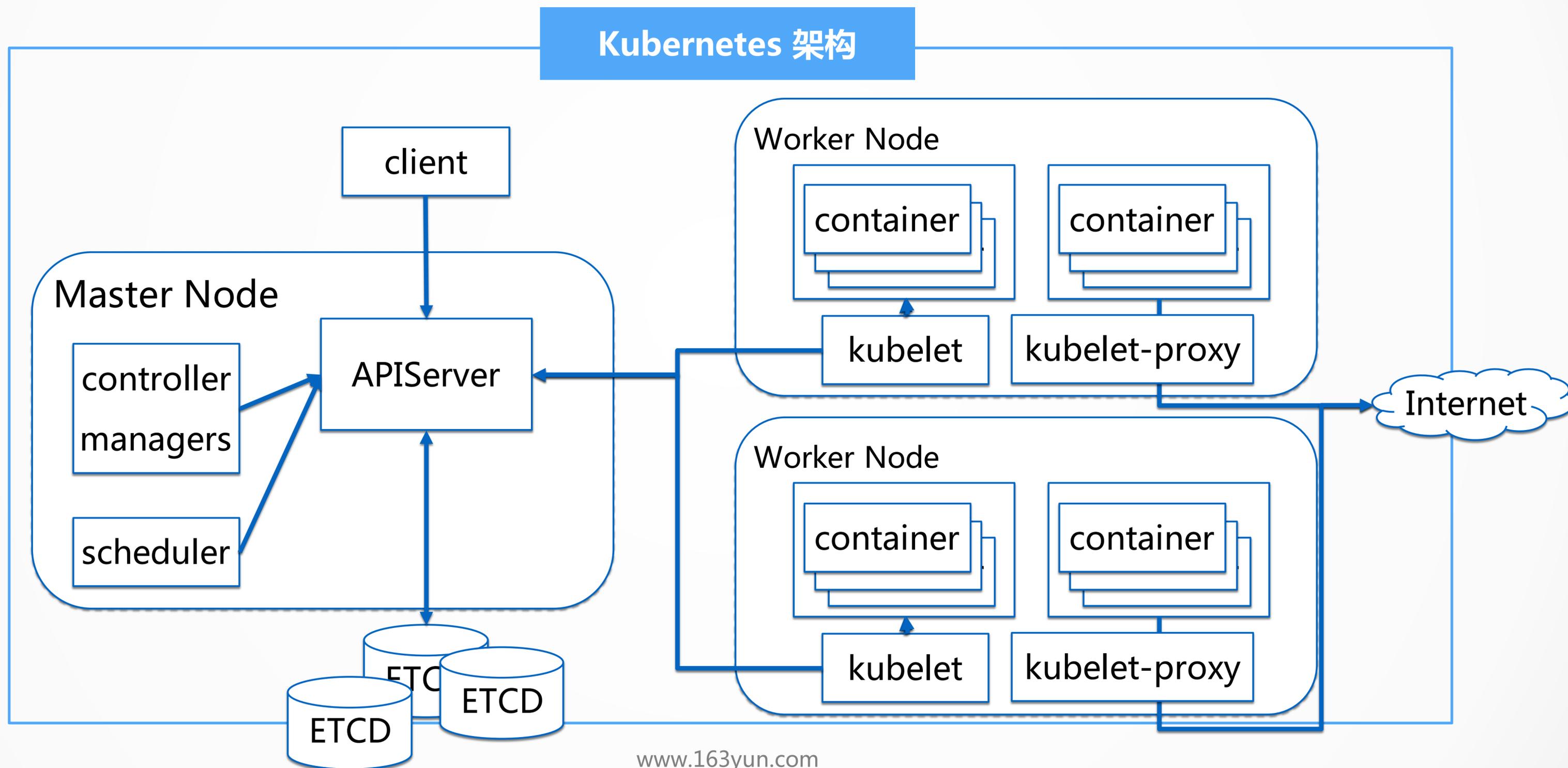
网易云 Kubernetes 性能优化实践

赖冬林, 2017-7-23

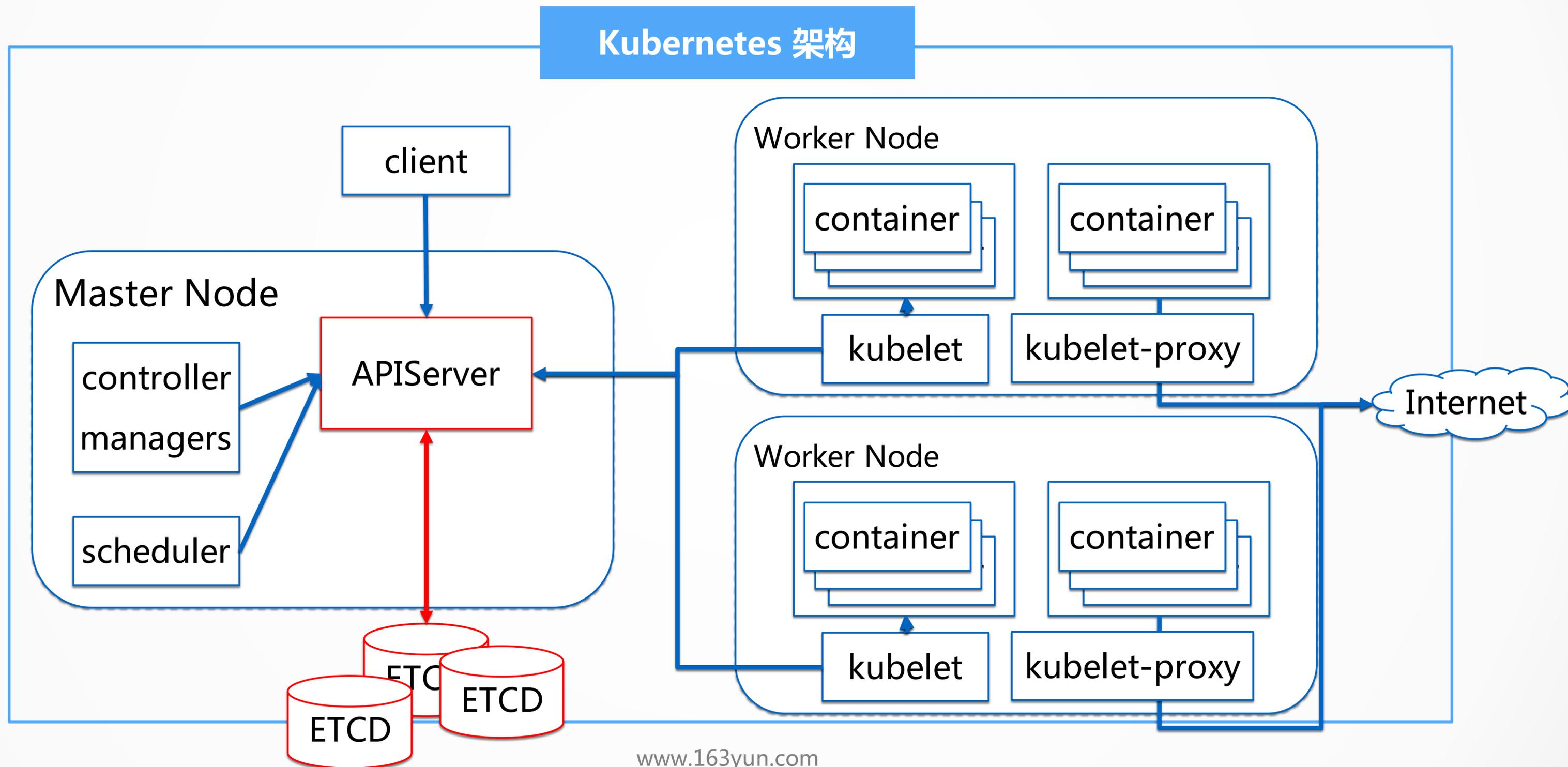


为什么要优化 Kubernetes?

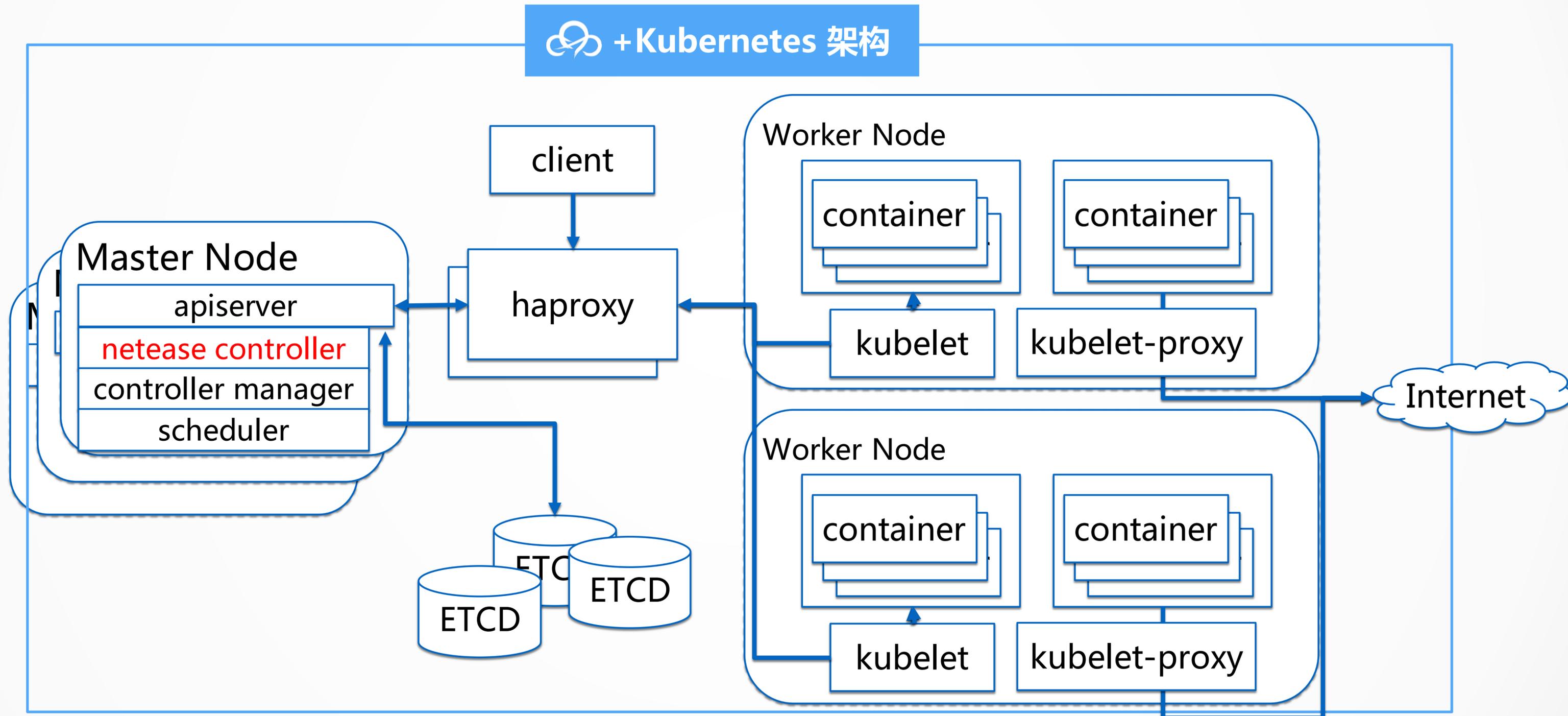
为什么要优化 Kubernetes ?



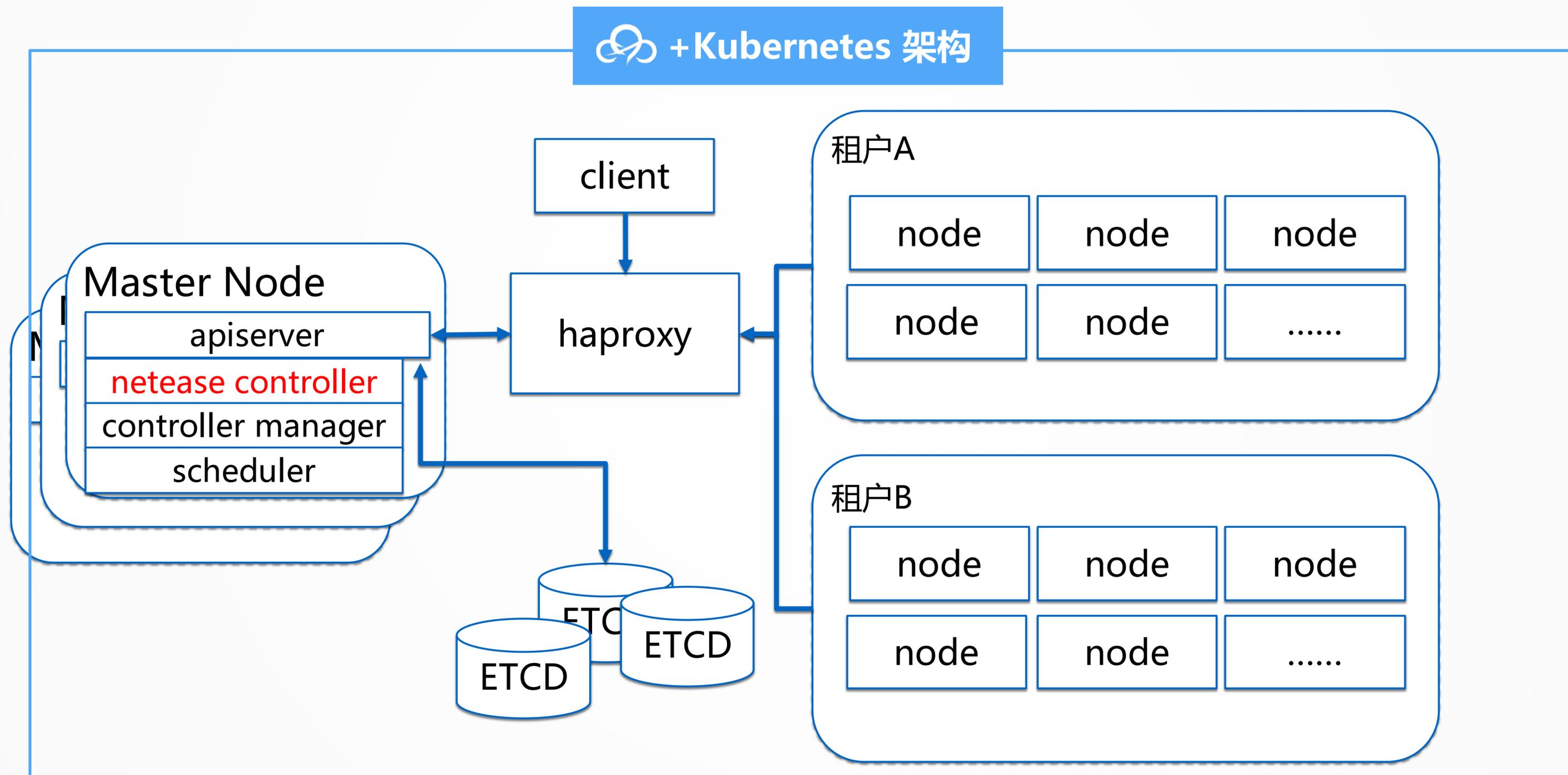
为什么要优化 Kubernetes ?



为什么要优化 Kubernetes ?



为什么要优化 Kubernetes ?



为什么要优化 Kubernetes ?

业务压力

- 服务于网易互联网产品
- 云音乐
- 考拉
- 网易新闻
- 云阅读
- 云课堂
-

产品需求

- 面向公有云
- 多租户
- 专属云
-

最重要的是：节省成本！为老板省钱！

为什么要优化 Kubernetes ?

原生版本性能

- 1.0 – 100 Node / 3000 Pod
- 1.3 – 2000 Node / 60k Pod
- 1.6 – 5000 Node / 150k Pod

都远达不到我们的目标是：单集群 **30k Node**

如何优化 Kubernetes?

如何优化 Kubernetes ?

基本原则

- 方法论 USE
- 指标收集
- 性能剖析
- 优化措施

如何优化 Kubernetes ?

方法论 USE

- Utility : CPU/内存/磁盘/网络利用率
- Saturation : 各种队列长度/饱和度
- Error : 协议栈错误、应用层错误日志



By Brendan Gregg

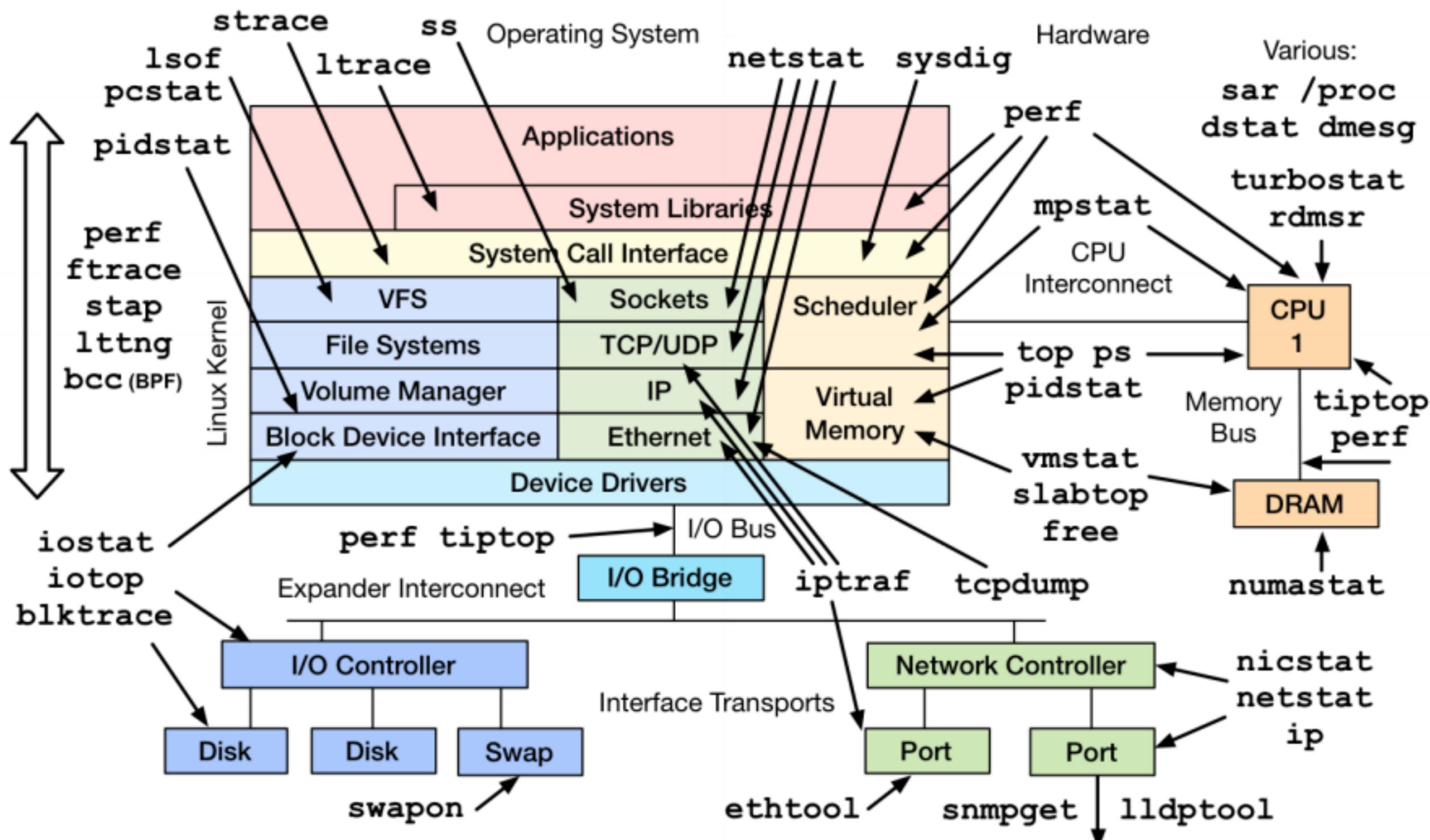
<http://www.brendangregg.com/usemethod.html>

如何优化 Kubernetes ?

指标收集

- 系统指标：top/sysstat 包等(debian)
- 应用指标：prometheus + grafana

如何优化 Kubernetes ?



如何优化 Kubernetes ?

Perf Analysis in 60s

- 1. uptime
- 2. dmesg | tail
- 3. vmstat 1
- 4. mpstat -P ALL 1
- 5. pidstat 1
- 6. iostat -xz 1
- 7. free -m
- 8. sar -n DEV 1
- 9. sar -n TCP,ETCP 1
- 10. top
- load averages
- kernel errors
- overall stats by time
- CPU balance
- process usage
- disk I/O
- memory usage
- network I/O
- TCP stats
- check overview

如何优化 Kubernetes ?

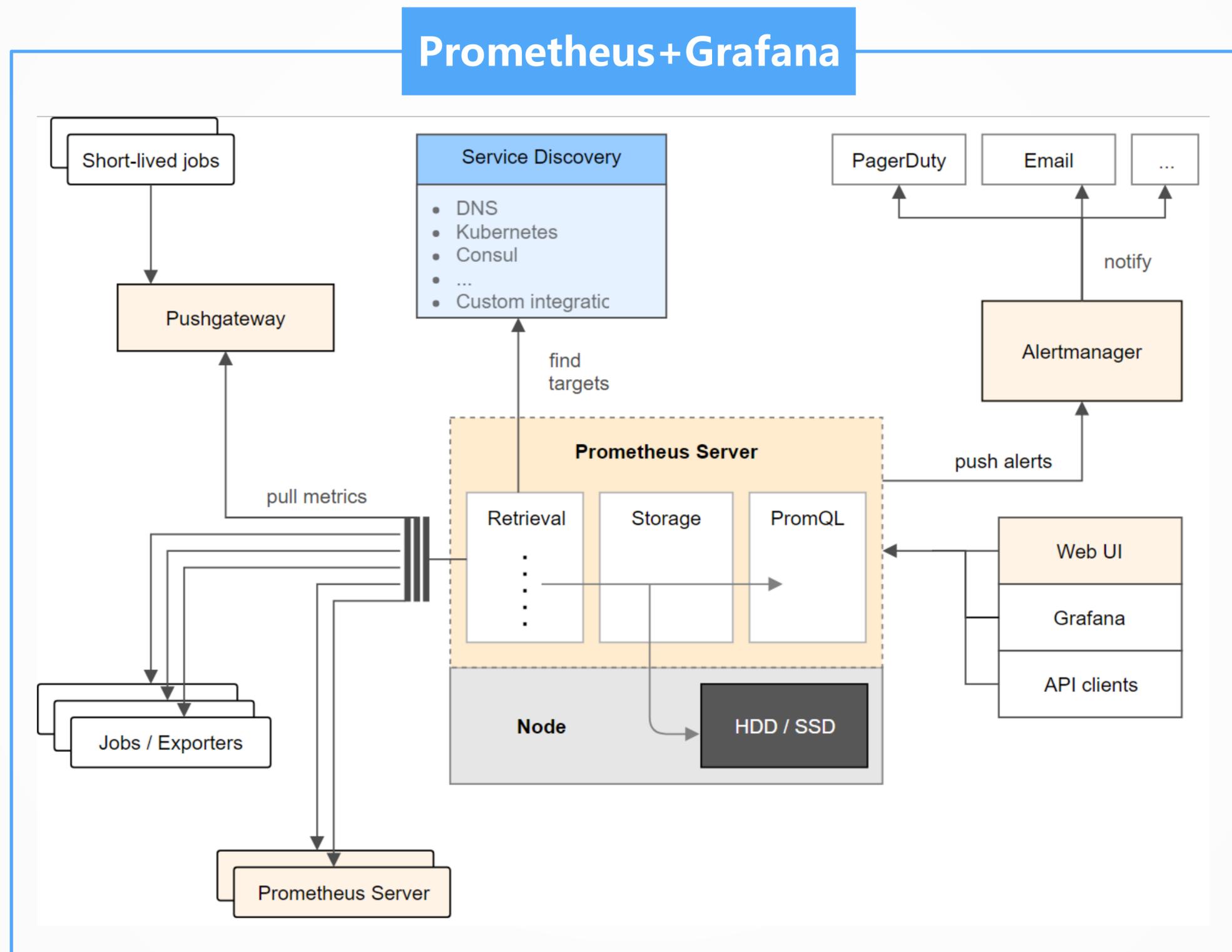
举个栗子

- netstat/sar 网络监控

```
TcpExt:
  867 SYN cookies sent
 1063 SYN cookies received
 2470 TCP sockets finished time wait in fast timer
  25 delayed acks sent
Quick ack mode was activated 150 times
 36984 times the listen queue of a socket overflowed
 36984 SYNs to LISTEN sockets dropped
  30 packets directly queued to recvmsg prequeue.
```

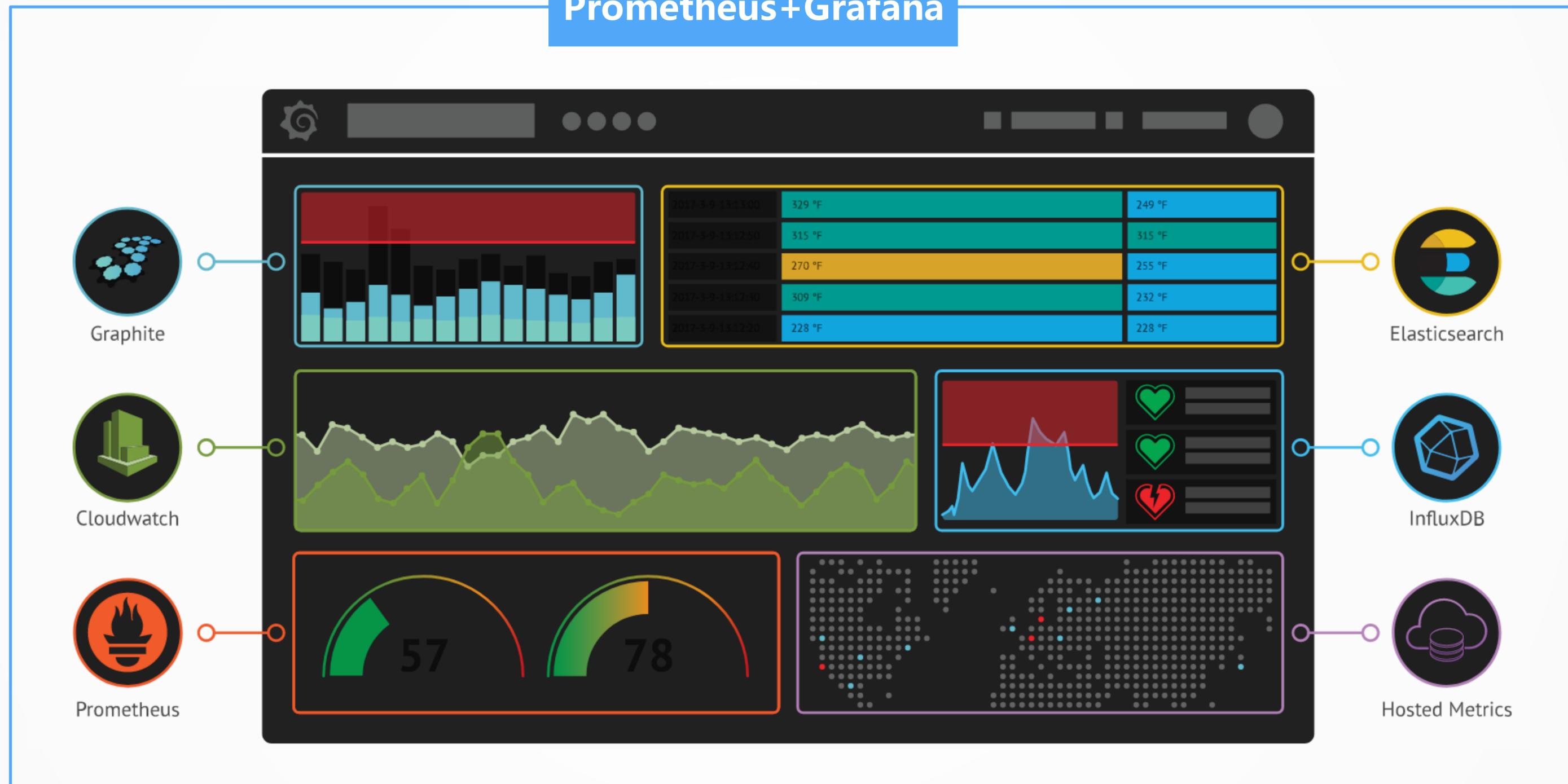
- 重启时发现异常全连接队列太小
- sar -n DEV,ETCP 1 通常会有很高的 retrans

如何优化 Kubernetes ?



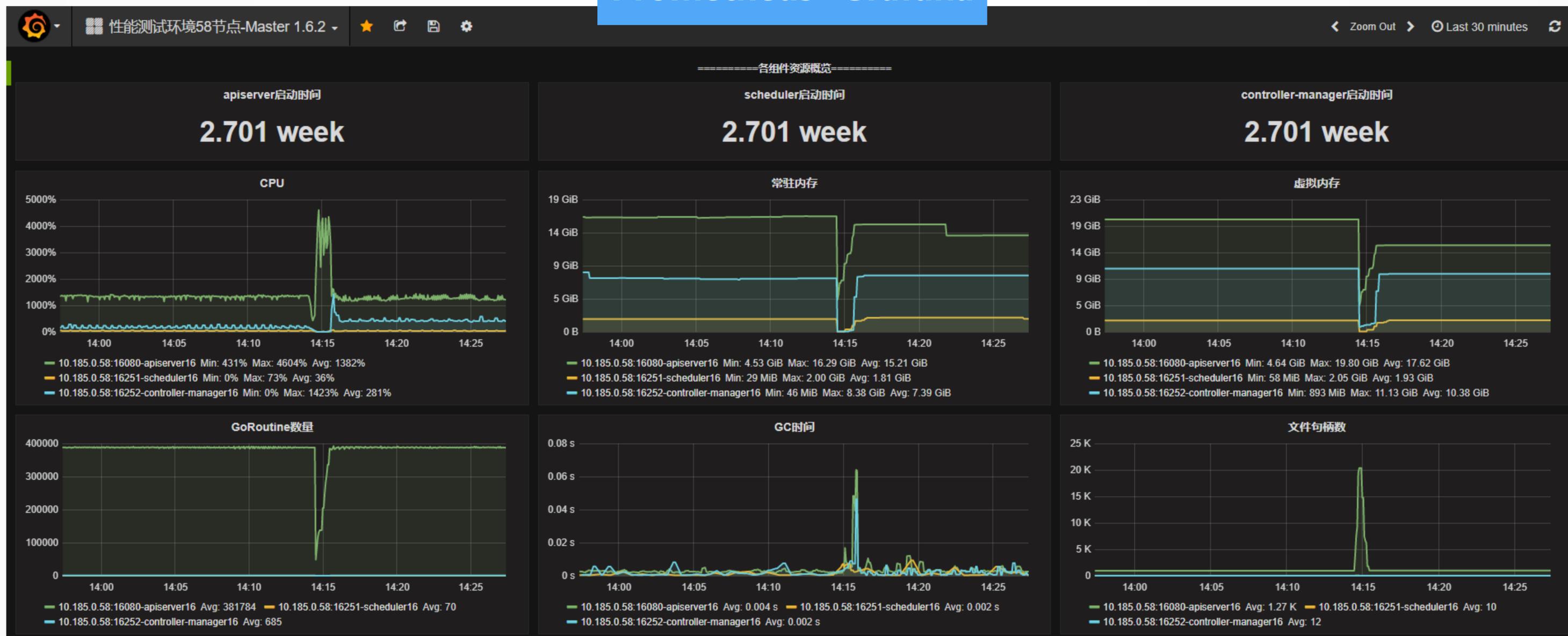
如何优化 Kubernetes ?

Prometheus+Grafana



如何优化 Kubernetes ?

Prometheus+Grafana



如何优化 Kubernetes ?

Prometheus+Grafana

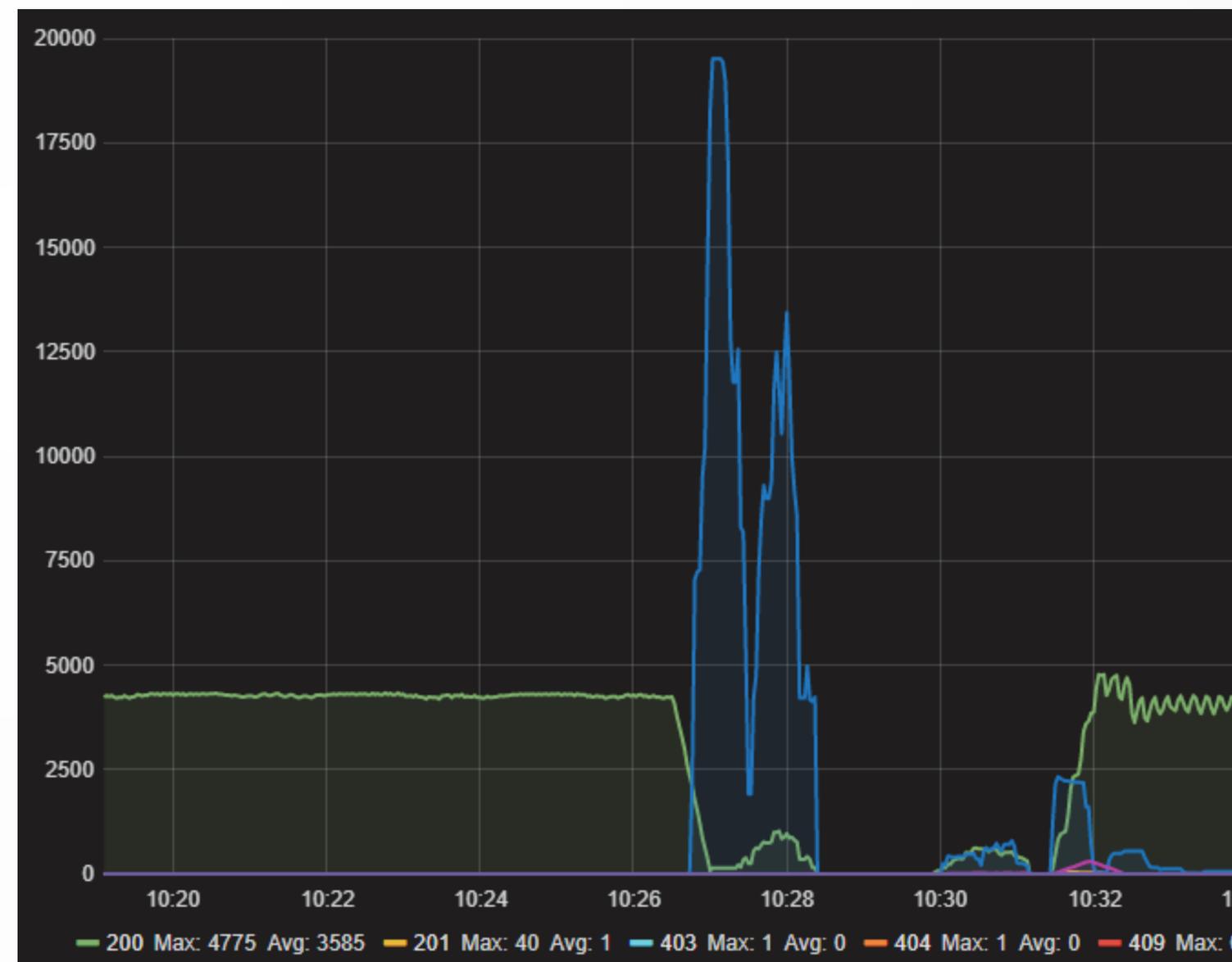
- 增加多种 metrics
- apiserver 与 etcd 访问的延迟 (operation&resource)
- apiserver 流控比例
- Controller manager Sync 延迟统计 (resource)
- Controller manager 队列饱和度
- Scheduler 队列饱和度
-

如何优化 Kubernetes ?

apiserver响应码统计

- 例如：APIServer 响应码统计
- 背景：Node – 2.5w / Pod – 15w
- 操作：重启 APIServer
- 现象：
 - 平时主要是200(绿色)，4600个每秒
 - 重启429(蓝色)异常增高，最多近2w个
 - 有段时间没有 200 的响应
 - 重启时间长达近 7 分钟

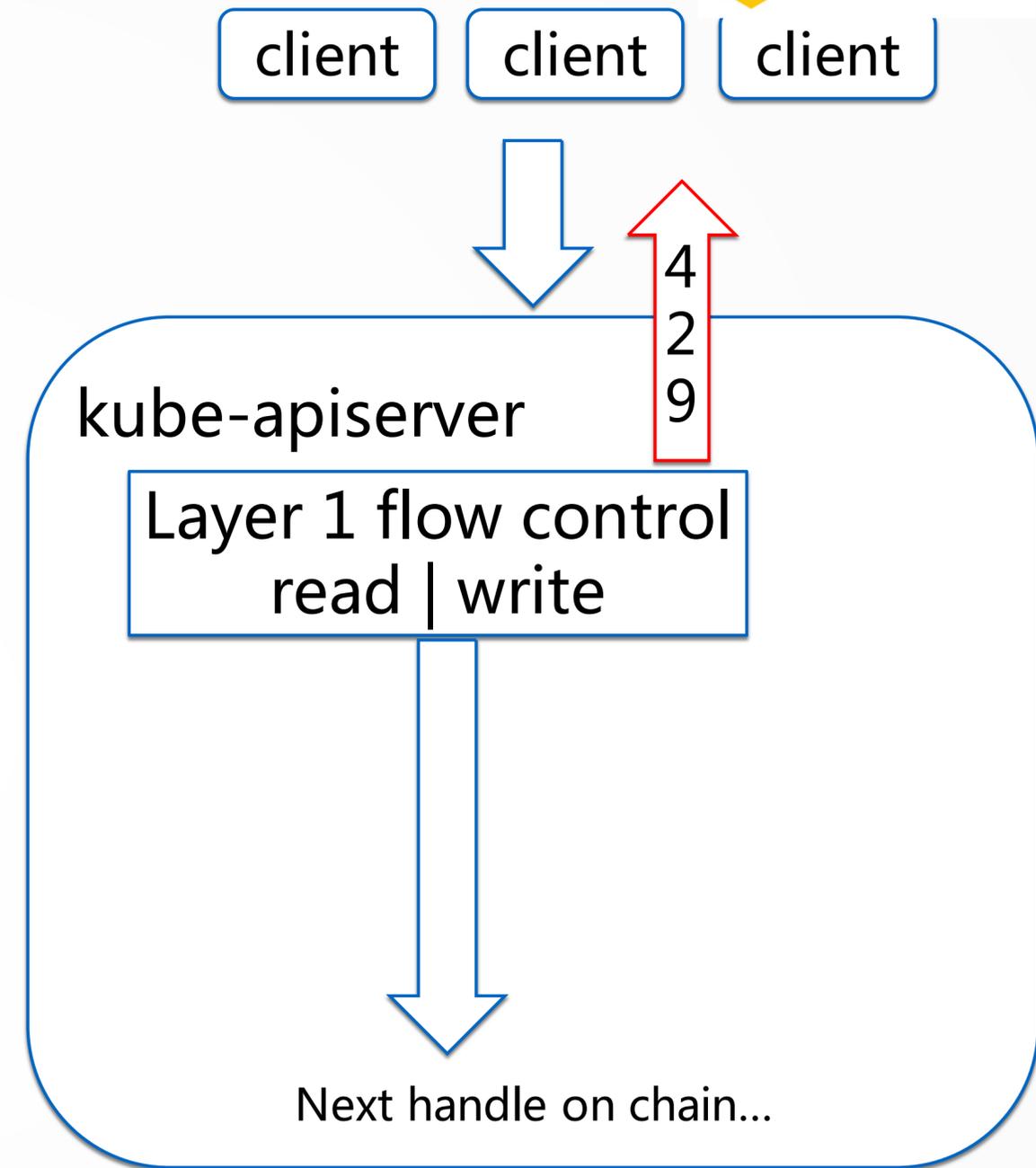
不符合我们的性能通过标准
(重启/升级停机时间 **< 3min**)



如何优化 Kubernetes ?

流控优化

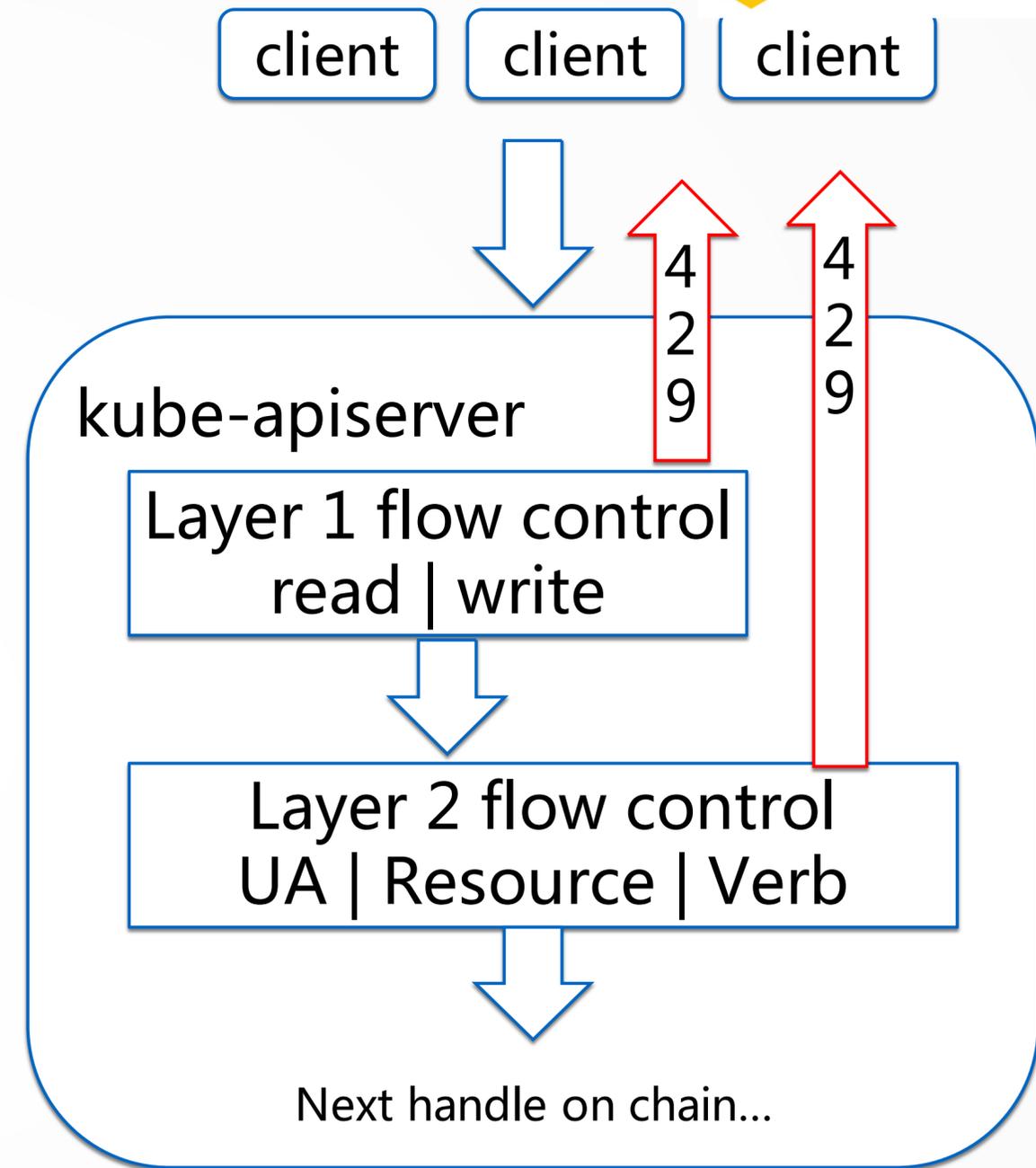
- 响应码 429 – 流控
- 默认流控重试时间 1s
- 机理：重启时所有 client 需要重新 list&watch 导致并发请求量过高（包含 bootstrap controller），超过流控。超过流控之后，返回响应码 **429**，同时在响应头里添加 **Retry-After** 告诉 client 1s 之后再试，造成 APIServer 长时间过载。



如何优化 Kubernetes ?

流控优化

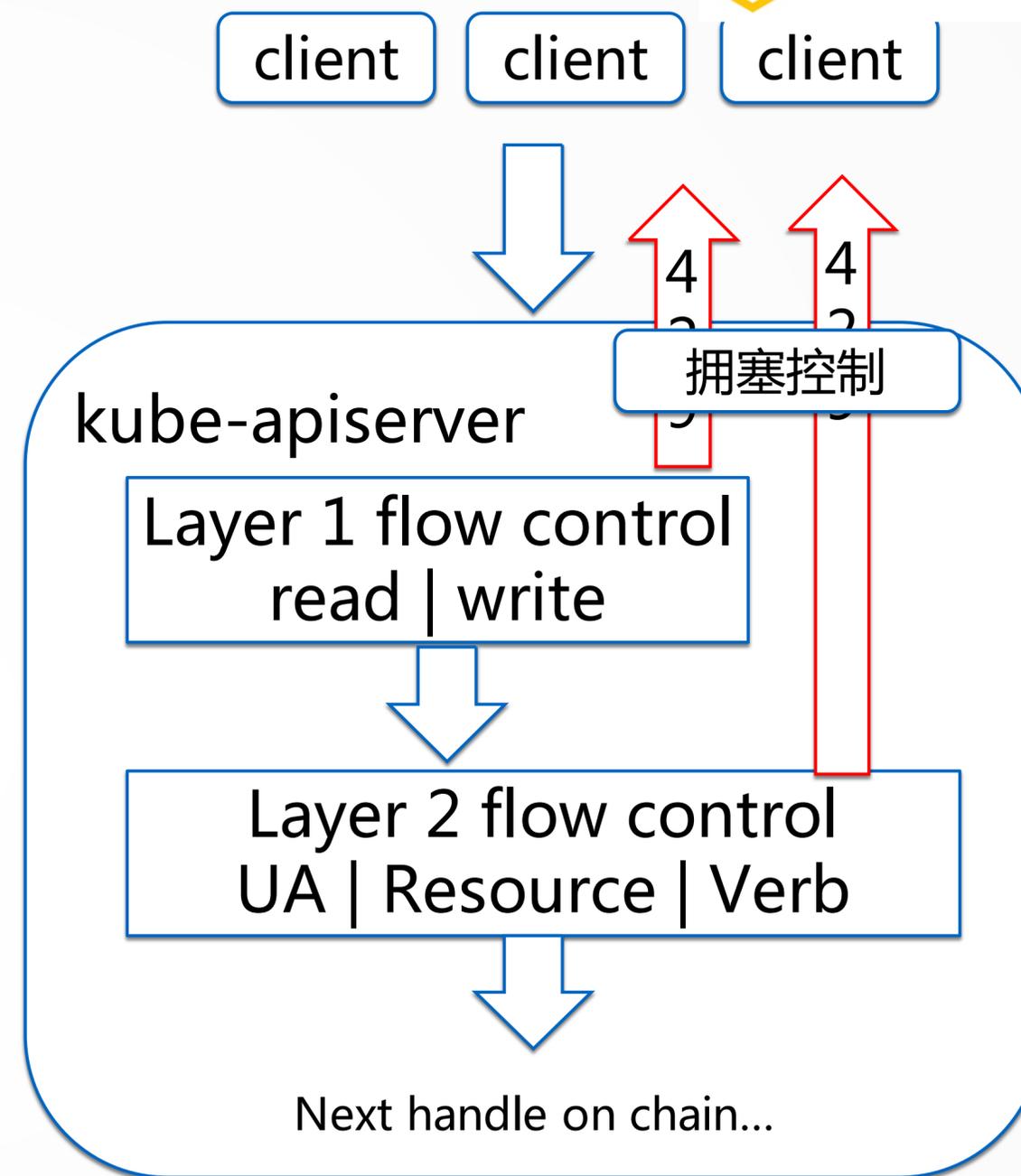
- 多级流控：在读写流控的基础上，根据 UserAgent、Resource、Verb 进行细粒度流控
- 例如：通过 UserAgent，限制 kubelet，优先允许 Master 组件初始化(API Server Bootstrap Controller、其他Controller 和 Scheduler 等)



如何优化 Kubernetes ?

流控优化

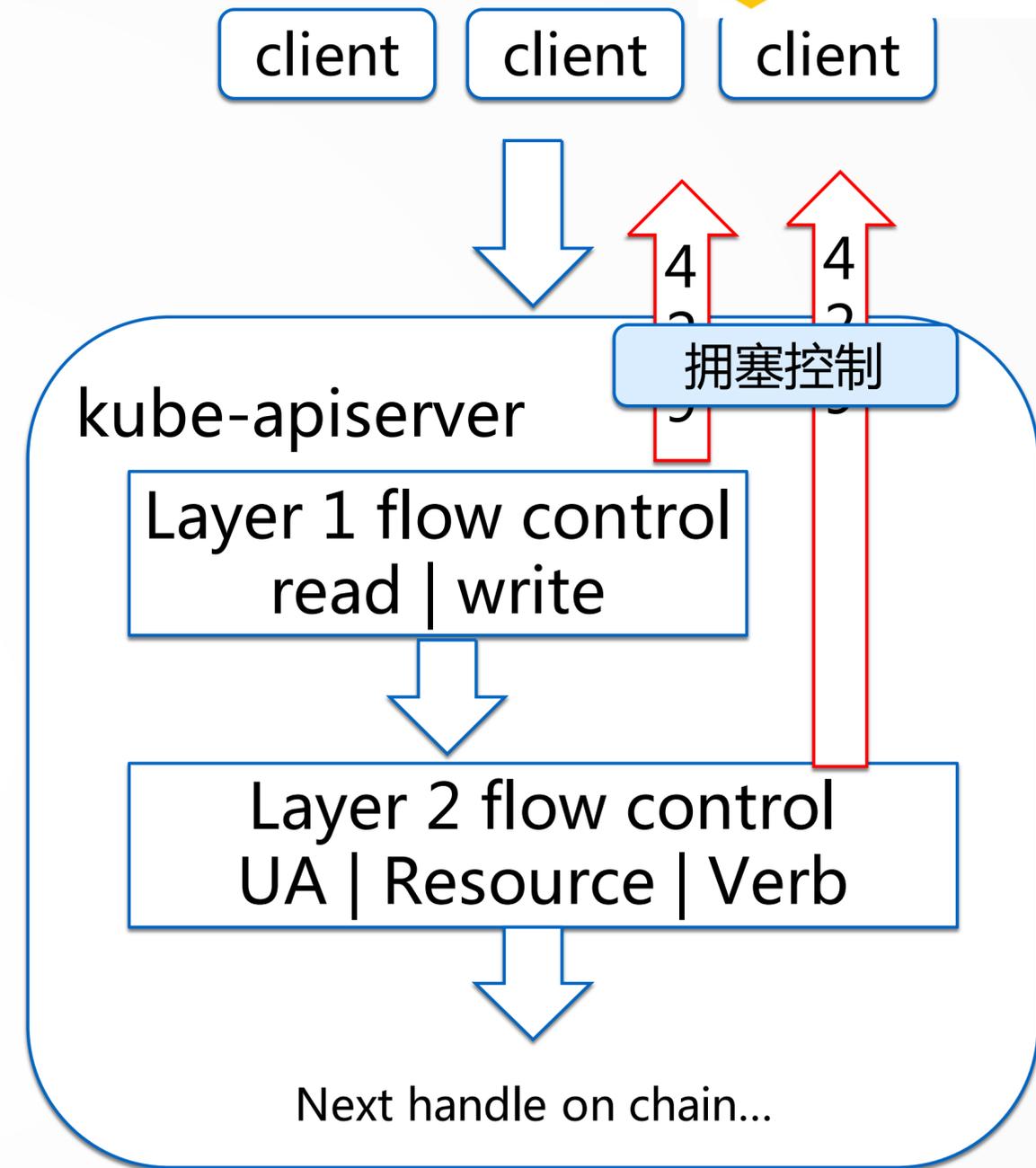
- 拥塞控制：静态 Retry-After 改为动态值，根据系统当前繁忙程度调节1到8s (token bucket)



如何优化 Kubernetes ?

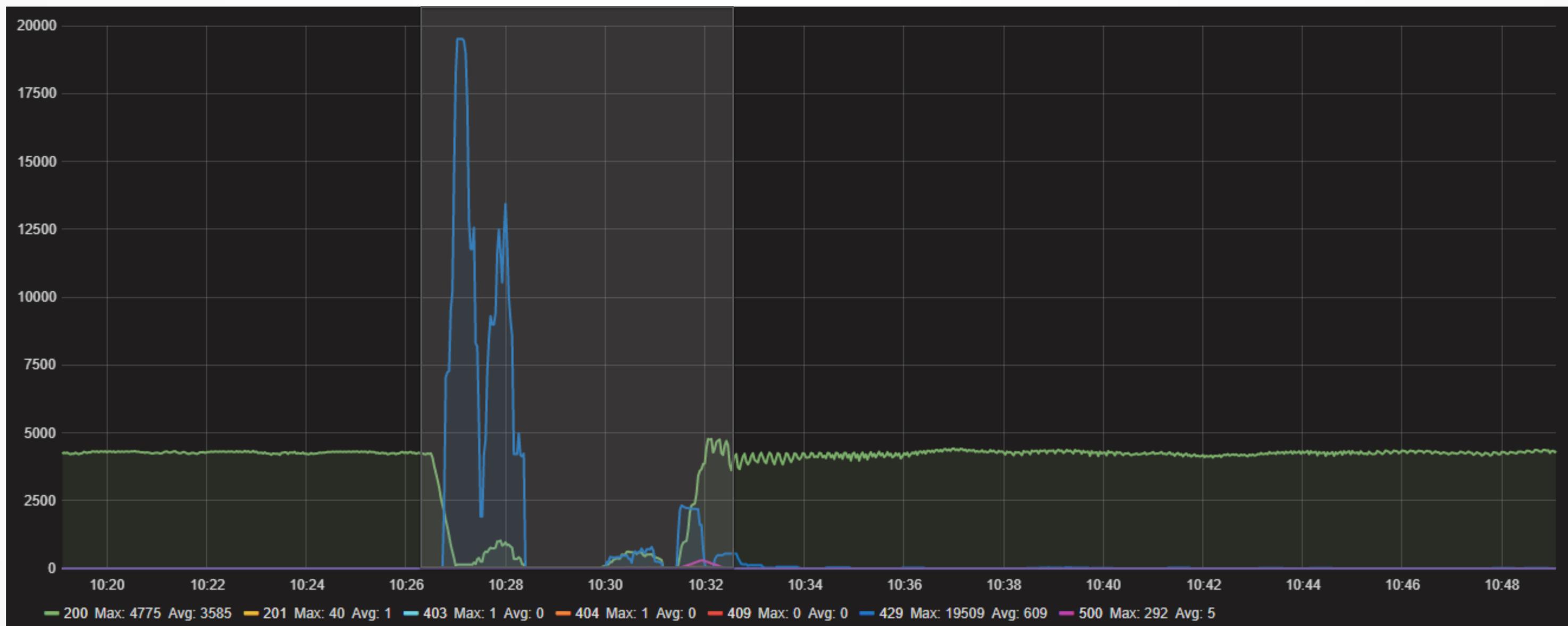
流控优化

- 拥塞控制：静态 Retry-After 改为动态值，根据系统当前繁忙程度调节1到8s (token bucket)
- 算法：初始化时间窗口为1s、容量 2^{10} 的 bucket，每次被流控尝试拿Token
 - 可用量 $> 1/2$ ，返回 1s
 - 可用量 $> 1/4$ ，返回 2s
 - 可用量 $> 1/8$ ，返回 4s
 - 可用量不足或者不可用，返回 8s



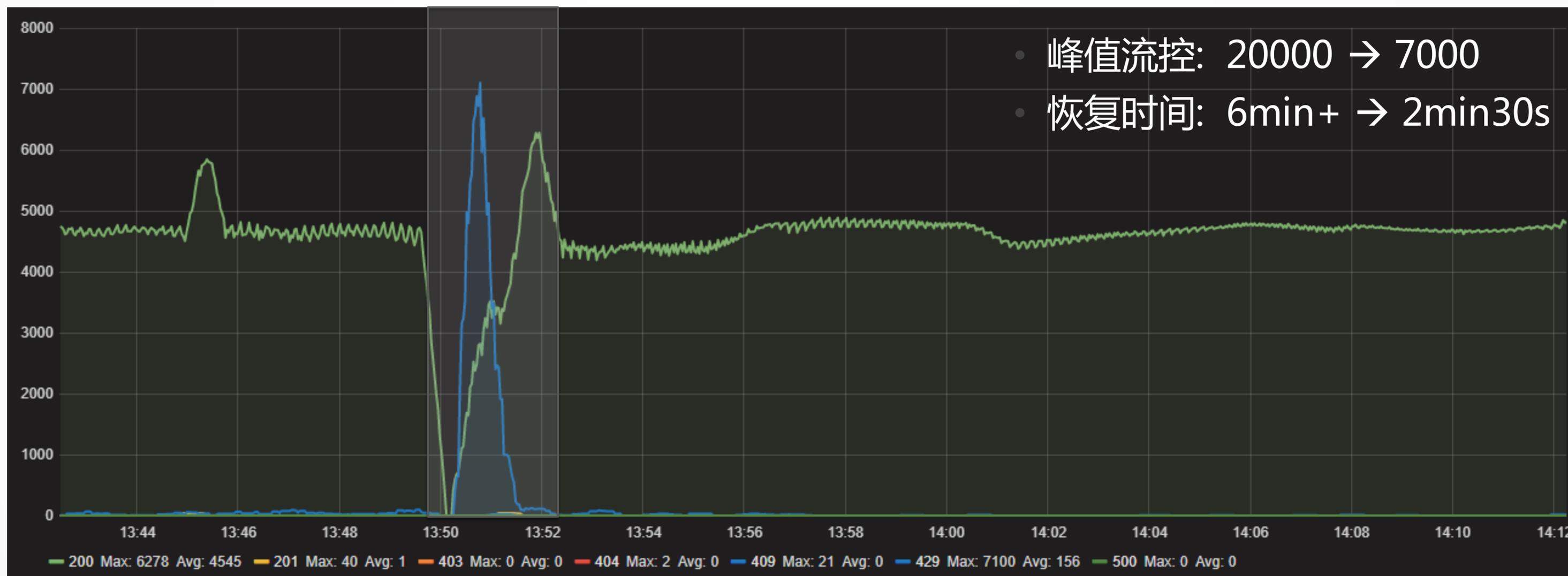
如何优化 Kubernetes ?

apiserver响应码统计



如何优化 Kubernetes ?

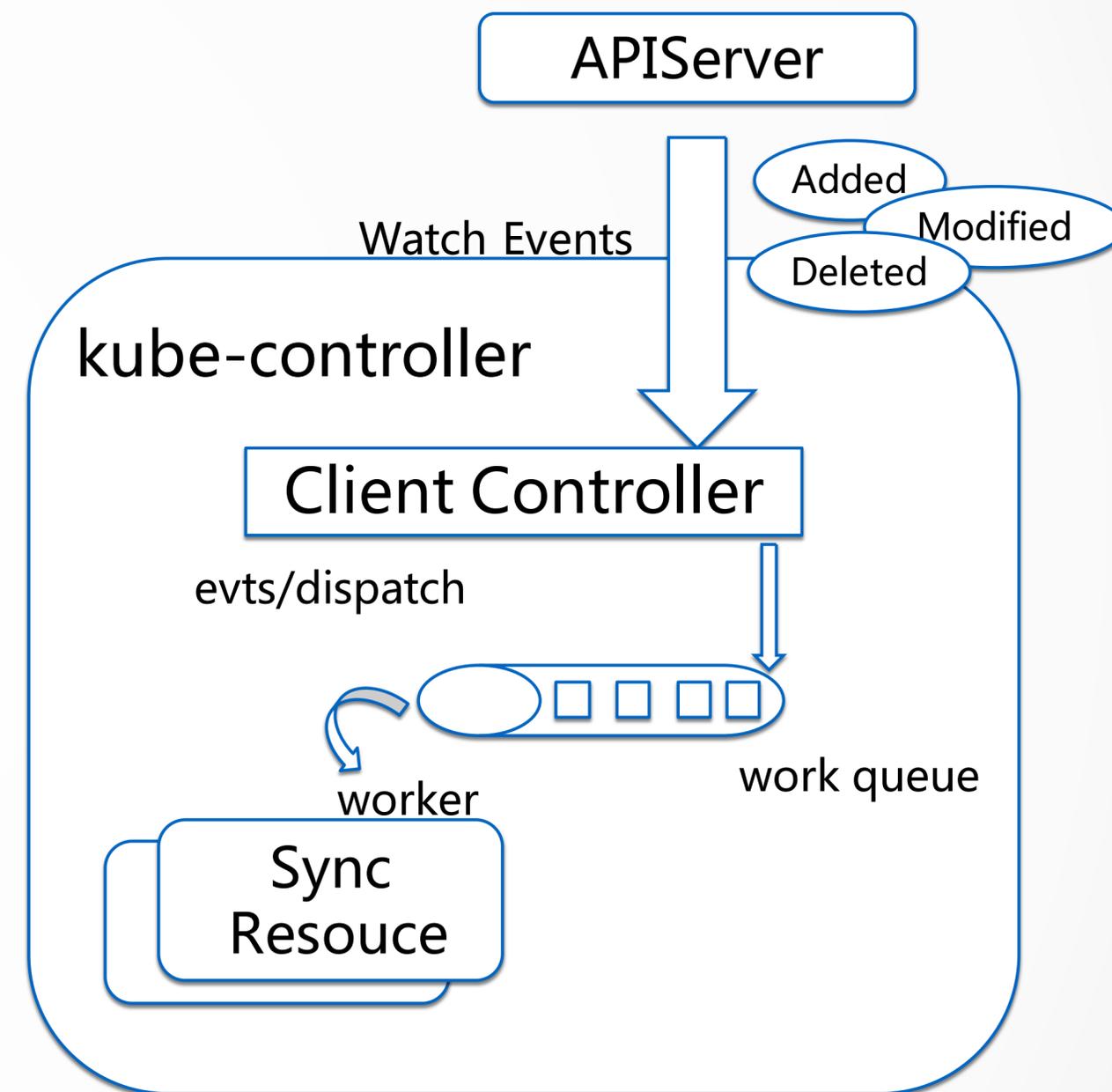
apiserver响应码统计



如何优化 Kubernetes ?

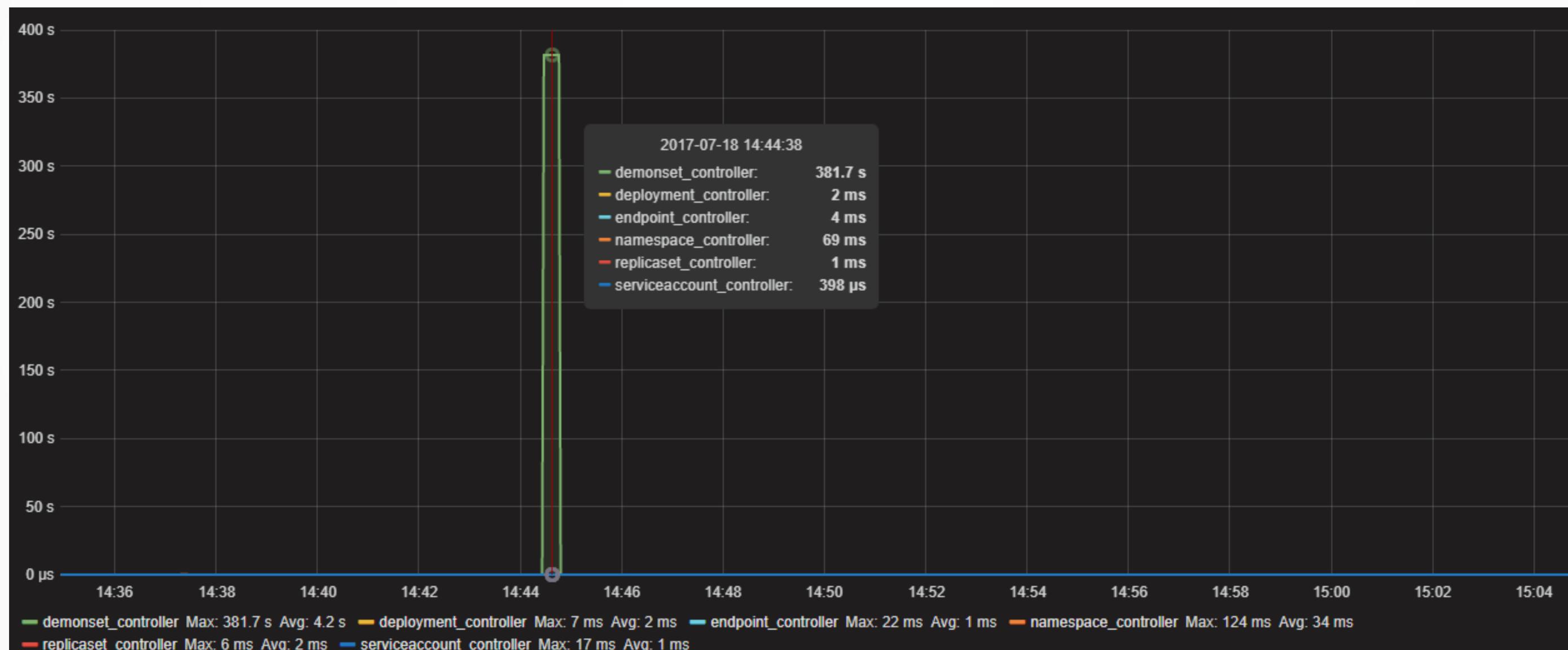
另一个例子

- Controller manager Sync 延迟统计 (resource)
- 什么是 Sync ?
- - 控制器在处理一种资源的过程称为一个 Sync (增删改)
- - 每隔一段时间(普通资源 12 ~ 24 小时、Namespace 5 min, Endpoint 30s) **全量同步**一次 (防止漏处理, 来自 etcd2 的毛病)
- 泡一杯咖啡坐等结果.....



如何优化 Kubernetes ?

Controller Sync Latency



如何优化 Kubernetes ?

Controller Sync Latency

- 发现 DaemonSet Controller 同步时间异常
- 原因：每次 DaemonSet 同步时，循环 node 时，在node内部重复 List Pod，时间复杂度 $O(\text{PodNum} * \text{NodeNum} * \text{DaemonSetNum})$
- 能否优化过程：过程无法去除，因为 DaemonSet Controller 需要检查 node 可调度状态，那么就必须要取得 Node 上所有 Pod 的状态
- 优化：空间换时间，对 pod list 一次，进行分组，node 只根据检查调度到自身的 pod，空间复杂度 $O(N)$ ，时间复杂度 $O(P * D)$
- 优化：加速 Pod List 的过程
- 优化：增加 Field 索引
- 结果：DaemonSet 峰值去除

如何优化 Kubernetes ?

加速 Pod List 过程

- 原理：Lister 接口接收 LabelSelector，但此时 Pod 传入 LabelSelector 为空，则无需经过 Label 的比较
- 实现：API Expansion 机制，k8s 允许我们自定义资源操作接口，而不对原生逻辑造成侵入，不会影响 client 代码生成
- 方式：`$SRC/pkg/client/listers/$VERSION/${RESOURCE}_expansion.go`

如何优化 Kubernetes ?

加速 Pod List 过程

- 实现：API Expansion 机制
- 方式：\$SRC/pkg/client/listers/\$VERSION/\${RESOURCE}_expansion.go

```
// PodListerExpansion allows custom methods to be added to
// PodLister.
type PodListerExpansion interface {
    ListAll() ([]*v1.Pod, error)
}

// ListAll provide a fast path to get pods without selector.
func (l *podLister) ListAll() ([]*v1.Pod, error) {
    src := l.indexer.List()
    dst := make([]*v1.Pod, len(src))
    for _, m := range src {
        dst = append(dst, m.(*v1.Pod))
    }

    return dst, nil
}
```

如何优化 Kubernetes ?

Prometheus+Grafana

- 不是万能
- 不足
 - - 采样抹掉毛刺(10s 踩一次), 漏掉异常波动
 - - 平滑算法, 可能因为采样区间问题导致图像完全不一致, 造成错误判断

如何优化 Kubernetes ?

采样+平滑算法bug



如何优化 Kubernetes ?

性能剖析

- 系统层面：perf、systemtap、bcc
- 应用层面：go tool pprof

如何优化 Kubernetes ?

系统层：举个例子

- 发现 Kubemark CPU 消耗太高(> 50%)
- go tool pprof 能够采集到的 CPU 消耗很低(10%+)
- 猜测：可能是 kernel 过程耗时过多 CPU
- 工具：perf top -p `pidof kubemark`

如何优化 Kubernetes ?

```
Samples: 94K of event 'cpu-clock', Event count (approx.): 9389641316
```

```
Overhead Shared O Symbol
35.24% [kernel] [k] copy_page_range
4.15% [kernel] [k] unmap_page_range
3.13% [kernel] [k] __x2apic_send_IPI_mask
2.75% [kernel] [k] copy_page
2.30% [kernel] [k] finish_task_switch
2.28% [kernel] [k] __do_page_fault
1.98% [kernel] [k] __lock_text_start
1.98% [kernel] [k] native_write_msr
1.53% [kernel] [k] vm_normal_page
1.39% kubemark [.] runtime.scanobject
1.38% [kernel] [k] smp_call_function_many
1.28% [kernel] [k] page_remove_rmap
1.25% [kernel] [k] handle_mm_fault
1.20% [kernel] [k] smp_call_function_single
1.14% kubemark [.] runtime.mallocgc
1.06% [kernel] [k] get_page_from_freelist
0.82% kubemark [.] runtime.selectgoImpl
0.82% [kernel] [k] clear_page_c_e
0.78% [kernel] [k] release_pages
0.66% kubemark [.] runtime.siftdownTimer
0.57% [kernel] [k] _raw_spin_lock
0.53% kubemark [.] runtime.heapBitsSetType
0.52% [kernel] [k] wp_page_copy
0.49% kubemark [.] runtime.mapiternext
0.47% kubemark [.] runtime.heapBitsForObject
0.45% [kernel] [k] flush_tlb_page
0.44% kubemark [.] runtime.memmove
```

如何优化 Kubernetes ?

系统层：举个例子

- 发现 Kubemark CPU 消耗太高(> 50%)
- go tool pprof 能够采集到的 CPU 消耗很低(10%+)
- 猜测：可能是 kernel 过程耗时过多 CPU
- 工具：perf top -p `pidof kubemark`
- 结果：copy_page_range → copy_page → fork → os.Exec ,
kubemark频繁调用 iptables 进程导致，mock 掉 iptables调用
- 结论：系统 trace 有时能帮我们发现隐藏的问题

<http://www.brendangregg.com/perf.html>

如何优化 Kubernetes ?

PProf 用法

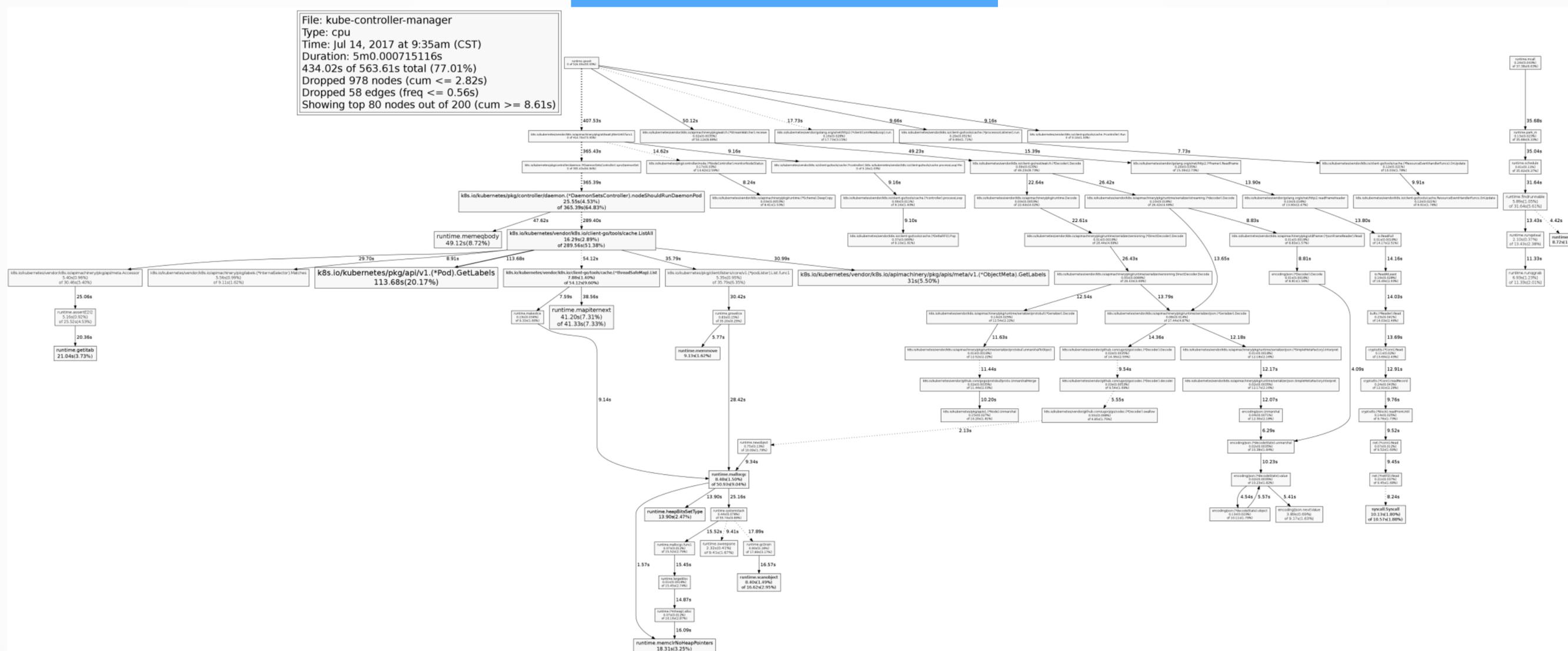
- pprof 用法
- `go tool pprof http://master/debug/pprof/profile`
- **坑**：由于 apiserver timeout 模块，导致最多只能抓1分钟即被断开
- **跳坑姿势**：可以用 http method hack，伪装成 watch，离线分析
- `curl -XWATCH`
`http://master/debug/pprof/profile?seconds=XXX -o output`
- `go tool pprof <bin> output`
- 举两个例子

<https://blog.golang.org/profiling-go-programs>

如何优化 Kubernetes ?

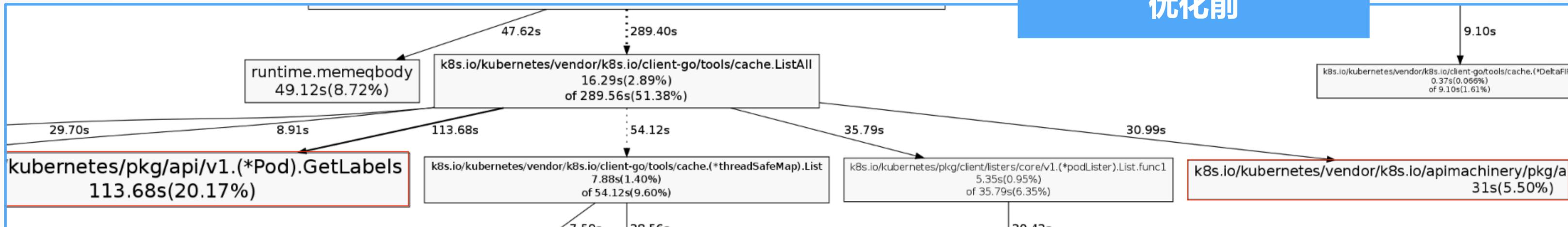
pprof CPU 热点图

File: kube-controller-manager
Type: cpu
Time: Jul 14, 2017 at 9:35am (CST)
Duration: 5m0.000715116s
434.02s of 563.61s total (77.01%)
Dropped 978 nodes (cum <= 2.82s)
Dropped 58 edges (freq <= 0.56s)
Showing top 80 nodes out of 200 (cum >= 8.61s)



如何优化 Kubernetes ?

优化前



分析

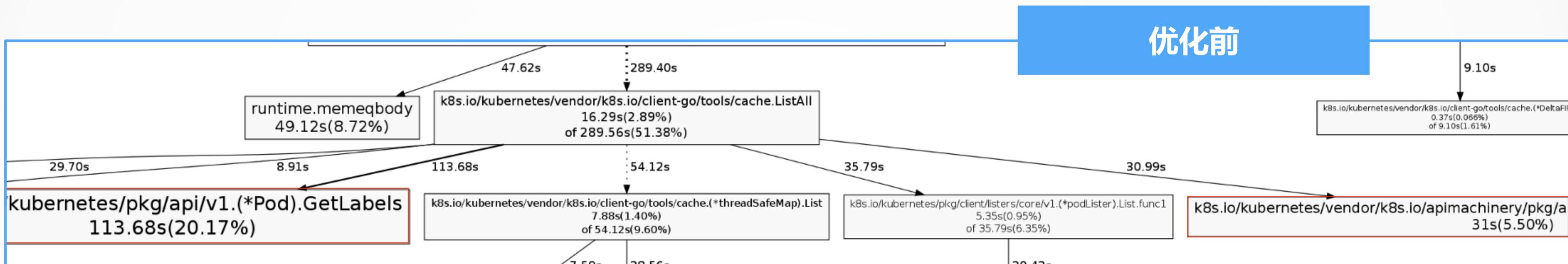
- ListAll 需要通过LabelSelector过滤
- 通过 GetLabels 取 labels 过滤
- 大部分请求 selector 均为空
- 方案：增加 selector.Empty() 时的 QuickPath

```

33 func ListAll(store Store, selector labels.Selector, appendFn AppendFunc) error {
34     for _, m := range store.List() {
35         metadata, err := meta.Accessor(m)
36         if err != nil {
37             return err
38         }
39         if selector.Matches(labels.Set(metadata.GetLabels())) {
40             appendFn(m)
41         }
42     }
43     return nil
44 }

```

如何优化 Kubernetes ?



分析

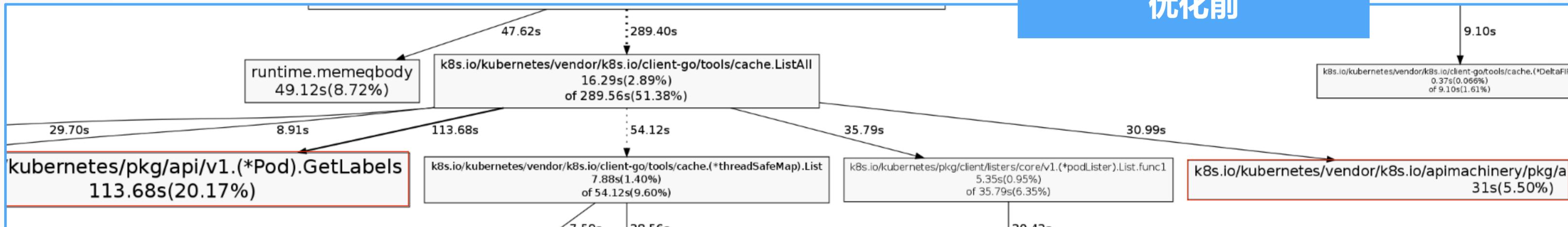
- ListAll 需要通过LabelSelector过滤
- 通过 GetLabels 取 labels 过滤
- 大部分请求 selector 均为空
- 方案：增加 selector.Empty() 时的 QuickPath

```

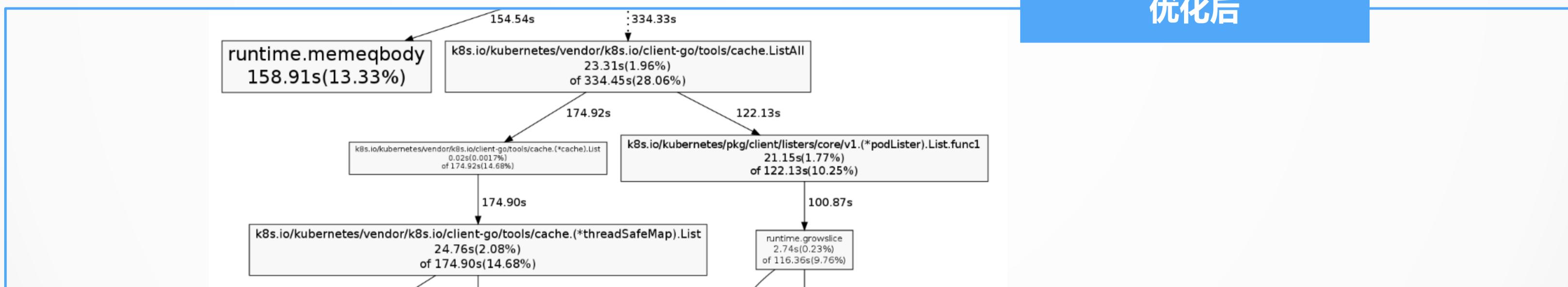
33 func ListAll(store Store, selector labels.Selector, appendFn AppendFunc) error {
34     for _, m := range store.List() {
35         if selector.Empty() {
36             appendFn(m)
37             continue
38         }
39         metadata, err := meta.Accessor(m)
40         if err != nil {
41             return err
42         }
43         if selector.Matches(labels.Set(metadata.GetLabels())) {
44             appendFn(m)
45         }
46     }
47     return nil
48 }
    
```

如何优化 Kubernetes ?

优化前

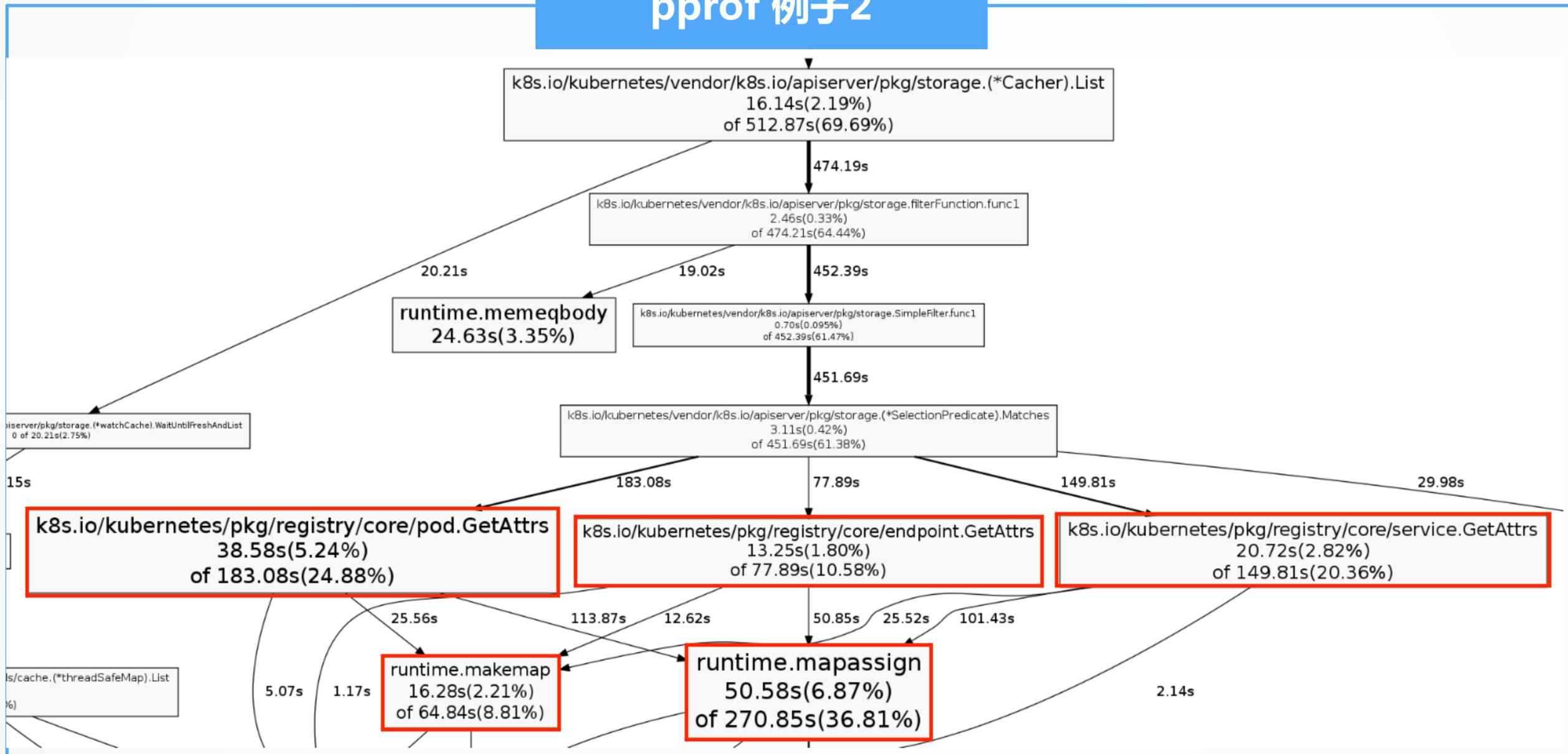


优化后



如何优化 Kubernetes ?

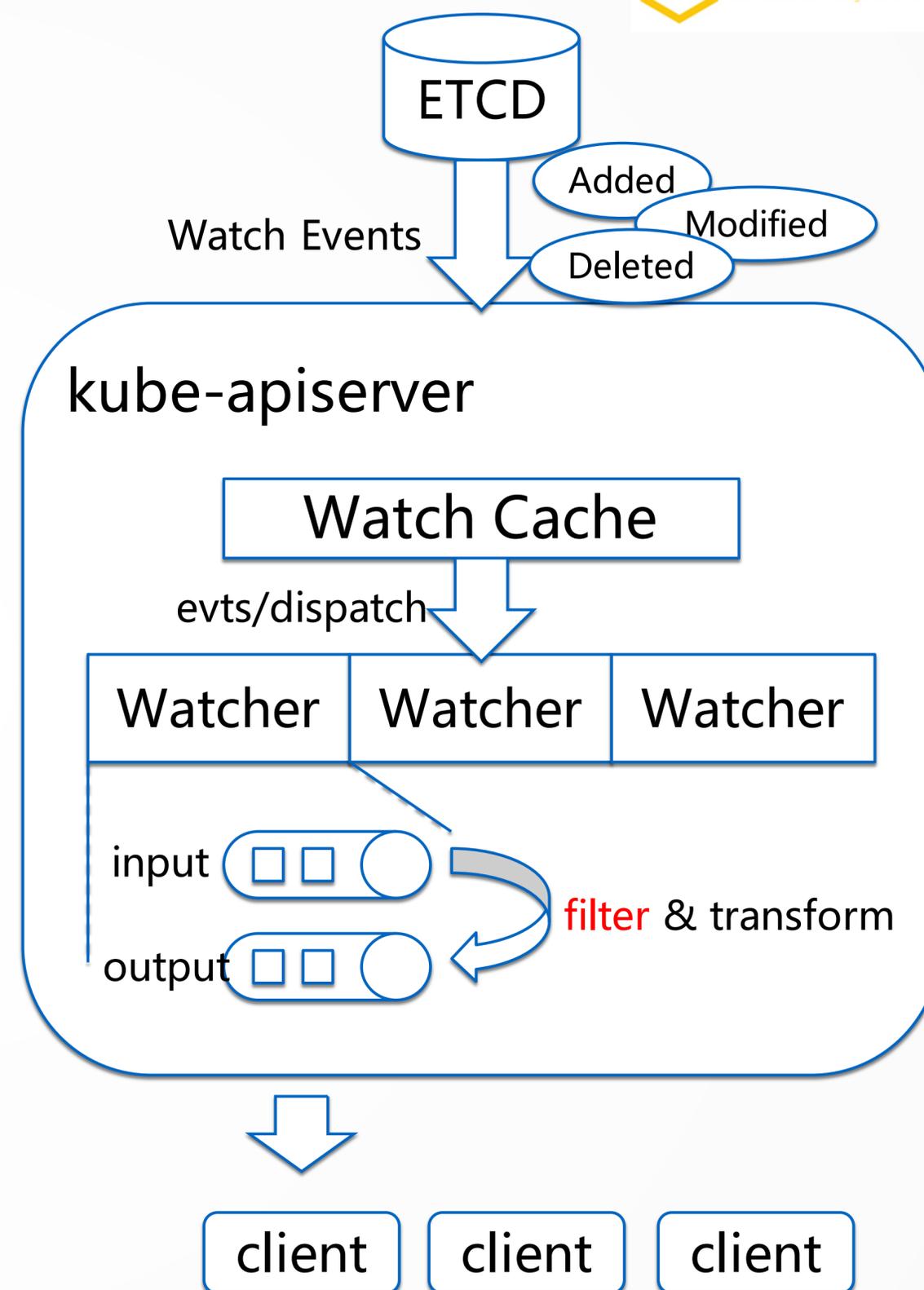
pprof 例子2



如何优化 Kubernetes ?

pprof 例子2

- 缓存层问题
- 缓存层：缓存和聚合来自 client 的 List&Watch，防止 etcd 压力过大(k8s1.0 引入)

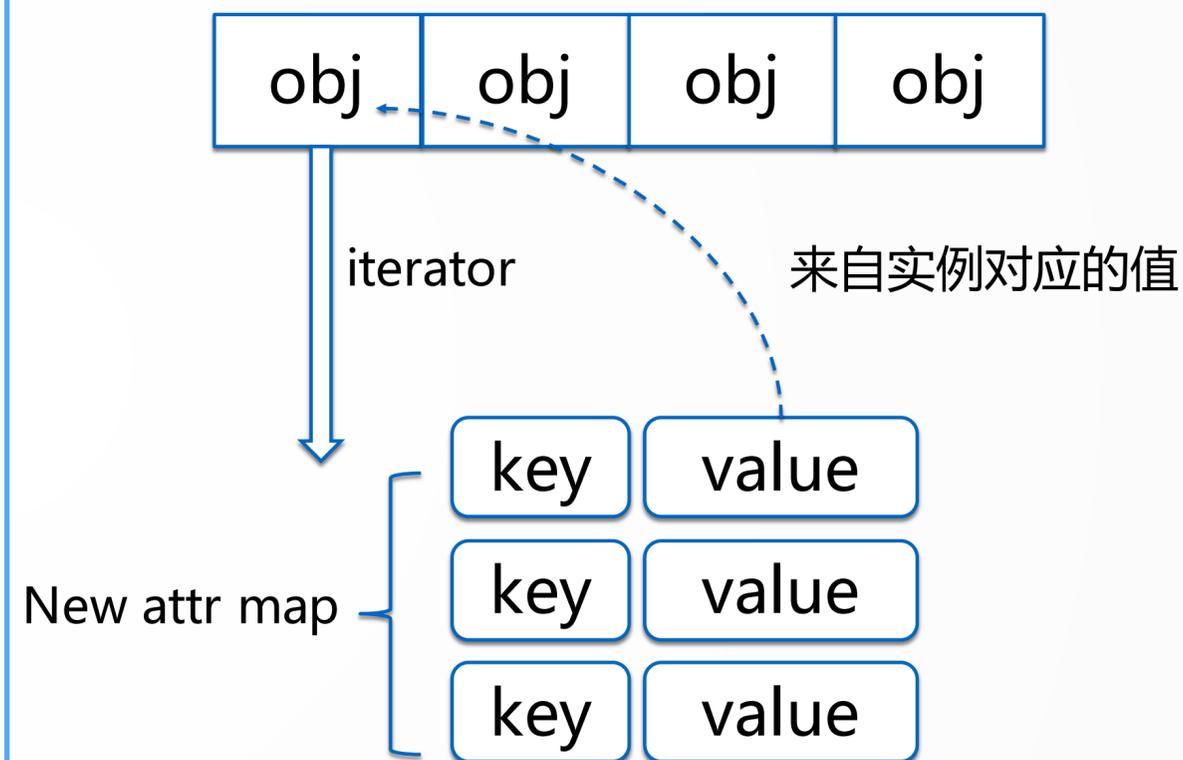


如何优化 Kubernetes ?

pprof 例子2

- 问题：每次过滤，从实例中取可过滤的attr map(由每个实例的GetAttr生成)，包含可过滤的 key 和 实例对应的属性值，N个实例就有N个临时Map生成
- 每种资源，attr map结构一样，key相同，只有value不一致

Filter Procedure



```

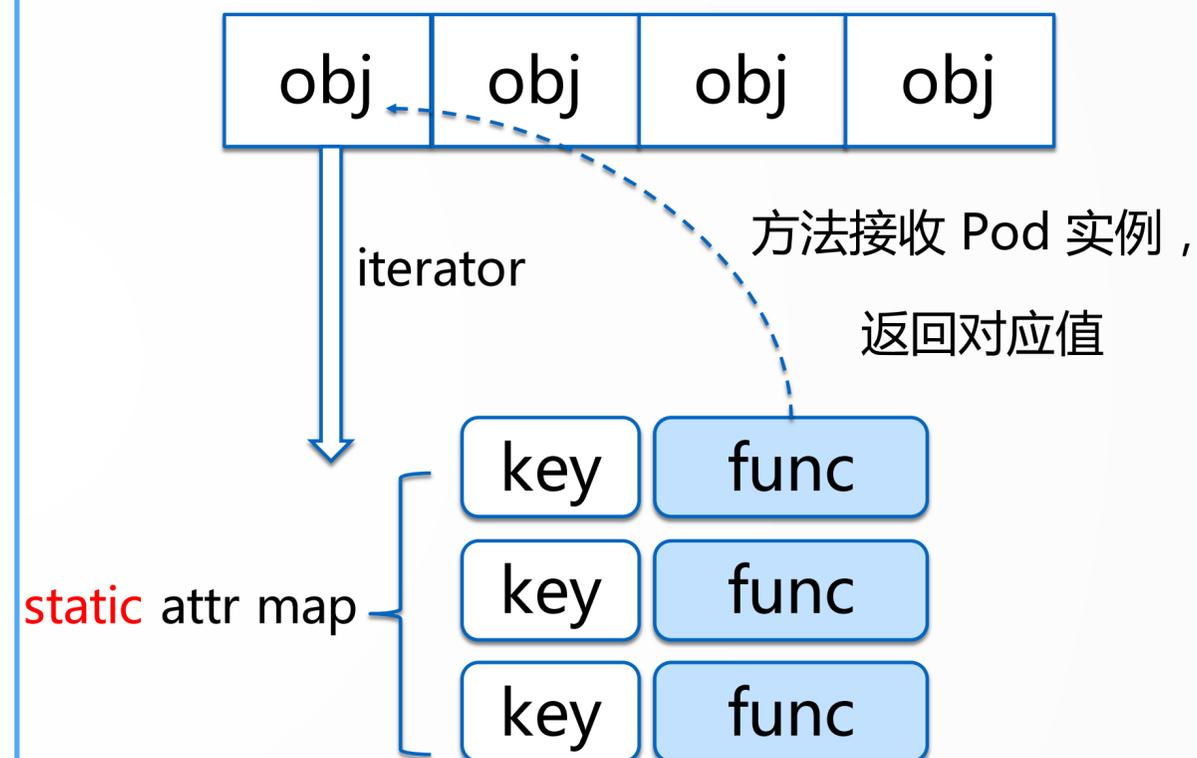
209 // PodToSelectableFields returns a field set that represents the object
210 // TODO: fields are not labels, and the validation rules for them do not apply.
211 func PodToSelectableFields(pod *api.Pod) fields.Set {
212     // The purpose of allocation with a given number of elements is to reduce
213     // amount of allocations needed to create the fields.Set. If you add any
214     // field here or the number of object-meta related fields changes, this should
215     // be adjusted.
216     podSpecificFieldsSet := make(fields.Set, 5)
217     podSpecificFieldsSet["spec.nodeName"] = pod.Spec.NodeName
218     podSpecificFieldsSet["spec.restartPolicy"] = string(pod.Spec.RestartPolicy)
219     podSpecificFieldsSet["status.phase"] = string(pod.Status.Phase)
220     return generic.AddObjectMetaFieldsSet(podSpecificFieldsSet, &pod.ObjectMeta, tr
221 }
    
```

如何优化 Kubernetes ?

pprof 例子2

- 优化：既然每个value都来自实例，那么就可以直接通过**方法**返回
- 例如：pod 对应的 filter 所需的 map

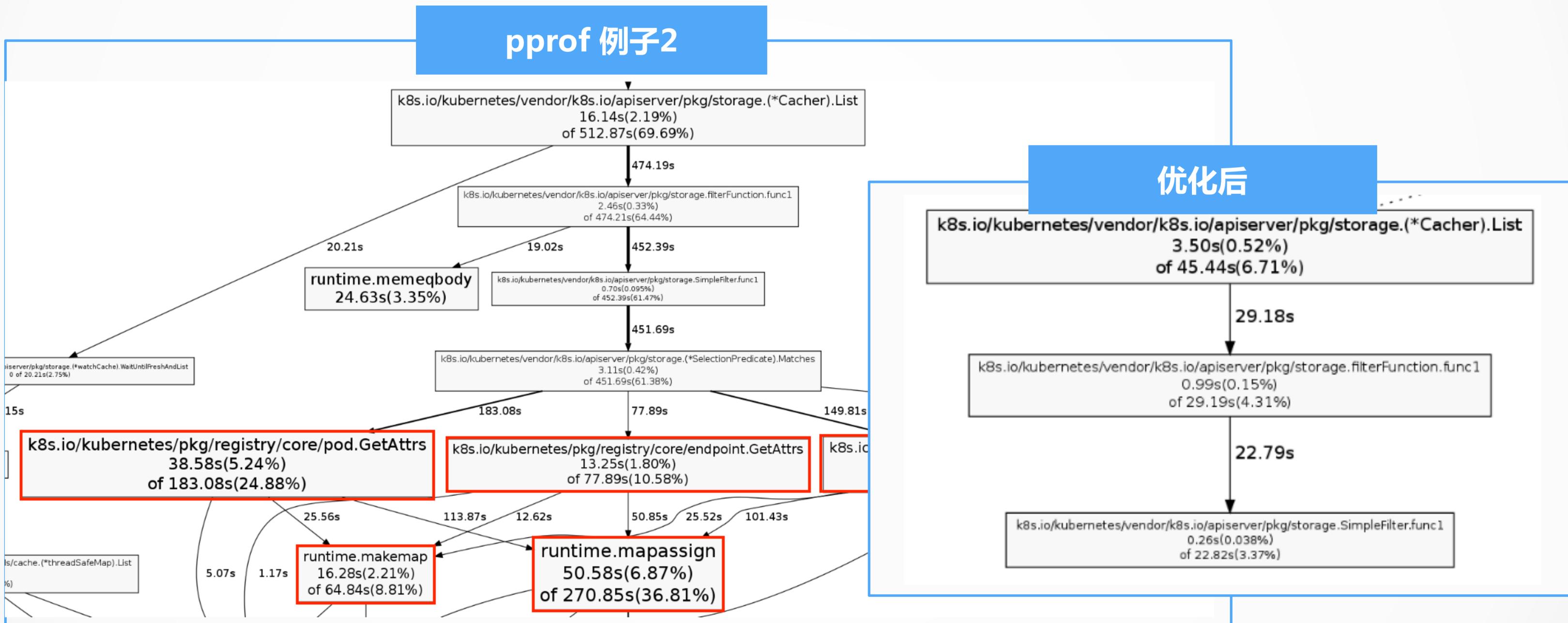
Filter Procedure



```

164 var podFieldCallbak = fields.CallbackSet{
165     "metadata.name":      func(obj runtime.Object) string { return obj.(*api.Pod).ObjectMeta.Name },
166     "metadata.namespace": func(obj runtime.Object) string { return obj.(*api.Pod).ObjectMeta.Namespace },
167
168     "spec.nodeName":      func(obj runtime.Object) string { return obj.(*api.Pod).Spec.NodeName },
169     "spec.restartPolicy": func(obj runtime.Object) string { return string(obj.(*api.Pod).Spec.RestartPolicy) },
170     "status.phase":       func(obj runtime.Object) string { return string(obj.(*api.Pod).Status.Phase) },
171 }
    
```

如何优化 Kubernetes ?



如何优化 Kubernetes ?

其他优化

- APIServer 等待缓存层同步完成再服务
- Kubemark 同时模拟多个 Node , 性能提升 10x , 满足线下性能测试需求
- 网络方案采用 sriov , 提高容器网络传输效率
- kubelet/kube-proxy CPU/内存/优化 , 例如 kubelet/proxy 只需关心少量资源存在 (List&Watch 资源时增加过滤)
-

未来的优化

未来的优化

接下来的优化

- 优化 Node 心跳汇报 – 利用 etcd Lease 机制，不发生状态变更时无需变更版本号；问题：依赖的 etcd 3.2 版本、改动原有心跳机制
- 高性能容器、网络优化
- 提 Proposals，回馈社区

想知道更多？

云原生应用架构实践

- 我司各种大牛之精华
- 集团10+ 年实践总结
- 非常有操作性和指导性
- 详解了云原生应用的内涵和要点，对实现云原生应用面临的功能和非功能（高性能、高可用、可扩展、安全性、高可靠等）的不同阶段需求和实现方案
- 了系统工程化、高性能数据库、分布式数据库、DevOps、微服务架构、服务化测试、多机房架构等
- 适合希望采用云计算帮助企业实现业务提升的 CTO、CIO、架构师等群体





网易云

共创云上精彩世界