如何优化 Kubernetes？

网易云

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？

网易云

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？

网易云

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？

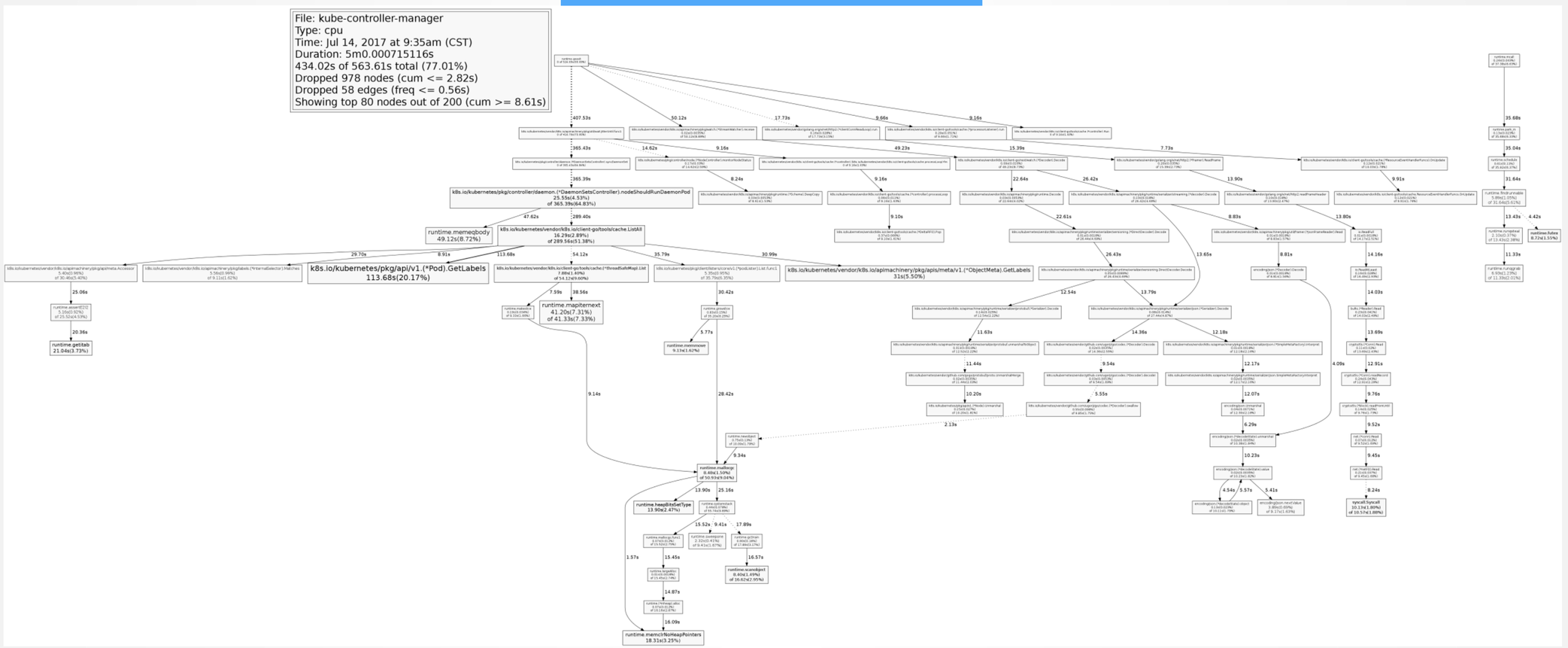
PProf 用法

- 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？

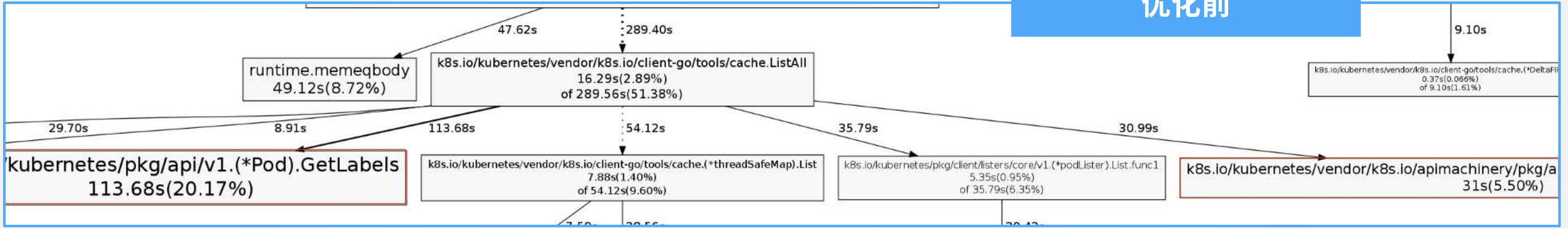
pprof CPU 热点图



如何优化 Kubernetes？

网易云

优化前



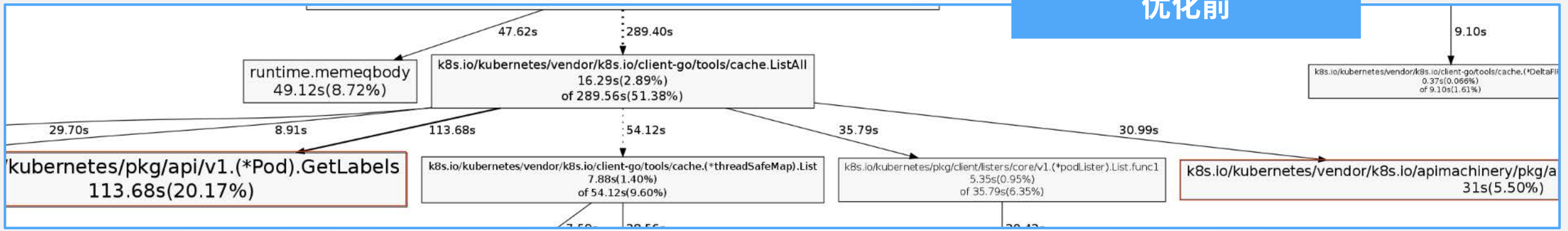
分析

- 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.Emtpy() 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？

优化前



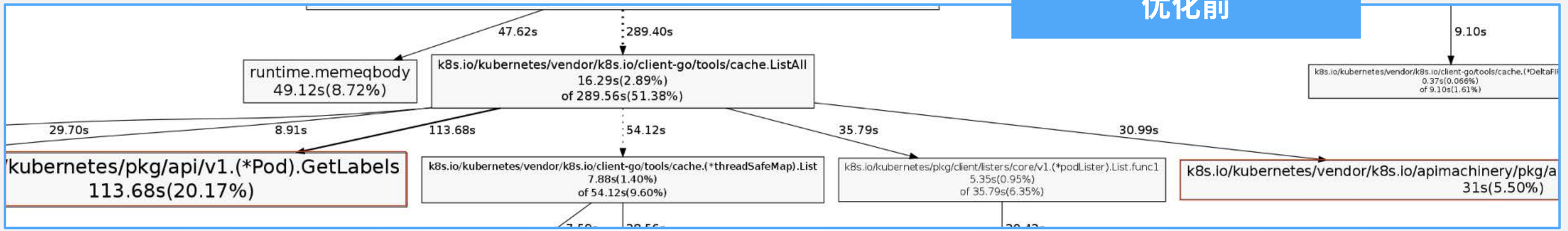
分析

- 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.Emtpy() 时的 QuickPath

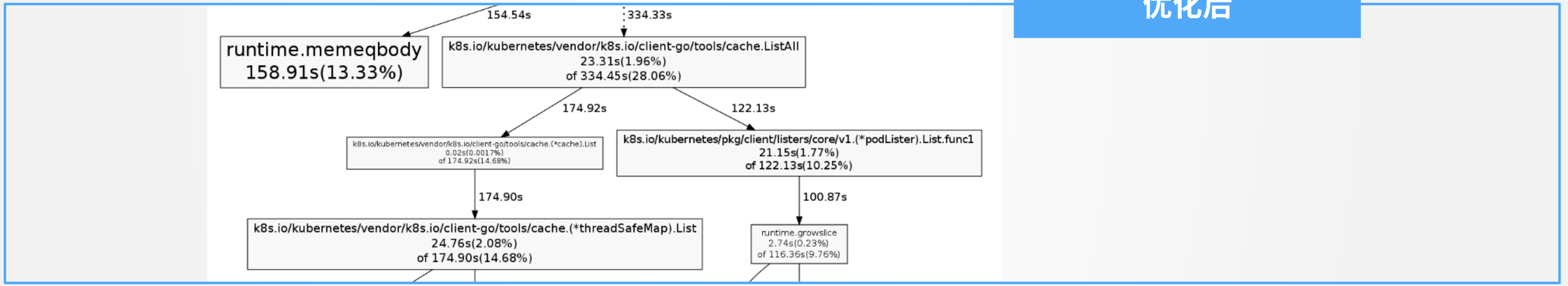
```
33 func ListAll(store Store, selector labels.Selector, appendFn AppendFunc) error {
34     for _, m := range store.List() {
35         if selector.Empty() {
36             appendFn(m)
37             continue
38         }
39         metadata, err := meta.Accessor(m)
40         if err != nil {
41             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？

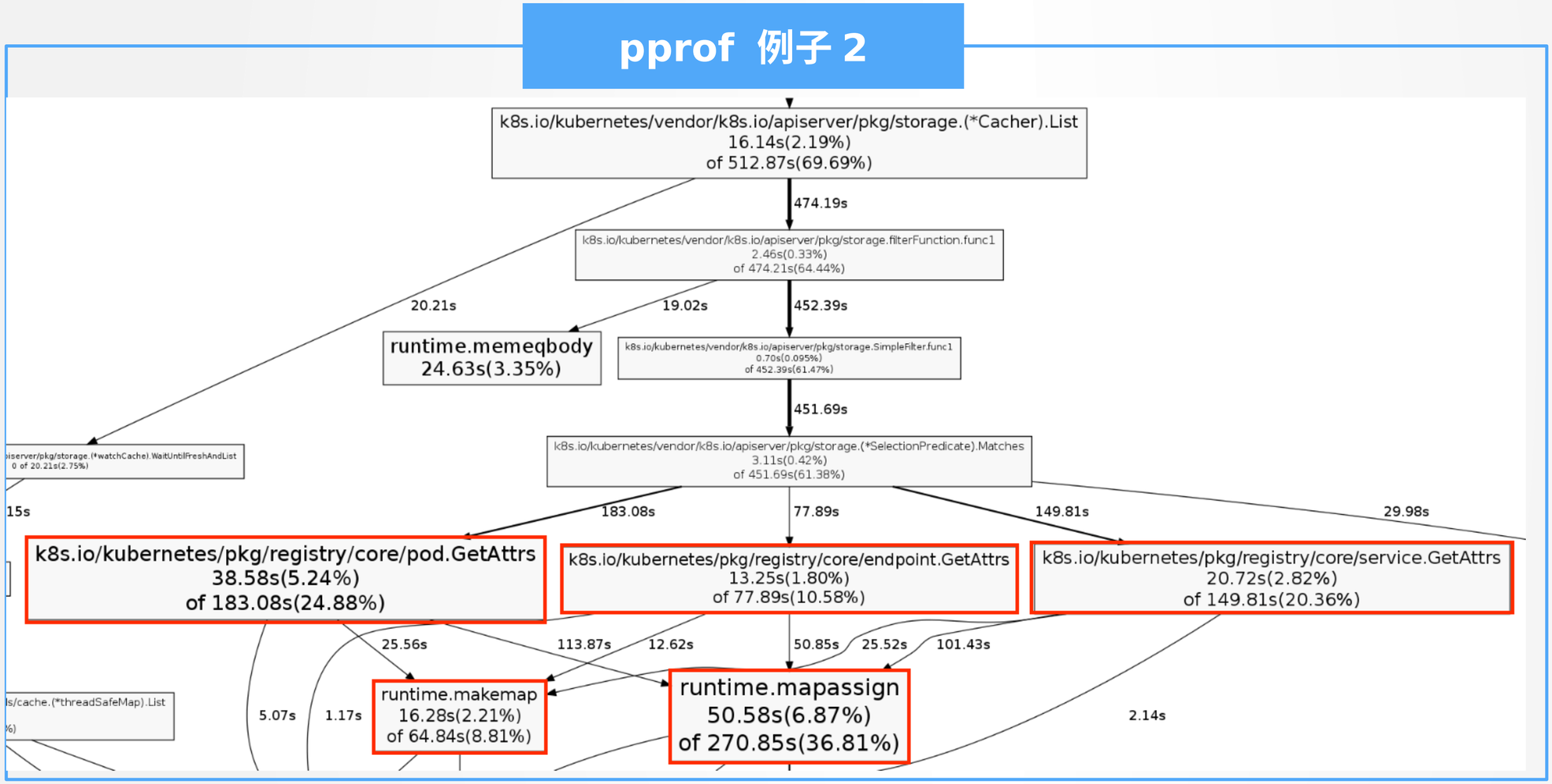
优化前



优化后



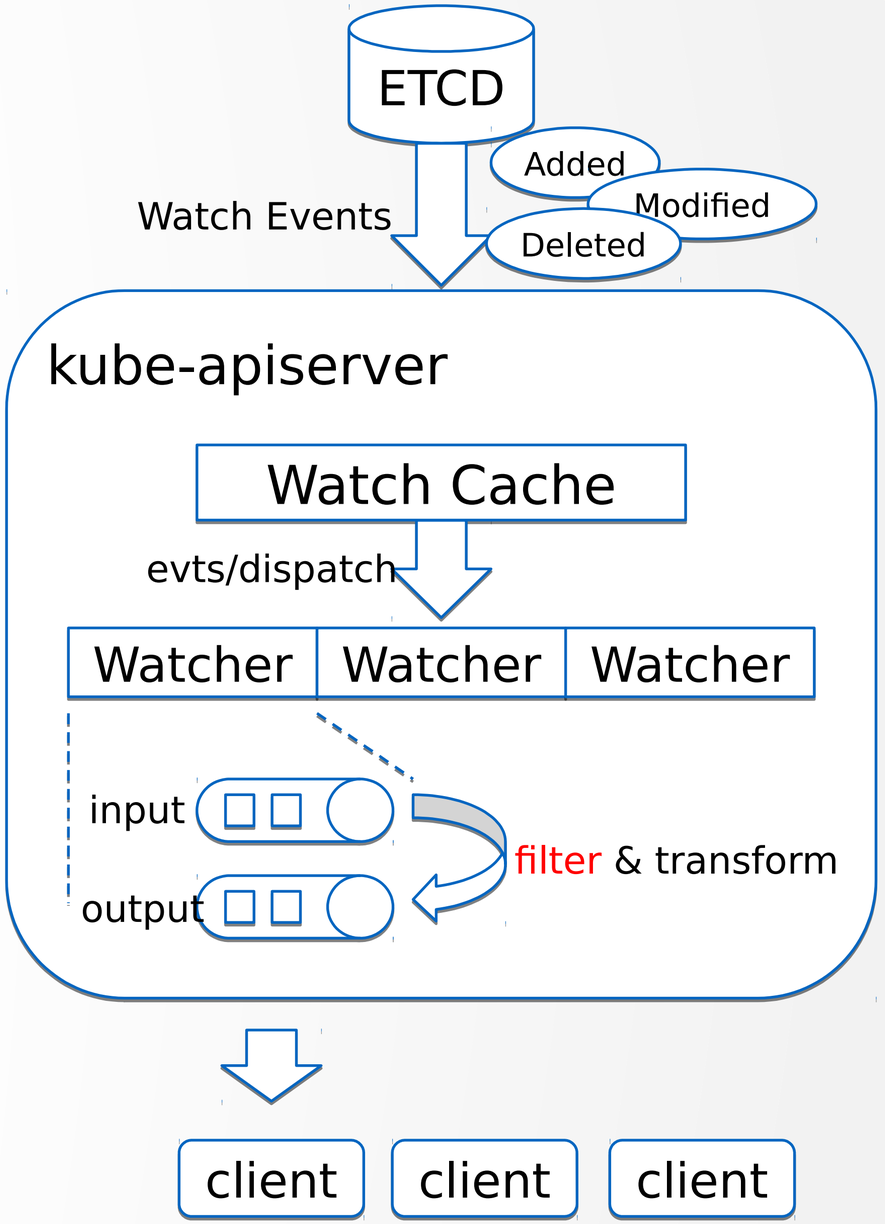
如何优化 Kubernetes？



如何优化 Kubernetes？

pprof 例子 2

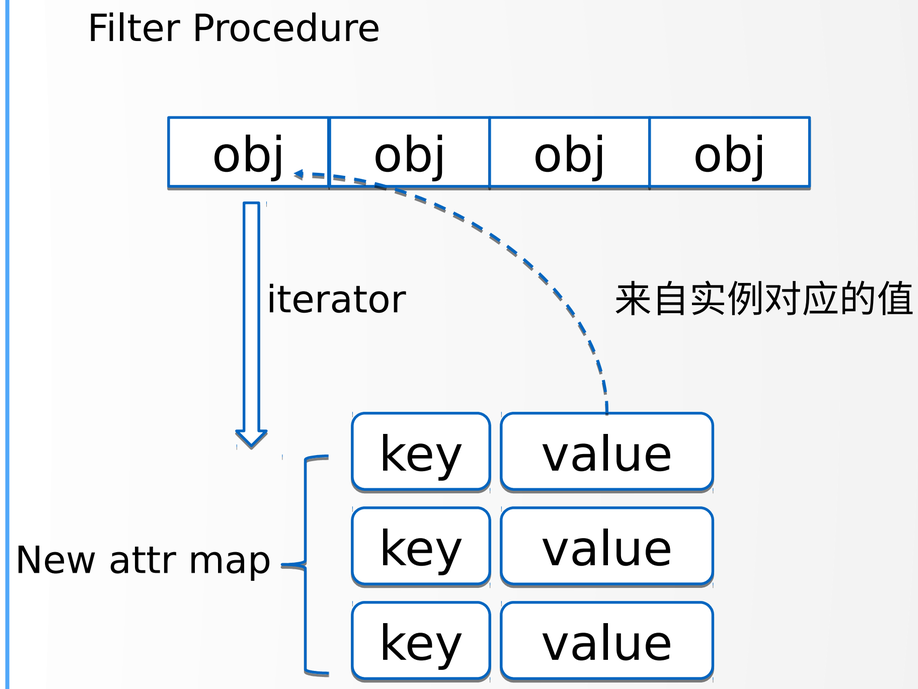
- **缓存层问题**
- 缓存层：缓存和聚合来自 client 的 List&Watch，防止 etcd 压力过大 (k8s1.0 引入)



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

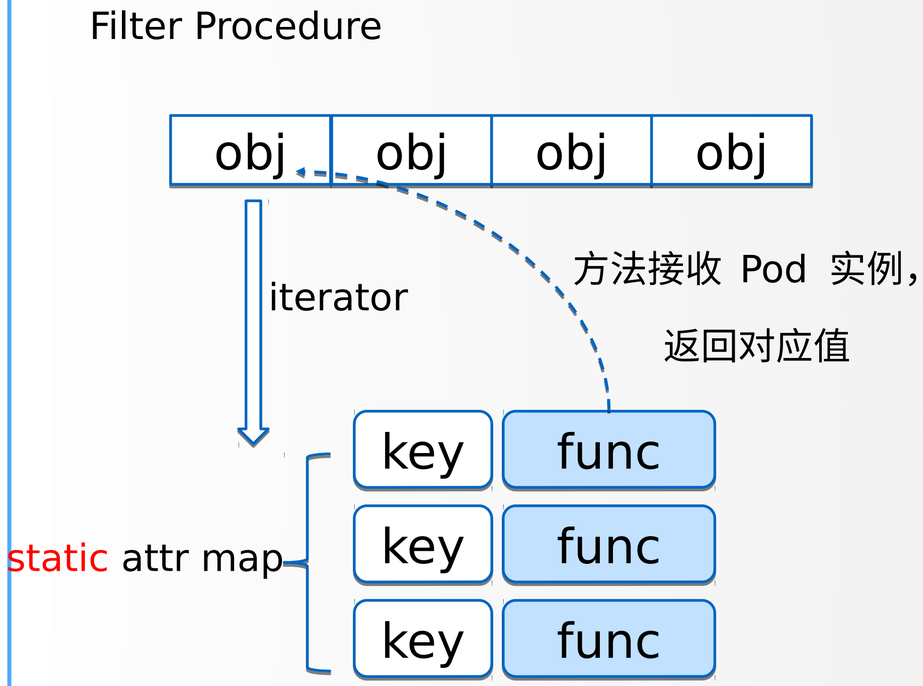



```
209 // PodToSelectableFields returns a field set that represents the object
210 // TODO: fields are not labels, and the validation rules for them do not apply.
211 func PodToSelectableFields(pod *api.Pod) fields.Set {
212     // The purpose of allocation with a given number of elements is to reduce
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？

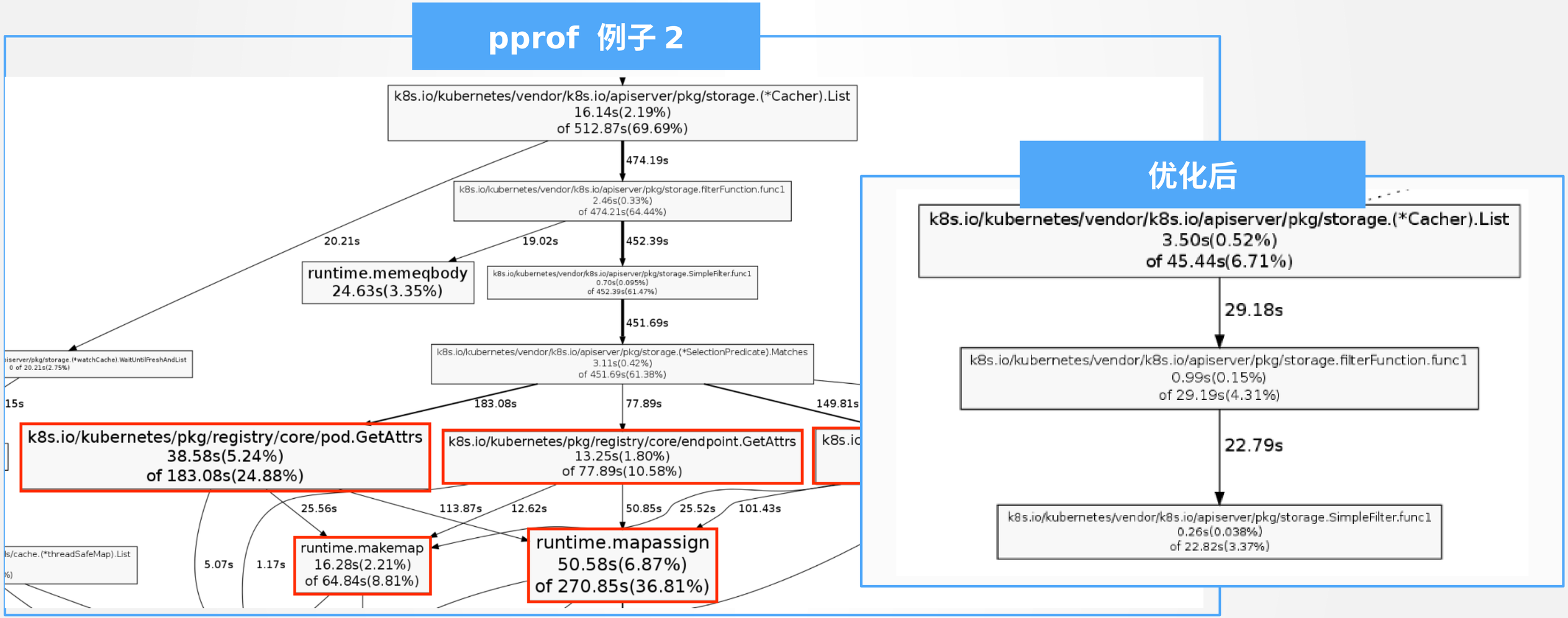
pprof 例子 2

- 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




```
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？

网易云

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

- 继续提升水位，满足一个机房容量需求
- 优化 Node 心跳汇报 / 各类控制器
- Client 端优化
- 高性能容器、网络优化
- 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
- 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
网易云
www.163yun.com