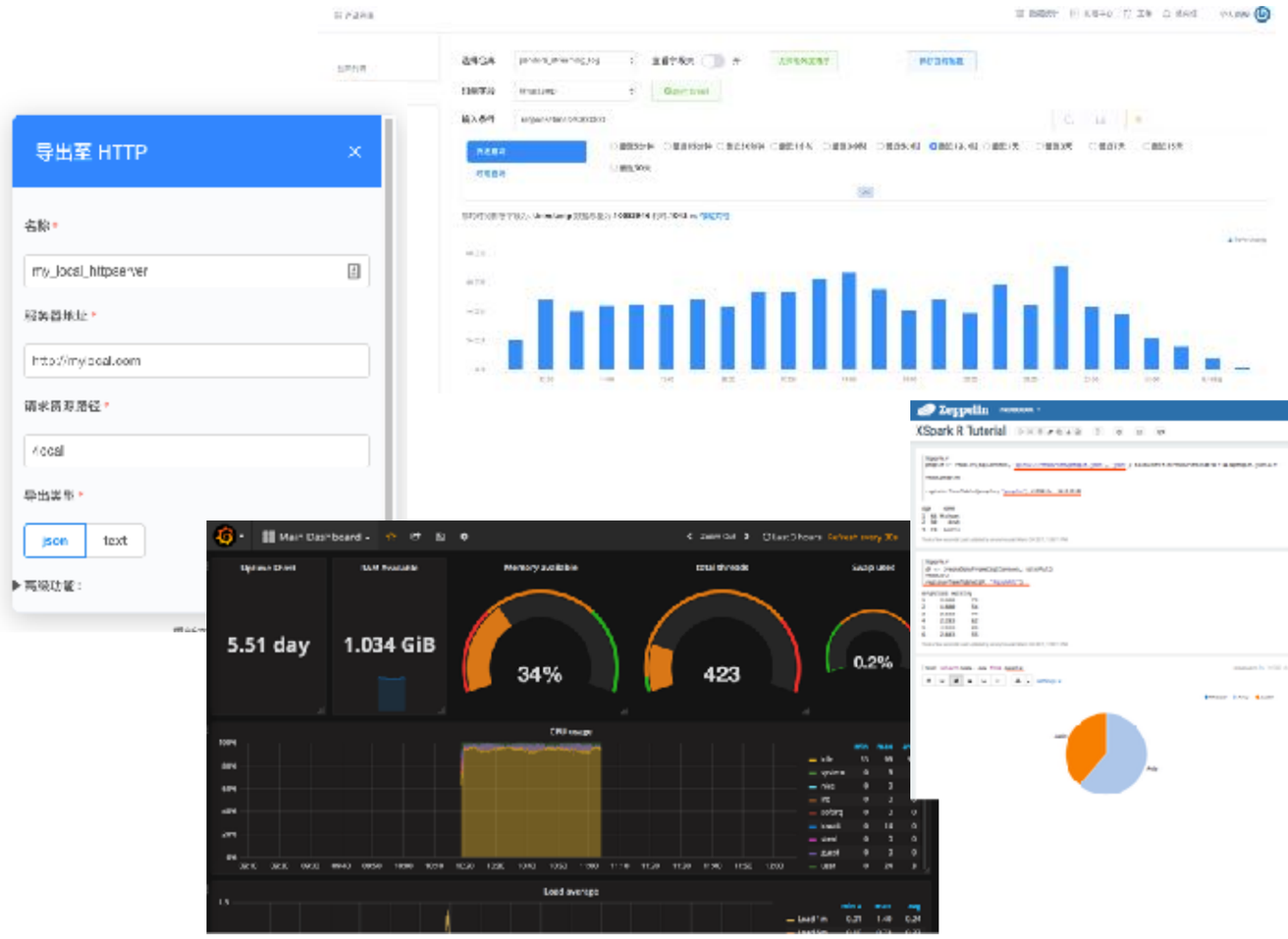
基于Go的大数据平台

七牛云－党合萱

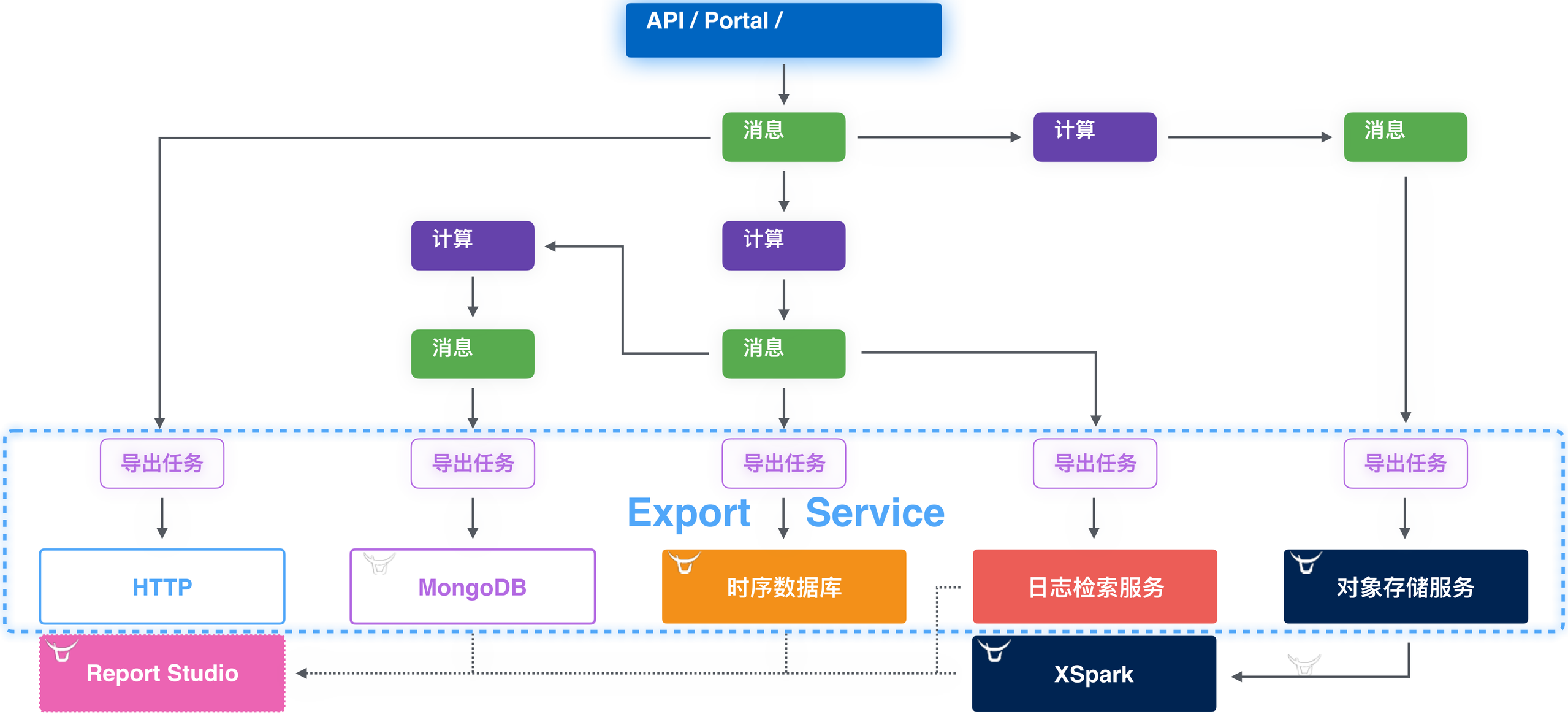
什么是Pandora


数据源
计算任务
消息队列
HTTP


导出至 HTTP
名称 *
my_local_httpserver
http://mylocal.com
/local
json
text
5.51 day
1.034 GiB
34%
423
0.2%
Load average

简单 · 可信赖

Pandora架构图

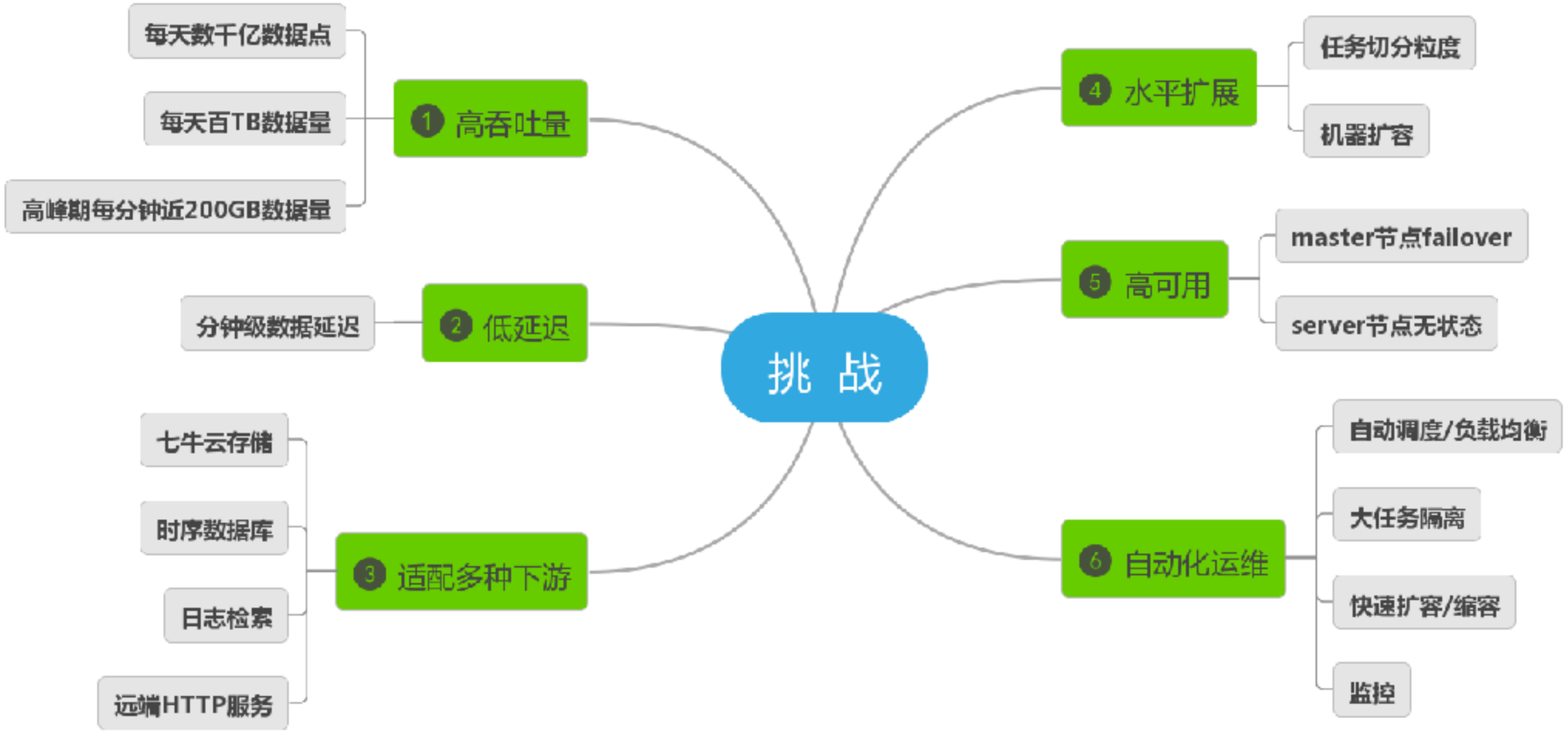


内容提要

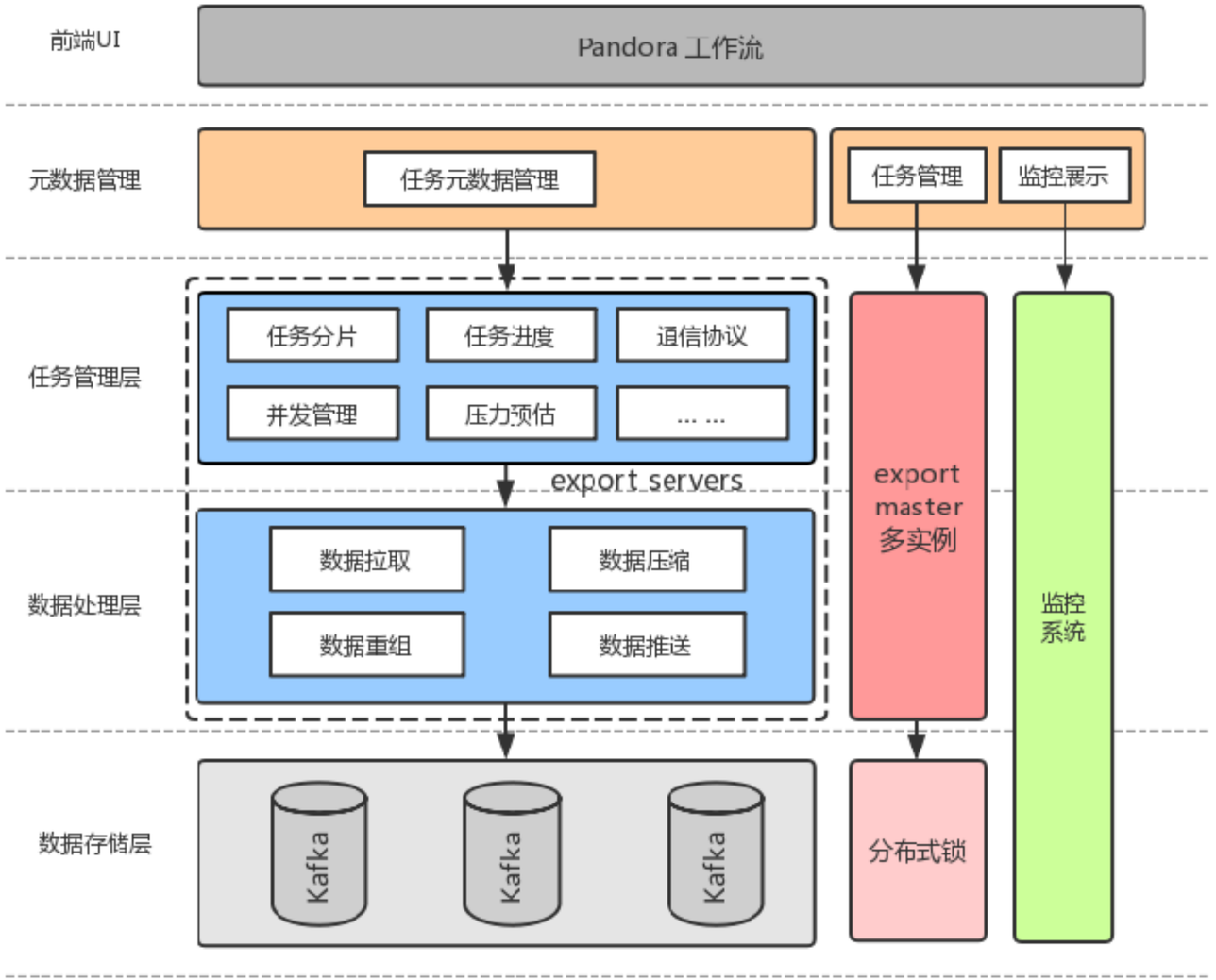
- 系统设计分析与架构
- 多种上下游适配
- 高吞吐/低延迟问题探究
- 高可用与水平扩展
- 自动化运维
- Go的应用

系统设计分析与架构

构建系统的挑战

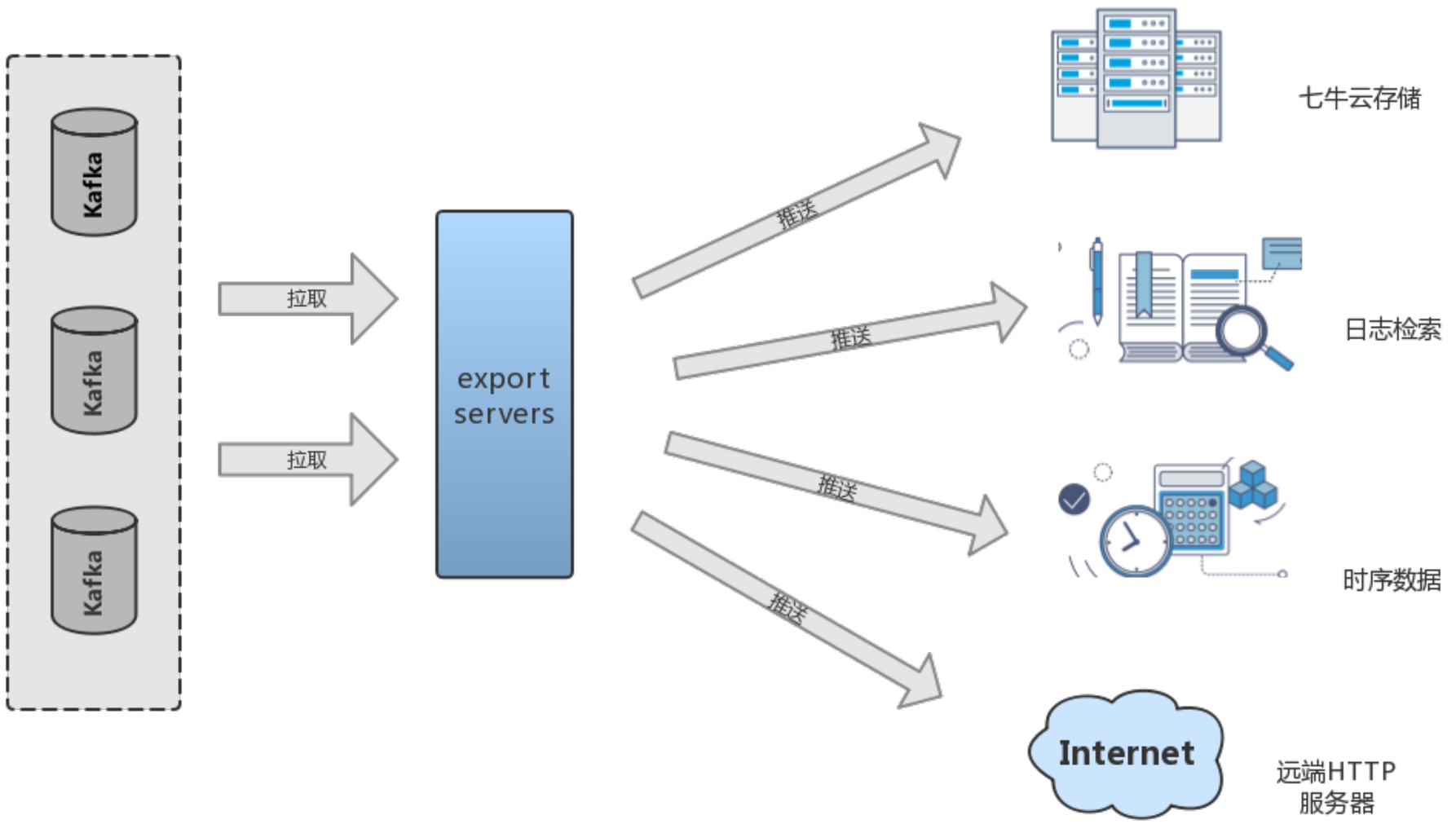

挑 战
1 高吞吐量
每天数千亿数据点
每天百TB数据量
高峰期每分钟近200GB数据量
2 低延迟
分钟级数据延迟
3 适配多种下游
七牛云存储
时序数据库
日志检索
远端HTTP服务
4 水平扩展
任务切分粒度
机器扩容
5 高可用
master节点failover
server节点无状态
6 自动化运维
自动调度/负载均衡
大任务隔离
快速扩容/缩容
监控

export service系统全貌

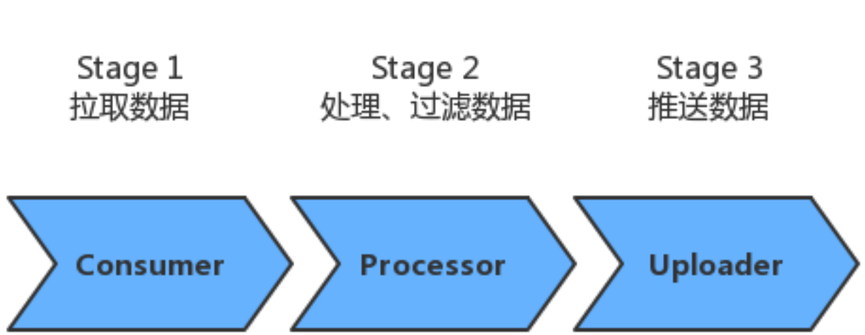


多种上下游适配

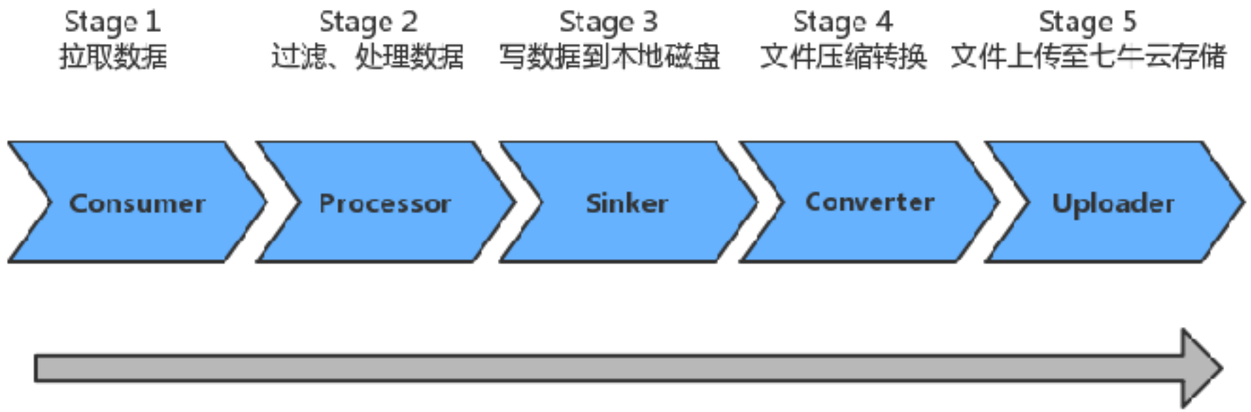
业务架构


Kafka
Kafka
Kafka
拉取
拉取
export
servers
推送
推送
推送
推送
七牛云存储
日志检索
时序数据
Internet
远端HTTP
服务器

导出模型



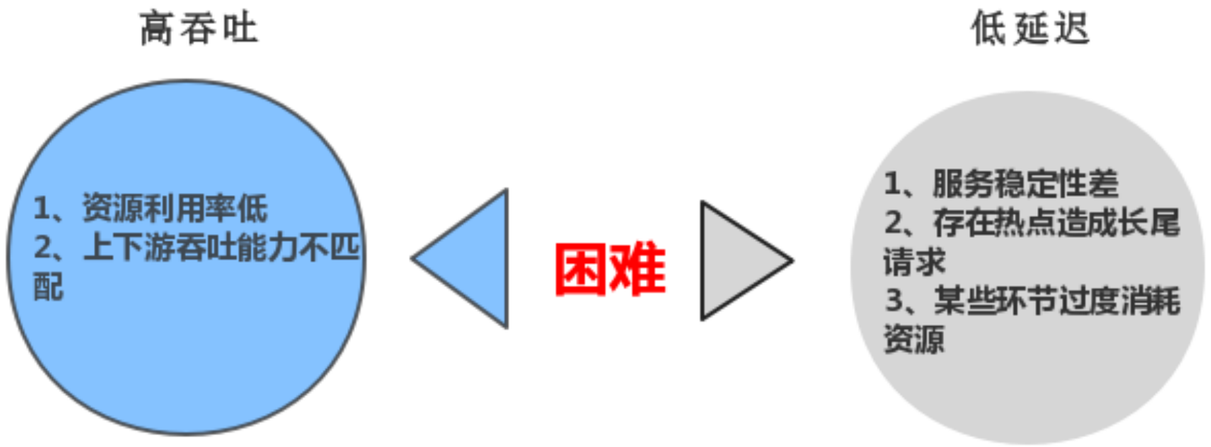
通用导出模型



云存储导出模型

高吞吐/低延迟问题探究

困难

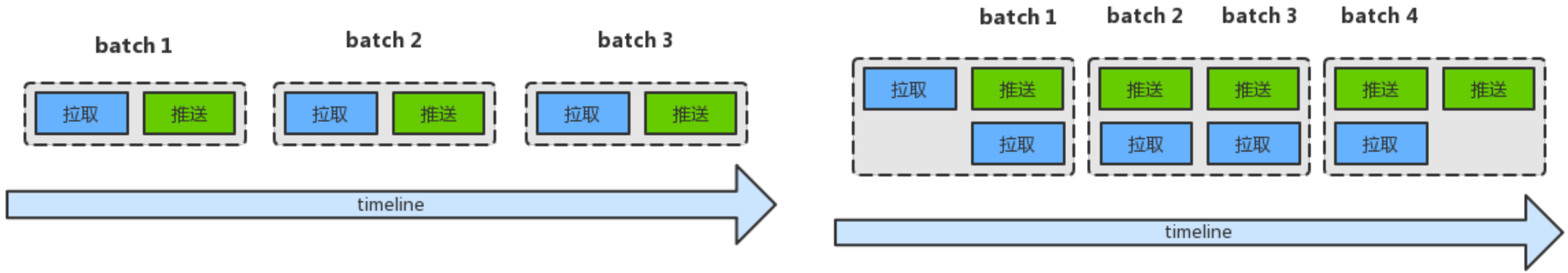


将导致Lag出现！

以上两者有很多共通点，除了这些，如果不能及时的发现上下游系统间的负载变化也会导致某一环节被打爆，而越是复杂的系统，问题也就越容易在某个不经意的环节出现！

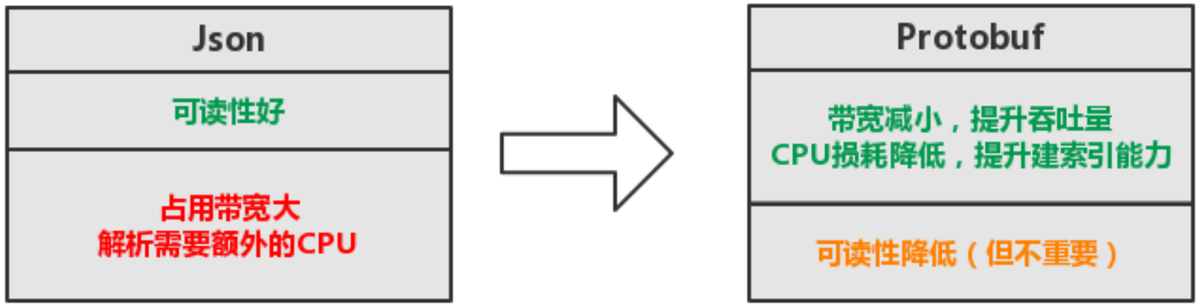
数据预取

- export server在向下游推数据的时候预先从上游拉数据回来，保证网络最大的利用率，同时也减小了等待时间，提升导出效率。
- 预取时如果无数据可取，则休眠1s再取数据，既然没有数据则休眠时间加倍……一直到32s为止，过程中如果取到数据，则休眠时间重置为1s，有效减少对底层存储的请求数量。



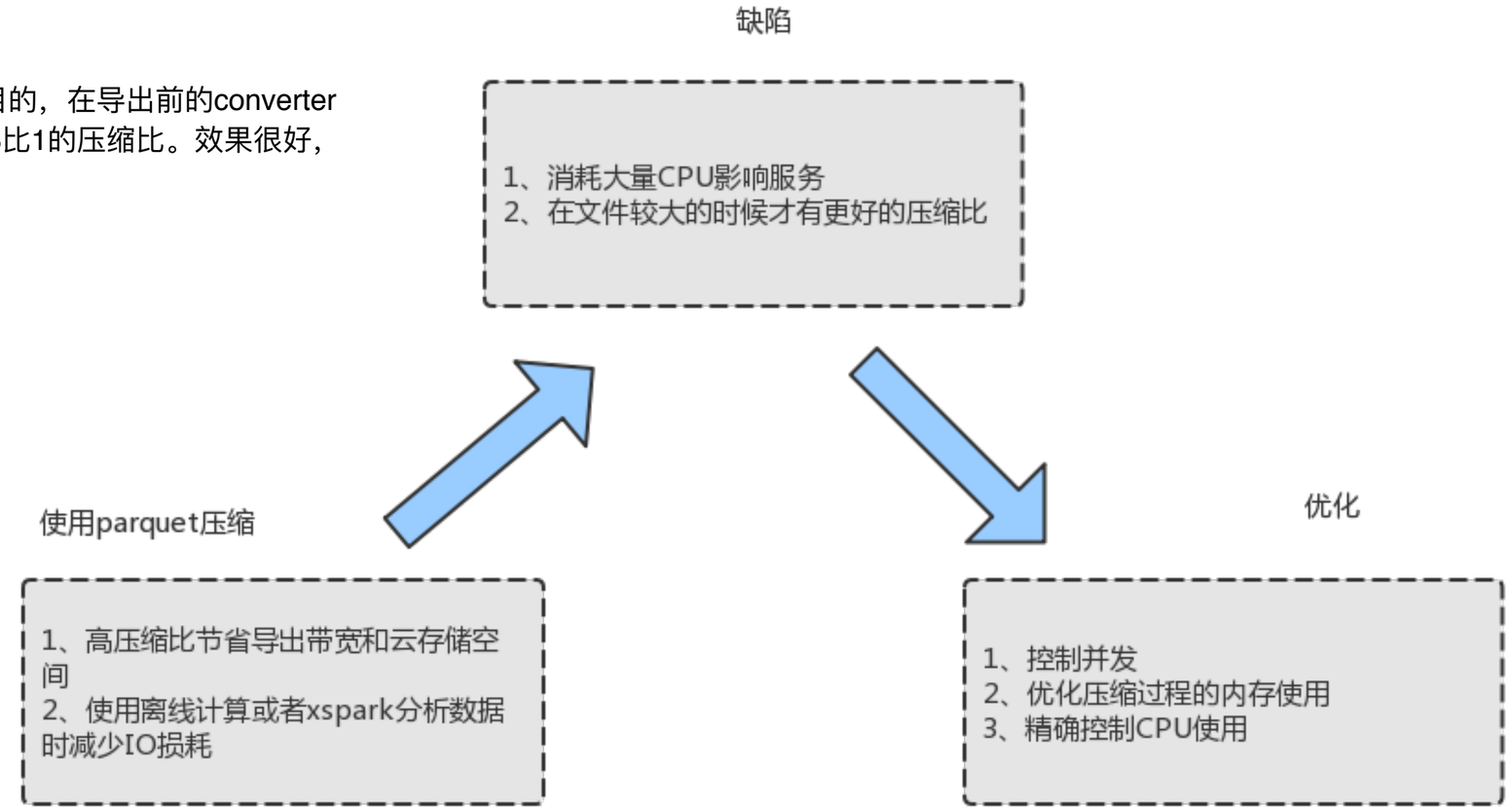
数据推送协议优化

- 优化export service与logdb之间的数据推送协议



资源使用优化

- kodo导出为了达到节省存储空间的目的，在导出前的converter这一步使用了parquet压缩，可以有8比1的压缩比。效果很好，但缺点也同样明显。



压力感知与退避算法

如何感知上下游压力

- 响应时间
- 特定错误码
- 超时错误

快速启动还是慢启动

- 速度的控制

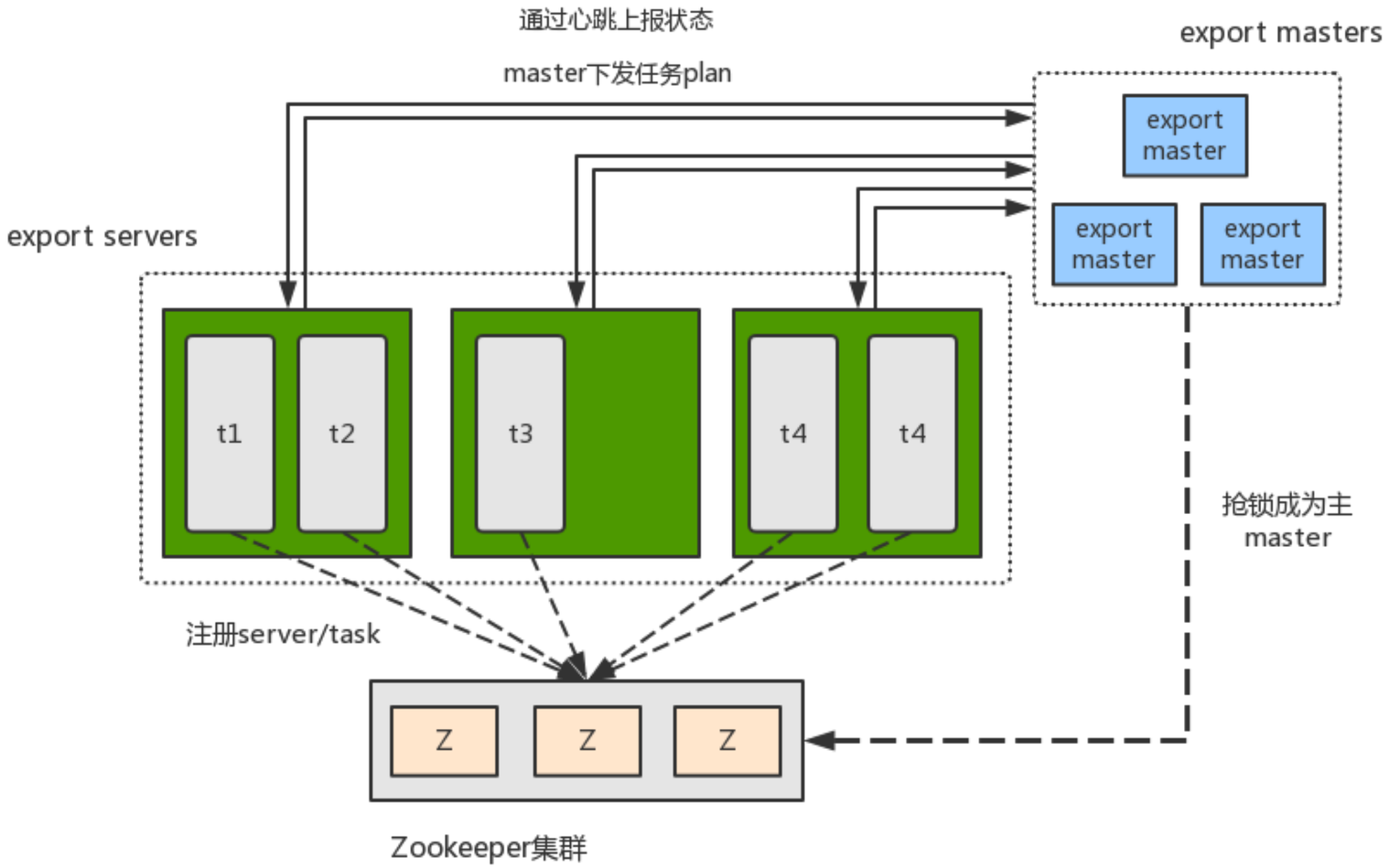
退避策略

- 起始阶段慢增长
- 指数增长

高可用与水平扩展

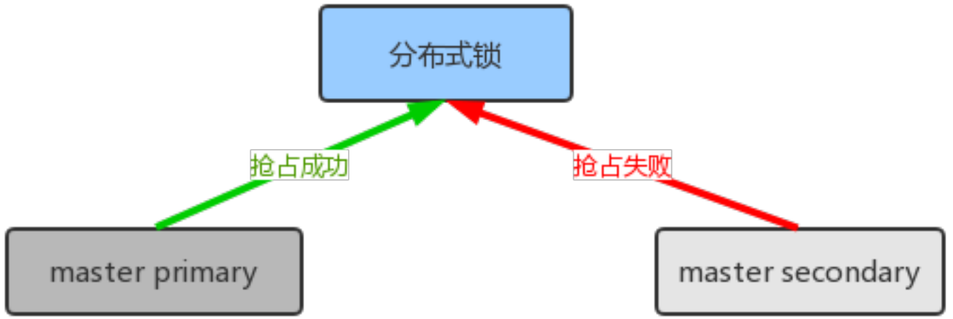
master/server架构

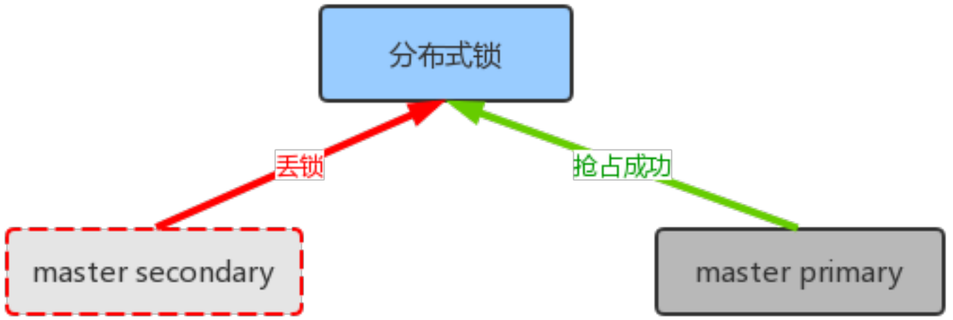
- master/server间采用golang rpc通信
- server上报心跳证明自己存活，并汇报所执行任务的情况
- master向server周期性下发任务，server管理自身任务决定哪些要执行哪些要丢弃



master高可用

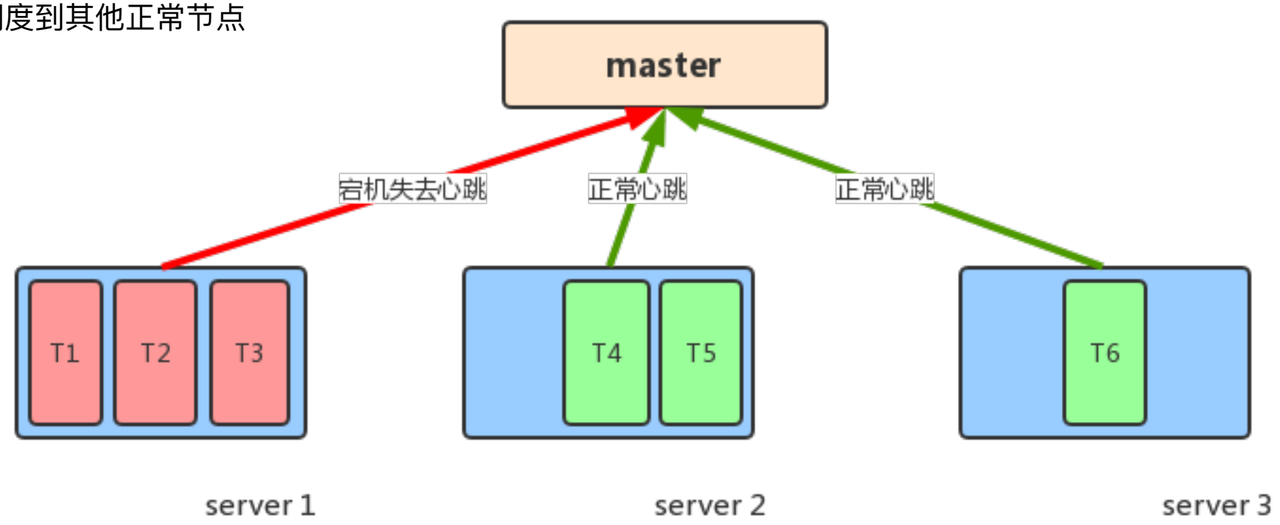
- master自身无状态，身份信息注册在zookeeper
- master failover时主备自动切换
- 主master丢锁会自杀，备master抢锁成为主master





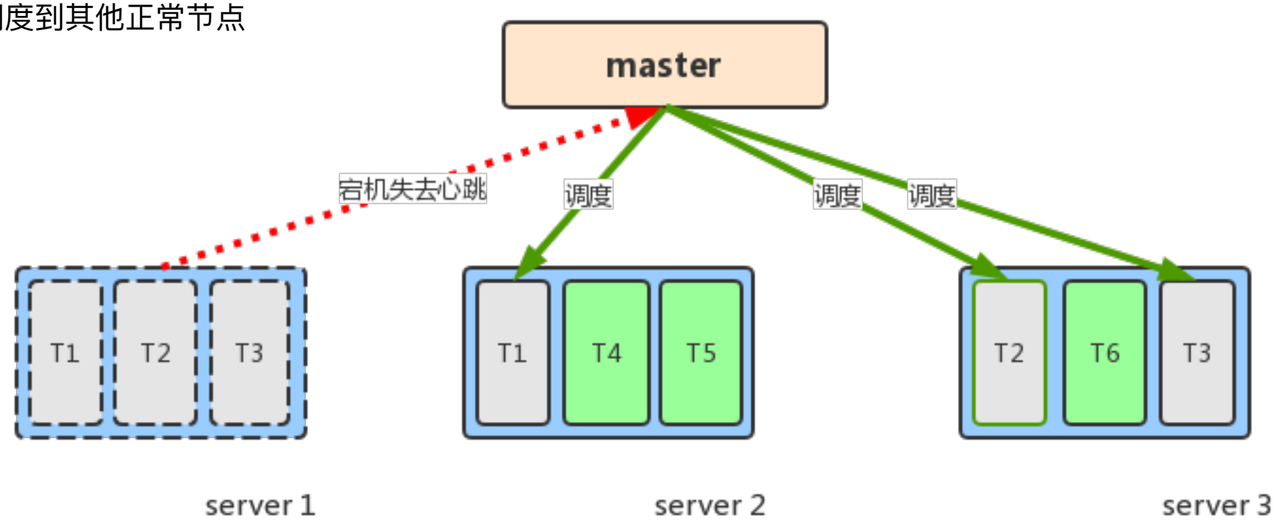
server高可用

- server注册自身，防止单机重复运行
- server注册每一个任务，防止任务被重复执行
- server高可用，节点故障时任务会被调度到其他正常节点



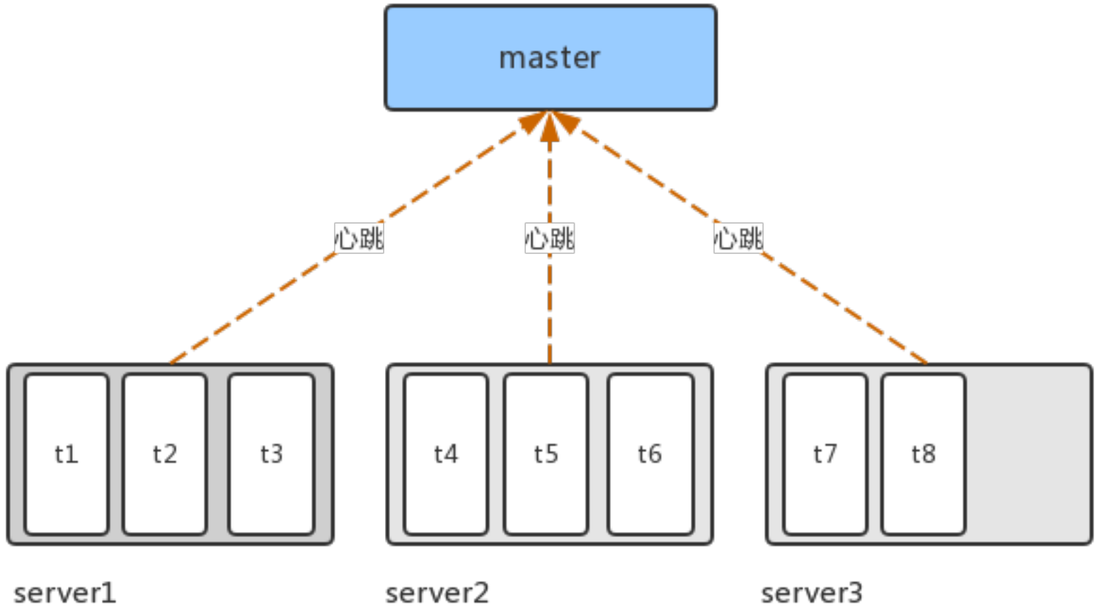
server高可用

- server注册自身，防止单机重复运行
- server注册每一个任务，防止任务被重复执行
- server高可用，节点故障时任务会被调度到其他正常节点



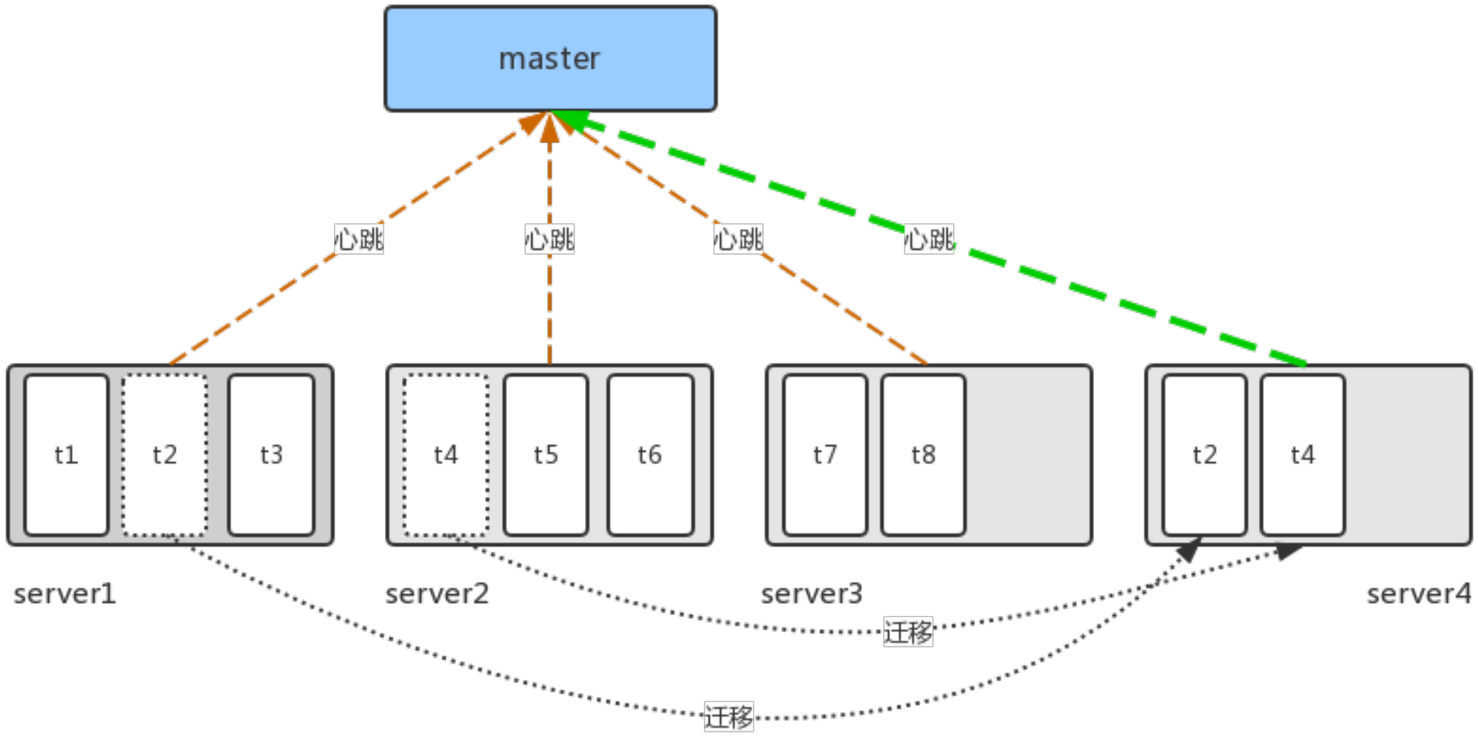
水平扩展

- 资源不足时加入新的机器作为新server
- 新server从zk上获取master身份信息
- 新server上报心跳给master
- 其他任务被调度至新server



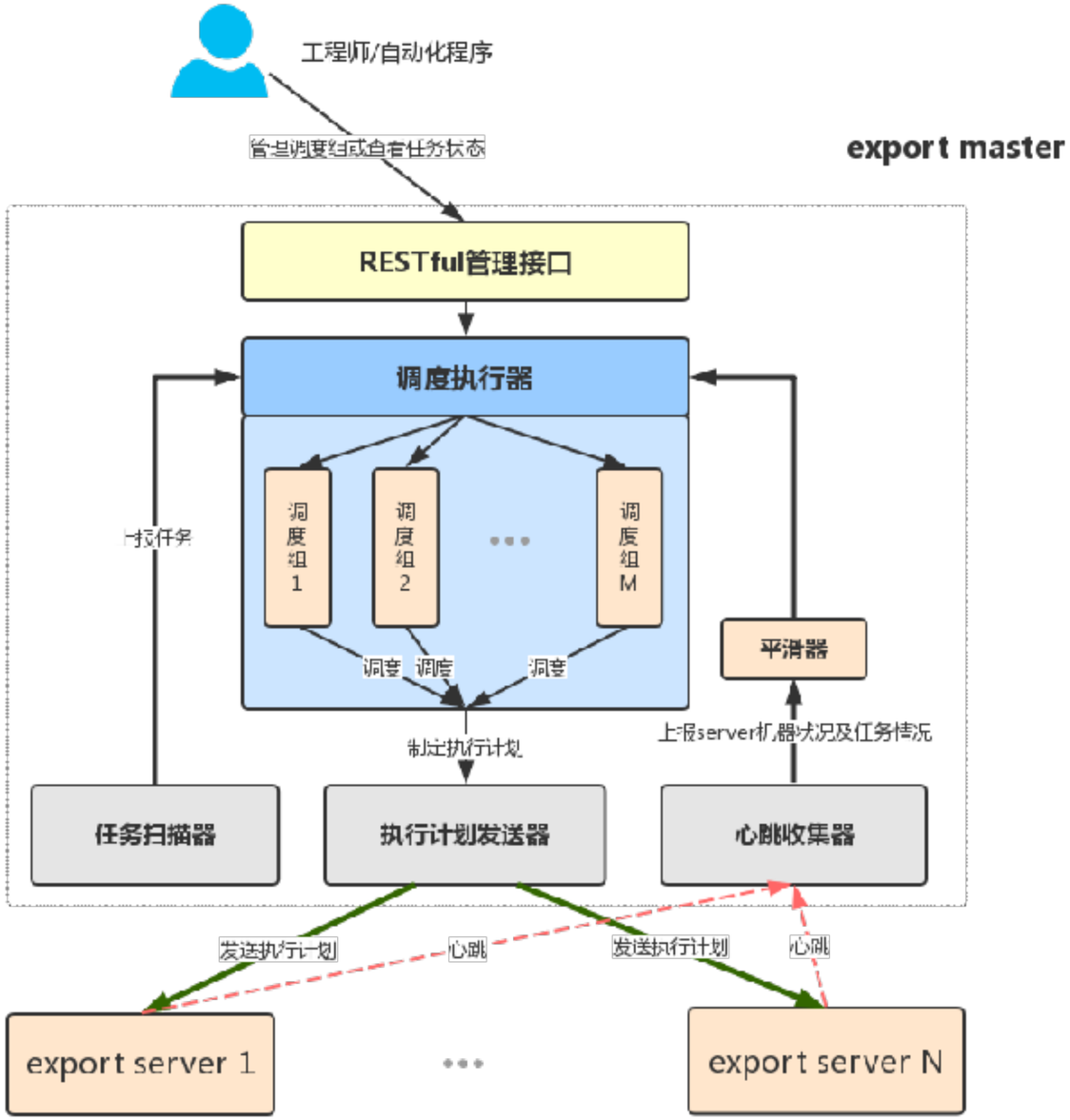
水平扩展

- 资源不足时加入新的机器作为新server
- 新server从zk上获取master身份信息
- 新server上报心跳给master
- 其他任务被调度至新server

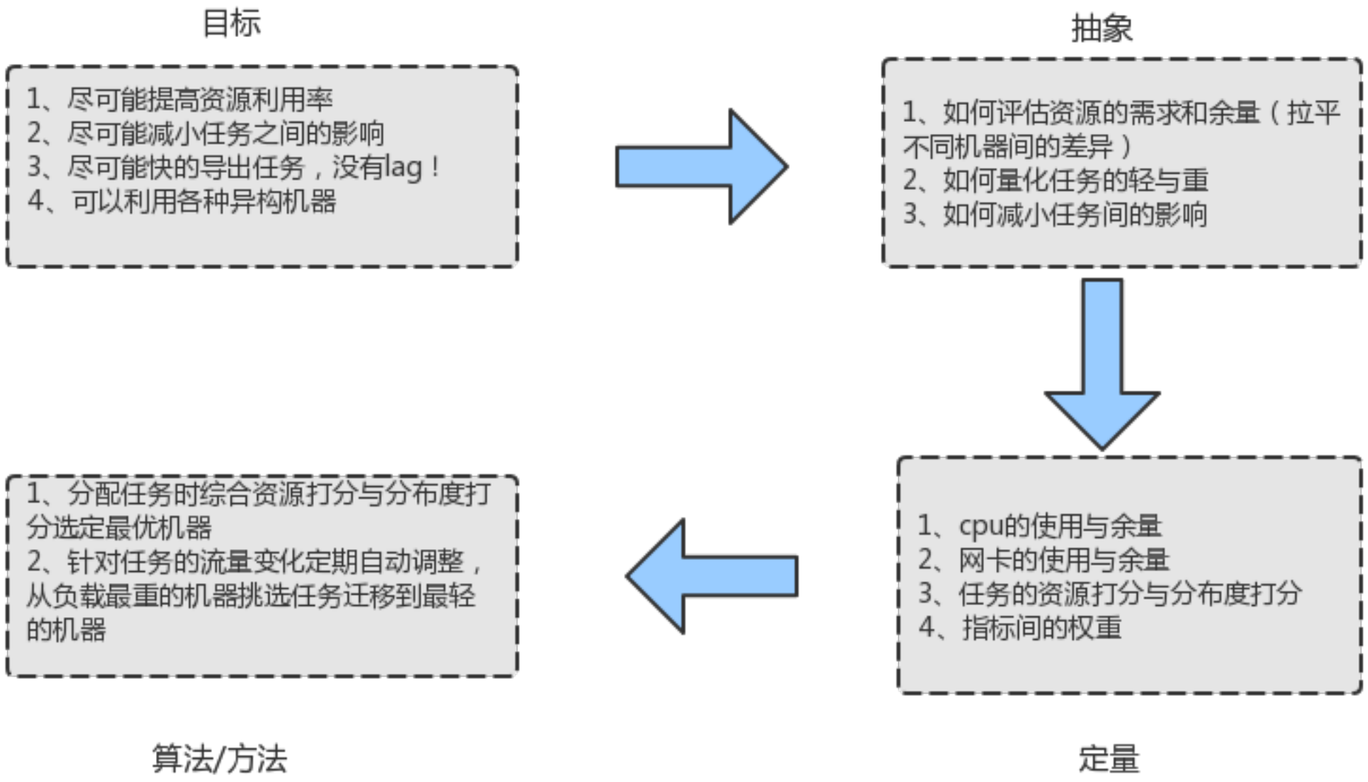


调度框架

- server感知
- 任务发现及粒度切分
- 数据平滑
- 调度组
- 管理接口



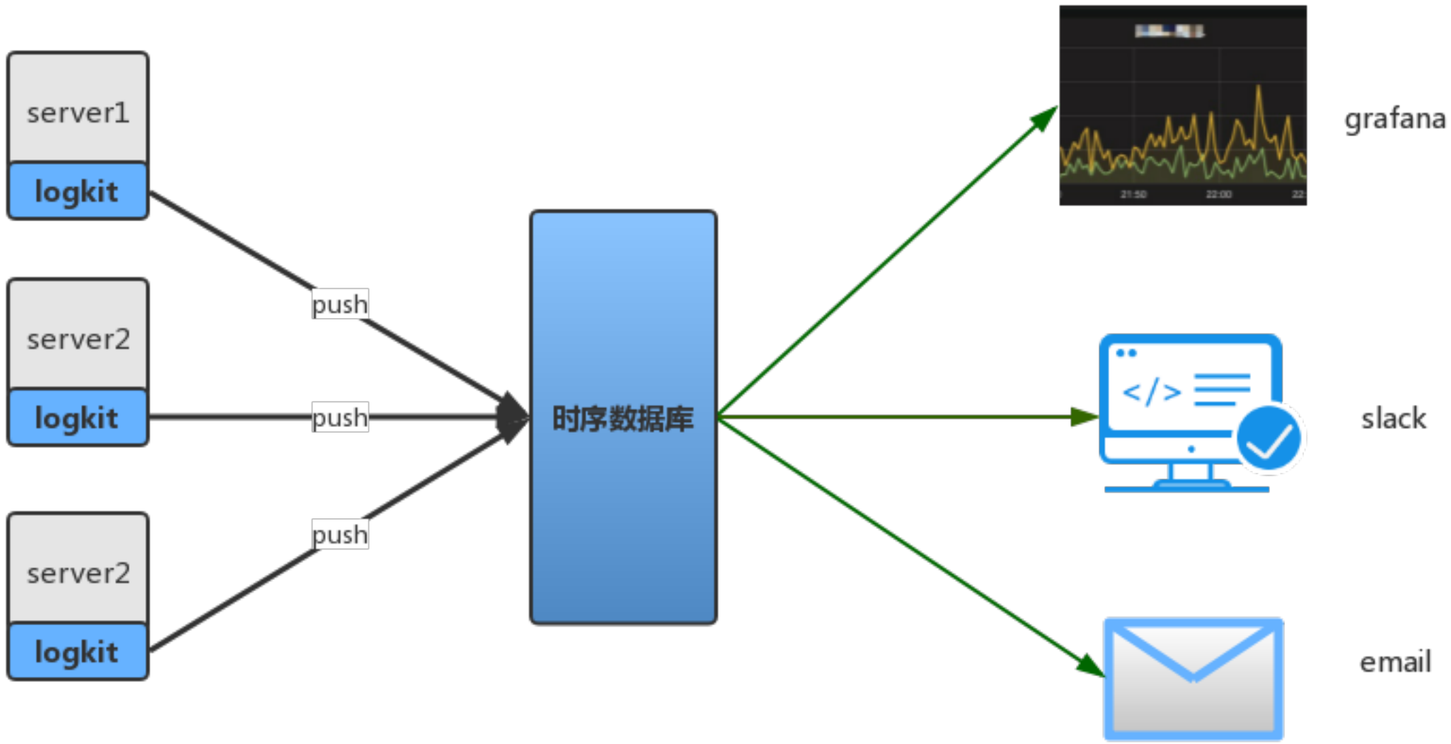
调度算法


目标
1、尽可能提高资源利用率
2、尽可能减小任务之间的影响
3、尽可能快的导出任务，没有lag！
4、可以利用各种异构机器
抽象
1、如何评估资源的需求和余量（拉平不同机器间的差异）
2、如何量化任务的轻与重
3、如何减小任务间的影响
1、cpu的使用与余量
2、网卡的使用与余量
3、任务的资源打分与分布度打分
4、指标间的权重
定量
1、分配任务时综合资源打分与分布度打分选定最优机器
2、针对任务的流量变化定期自动调整，从负载最重的机器挑选任务迁移到最轻的机器
算法/方法

自动化运维

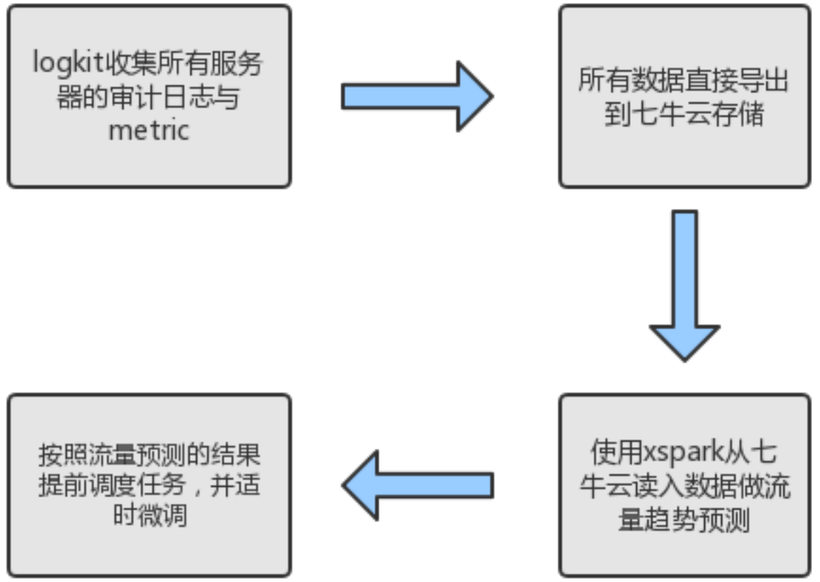
监控方案

- **logkit**：七牛pandora团队开发的纯go语言数据收集、推送工具，支持多种数据源，高效易用
- 时序数据库（TSDB）：兼容influxdb，适配grafana



系统热点自动感知与调整

- 一方面依靠日志对服务做审计、趋势预测，宏观上预知热点所在
- 另一方面依靠服务自身的状态反馈实时微调，修正宏观预测结果



线上系统现状

- 每日处理超过千亿数据点
- 每日处理百TB级别的数据量
- 线上导出延迟在1分钟以内
- 较少的人工介入
- 秒级扩容
- 实时的可视化监控系统
- 易用的报警系统
- 自动生成线上日报

Go的应用

我们用Golang做了些什么

- 流式计算、离线计算、日志检索、时序数据库等**一整套服务的核心代码**都使用Golang开发

- 简单、高效、易用的数据接入工具**logkit**，除了pandora之外可以接入多种数据库、kafka、机器metric信息等等

- 全套监控工具

为什么选择Golang

- 易上手，入门快
- 降低心智负担，集中精力在业务上
- 更简单高效的并发模型
- 丰富的库
- 七牛技术栈

Thank you！

- 了解更多的大数据玩法，尽在 https://qiniu.github.io/pandora-docs

简 单 · 可 信 赖