

Golang在百万亿查询引擎中的应用

Poseidon

郭军

360核心安全.云引擎开发组

<https://github.com/guojun1992>



个人简历简介

- 2013-2014 新浪
负责新浪微博游戏平台的开发
- 2014-至今 奇虎360
负责360会员中心架构设计和核心开发工作，增长至**1.5亿**用户，日活**1500万**，为公司**80**多个产品
推量

协同设计云控配置系统，为加速球和安全卫士提供服务，每天超过**100亿**次查询

初始设计并实施雷电OS用户中心

在代码库中实现了很多基础类库和组件

协同开发维护poseidon系统，能处理**一百万亿**行(**100PB**)日志的搜索引擎，已接入安全业务线**30**个子业务，已开源：<https://github.com/Qihoo360/poseidon>

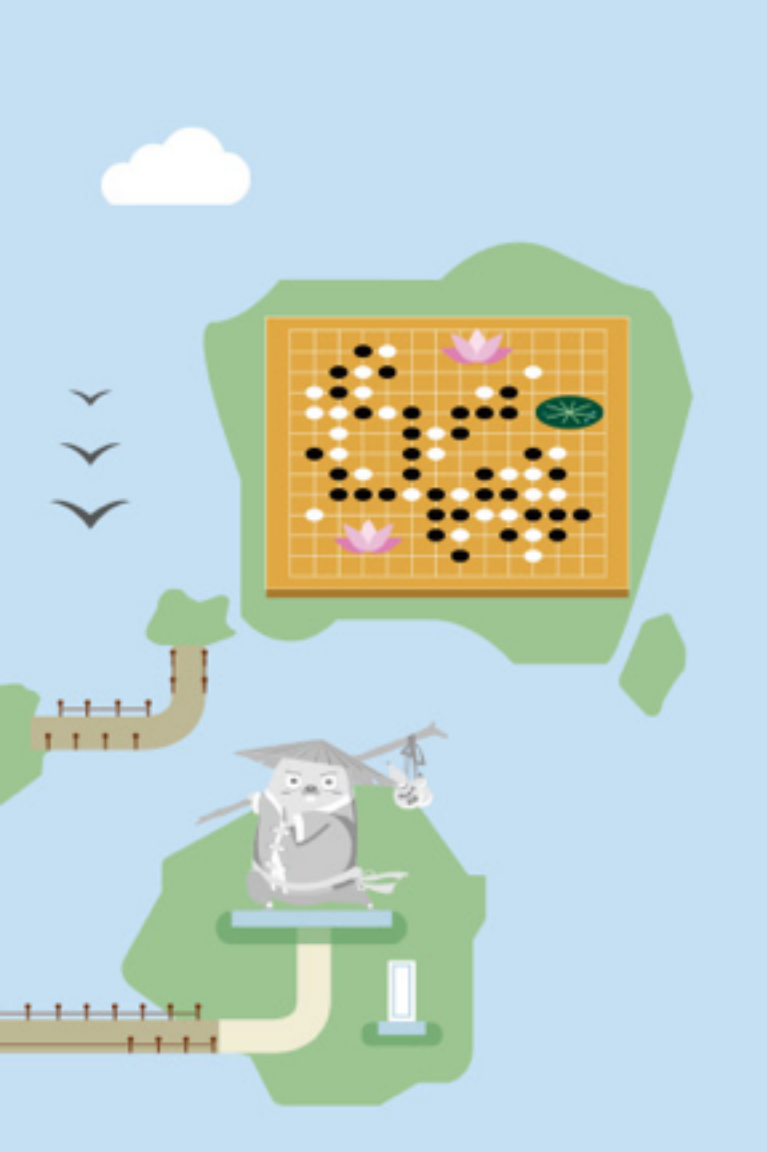
协同设计和开发软件替换引擎服务，为超过**2亿**用户提供服务

协同设计和实施云引擎基础监控设施

向国家专利局提交发明专利**43**件

目录

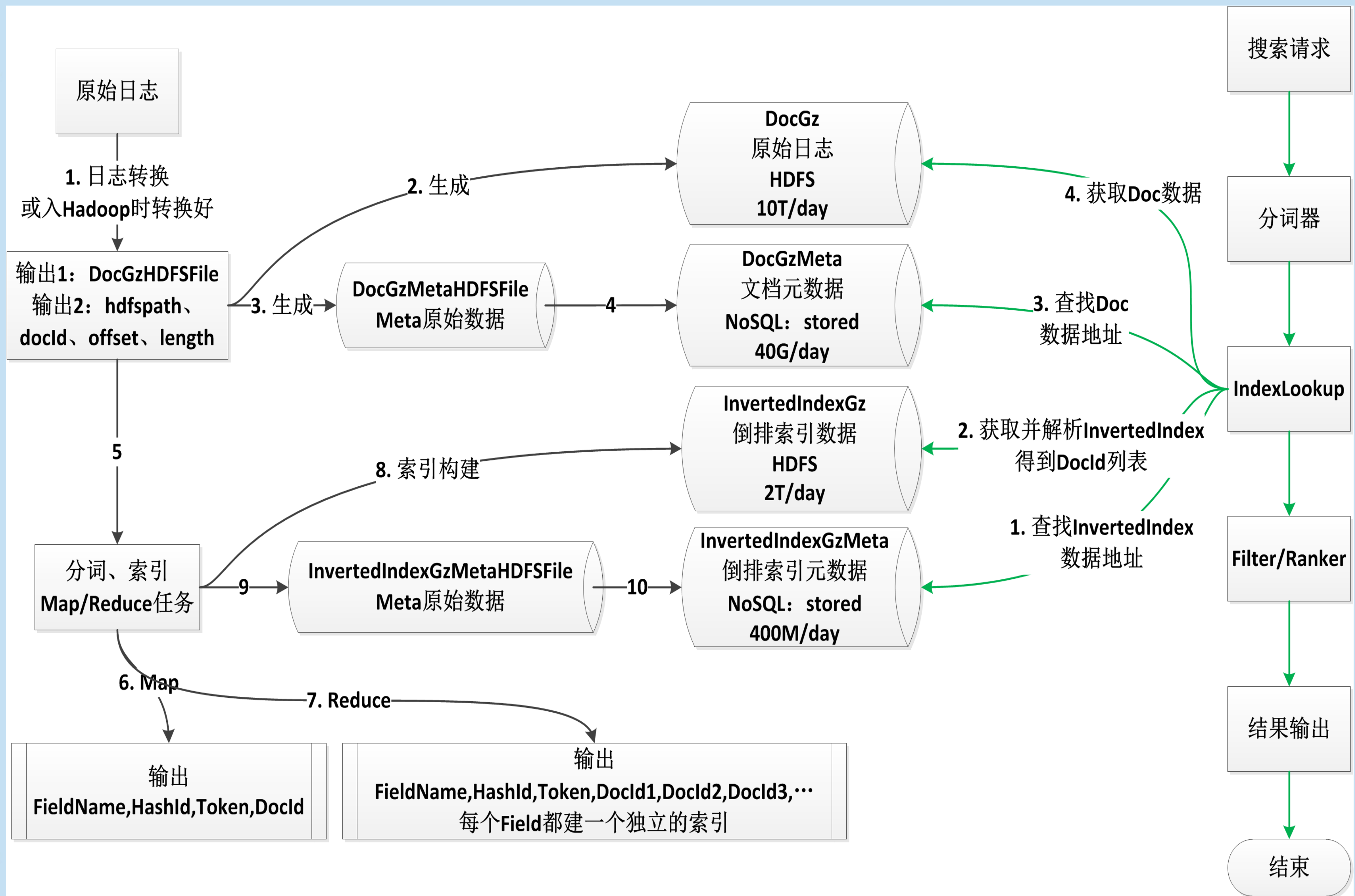
- 设计**目标**
- Go应用**场景**与遭遇的**挑战**
- 怎样**应对**?
- **开源**的改变
- 总结



设计目标



- 总数据量：保留**3年**历史数据，百万亿条、大小**100PB**
- **秒级**交互式搜索响应
- 每天支持**2000亿**增量数据灌入
- 故障转移，节点负载均衡，自动恢复
- 支持单/多天**批量查询，批量下载**



架构设计

在线引擎

web查询平台 (PHP)

Proxy集群 (Golang)

Searcher集群 (Golang)

ReadHDFS (JAVA)

分词、创建索引-离线

客户端请求, 生成日志

idgenerator (Golang)

离线生成DocGZ、
DocGZmeta

分词、倒排索引MR任务
元数据入NoSQL

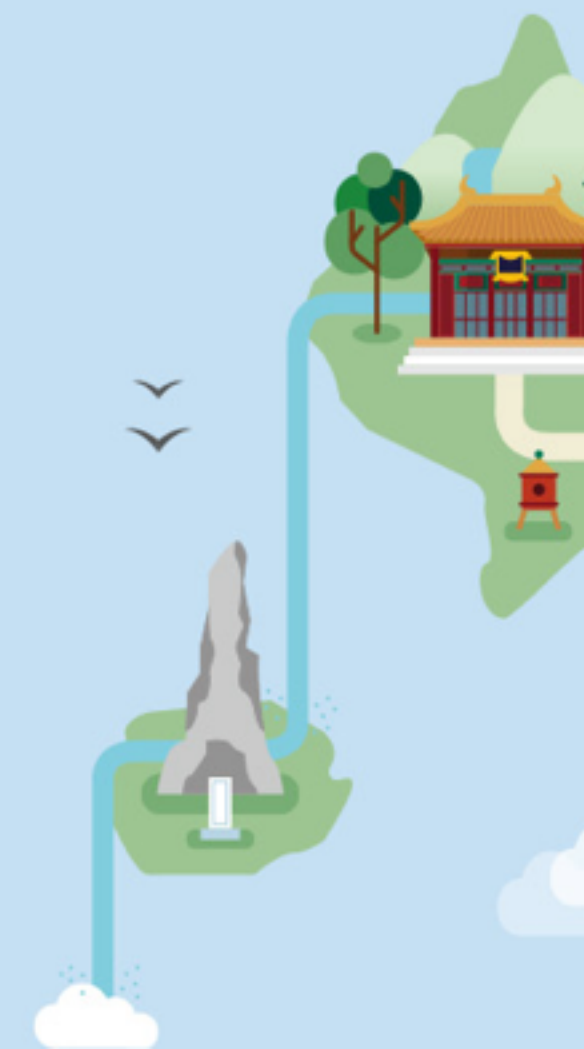
- 1、Mysql权限校验
- 2、模糊搜索 (可选)
- 3、查询四级索引**
- 4、过滤、去重、分页
- 5、返回原始数据

Hadoop集群

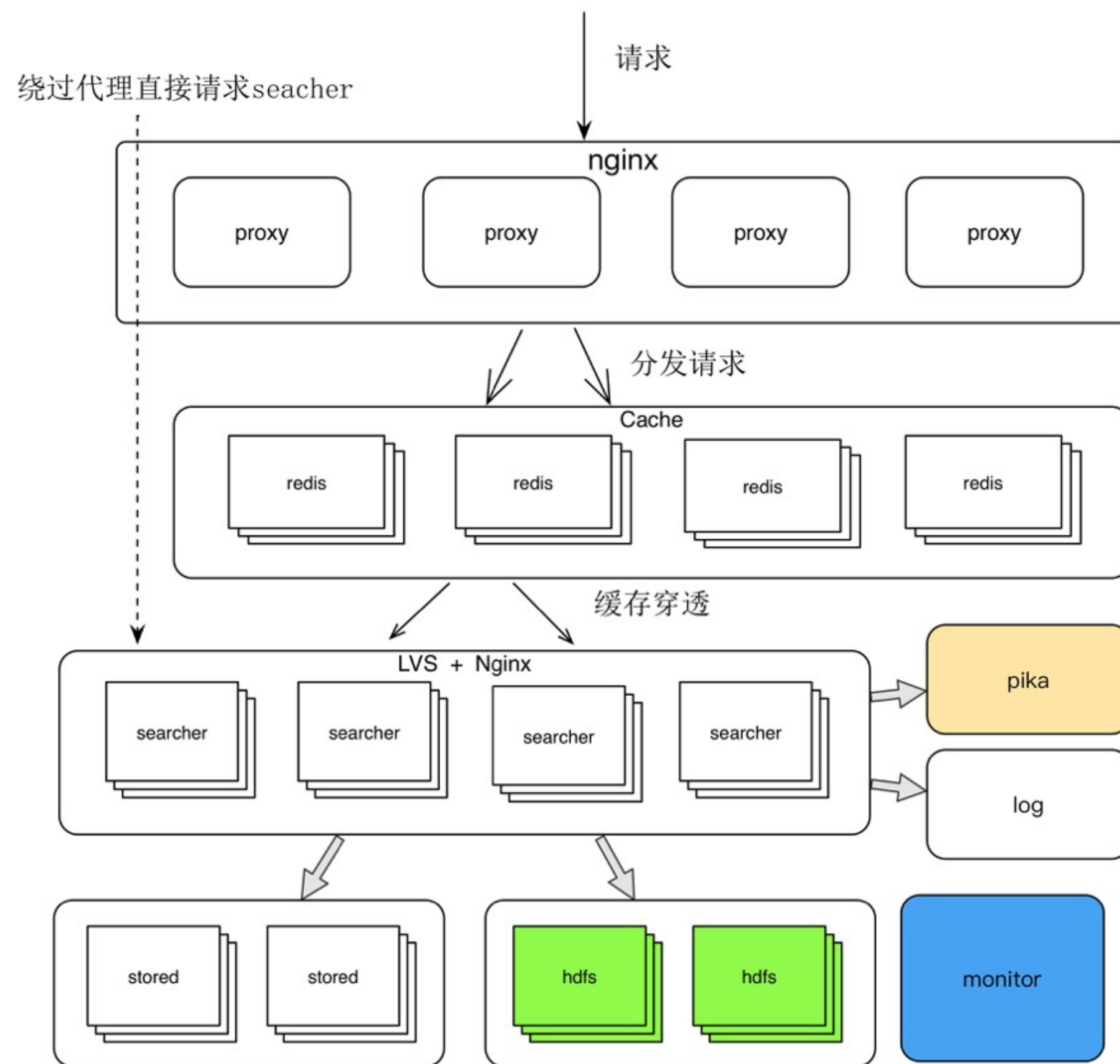
hdfs

hdfs

hdfs

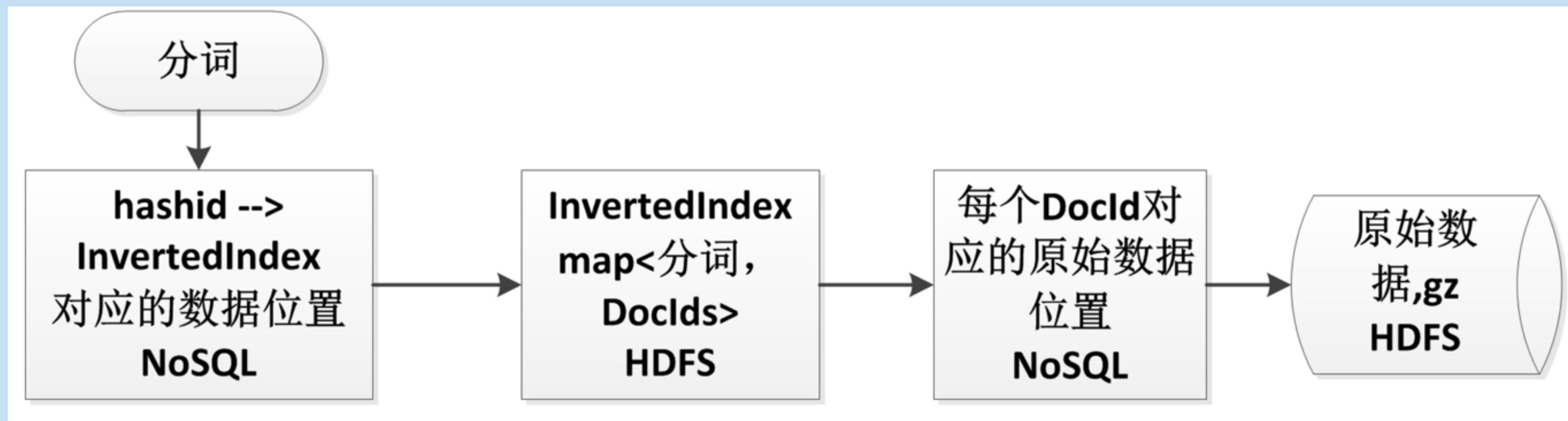


Proxy/Searcher详细设计



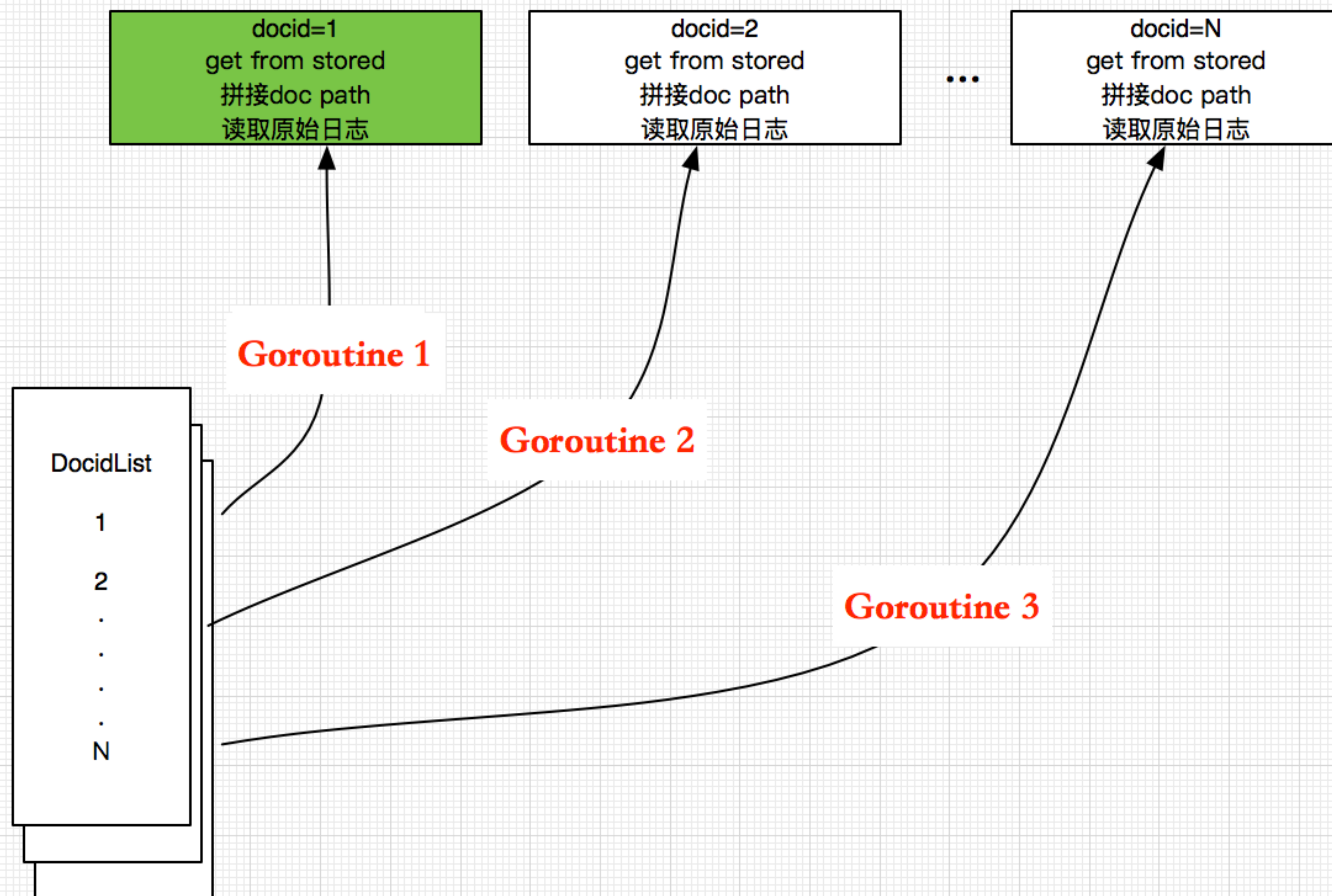
弥补倒排索引的缺陷—四级索引

- 关键字 -> **DocidList** Docid -> Doc



- “计算机科学领域的任何问题都可以通过增加一个间接的中间层来解决”

Searcher并发模型



问题与瓶颈

- 基础组件是c++, c++ -> c -> **CGO** -> Go
- gdb调试 进程过多

```
> SIGABRT: abort
> PC=0x32f5632625 m=45
> signal arrived during cgo execution
>
> goroutine 2289 [syscall, locked to thread]:
> runtime.cgocall(0x5334f0, 0xc820e758d0, 0x0)
> /usr/lib/golang/src/runtime/cgocall.go:120 +0x11b fp=0xc820e75888 sp=0xc820e75858
> golib/cgo/symc._Cfunc_symc_get(0x7fed79659840, 0xc820370520, 0x1, 0xc820370518, 0xc822f3e260, 0xc820e75888)
> golib/cgo/symc/_obj/_cgo_gotypes.go:106 +0x36 fp=0xc820e758d0 sp=0xc820e75888
>
>
```

core dump

解决方案

- 用Go重新实现一遍
- 将组件作为http服务，Go Client调用

问题与瓶颈

- 大量使用goroutine，子协程panic在主协程不能被recover，如何统一处理？



解决方案

- 通道类型为struct，封装正常数据和error，在主协程从通道取出数据，统一做处理

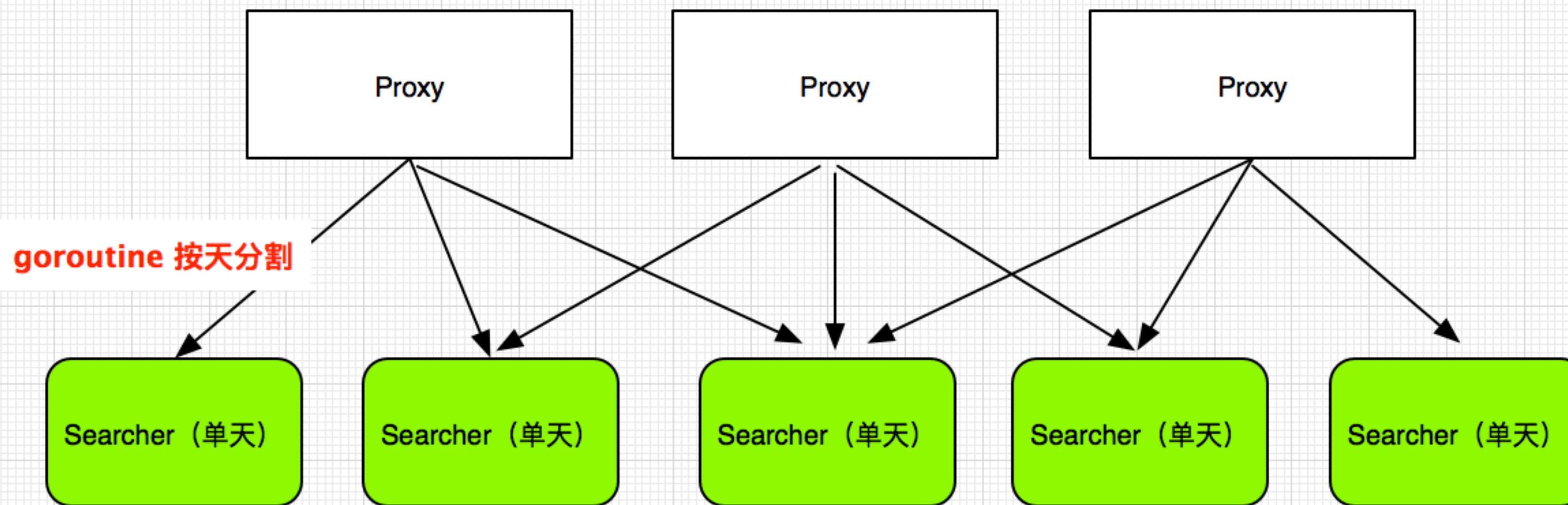
```
type DocDataResult struct {  
    DocId      int64  
    RowIndex   int32  
    Data       []byte  
    Err         error  
}
```

经验小结

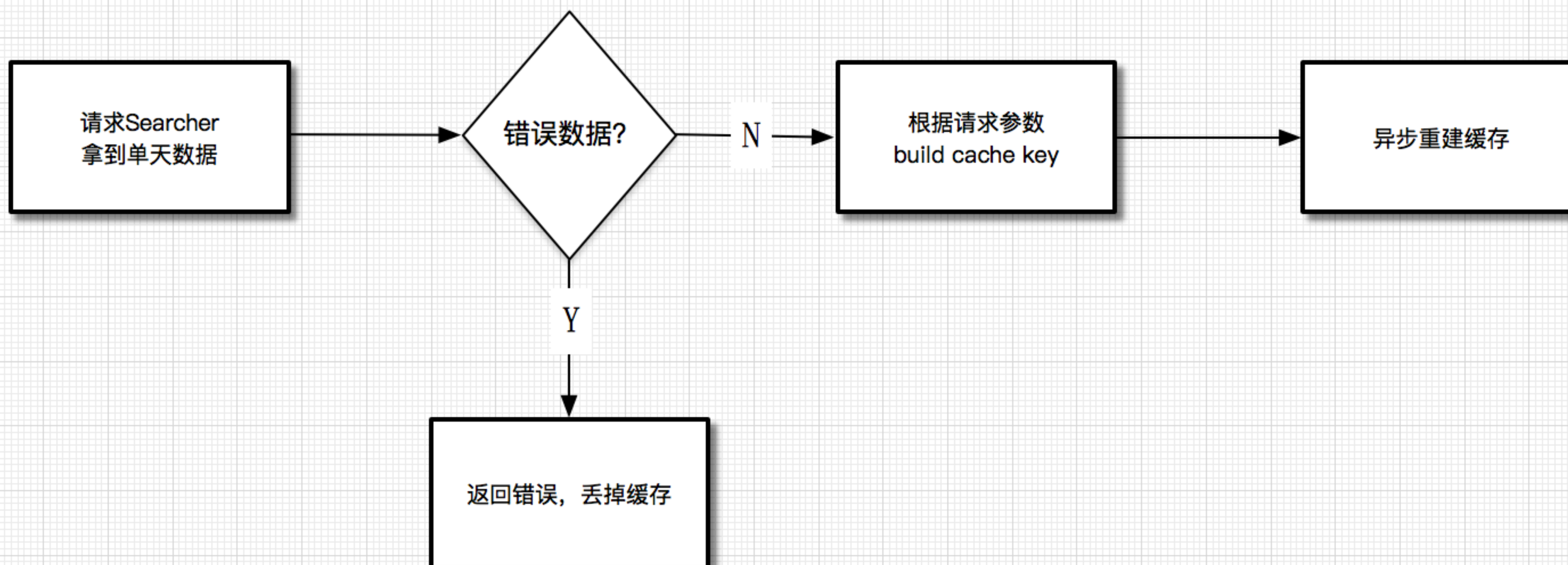
- 即使精通多种语言，最好**不要混用**，**谨慎**引入其他语言的解决方案
- 不要完全相信recover，它**不能恢复**runtime的一些panic

```
defer func() {  
    if err := recover(); err != nil {  
        in = []reflect.Value{reflect.ValueOf(err)}  
        method := vc.MethodByName("OutputError")  
        method.Call(in)  
    }  
}()
```


Proxy多天并发查询设计



Proxy多天并发Build Cache设计



索引数据变“冷”

- 同一份数据在缓存时间内不会走整个readhdfs流程
- build index 程序挂了会报警感知，但是数据错误却是未知
- 修复数据重建索引需要时间

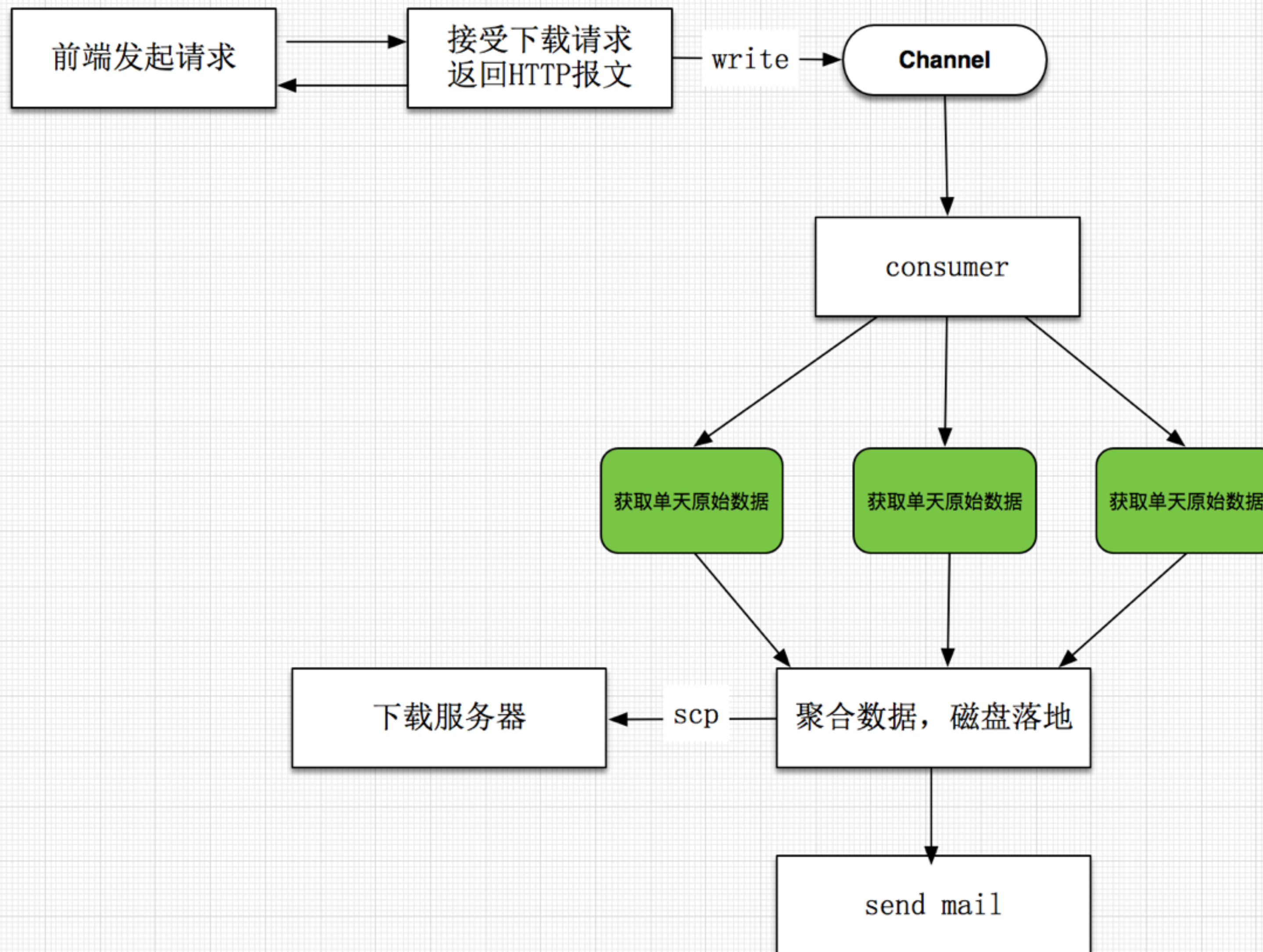
解决方案

- 减少缓存时间，在可容忍错误数据的时间内，用户查询及时发现问题
- 参考NSQ，利用**for+select**的不确定性来分流，随机流量到cache和hdfs做**热测试**，缺点：**开发成本较高**

经验小结

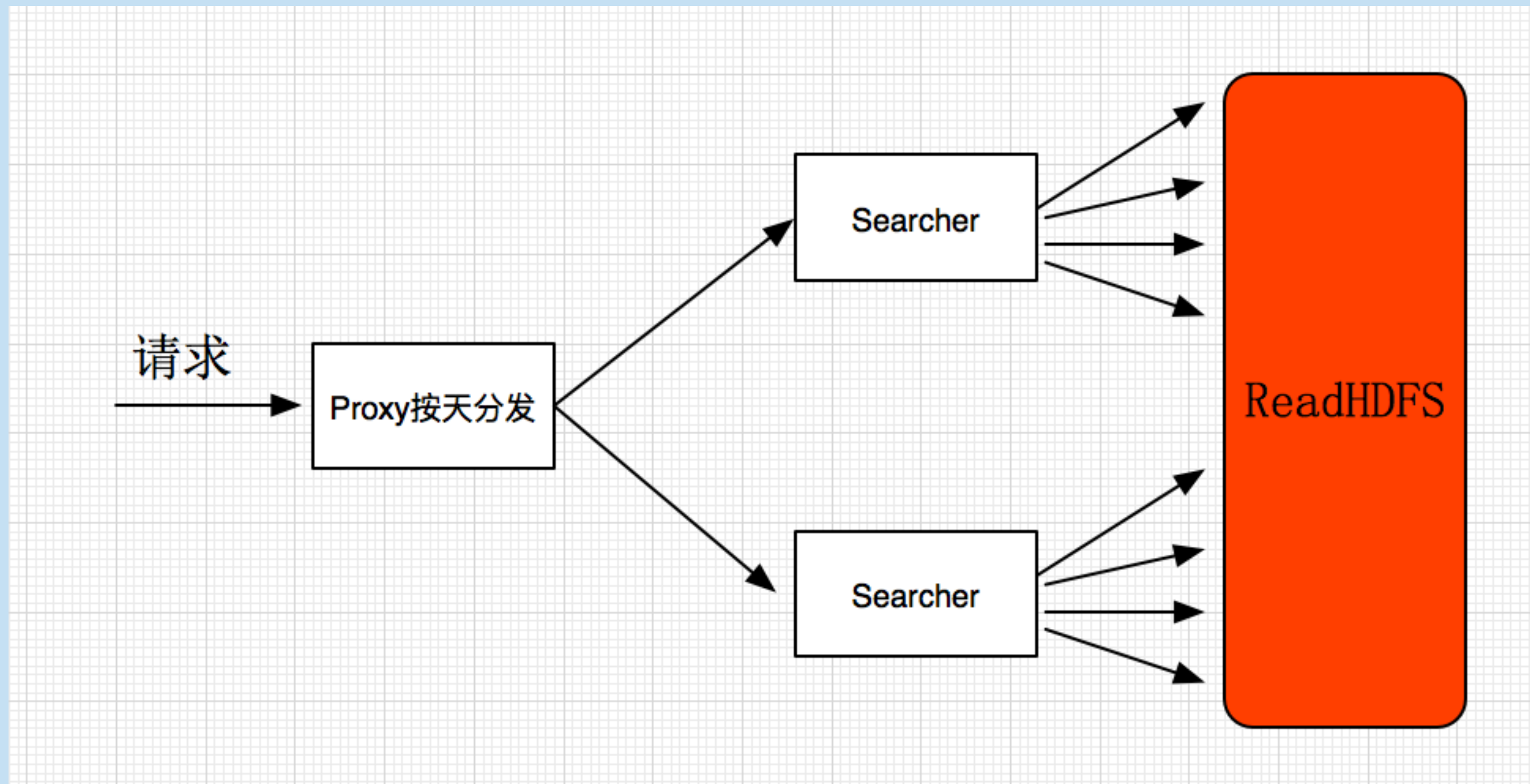
- 选择简单、有效、够用的解决方案
- 利用goroutine设计并发程序很方便，但是程序的并发运行模型要hold住

Proxy多天异步下载



ReadHDFS雪崩效应

- goroutine太多，底层readhdfs挂掉

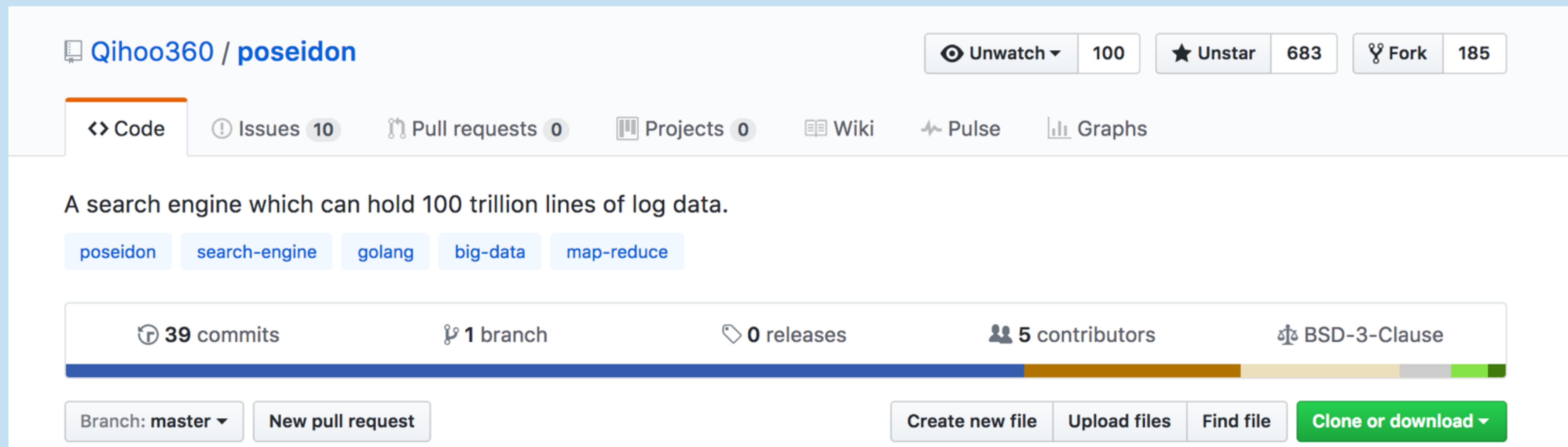


解决方案

- 连接池
- 熔断机制

开源

- 依赖组件太多，AllinOne想法



总结回顾

- 开发体验好，性能高，服务稳定，可以满足大部分场景的性能需求
- go语言的程序开发需要在代码可读性与性能之间做好平衡取舍，应用程序并发模型要在控制之内
- 谨慎与其他语言结合使用，即使对两种语言都很熟
- 视情况而定是否需要加前端代理，比如：nginx



Q & A



谢 谢

北京奇虎360科技有限公司

