中华数据库行业协会

2014中华架构师大会

11.15-11.16

联系我们：
联系人：朱小姐
联系电话：136 5197 9898
联系QQ：378091820

360
www.360.cn

集中式配置管理系统QConf

朱超

传统的配置管理方式

- **配置文件**
  - 存储在SVN/Git里
  - 每个机房一个
  - 上线脚本根据机器的hostname生成软链指向相应机房

```
lrwxrwxrwx 1 zhuchao zhuchao 12 11-15 16:07 config.php -> config.php.b
-rw-rw-r-- 1 zhuchao zhuchao 22 11-15 16:08 config.php.a
-rw-rw-r-- 1 zhuchao zhuchao 22 11-15 16:08 config.php.b
-rw-rw-r-- 1 zhuchao zhuchao 22 11-15 16:08 config.php.c
```

- **配置更新**
  - 修改配置文件
  - 提交SVN/Git
  - 重新上线

传统方式的不足

- **配置文件方式的缺点**
  - 繁琐、不直观、易出错
  - SVN/Git里存在很多没有实际意义的提交
  - 每增加一个新机房就需要增加新的配置文件
  - 上线过程中有可能失败
  - 配置出错不能快速恢复

QConf的目标

- **目标就是解决上述问题**
  - 配置内容与程序代码完全分离，SVN/Git里只保存代码逻辑
  - 配置集中存储，统一管理，不再需要配置文件
  - 配置值一经修改，实时同步到所有机器
  - 更多方便快捷的附加功能
    - 查看每台机器的配置更新状态
    - 快速回滚到旧配置
    - 一键复制多个配置项到新机房

ZooKeeper

- **存储所有配置信息**
  - 节点(node)：节点名(path)、节点值(data)
  - 一个节点代表一个配置项
  - 多个节点的树形层级结构类似文件系统
  - 只使用普通节点，不使用临时节点
- **回调通知机制**
  - 客户端注册对感兴趣事件的监视(watcher)
  - 事件发生后服务端通知客户端，回调函数被执行
  - 监视是一次性的
- **集群**
  - 三台以上ZK组成
  - 数据强一致
  - 每个机房部署独立集群

方案1

- **以PHP扩展形式实现**
  - 扩展加载时连接ZK服务端
  - 每次读取配置值都要通过网络，延迟高

方案2

- **目标是降低读操作的延迟**
  - 扩展加载时连接ZK服务端
  - 初次读取到的配置值放入共享内存缓存起来
  - 再次读取配置时，先去共享内存中检索，延迟大大降低
  - 配置值改变时，由ZK回调通知fpm进程，去更新共享内存里的旧值
  - 每个php-fpm进程都维持到ZK的长连接，一台客户端机器上默认有128个fpm，ZK服务端连接数 = 128 * 客户端机器数，不堪重负

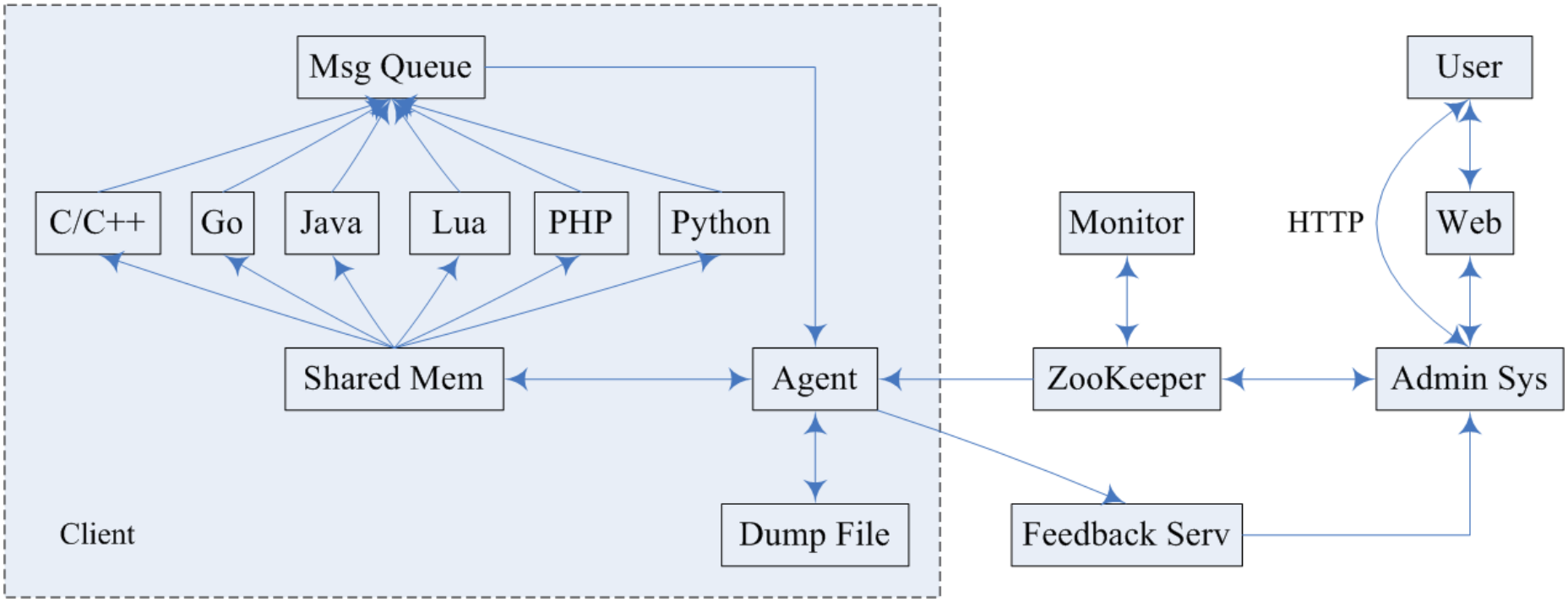
方案3

- **目标是减少到ZK的连接数**
  - 每个key只有一个fpm进程连接ZK，其他进程只读
  - fpm进程互相是对等的，因此必须设置锁，抢到锁的进程成为写进程，其他进程从共享内存里取值
  - 为防止内存泄漏，每个fpm进程处理完N个请求后会自动被kill掉，写进程被kill掉后会重新抢锁
  - 每个key可能被不同的写进程负责维护，锁数量较多
  - 逻辑复杂，有死锁可能

现方案

- **目标是降低复杂度**
  - 独立进程agent负责与ZK通讯并写共享内存
  - 客户端SDK只从共享内存里读值，对ZK一无所知
  - SDK需要的key值通过消息队列发给agent
  - 优点1：agent不像fpm进程一样不定期退出，也不需要抢锁，复杂度大大降低
  - 优点2：一台客户端机器到ZK只有一个长连接
  - 优点3：多语言支持很方便，因为大部分逻辑移到了agent里

整体架构


Msg Queue
C/C++
Go
Java
Lua
PHP
Python
Shared Mem
Agent
Dump File
Client
Monitor
ZooKeeper
User
HTTP
Web
Admin Sys
Feedback Serv

SDK

- **为用户提供读取配置接口**
  - 基本接口只有一个：get(key, idc=null)
  - key代表节点名(path)，idc为空时从本机房ZK集群取值，否则取指定机房
  - 从共享内存里检索key，找到即返回
  - 否则将key写入消息队列，等待agent将值写入共享内存后再读出值返回
  - 目前支持7种语言：C/C++、Java、PHP、Python、Lua、Go、Shell

agent

- 负责数据获取与更新
  - 与ZK保持长连接
  - 注册watcher
  - 监视消息队列有无新内容
  - 问题1：刚上线时共享内存空白，大量key被扔到消息队列里，而agent从ZK取值相对慢，队列积压最终被写满
  - 解决：agent创建独立线程，负责将key从队列里读到本进程内存里
  - 获取或更新数据后写入共享内存，并汇报给反馈服务器
  - 问题2：watcher是一次性的，有极小概率丢失事件
  - 解决：周期性扫描共享内存，并与ZK上的数据比对
  - 问题3：agent崩溃或断网，此时间段内的更新事件被遗漏
  - 解决：agent启动或网络恢复后，立即扫描共享内存并与ZK比对
  - 问题4：断网的同时机器宕机，重启后共享内存空白且无法连上ZK
  - 解决：周期性把所有数据持久化到磁盘上

共享内存

- 在本机上缓存配置信息
  - 由agent在启动时创建
  - 作为agent向SDK传递信息的通道
  - 避免每次通过网络读取ZK
  - 数据以哈希表形式存储
  - 问题5：agent和SDK都要操作共享内存，需加锁，并发度低
  - 解决：去锁，节点值和MD5同时写入共享内存，SDK读出值和MD5后校验，失败则重读

消息队列

- **请求agent获取指定节点的值**
  - 由agent在启动时创建
  - 作为SDK向agent传递信息的通道
  - SDK在共享内存中未检索到节点，将节点名写入消息队列
  - agent从消息队列中得到节点名，去ZK读取节点值

monitor

- **监控服务存活**
  - 对服务类配置进行周期性的存活检测
  - 默认检测方式为连接IP:port并重试若干次
  - 也可自定义检测脚本
  - 检测结果写入ZK，SDK只会获取到可用的服务

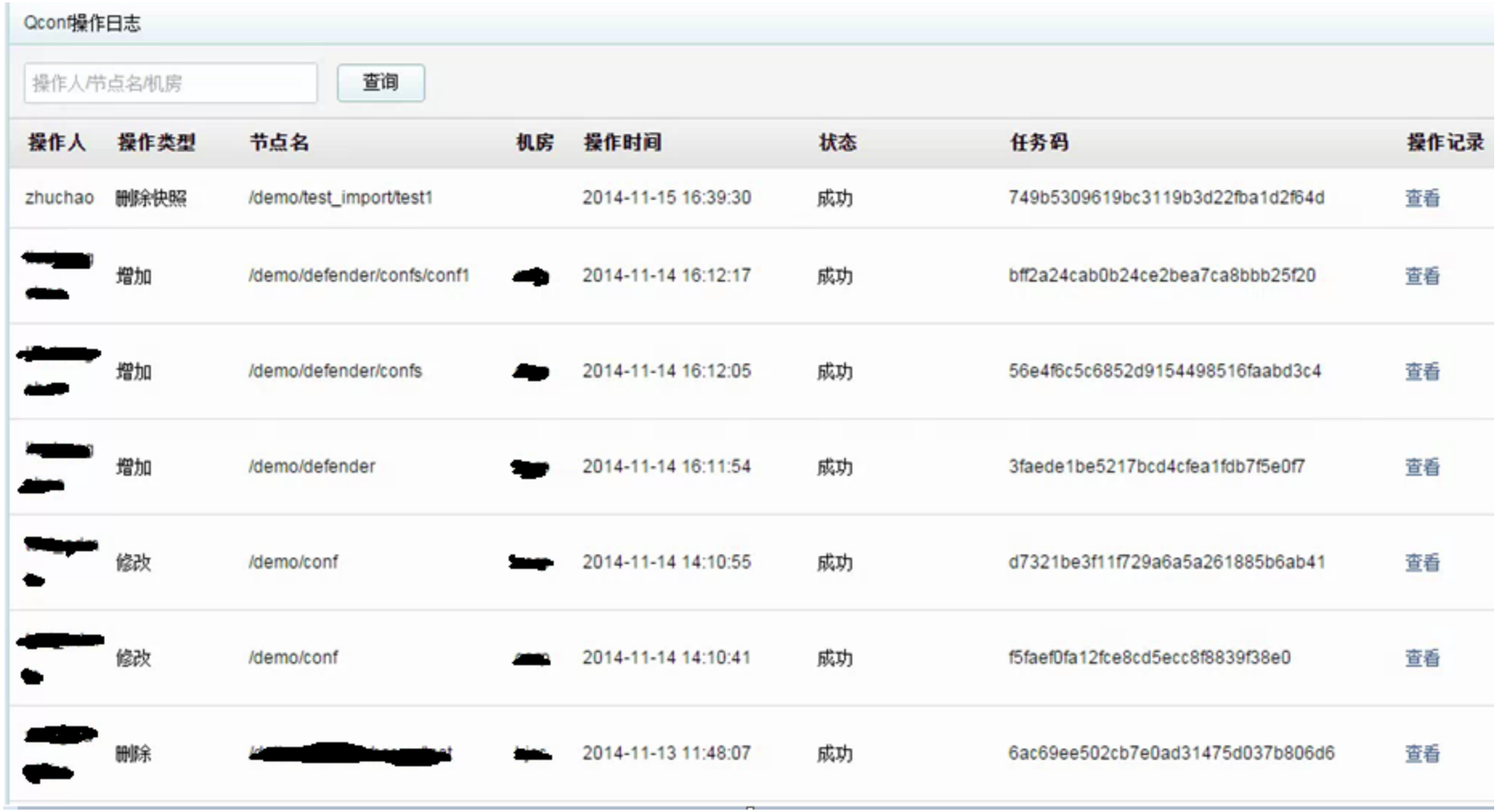
反馈服务器

- **接收反馈信息**
  - agent每次从ZK读取数据后，都将该信息汇报
  - 记录汇报的信息，供用户查看
  - 哪些机器读取了哪些配置
  - 各台机器上的各项配置是否已与ZK同步到最新

Web界面

- **提供方便友好的操作界面**
  - 节点树状结构的图形化展示
  - 节点值的可视化显示与创建、删除、修改等操作
  - 日志显示、快照(回滚)管理等

Web界面



Qcon操作日志

操作人/节点名/机房 查询

操作人	操作类型	节点名	机房	操作时间	状态	任务码	操作记录
zhuchao	删除快照	/demo/test_import/test1		2014-11-15 16:39:30	成功	749b5309619bc3119b3d22fba1d2f64d	查看
[redacted]	增加	/demo/defender/confs/conf1	[redacted]	2014-11-14 16:12:17	成功	bff2a24cab0b24ce2bea7ca8bbb25f20	查看
[redacted]	增加	/demo/defender/confs	[redacted]	2014-11-14 16:12:05	成功	56e4f6c5c6852d9154498516faabd3c4	查看
[redacted]	增加	/demo/defender	[redacted]	2014-11-14 16:11:54	成功	3faede1be5217bcd4cfea1fdb7f5e0f7	查看
[redacted]	修改	/demo/conf	[redacted]	2014-11-14 14:10:55	成功	d7321be3f11f729a6a5a261885b6ab41	查看
[redacted]	修改	/demo/conf	[redacted]	2014-11-14 14:10:41	成功	f5faef0fa12fce8cd5ecc8f8839f38e0	查看
[redacted]	删除	[redacted]	[redacted]	2014-11-13 11:48:07	成功	6ac69ee502cb7e0ad31475d037b806d6	查看

管理服务器

- **总括所有管理类操作**
  - 对ZK上的数据执行增删改查等操作
  - 用户权限控制
  - 记录每个用户的操作日志
  - 实现导入、回滚等附加功能
  - 用户可通过Web界面操作，也可直接调用管理服务器的HTTP接口

管理功能：从文件导入

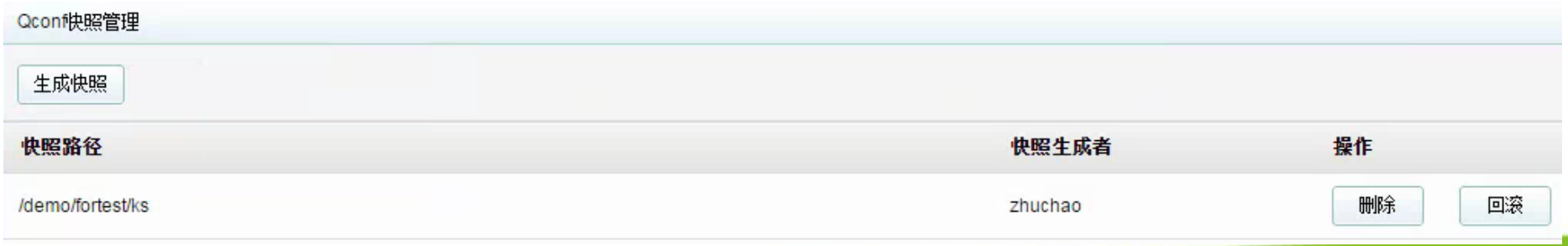
- **背景**
  - 现有业务使用配置文件，转换到QConf需要手工建立节点并填写配置值
  - 工作量较大，易出错
- **目标**
  - 自动解析已存在的配置文件，并导入到QConf系统中
  - 省去手动操作，确保配置正确无误

管理功能：查看客户机

- **背景**
  - 用户希望知道哪台客户机读取了哪些配置项
  - 以及每台客户机是否已经更新到最新的配置值
- **原理**
  - 每台客户机在首次读取或更新某配置项后，将信息反馈给反馈服务器
  - 用户通过管理服务器查询即可

管理功能：回滚

- **背景**
  - 新配置可能未经长时间生产环境检验，上线后有可能出问题
- **目标**
  - 新配置出错时能快速恢复为旧的稳定配置
- **原理**
  - 用户基于旧配置生成一份快照
  - 修改成新配置值
  - 新配置出错，利用快照数据将变更的配置项修改为旧值



管理功能：机房间复制

- **复制当前节点**
  - 把某个节点复制到其他机房
- **复制子节点树**
  - 把某个节点及其所有下层节点复制到其他机房
- **批量修改节点值**
  - 把多个机房的同名节点批量修改为某值

未来发展

- **权限控制细化**
  - 为每个人对每个节点的操作赋予权限，如读写、只读、不可见等
- **自定义脚本**
  - 事件发生后，调用自定义脚本，来执行某些操作
- **类事务操作**
  - 多个配置项的修改以事务方式进行
- **灰度发布**
- **支持HTTP方式读取**
- **客户端可写入?**
- **多数据结构?**
- **开源(预定15年Q1)**

我们的团队-基础架构组

- MySQL中间件Atlas
  - https://github.com/Qihoo360/Atlas
  - 业界43家公司生产环境采用
  - 开源QQ群326544838
- 分布式消息队列QBus
  - 基于Kafka二次开发
- NoSQL数据库Bada
  - 基于LevelDB存储引擎的分布式key-value型数据库
- Subrange模块for Nginx
  - 将大数据量的HTTP请求切分为多个子请求
  - https://github.com/Qihoo360/ngx_http_subrange_module
- Mario库
  - 将同步写自动转化为异步写
  - https://github.com/Qihoo360/Mario
- Erlang日志库
  - https://github.com/Qihoo360/elog

基础项目发展体会

- 业务成本
  - 部署成本
    - 角色多少
    - 依赖项是否简单
    - 能否自动化
  - 升级成本
    - 是否需要业务工程师配合
    - 对业务有无影响，影响多大
  - 学习成本
    - 使用是否简便
  - 风险成本
    - 出问题怎么办
    - 影响面多大
    - 是否有预案
    - 是否可控

基础项目发展体会

- **基础服务定位**
  - **基础服务的特点**
    - 没有最终用户，用户是业务部门的开发工程师
    - 三无产品：无专职产品经理，无专职测试，无专职运营
    - 稳定性极其重要，服务长期运行，支撑多个业务，升级成本较高
  - **类1（必须）：实现功能或解决问题**
    - 例：需要存储数据，必须配备MySQL等数据库
    - 例：需要生成全局唯一序号，必须配备idgen等序号生成器
  - **类2（改进）：提高开发效率或降低运维成本**
    - 例：使用配置文件，繁琐易错，改用QConf，简便直观
    - 例：DBA切换DB需要协调业务，使用Atlas后，业务不感知
  - **比较优势**
    - 与现存成熟方案的比较，是否有优势？
  - **基础服务成功的条件**
    - 若属类1且不存在竞品，则成功基本无压力
    - 若属类2且不存在竞品，则应满足收益 > 四项成本之和
    - 若存在竞品，则应满足收益 > 业务成本之和 且 比较优势 > 0

联系我们

- **邮件**
  - 个人： zhuchao@360.cn
  - 项目组：g-qconf@360.cn
  - 团队：g-infra@360.cn
- **Github**
  - https://github.com/qihoo360/

360 www.360.cn

微信公众号

中华数据库行业协会

敬请期待：
2015中华数据库大会
时间：2015.05.16
报名时间：2015.02.14
报名网址：meeting.zhdba.com

联系我们：
联系人：朱小姐
联系电话：136 5197 9898
联系QQ：378091820

360
www.360.cn