腾讯微信技术总监周颢：一亿用户增长背后的架构秘密

2012-05-15 07:56 来源：CSDN 专稿 作者：付江

[CSDN.NET 专稿 付江/文] 微信——腾讯战略级产品，创造移动互联网增速记录，10 个月 5000 万手机用户，433 天之内完成用户数从零到一亿的增长过程，千万级用户同时在线，摇一 摇每天次数过亿...在技术架构上，微信是如何做到的？日前，在腾讯大讲堂在中山大学校园宣讲活动上，腾讯广研助理总经理、微信技术总监周颢在两小时的演 讲中揭开了微信背后的秘密。

周颢，2001 年毕业于华南理工大学，计算机专业硕士。2005 年加入腾讯广州研发部，历任 QQ 邮箱架构师，广研技术总监，T4 技术专家，微信中心助理总经理。

（腾讯广研助理总经理、微信技术总监 周颢 CSDN 配图）

周颢把微信的成功归结于腾讯式的**“三位一体”策略：即产品精准、项目敏捷、技术支撑。**微信的成功是在三个方面的结合比较好，能够超出绝大多数同行或对手，使得微信走到比较前的位置。所谓产品精准，通俗的讲就是在恰当的时机做了恰当的事，推出了重量级功能，在合适的时间以最符合大家需求的方式推出去。他认为在整个微信的成功中，产品精准占了很大一部分权重。

敏捷是一种态度 敏捷就是试错

微信研发团队里鼓励一种**试错的信仰：他们坚信，在互联网开发里，如果能够有一个团队在更短的时间内尝试了更多机会(并能改进过来)，就能有(更多的)机会胜出。**敏捷是一种态度，在软件开发过程中，项目管理者都会非常忌讳“变更”这个词，但是在微信的项目运作中是不可以的。**因为微信必须要容忍说哪怕在发布前的十分钟，也要允许他变更。这是非常大的挑战，因为打破了所有传统项目开发的常识。所有人都说不可能做到的，但微信做到了。研发团队所做的一切都是要给产品决策者有最大的自由度，而这个决策正是微信能够胜出的关键。**

海量系统上的敏捷 无异于悬崖边的跳舞

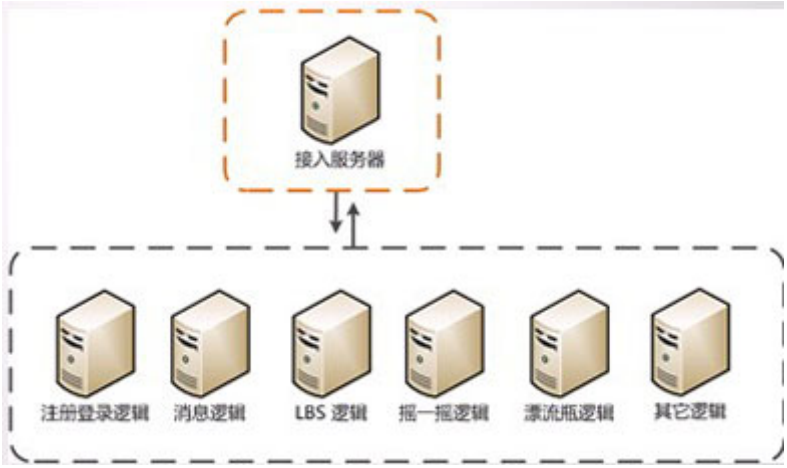
敏捷有很多困境，如果做一个单机版程序，是可以做到很敏捷的，但是**腾讯正在运作的是一个海量系统，有千万级用户同时在线，在一个单 独的功能上每天有百亿级的访问，同时还要保证 99.95%的可用性。在海量系统上应对项目开发会有很严谨的规范，都说要尽可能少的变化，因为 90%-95%的错误都是在变更中产生的，如果系统一直不变更会获得非常高的稳定度，但是微信就是要在悬崖边跳舞。**微信的研发团队要做一些事情，让敏捷开发变得更简单。

如何做到这一切？周颢认为，首先，必须建立起一种**狂热的技术信念，就是一定是可以做到的。**然后，需要用一些稳固的技术(理念)来支撑，例如**大系统小做、让一切可扩展、必须有基础组件、轻松上线**(灰度、灰度、再灰度；精细监控；迅速响应)...等等来支撑。

四大法器：大系统小做、让一切可扩展、要有基础组件、轻松上线

- 大系统小做：当设计庞大系统的时候，应该尽量分割成更小的颗粒，使得项目之间的影响是最小的。
- 一切可扩展：在高稳定度、高性能的系统中间，为了稳定性能把它设计成不变化的系统，但为了支持敏捷需要让一切的东西都要变得可以扩展。
- 必须建立基础组件：要解决复杂问题的时候，需要将已有的经验固化下来，固化下来的东西会成为系统中的一部分。
- 轻松上线：当做了变化并把它从开发环境中部署到现有的运营环境中去，在这个过程中，“灰度”这个词非常关键，就是在黑和白之间的选择，必须要变成一种小规模尝试，再逐步扩展到海量过程中的一个问题。

大系统小做——仅仅把模块变得更为清晰，这在海量系统设计开发中是不够的，还需要在物理环境上进行分离部署，出现问题的时候可以快速发现，并且在最快的情况下解决掉。



大系统小做 混搭模式

将不同的应用逻辑物理分割独立出来，用户注册登录、LBS 逻辑、摇一摇逻辑、漂流瓶逻辑、消息逻辑独立开来。把关键的逻辑混搭在一起，当所有的逻辑部署在同一个服务器上，确实也会带来很大敏捷上的好处，因为不需要额外的考虑部署和监控的问题。**在整个微信的逻辑中，可能现在已经有上百种不同的逻辑，因为会在逻辑的分割上拆分成 8-10 种做分离部署。**

一切可扩展——网络协议可扩展、数据存储可扩展

扩展的关键点有两块。一个是网络协议需要扩展，当要升级一个新功能的时候，会有一些比较大的困难，所以所有协议设计都比较向前兼容，但是向前兼容还 是不够的，因为网络协议设计本身有非常多的功能也会有比较大的字段，相关的代码可能会有数千行，这一块不能通过手写方式完成。可以通过 XML 描述，再通过 工具自动生成所有的代码，这是微信获得快速开发的一个重要的点。

另外一块就是在数据存储方面是必须可扩展的。在 2005 年绝大多数海量系统的设计都是采用固定字段的存储，但是在现代系统中会意识到这个问题，会采用 KV 或者 TLV 的方式，微信也做了不同的设计。

把复杂逻辑都固化下来，成为基础软件。在微信后台会有几种不同的基础组件。大致包括：

- Svrkit——Client/Server 自动代码生成框架:10 分钟搭建内部服务器
- LogicServer——逻辑容器：随时添加新逻辑
- OssAgent——监控/统计框架：所见即所得的监控报表
- 存储组件——屏蔽容灾/扩容等复杂问题

灰度、灰度、再灰度

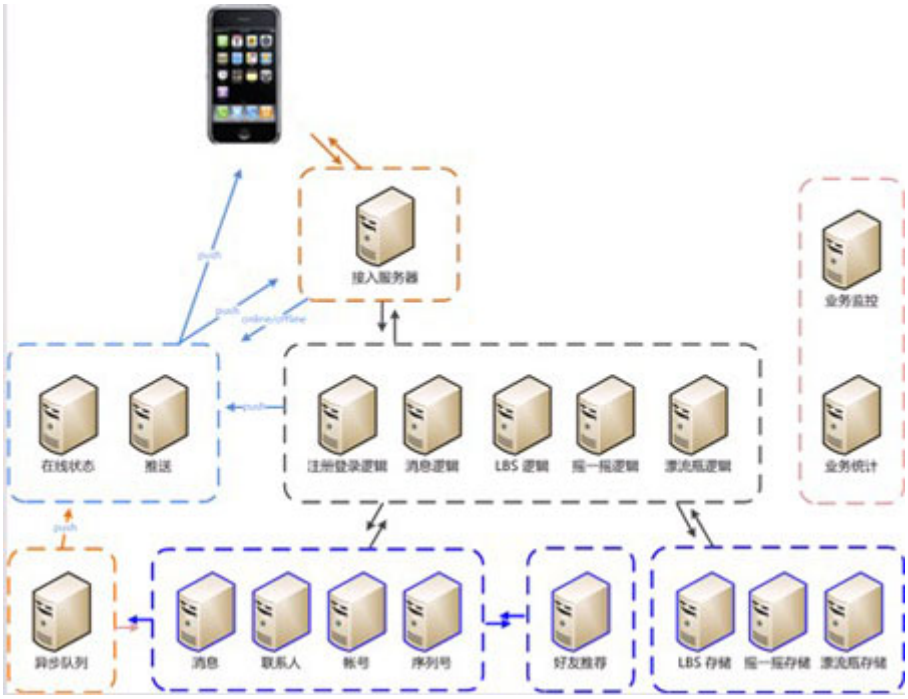
在变更后的部署方式上，微信在一些规则会限定不能一次把所有的逻辑变更上去，每一次变更一小点观察到每一个环节没有问题的时候，才能布局到全网上去。**微信后台每一天可以支撑超过 20 个后台变更，在业界来说，通常做到 5 个已经是比较快了，但是微信可以做到快 4 倍。**

腾讯内部的上线系统

而所谓**灰度发布**，是指在黑与白之间，能够平滑过渡的一种发布方式。AB test 就是一种灰度发布方式，让一部用户继续用 A，一部分用户开始用 B，如果用户对 B 没有什么反对意见，那么逐步扩大范围，把所有用户都迁移到 B 上面 来。灰度发布可以保证整体系统的稳定，在初始灰度的时候就可以发现、调整问题，以保证其影响度。(在腾讯，灰度发布是最常采用的发布方式之一)

孙子兵法：古之所谓善战者，胜于易胜者也

常识上，解决一个复杂问题的时候，会用高明的技巧解决复杂的问题，这个不是微信团队的目标，他们追求的要做到让所有问题很自然和简单的方式解决掉。**在周颢看来，微信架构的技术复杂点在四个要点：协议、容灾、轻重、监控。**



微信架构

- 协议。手机终端跟后台服务器之间的交互协议，这个协议的设计是整个系统的骨架，在这一点做好设计可以使得系统的复杂度大大降低。
- 容灾。当系统出现了若干服务器或若干支架(宕机的时候)，仍然需要让系统尽可能的提供正常的服务。
- 轻重。如何在系统架构中分布功能，在哪一个点实现哪一个功能，代表系统中间的功能配置。
- 监控。为系统提供一个智能仪表盘。

在协议设计上，移动互联网和常规互联网有很大的区别。首先有 CMWAP 和 CMNET 的不同，在中国现在有相当多的手机用户使用 WMWAP 连接，还有 就是在线和离线的概念，当 QQ 下线的时候叫离线，当你登录的时候叫在线。但是在移动互联网这两个概念比较模糊。从微信的设计中，不管在线还是离线系统表现 都应该是一致的。还有一个是连接不稳定的问题，由于手机信号强弱的变化，当时信号很好，5

秒钟走到信号不好的地区，连接就必须断掉。这个中间带来不稳定的 因素为协议设计带来较大困难。此外就是资费敏感的问题，因为移动互联网是按照流量计费的，这个计费会使得在协议设计中如何最小化传输的问题。最后就是高延 迟的问题。

对此，业界标准的解决方案：Messaging And Presence Protocol：1)XMPP;2)SIP/SIMPLE。它的优点是简单，大量开源实现。而缺点同样明显：1)流量大：状态初始化；2)消息不可靠。

微信在系统中做了特殊设计，叫 SYNC 协议，是参考 Activesyec 来实现的。特点首先是基于状态同步的协 议，假定说收发消息本身是状态同步的过程，假定终端和服务器状态已经被迟了，在服务器端收到最新的消息，当客户端、终端向服务器对接的时候，收取消息的过 程实际上可以简单的归纳为状态同步的过程，收消息以及收取你好友状态更新都是相同的。在这样的模式之下，我们会也许会把交互的模式统一化，只需要推送一个 消息到达的通知就可以了，终端收到这个通知就来做消息的同步。在这样的简化模式之下，安卓和塞班都可以得到统一。这样的系统本身的实现是更为复杂的，但是 获得很多额外的好处。

让剩下系统实现的部分更加简单，简化了交互模式，状态同步可以通过状态同步的差值获得最小的数据变更，通过增量的传输得到最小的数据传输量。通过这 样的协议设计，微信可以确保消息是稳定到达的，而且是按序到达。引用一句俗话：比它炫的没它简单，比它简单的没它快，没谁比他更快，哪怕在 GPRS 下，微 信也能把进度条轻易推到底。

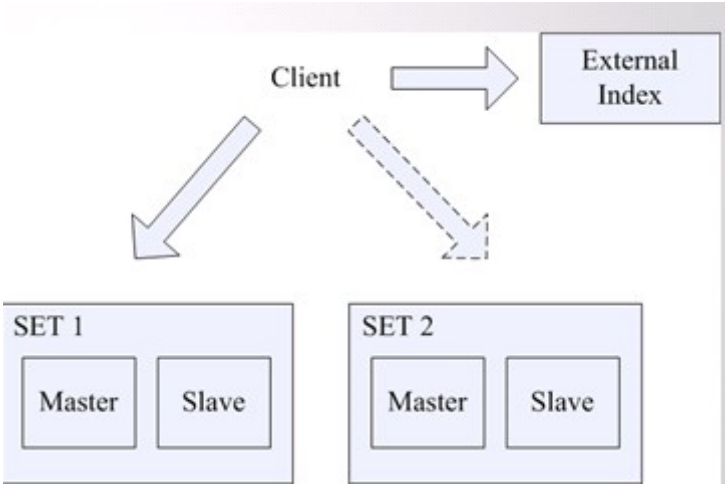
追求完美设计的团队不能胜任海量服务

在容灾之前面向最坏的思考，如果系统真的挂了，需要做一些事情，首先是防止雪崩，避免蝴蝶效应。**如果关注春节订火车票就知道了，用 户的请求量会因为系统服务不了而不断的重试，意味着发生雪崩的时候，系统可能会承载原先 3-10 倍的流量，使得所有的事情更加恶化。所以微信有很多“放 雪”功能的设计。第二个词是柔性可用，在任何的系统中不要追求完美设计，追求完美设计的是团队是不能胜任海量服务的。如果在一个系统出现问题的时候，这个 系统就挂了，那么这是一个不好的设计，最好的做法是提供 0-1 中间的选择。**举一个例子，当一个用户向另外一个用户发消息的时候，可能会 通过一个垃圾信息过滤的检测，如果垃圾信息过滤这个模块突然挂掉了，这个消息难道就不能达到了吗？在这样的情况下，要忽略掉这个错误，使得消息正常达到对 方。要精确定位出哪一个环节是最为重要的，把不是重要的错误尽可能的忽略掉。当不能做到完美的时候，尽可能为用户提供服务。另外一个重要方面叫做“保护点 前置”，最前的一个点就是终端，在手机终端上蕴埋更多的保护点，这样会为用户系统赢得更大的处理空间。如果终端具备这样的能力，会获得更大的反应空间。

周颢介绍了在微信上具体容灾设计的做法。在所有的容灾中存储层的容灾是最难的，一个系统的设计分为三层：接入层、逻辑层、存储层。接入层和逻辑层的 容灾都有比较成熟的方案。逻辑层的容灾相对来说比较简单，尽量不要有状态的设计，比如说当你做上一个请求的时候，会保持一些状态，要使得下一个请求发到下 一个服务器。如果任何一个请求之间互相不关联的话，这个就是无状态的设计，只要做到这一点逻辑层的容灾可以随意的切换。在回到存储层本身的容灾设计上，

相 对来说困难一些，但是**微信研发团队采用了一些技巧，叫分而治之，分离业务场景，寻求简单的设计，并不会寻求大而同一的解决方案，因为这样会使得系统的复杂度大幅度上升，而微信会尽可能把产品拆细，寻求简化的设计。**

首先是主备容灾，这是最常见的方案。在有一些业务场景中是可以容忍最终一致性的，比如账号系统的设计，每天写入账号系统的请求是非常少的，但是访问 的请求非常多，这个差异可能会达到数万倍的规模，在这样的场景下，微信会在账号系统中采用简化的方案，也可以获得比较大的稳定度。



SET 模型＋双写

第二种容灾的模式叫双写，两台 Master 的机器，当一台机故障的时候，另外一台机还是可以接收到写请求，当两台机交错启动的时候，会得到数据的丢 失。但是有一些场景是可以容忍轻度数据丢失的，比如说会有一个存储专门记录用户终端的类型，比如说安卓还是塞班以及他们使用终端的微信版本是什么，这样的 数据是可以容忍轻度数据丢失的，因为偶尔有一些丢失的话，下一次访问会把这些数据带上来，会尽快的修复所有的数据。双写也是非常简单的模式。

微信的研发团队做了一个叫 Simple Quorum 的机制，在微信的后台中，同步协议有一个很重要的基石叫序列发生器，这样的一个序列发生器需要有极高的稳定度。首先可以看到序列号有一个特点 永远是递增的，用递增方式往前推进的时候，最大的序列号就是最新的系列号。有一个毕业才加入广研的毕业生想到一个绝佳的方案，按 SET 分布，从 2G 减到 200K。

前轻后重 功能点后移

周颢还谈到了轻重的概念。这个概念的提出主要是从终端本身的一些困境所带来的。首先在终端上需要表现最多的一个产品的逻辑，逻辑非常复杂，变更的成 本也非常高，当需要修复的时候必须发布一个新版本，这个新版必须由自己下载才能完成，下载的成本非常高。在这样的前提下，如果手机终端产生了任何变化的时 候，如果这个变化有非常大的问题就会有极大的困境，所以需要在每一个发布之前做一些充分的数据，确保不会发生致命问题。如果一旦出现致命问题难以修复，需 要把关键的点从终端移到后台实现，把功能点后移，来充分发挥后台快速变更的能力。

接入优化：从 GSLB 到 IP 重定向

在接入层的优化，速度很重要的因素，是不是能够就近接入一个最优的节点，比如说移动用户最好接入移动的节点，海外的用户可能需要寻找更佳的路由，有 的时候可能无法自动做到这一点，一点是在终端上做测速，微信会通过在后台 IP 逆向的能力，通过后台指挥微信终端联网的能力，寻找最优的接入点。上图就是每 分钟收到同一项指令曲线的报表。

如何解决“偷流量”的问题——当国内类微信类产品发布的时候出现一个大的问题就是“偷流量”，当用户在某一些逻 辑下进行一个死循环，不断访问某一些数据，这样的死循环是非常可怕的，如果在用户不知觉的情况之下，可能会在一个小时之内偷到数 10 兆甚至数百兆的流量。 有非常多业内的同行都需要花大量的精力解决这个问题，微信研发团队用了非常强大的方式解决它。通过在后台建立起严厉的监控系统，对每一个用户的行为做一个 监控，当发现异常的时候，后台会给终端发出指令，使得微信终端在一段时间无法联网，但是可以保证用户流量不会白白的使用掉。

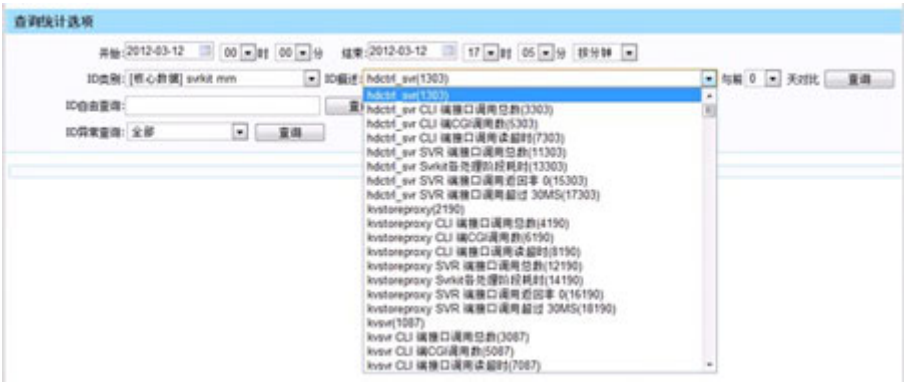
功能适配的例子——第一期微信版本发布的时候，当时没有群聊的功能，第二版发布的时候做了这个功能。当时有两个 选择，对于早期版本的用户，因为不支持群聊，就无法享用到这个功能，但是微信希望提供更好的选择，想让早期不支持群聊的版本，也可以被拉到一个群里面收消 息、发消息，通过后台功能的适配也能做到这个事情。

分而治之 把监控嵌入基础框架

对于一个海量系统来说，一个精密的仪表盘非常重要。监控是非常痛苦的，对于这样一个系统来说，每小时会产生数百 G 的监控日志。微信希望在 1 分钟之内监控的数据就能够显示在报表上，因为只有这样的精准和实时度才能够赢得处理故障的时间。**微信会做关联统计，通过摇一摇加了好友，他们活跃度如何，过了一段时间他们的活跃度变化情况又是如何。这种需求是需要通过大量日志的关联统计来获得的。研发团队也花了一段时间来理解这个问题，发现了中间一个重要的经验叫做“鱼和熊掌不能兼得”。**

为了让监控数值更敏感，需要把监控细化再细化，上面数据表示每一栏子系统的数据，下面这个是按微信版本号来划分的，这里的数据项是非常多。

微信还需要采集一些异常的点，如果有异常的话会发布紧急的版本，尽可能快的替换它。对收发消息延时做的监控，比如说 0—1 秒端到端的速度，会对不同 的区段做一些统计，当某一个环节出现异常的时候，通常会在中间的延时上体现出来。有一个很重要的点叫自动报警，现在有数千项的数据，不可能每一项都靠人工 去看的，必须要跟自动报警相关联，微信有一些智能的算法，是不是在正常的范围内，跟历史的数值进行对比，如果有异常的话，会通过短信、邮件还有微信本身来 发出报警信息。



把监控嵌入基础框架

微信会把监控嵌入到基础框架里面去，因为并不是每一个人都会意识到在需要的地方嵌入一个监控点，所以在基础框架本身内置很重要的监控点，比如说这个 表上的栏目，非常多的栏目大概会有数百项的栏目，都不需要程序员自己去写，当用基础组件搭建一个系统的时候，就可以直接观测系统数据。

在谈到微信未来的技术挑战时，周颢首先希望能够让微信成为可用性 99.99%的系统；设计出面向现在 10 倍容量的系统以及完全的 IDC 容灾。

网上盛传的凌晨两点，腾讯大厦那多层大片大片的灯光和楼下那长长的出租车队伍说明了一切。引用一句话做结尾，可怕的不是微信，真正可怕的是，比你领先比你更有天赋的团队比你更努力。

特别鸣谢：腾讯大讲堂（djt.qq.com）对本篇报道的内容支持

附录：腾讯微信技术总监周颢演讲 PPT

微信之道－至简

广州研发部　harveyzhou

腾讯大讲堂 http://djt.qq.com

关于我

- 周颢（harveyzhou）
- 2001年毕业于华南理工大学，计算机专业硕士
- 2005年加入腾讯广州研发部
- 历任QQ邮箱架构师，广研技术总监，T4技术专家，微信中心助理总经理

关于微信

- 移动互联网的探索者
- 10个月5000万手机用户
- 创造移动互联网用户增速的记录
- 千万级在线
- 苹果中国区AppStore月下载量第一
- 摇一摇每天次数过亿
- 腾讯战略级产品

腾讯大讲堂 http://djt.qq.com

微信的历程

腾讯大讲堂 http://djt.qq.com

微信的三位一体

- 产品的精准
- 项目的敏捷
- 技术的支撑

产品的精准—用简单规则构造复杂世界

- 张小龙，腾讯副总裁，广研灵魂人物
- 从第二代程序员旗手，到领军者，到产品传奇人物
- 从**Foxmail**，到**QQ**邮箱，到微信

微信的三位一体

- 产品的精准
- 项目的敏捷
- 技术的支撑

什么是敏捷

- 项目管理的技巧？**Scrum**？
- 矿工？

项目的敏捷

- 敏捷就是试错法
- 敏捷是一种态度
  - 产品决策是成功的第一因素
  - 允许发布前十分钟的变更
  - 给予产品决策以最大自由度

敏捷的困境

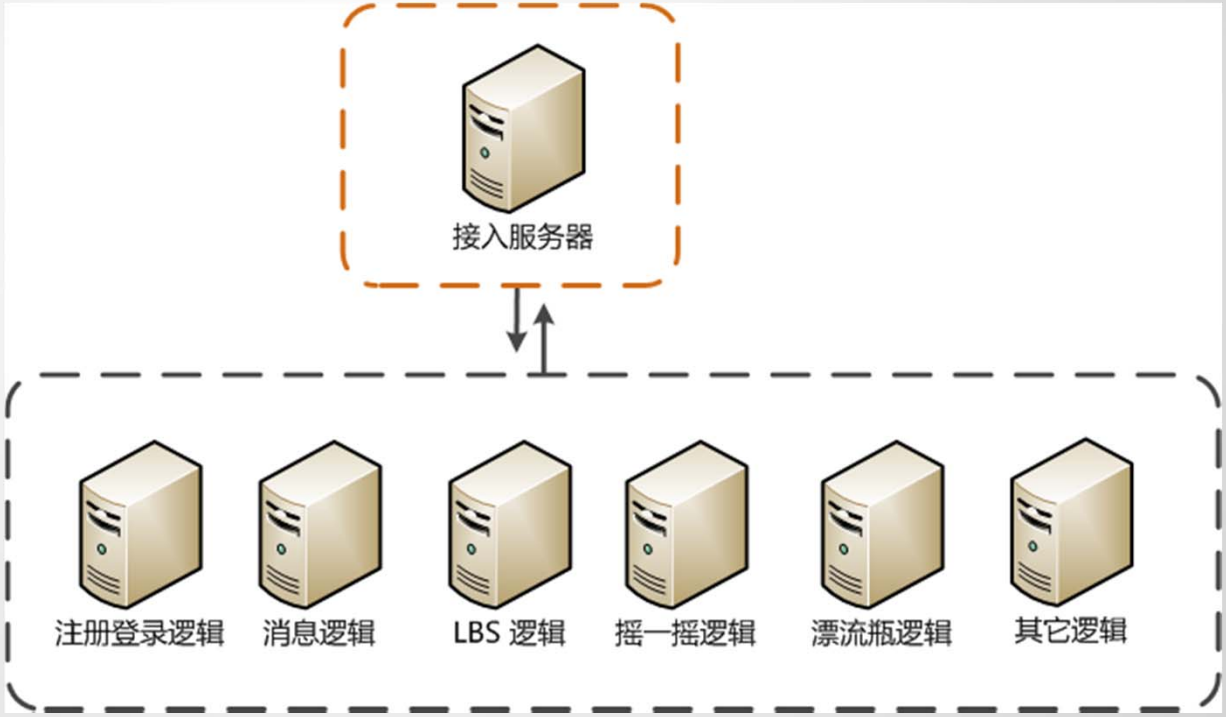
- 海量系统的复杂度
  - 千万级同时在线
  - 亿级摇一摇
  - 单集群百亿级服务请求
  - **99.95%**的可用性
- 海量系统上的敏捷，无异于悬崖边的跳舞

让敏捷变得简单

- 狂热的信念，**You can do it !**
- 稳固的技术支撑
  - 大系统小做
  - 让一切可扩展
  - 要有基础组件
  - 轻松的上线
    - 灰度，灰度，再灰度
    - 精细的监控
    - 迅速的响应

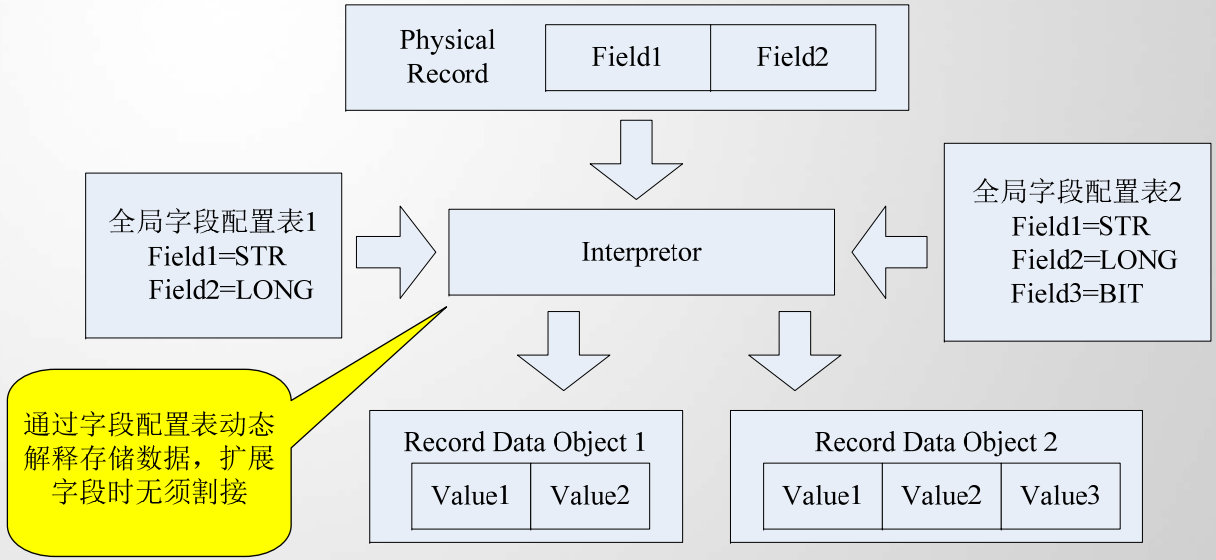
大系统小做

- 从代码分模块，到分离部署
- 灵活的折衷：混搭模式（重要/复杂逻辑分离，其它混合部署）



让一切可扩展

- 网络协议可扩展
  - **XML描述**
  - 向前兼容
  - 代码自动生成（**ProtocolBuffer & TLV**）
- 数据存储可扩展
  - **KV or TLV**
  - 字段配置表
  - 类SQL处理



要有基础组件

- **Svrkit：Client/Server自动代码生成框架**
  - **10分钟搭建内部服务器**
- **LogicServer：逻辑容器**
  - 随时添加新逻辑
- **OssAgent：监控/统计框架**
  - 所见即所得的监控报表
- 存储组件
  - 屏蔽容灾/扩容等复杂问题

灰度，灰度，再灰度


模块：mmsynclogicsvr
若找不到模块或服务器，请先找运维负责人，确认服务器状态是运营中
搜索：
已选5台服务器
还可选●台服务器
首页 上页 1 2 3 4 5 下页 末页
mmsynclogicsvr1
mmsynclogicsvr2
mmsynclogicsvr3
mmsynclogicsvr4
mmsynclogicsvr5
mmsynclogicsvr6
mmsynclogicsvr7
mmsynclogicsvr8
mmsynclogicsvr9
mmsynclogicsvr10
mmsynclogicsvr11
mmsynclogicsvr12
mmsynclogicsvr13
mmsynclogicsvr14
mmsynclogicsvr15
mmsynclogicsvr16
mmsynclogicsvr17
mmsynclogicsvr18
mmsynclogicsvr19
mmsynclogicsvr20
mmsynclogicsvr21
mmsynclogicsvr22
mmsynclogicsvr23
mmsynclogicsvr24
mmsynclogicsvr25
>>
<<
mmsynclogicsvr1
mmsynclogicsvr2
mmsynclogicsvr3
mmsynclogicsvr4
mmsynclogicsvr5
总计：●台，已灰度：21台
后台单至少灰度两次，且50%以上主机灰度，才能全面上线
全选所有
一次选够
灰度上线
返回
清空所有

You can do it !

>20个后台变更/天

每页显示 20 条　　首页 上页 1 2 3 4

单号	模块名	建单人	测试人	任务名	结束时间	状态
qqmail-120326-1654-0920	mmsynclogicsvr	bierhuang		mmsynclogics	2012-03-26 17:15:54	灰度完成
qqmail-120326-1635-4059	mmhdiconsvr	eddyzeng		mmhdiconsvr	2012-03-26 16:55:00	全上完成
qqmail-120326-1628-3189	ALL	paulohuang		mmlangconf	2012-03-26 16:39:18	全上中断[完成率:99.98%]
qqmail-120326-1627-3418	mmsynclogicsvr	bierhuang		mmsynclogics	2012-03-26 16:46:18	灰度回退完成
qqmail-120326-1605-5730	mmiconsvr	eddyzeng		mmiconsvr 修改	2012-03-26 16:45:42	全上完成
qqmail-120326-1544-2510	ALL	francowu		mmasyncmq7切走	2012-03-26 15:53:36	全上中断[完成率:99.98%]
qqmail-120326-1542-2150	mmopenappinfo	berryxie		mmopenappinf	2012-03-26 15:44:00	全上完成
qqmail-120326-1538-2239	ALL	junechen		mmadpush_cli	2012-03-26 15:54:59	全上中断[完成率:99.96%]
qqmail-120326-1536-2079	mmimglogicsvr	lynncui		mmimglogicsv	2012-03-26 15:38:21	全上完成
qqmail-120326-1532-2741	mmbottlelogicsvr	andrewang		更新 bottlelog	2012-03-26 16:53:04	全上完成
qqmail-120326-1528-2676	mmproxy	sharonzhang		mmproxy_cgi.	2012-03-26 15:34:43	灰度完成
qqmail-120326-1507-2070	mmbizindex	sharonzhang		mmbizindex上线	2012-03-26 15:10:33	全上完成
qqmail-120326-1505-4735	mmlogcenter	mariohuang		mmlogcenter重	2012-03-26 15:10:32	灰度中断[完成率:0.00%]
qqmail-120326-1412-1569	mmnewaddrbook	delphiliu		kvsvr 变更	2012-03-26 14:13:08	已建单
qqmail-120326-1411-3727	mmuserattr	stevenshe		kvstoreproxy	2012-03-26 14:30:27	全上完成
qqmail-120326-1410-0980	mmqqmsg	tommytang		配置QQ尾巴一条一条	2012-03-26 14:20:53	全上完成
qqmail-120326-1354-0514	mmadpush	junechen		mmadpush	2012-03-26 13:54:54	全上完成
qqmail-120326-1347-5984	mmlbslogicsvr	junechen		mmlbslogicsv	2012-03-26 17:13:40	全上完成

腾讯大讲堂 http://djt.qq.com

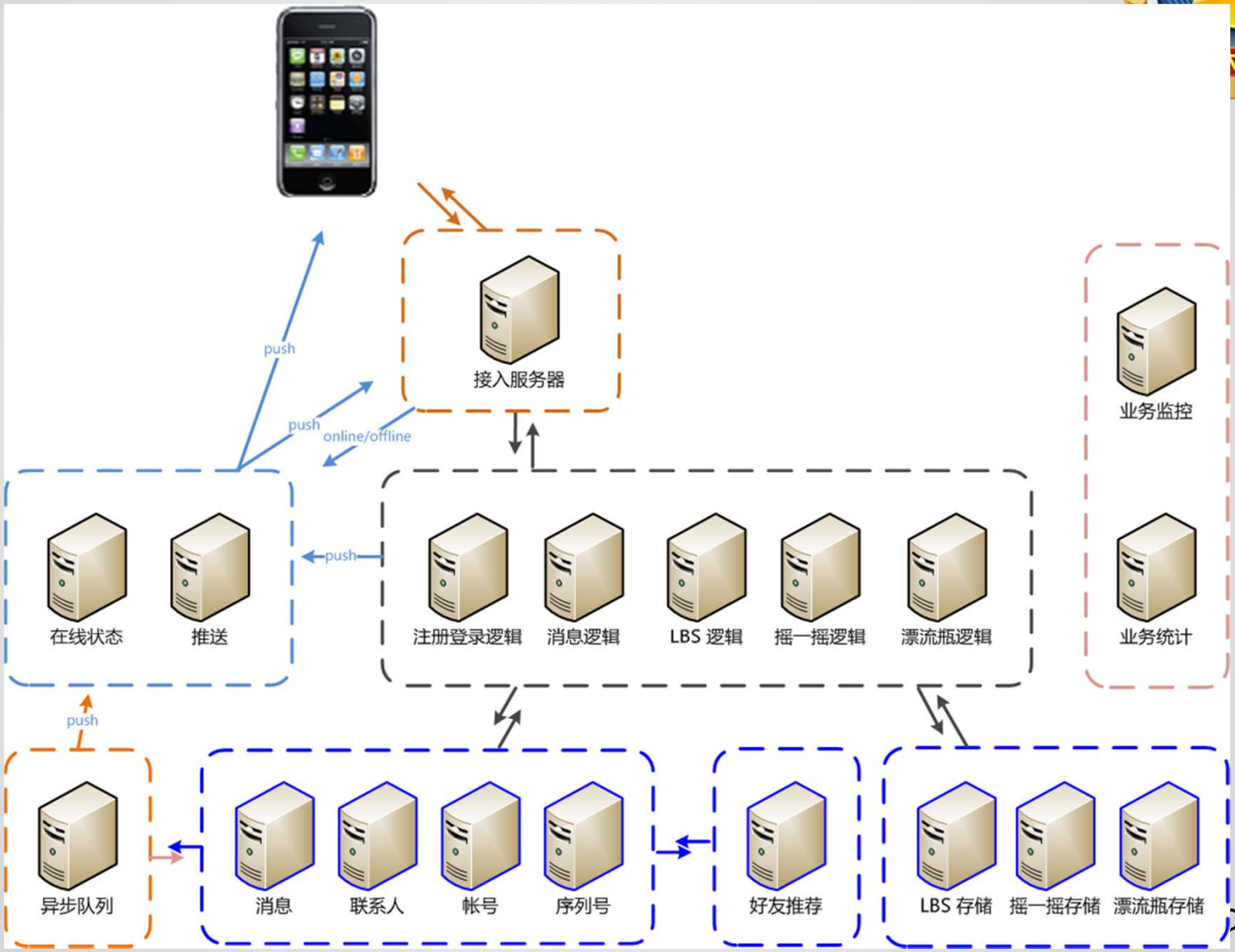
微信的三位一体

- 产品的精准
- 项目的敏捷
- 技术的支撑

技术的支撑—剥离复杂，让剩下的更简单

- 孙子兵法：古之所谓善战者，胜于易胜者也

微信架构


push
push
online/offline
接入服务器
业务监控
push
在线状态
推送
注册登录逻辑
消息逻辑
LBS 逻辑
摇一摇逻辑
漂流瓶逻辑
业务统计
push
异步队列
消息
联系人
帐号
序列号
好友推荐
LBS 存储
摇一摇存储
漂流瓶存储

关注复杂点

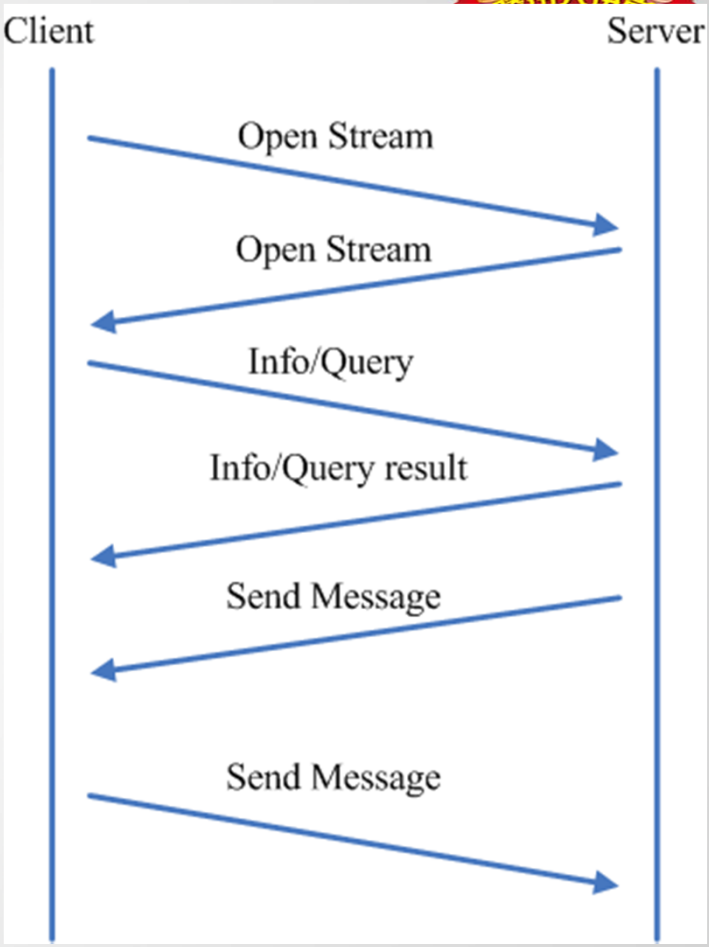
- 协议
- 容灾
- 轻重
- 监控

移动互联网的复杂性

- **CMWAP vs. CMNET**
- 在线 **vs.** 离线
- 连接不稳定
- 资费敏感
- 高延迟

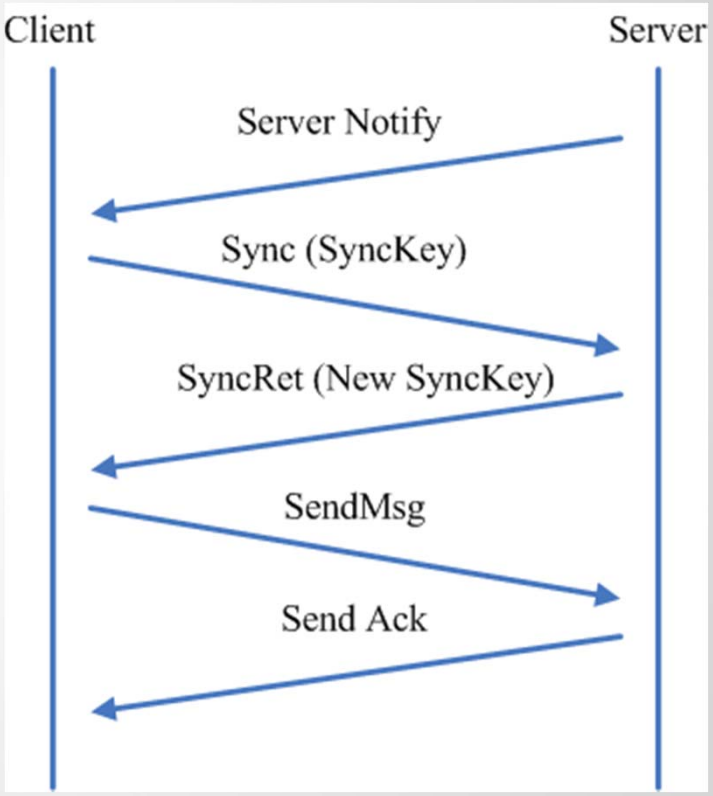
业界标准方案

- **Messaging And Presence Protocol**
  - **XMPP**
  - **SIP/SIMPLE**
- 优点：
  - 简单，大量开源实现
- 缺点
  - 流量大：状态初始化
  - 消息不可靠
- 把简单留给自己，把复杂留给别人？



SYNC协议

- 参考**ActiveSync**
- 状态同步：**Sync by SyncKey**
- 模式简化：**Notify & Client Pull**
- 实现更复杂，但 …



让剩下的更简单

- 简化交互模式
- 最小增量传输
- 最优重传控制
- **More important：** 消息可靠传输 **&** 按序到达

- *和菜头 @ 2011/11/26*
  比它炫的没它简单，
  比它简单的没它快，
  没有谁比它更快，
  哪怕在GPRS下，
  微信也能把进度条轻易推到底。

关注复杂点

- 协议
- 容灾
- 轻重
- 监控

在容灾之前－面向最坏的思考

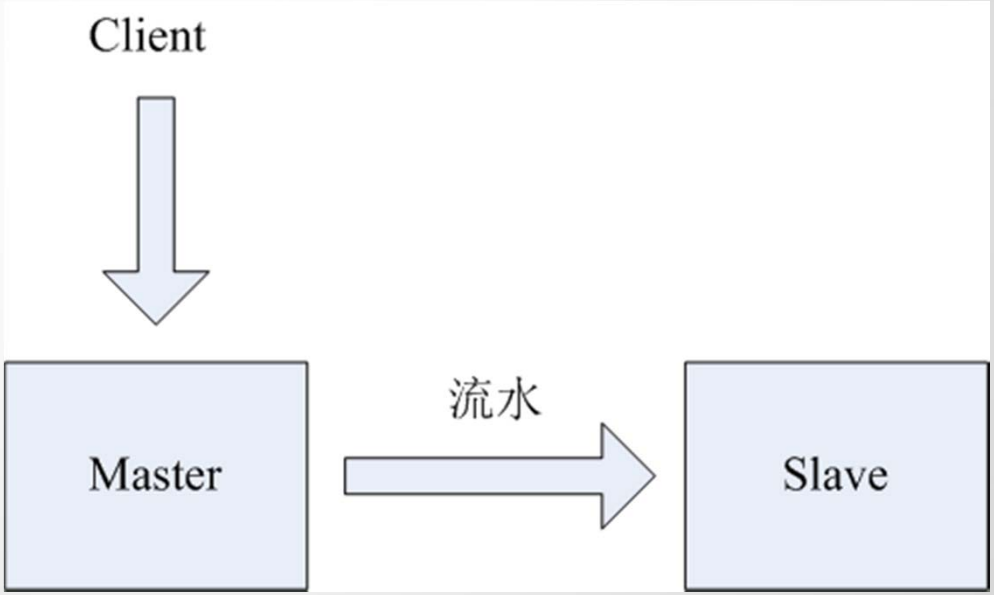
- 如果真的挂了 …. …
  - 防止雪崩，避免蝴蝶效应
    - 把防雪崩内置到组件
  - 柔性可用，追求不完美
    - 只求完美的团队，
      不能胜任海量服务。
      0/1完美 = **60**分
  - 保护点前置，赢得处理空间
    - 终端配合的容灾

存储层容灾－分而治之

- 与接入层/逻辑层对比
  - 接入层：**GSLB, LVS, IP redirect, Client Retry**
  - 逻辑层：无状态设计
- 存储层容灾是海量系统最复杂的设计
- 分而治之
  - 分离业务场景，寻求简单设计

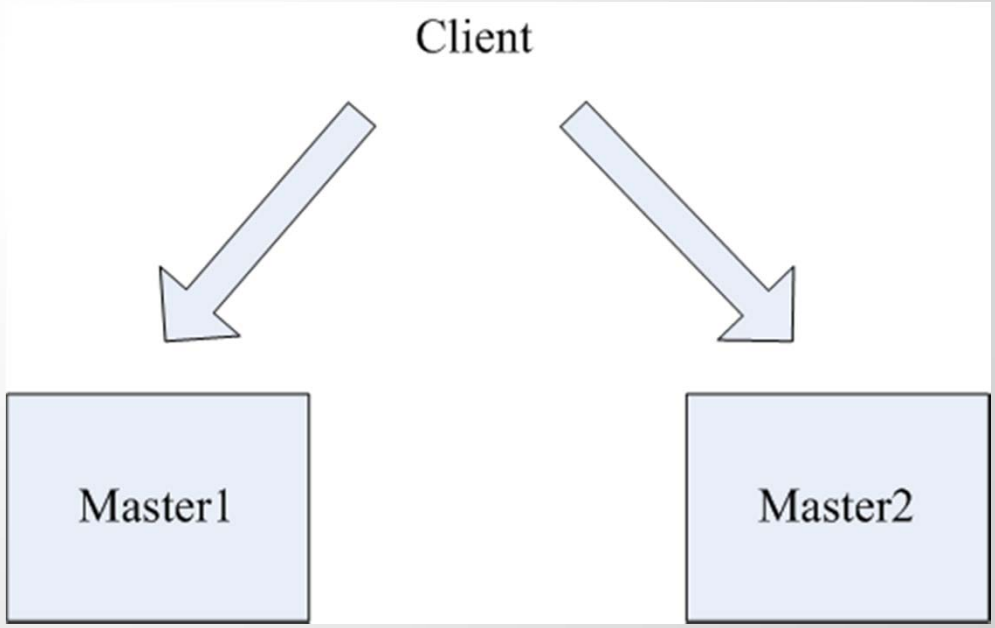
主备

- 实现简单
- 局限
  - 容忍最终一致性
  - 故障时不可写
- **Example**
  - 帐号系统



双写

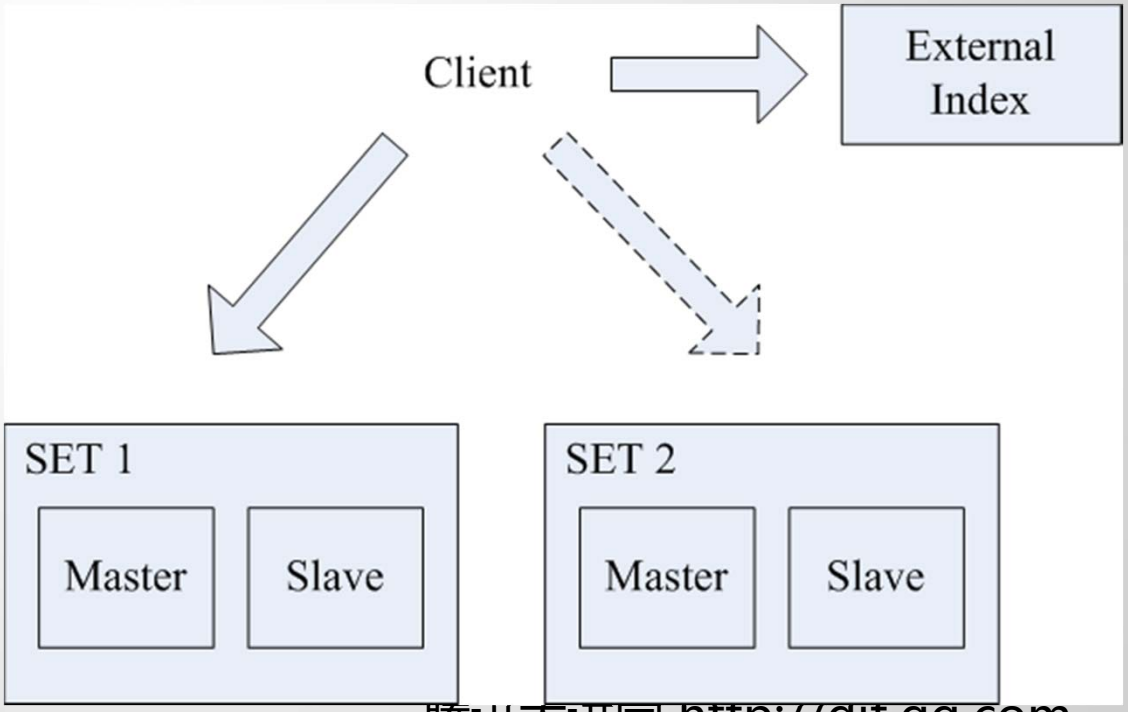
- 实现简单
- 故障时可写
- 局限
  - 容忍轻度数据丢失
- **Example**
  - 用户终端类型记录



SET模型＋双写

- 实现简单（注：**SET1**写入不成功时，切换到**SET2**写入）
- 完全一致的备份副本
- 局限
  - 只支持追加写
  - 需要外部索引
- 简化版**Google FS**

- **Example**
  - 语音/图片存储



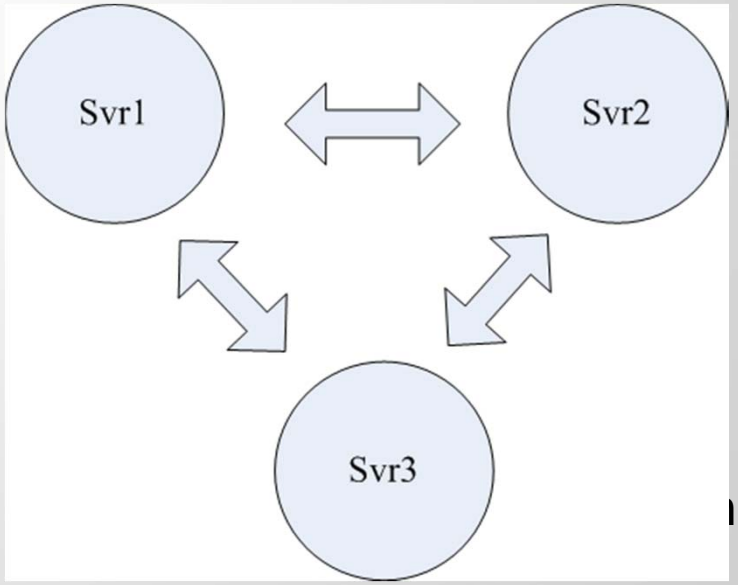
Quorum

- 分布式理论
  - **CAP理论**
  - **Paxos**
    - **Leslie Lamport**
    - **Chubby，ZooKeeper**
  - **Quorum：Amazon Dynamo**
    - **R+W>N**
    - **Vector Clock：解决冲突**
    - **Merkle Tree：节点恢复**

Simple Quorum

- 实现SYNC协议的序列发生器
- 极高稳定度要求

- 避免Vector Clock：递增的序列号
- 避免Merkle Tree：全量加载
  - 一位毕业生的创意：按SET分布，全量数据从2G减到200K



关注复杂点

- 协议
- 容灾
- 轻重
- 监控

终端的迷思

- 复杂的逻辑
- 高昂的变更
- 致命的风险

前轻后重

- 功能点后移，发挥后台快速变更的优势

接入优化：从GSLB到IP重定向

“偷流量”防御：屏蔽流量异常的终端

后台适配

- 群聊的例子
- 二维码扫描的例子

腾讯大讲堂 http://djt.qq.com

关注复杂点

- 协议
- 容灾
- 轻重
- 监控

监控的痛苦

- 海量的日志：数百G/小时
- 实时的图表：1分钟
- 灵活的需求：复杂的关联统计

- 鱼与熊掌不可兼得也

分而治之

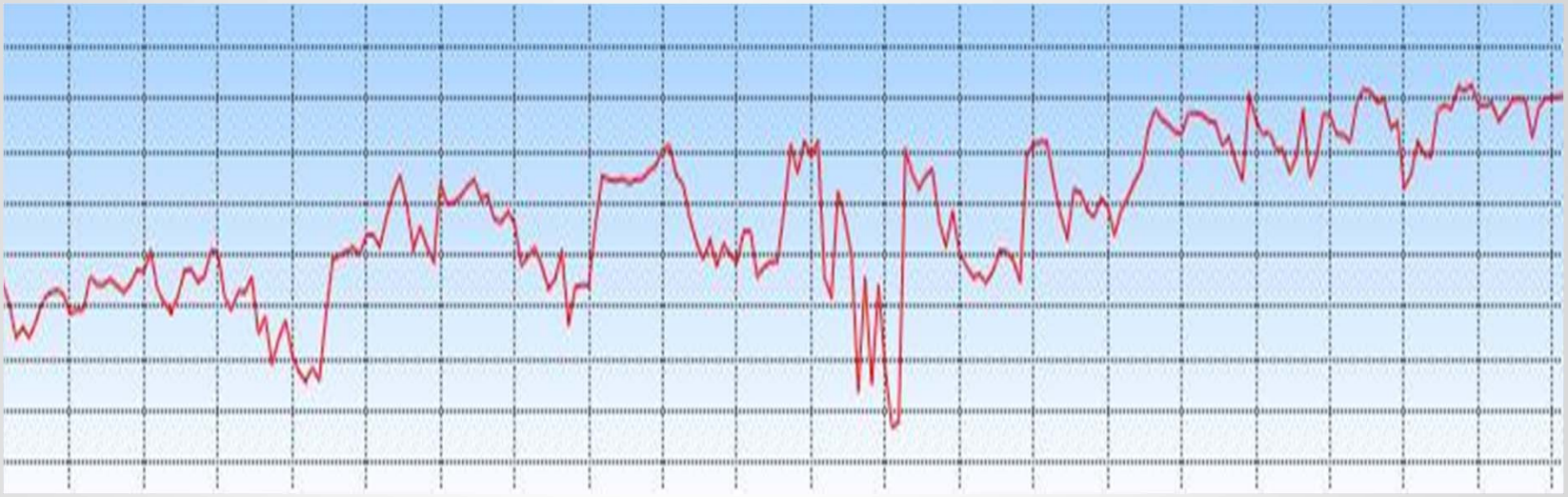
- 监控 != 统计
- 监控
  - 反映系统运行状态
  - 关联实时报警
  - 可靠性与业务系统等同
- 统计
  - 反馈业务指标
  - 非实时（数小时~一天）
  - 灵活变化
- 90%以上数据项属于监控。需要专用监控系统

监控

- 监控系统
  - 极致的简单
  - **AttrAPI**：单一数值取样接口，易于添加，所见即所得
  - 数千监控项
- 统计系统
  - 极致的灵活
  - **OssLog**：日志汇总接口
  - 大日志量，数百统计项
  - **Hadoop**

- 在故障可被用户感知前排除它

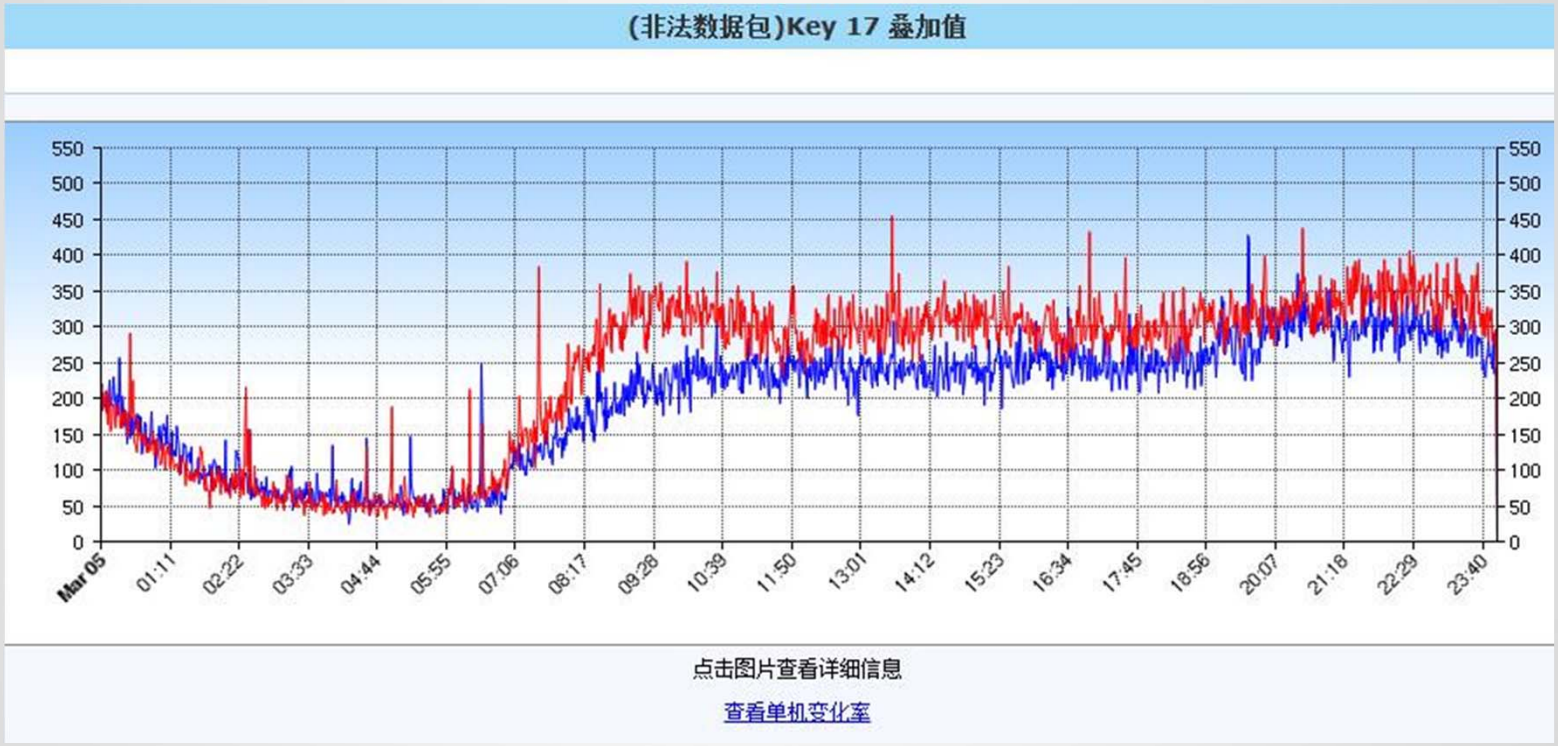
春晚时段监控曲线



LBS地图

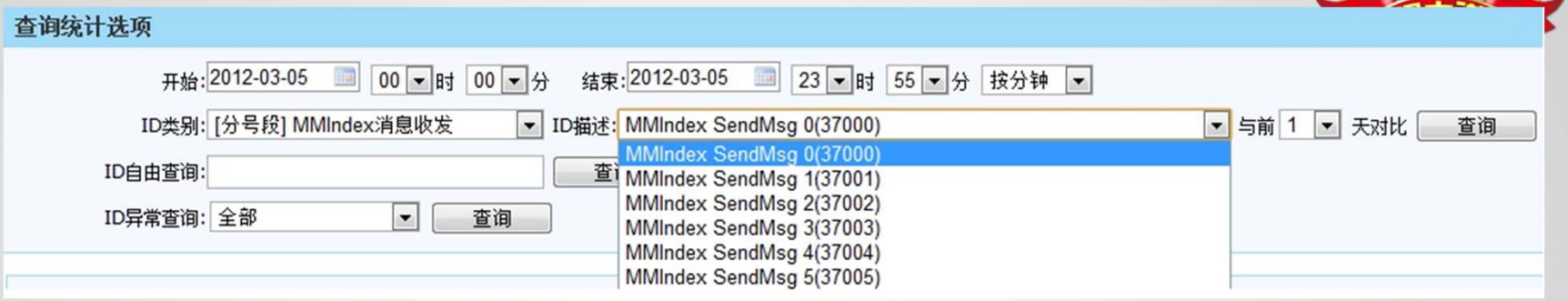
djt.qq.com

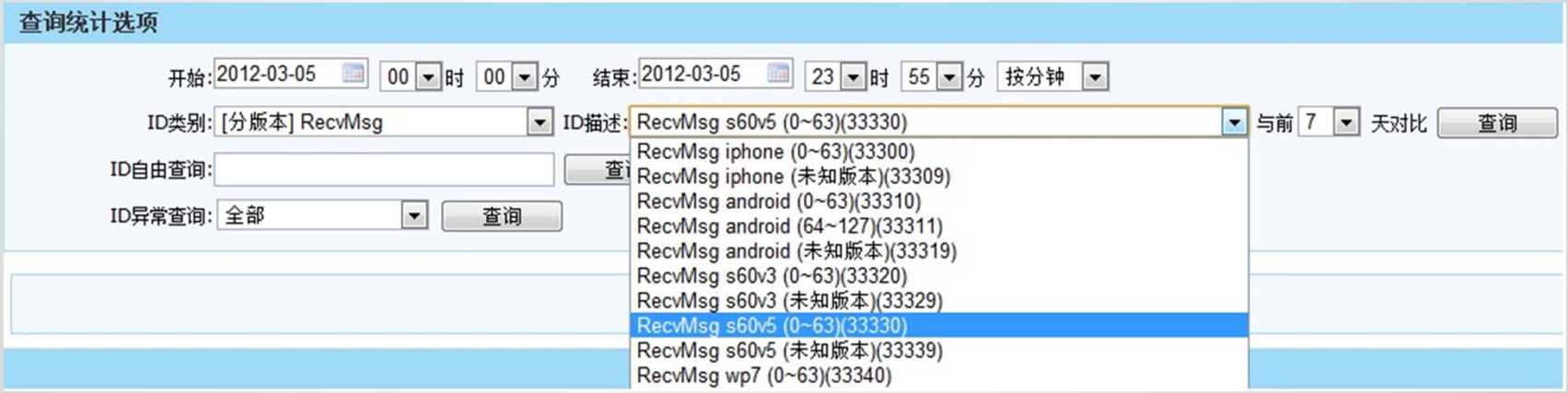
让监控更灵敏—捕捉异常



腾讯大讲堂 http://djt.qq.com

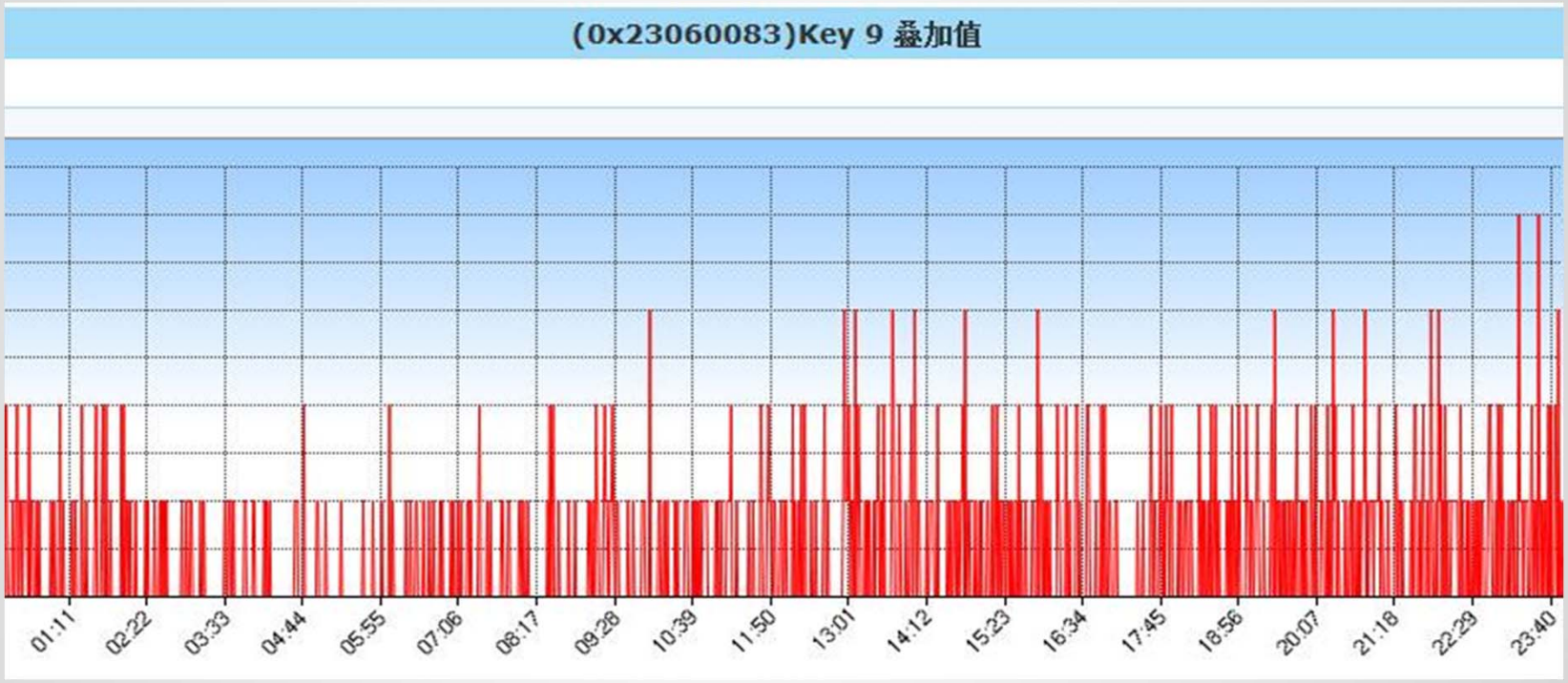
让监控更灵敏－分段监控


查询统计选项
开始:2012-03-05 00 时 00 分
结束:2012-03-05 23 时 55 分 按分钟
ID类别: [分号段] MMIndex消息收发
ID描述: MMIndex SendMsg 0(37000)
与前 1 天对比
查询
ID自由查询:
ID异常查询: 全部
查询
MMIndex SendMsg 0(37000)
MMIndex SendMsg 1(37001)
MMIndex SendMsg 2(37002)
MMIndex SendMsg 3(37003)
MMIndex SendMsg 4(37004)
MMIndex SendMsg 5(37005)


查询统计选项
开始:2012-03-05 00 时 00 分
结束:2012-03-05 23 时 55 分 按分钟
ID类别: [分版本] RecvMsg
ID描述: RecvMsg s60v5 (0~63)(33330)
与前 7 天对比
查询
ID自由查询:
ID异常查询: 全部
查询
RecvMsg iphone (0~63)(33300)
RecvMsg iphone (未知版本)(33309)
RecvMsg android (0~63)(33310)
RecvMsg android (64~127)(33311)
RecvMsg android (未知版本)(33319)
RecvMsg s60v3 (0~63)(33320)
RecvMsg s60v3 (未知版本)(33329)
RecvMsg s60v5 (0~63)(33330)
RecvMsg s60v5 (未知版本)(33339)
RecvMsg wp7 (0~63)(33340)

让监控更灵敏－灰度的利器

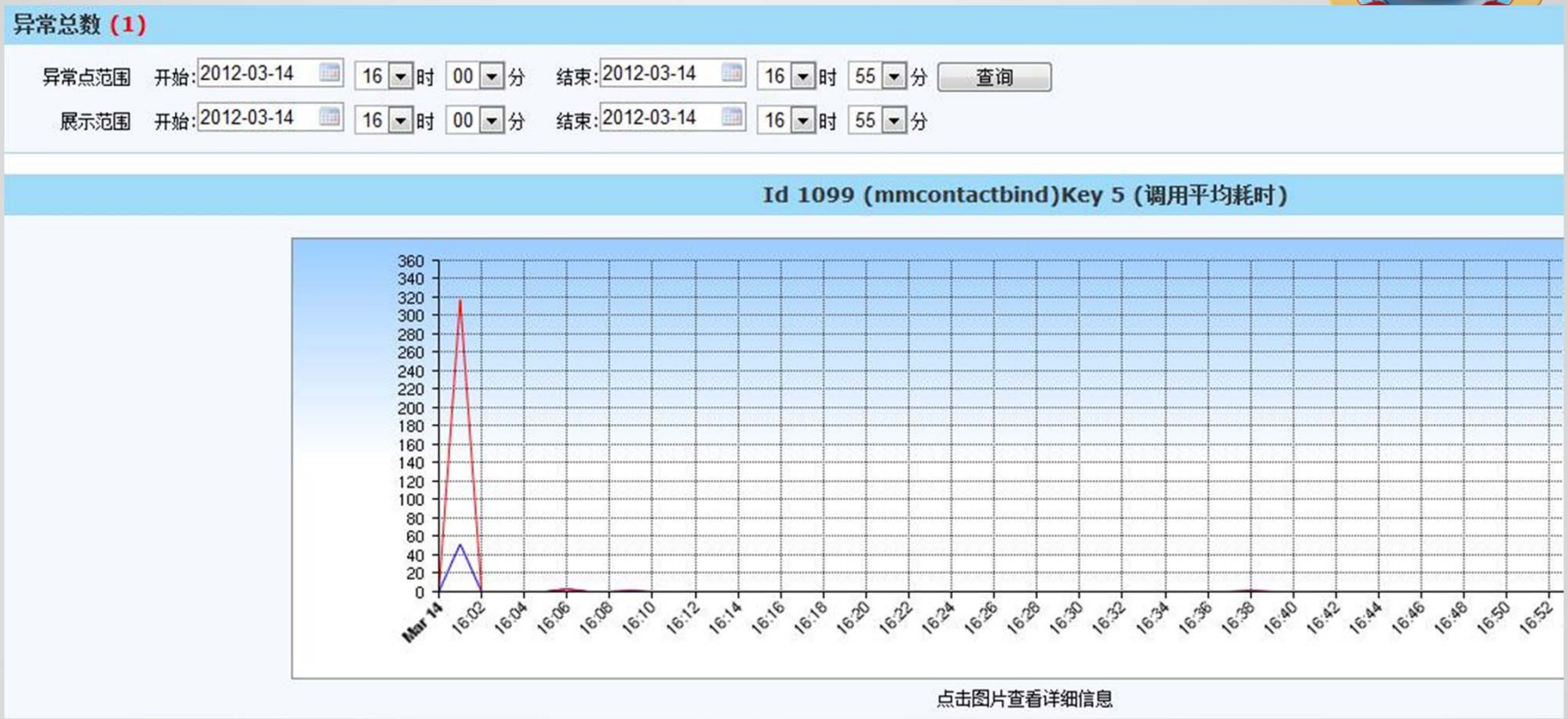
Example: Android Crash Report



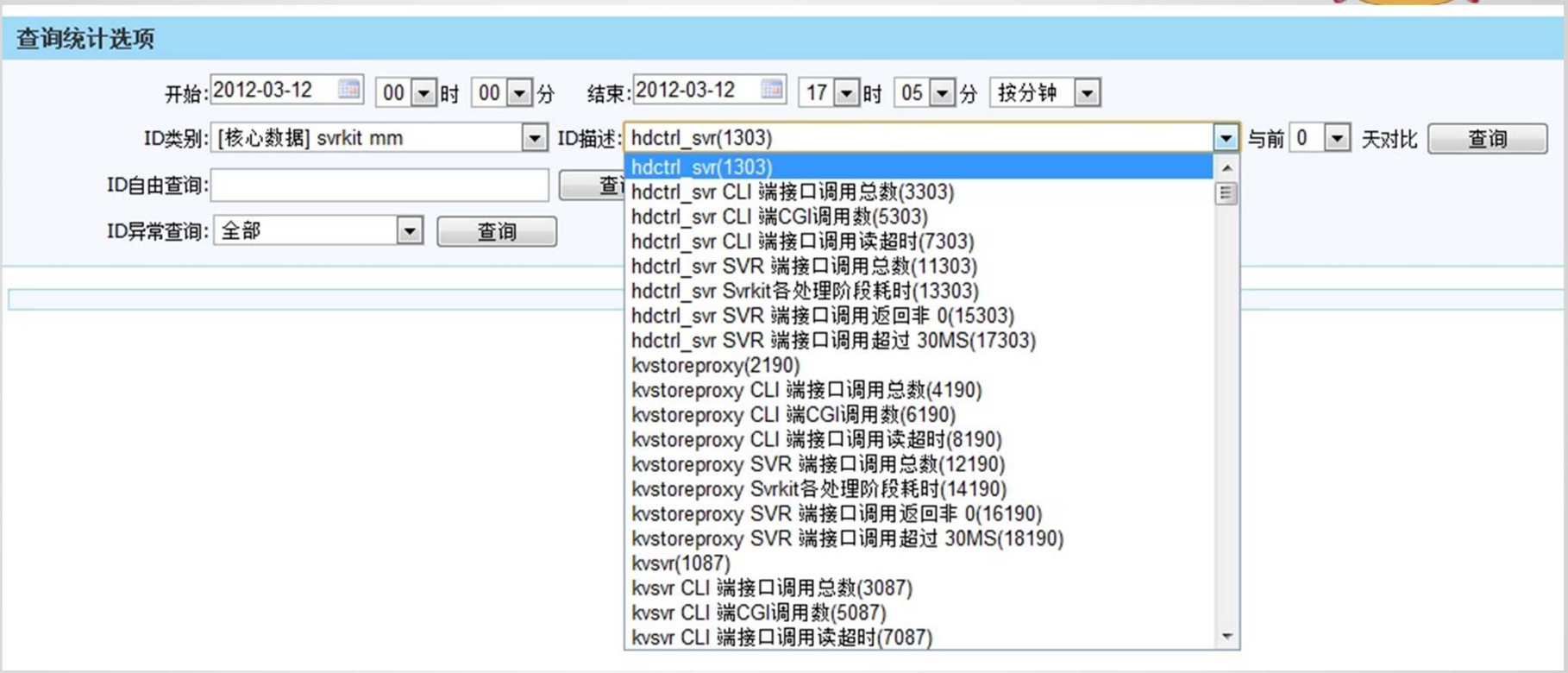
让监控更准确－监控点前移

	键值key	键值描述
☑	0	iphone平台收取消息延时0-1秒
☑	1	iphone平台收取消息延时1-2秒
☑	2	iphone平台收取消息延时2-3秒
☑	3	iphone平台收取消息延时3-5秒
☑	4	iphone平台收取消息延时5-10秒
☑	5	iphone平台收取消息延时10-20秒
☑	6	iphone平台收取消息延时20-30秒
☑	7	iphone平台收取消息延时30-60秒
☑	8	iphone平台收取消息延时60-120秒
☑	9	iphone平台收取消息延时120-180秒
☑	10	iphone平台收取消息延时180-300秒
☑	11	iphone平台收取消息延时300-600秒
☑	12	iphone平台收取消息延时600-1200秒
☑	13	iphone平台收取消息延时1200-1800秒
☑	14	iphone平台收取消息延时1800-3600秒
☑	15	iphone平台收取消息延时大于3600秒

自动报警


异常总数 (1)
异常点范围 开始:2012-03-14 16 时 00 分 结束:2012-03-14 16 时 55 分 查询
展示范围 开始:2012-03-14 16 时 00 分 结束:2012-03-14 16 时 55 分
Id 1099 (mmcontactbind)Key 5 (调用平均耗时)
360 340 320 300 280 260 240 220 200 180 160 140 120 100 80 60 40 20 0
Mar 14 16:02 16:04 16:06 16:08 16:10 16:12 16:14 16:16 16:18 16:20 16:22 16:24 16:26 16:28 16:30 16:32 16:34 16:36 16:38 16:40 16:42 16:44 16:46 16:48 16:50 16:52
点击图片查看详细信息

把监控嵌入基础框架


查询统计选项
开始:2012-03-12 00 时 00 分
结束:2012-03-12 17 时 05 分 按分钟
ID类别: [核心数据] svrkit mm
ID描述: hdctrl_svr(1303)
与前 0 天对比
查询
ID自由查询:
ID异常查询: 全部
查询
hdctrl_svr(1303)
hdctrl_svr CLI 端接口调用总数(3303)
hdctrl_svr CLI 端CGI调用数(5303)
hdctrl_svr CLI 端接口调用读超时(7303)
hdctrl_svr SVR 端接口调用总数(11303)
hdctrl_svr Svrkit各处理阶段耗时(13303)
hdctrl_svr SVR 端接口调用返回非 0(15303)
hdctrl_svr SVR 端接口调用超过 30MS(17303)
kvstoreproxy(2190)
kvstoreproxy CLI 端接口调用总数(4190)
kvstoreproxy CLI 端CGI调用数(6190)
kvstoreproxy CLI 端接口调用读超时(8190)
kvstoreproxy SVR 端接口调用总数(12190)
kvstoreproxy Svrkit各处理阶段耗时(14190)
kvstoreproxy SVR 端接口调用返回非 0(16190)
kvstoreproxy SVR 端接口调用超过 30MS(18190)
kvsvr(1087)
kvsvr CLI 端接口调用总数(3087)
kvsvr CLI 端CGI调用数(5087)
kvsvr CLI 端接口调用读超时(7087)

总结

- 三位一体
  - 产品的精准
  - 项目的敏捷
  - 技术的支撑
- 剥离复杂，让剩下的更简单
  - 协议
  - 容灾
  - 轻重
  - 监控
- 一些原则
  - 大系统小做
  - 面向最坏的思考，柔性可用
  - 分而治之

最后，让剩下的更简单

- 摇一摇 & 漂流瓶，一周完成
- 3个月30个内部发布
- 每天20个后台变更
- 99.95%的可用性

未来的技术挑战

- **99.99%**
- 面向**10**倍的架构提升
- 完全的**IDC**容灾

技术的追求

- 其疾如风，其徐如林，侵掠如火，不动如山

Q & A