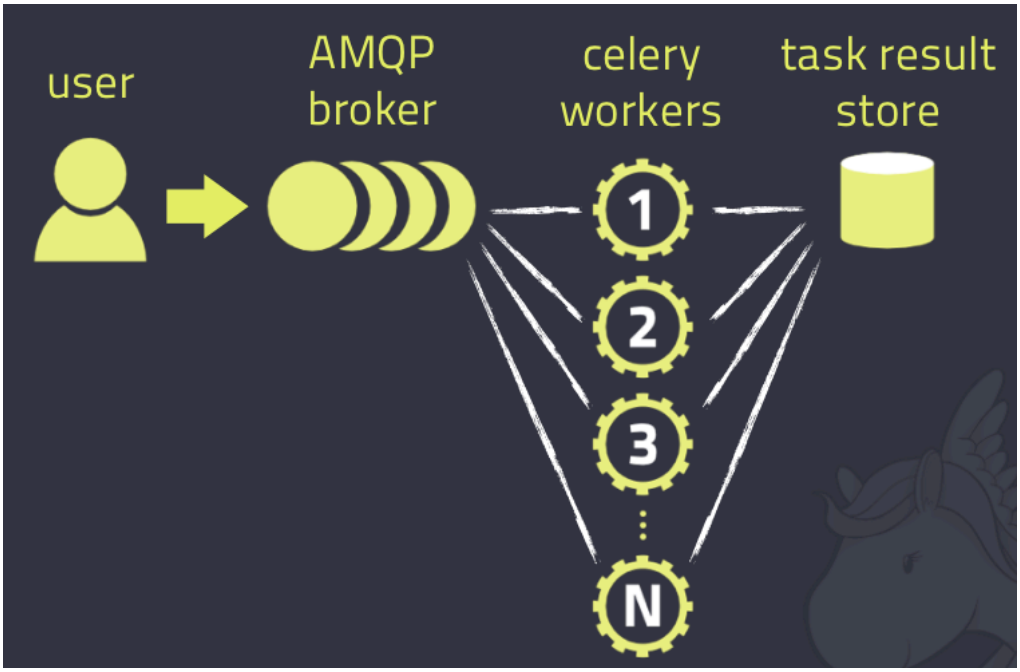
讨论课2

并行分布式框架：Celery（Python）

Celery 是一个简单、灵活且可靠的，处理大量消息的分布式系统，并且提供维护这样一个系统的必需工具。它是一个专注于实时处理的任务队列，同时也支持任务调度。Celery 需要一个发送和接受消息的传输者。RabbitMQ 和 Redis 中间人的消息传输支持所有特性，但也提供大量其他实验性方案的支持，包括用 SQLite 进行本地开发。Celery 可以单机运行，也可以在多台机器上运行，甚至可以跨越数据中心运行。



任务队列

任务队列是一种在线程或机器间分发任务的机制。

消息队列的输入是工作的一个单元，称为任务，独立的职程（Worker）进程持续监视队列中是否有需要处理的新任务。

Celery 用消息通信，通常使用中间人（Broker）在客户端和职程间斡旋。这个过程从客户端向队列添加消息开始，之后中间人把消息派送给职程。

Celery 系统可包含多个职程和中间人，以此获得高可用性和横向扩展能力。

Celery 是用 Python 编写的，但协议可以用任何语言实现。迄今，已有 Ruby 实现的 RCelery 、node.js 实现的 node-celery 以及一个 PHP 客户端 ， 语言互通也可以通过 using webhooks 实现。

特点

- 简单

Celery 易于使用和维护，并且它不需要配置文件 。

下面是一个可以实现的最简应用：

```python
from celery import Celery

app = Celery('hello', broker='amqp://guest@localhost//')

@app.task def hello(): return 'hello world'
```

- 高可用性

倘若连接丢失或失败，职程和客户端会自动重试，并且一些中间人通过 主/主 或 主/从 方式复制来提高可用性。

- 快速

单个 Celery 进程每分钟可处理数以百万计的任务，而保持往返延迟在亚毫秒级（使用 RabbitMQ、py-librabbitmq 和优化过的设置）。

- 灵活

Celery 几乎所有部分都可以扩展或单独使用。可以自制连接池、序列化、压缩模式、日志、调度器、消费者、生产者、自动扩展、 中间人传输或更多。

框架集成

Celery 易于与 Web 框架集成，其中的一些甚至已经有了集成包：

- Django django-celery
- Pyramid pyramid_celery
- Pylons celery-pylons
- Flask 不需要
- web2py web2py-celery
- Tornado tornado-celery

轻量级分布式通信框架:ZeroRPC

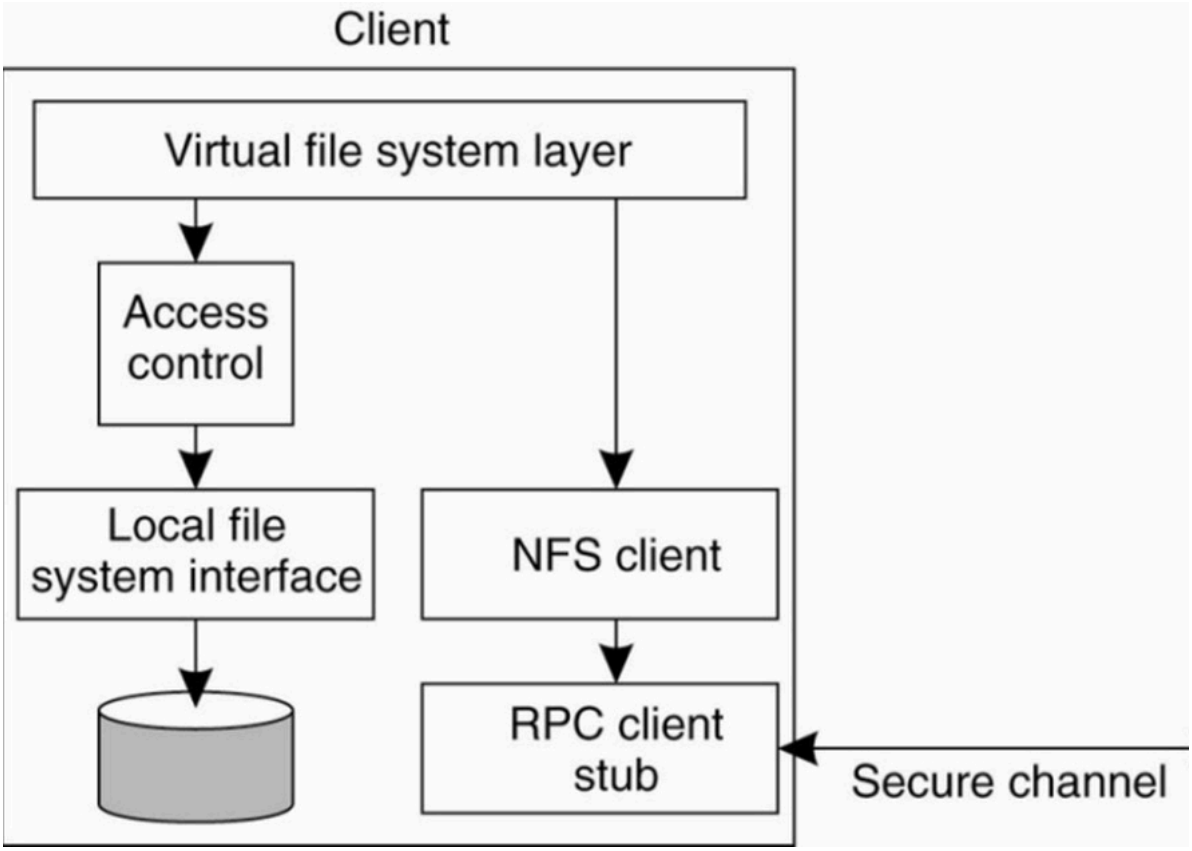
分布式系统的核心是分布式通信，而传统上开发一套支持上千台规模集群，可靠性非常高的分布式通信框架，需要不少的精力投入。而在多数情景下，我们（特别是时间宝贵的OP）并不是非常关注技术实现的细节，而是希望有一套成熟、轻量、可靠性高、使用方便而且易于调试的分布式通信框架，可以直接使用，从而把时间放在具体业务逻辑上。

在PyCon 2012大会上，dotcloud公司开源了一套基于ZeroMQ和MessagePack的分布式通信框架（或者说是协议+Python实现）。该框架因为基于ZeroMQ，使用方法是RPC，所以被命名为ZeroRPC。ZeroRPC的特点在其官网的介绍中一目了然

> *ZeroRPC is a light-weight, reliable and language-agnostic library for distributed communication between server-side processes.*

ZeroRPC的设计初衷是『simple tool to solve a simple problem』，dotcloud工程师在PPT中提到了他们设计目标：

- 最小的使用成本。比如单机时我们这么用import foo; foo.bar(42)，分布式后我们希望可以这么用：foo=RemoteService(...) ; foo.bar(42) 自文档。不需要看远程服务的源代码，我们能够获得远程服务所支持的方法，docstring等等。
- 优雅的处理异常。调用远程方法产生的异常，不应该中断远程服务，而应该在本地抛出。（异常比返回状态更方便，更简洁）
- 语言无关。同一套协议有多种实现（目前有Python和Node.js两种实现），而且实现简单，目前Python实现只有2000行代码。
- 高可用性，快速。目前Python实现基于gevent+zmq，性能非常高。 良好的debug和profile工具
- MessagePack ZeroRPC选择MessagePack[5]作为序列化工具，相比于JSON/BSON/YAML等格式，MsgPack提供20~50倍的序列化速度以及1/2大小的序列化产出。



ZeroRPC协议

ZeroRPC分为三层：

Wire (or transport) layer 利用ZeroMQ和MessagePack进行通信 Event (or message) layer 最复杂的一层，处理心跳，事件等 RPC layer RPC Request+Response

超时和心跳

由于ZMQ隐藏了Socket细节，所以无法感知断线，于是ZeroRPC采用心跳包的方式进行在线检测。Client端在断线后会抛出LostRemote异常。默认断线超时为30s，可设定。

身份验证和传输加密

目前版本的ZeroRPC没有实现任何形式的身份验证，传输加密官方建议通过SSL实现。

中间件

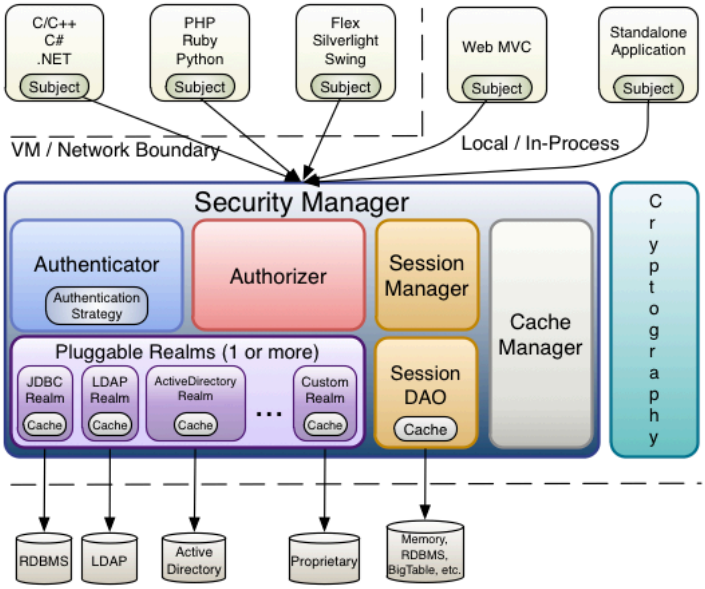
最新版本的ZeroRPC支持middleware API，比如StatsD Middleware[6]，可以跟踪Request/Response的耗时并传送给StatsD，以分析系统性能。

Java安全框架：Apache Shiro

Apache Shiro(发音为“shee-roh”，日语“堡垒(Castle)”的意思)是一个强大易用的Java安全框架，提供了认证、授权、加密和会话管理功能，可为任何应用提供安全保障 - 从命令行应用、移动应用到大型网络及企业应用。

Shiro为解决下列问题提供了保护应用的API：

```
认证  -用户身份识别，常被称为用户“登录”；
授权 - 访问控制；
密码加密 - 保护或隐藏数据防止被偷窥；
会话管理 - 每用户相关的时间敏感的状态。
```



特性

- 易于使用 - 易用性是这个项目的最终目标。应用安全有可能会非常让人糊涂，令人沮丧，并被认为是“必要之恶”【译注：比喻应用安全方面的编程。】。若是能让它简化到新手都能很快上手，那它将不再是一种痛苦了。
- 广泛性 - 没有其他安全框架可以达到Apache Shiro宣称的广度，它可以为你的安全需求提供“一站式”服务。
- 灵活性 - Apache Shiro可以工作在任何应用环境中。虽然它工作在Web、EJB和IoC环境中，但它并不依赖这些环境。Shiro既不强加任何规范，也无需过多依赖。
- Web能力 - Apache Shiro对Web应用的支持很神奇，允许你基于应用URL和Web协议(如REST)创建灵活的安全策略，同时还提供了一套控制页面输出的JSP标签库。 可插拔 - Shiro干净的API和设计模式使它可以方便地与许多的其他框架和应用进行集成。你将看到Shiro可以与诸如Spring、Grails、Wicket、Tapestry、Mule、Apache Camel、Vaadin这类第三方框架无缝集成。
- 支持 - Apache Shiro是Apache软件基金会成员，这是一个公认为了社区利益最大化而行动的组织。项目开发和用户组都有随时愿意提供帮助的友善成员。像Katasoft这类商业公司，还可以给你提供需要的专业支持和服务。

阿里服务互联网金融的关系数据库：OceanBase

1、OceanBase是阿里巴巴自主研发支持海量数据的高性能分布式数据库系统。

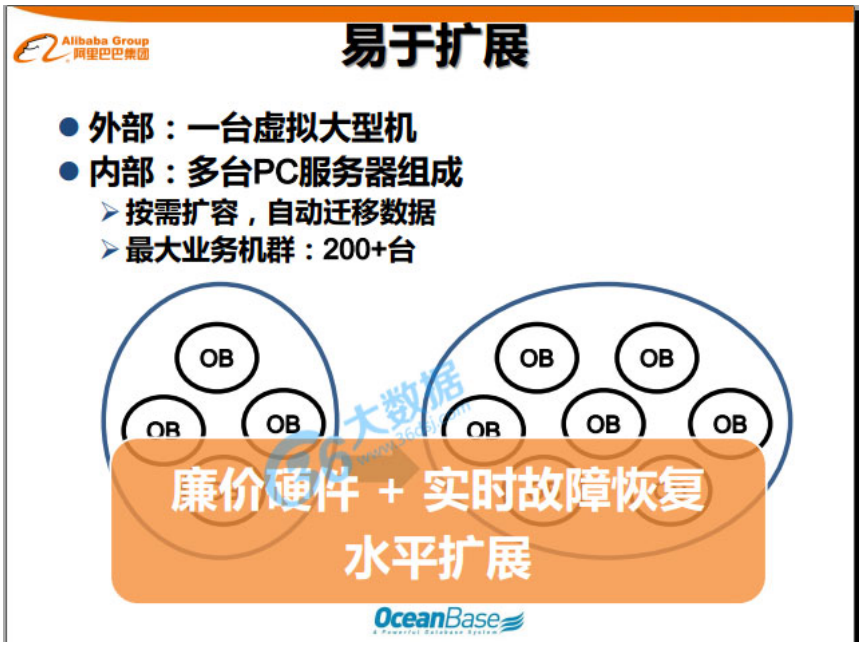
2、比传统的关系数据库，谈及OceanBase的最大亮点，当属可自动扩展的特点，它不仅仅可以扩展到一个数据中心，乃至同城，在未来，OceanBase能成为跨地域多数据中心的全球数据库。当然，OceanBase还有一些特性，比如强一致性，能够自动容忍一台服务器甚至是整个数据中心故障，而不会丢失一条记录。值得强调的是，支付宝交易之所以会选择OceanBase而不是开源的MySQL，正是因为只有OceanBase能够做到完全不丢数据。

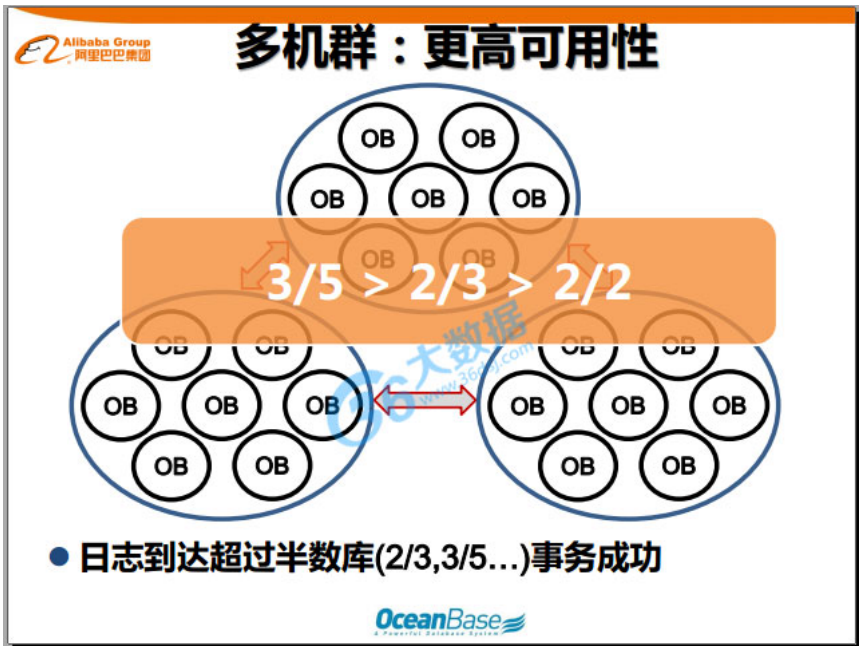
3、OceanBase内部会自动把数据切分为一个个比较小的分片，每台机器服务若干个分片，当某台服务器的分片成为热点时，会自动触发迁移操作，将分片从负载较高的服务器迁移出去，这样，就避免了“木桶效应”。这也是OceanBase的一个优势，无论双十一之前热点怎么变，OceanBase都可以很快把热点数据均衡到整个集群，而不是因为一两台服务器把整个集群压垮了。

4、OceanBase是真正的分布式关系数据库，不仅仅支持分布式，还支持SQL、事务、并发控制这样的关系数据库特性，用户使用起来和MySQL没有任何差别。

5、OceanBase的事务引擎面向内存设计的，它的特点就是快，而且避免了传统数据库的写入放大问题，这样，OceanBase的锁等待天生就要更少。另外，OceanBase还提出并第一个实现了提前解锁、排队等待等优化思想，这些思想后来也应用到集团MySQL数据库的patch中，成为阿里双十一应对热点问题的标配。

6、和MySQL的不同点在于，MySQL主备同步是有丢数据风险的，而OceanBase通过Paxos选举协议实现强一致性。无论是一台服务器，还是数据中心整体故障，OceanBase都能够自动恢复，而且做到完全不丢数据。OceanBase的这种做法和Google、Microsoft以及Amazon云存储系统的底层原理都是共通的。

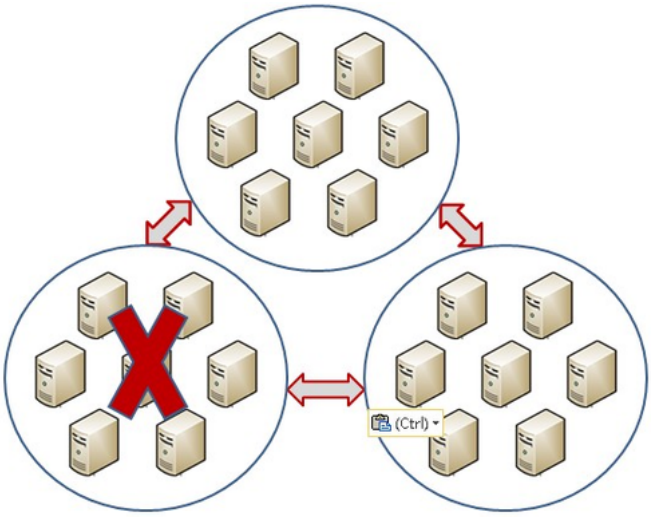




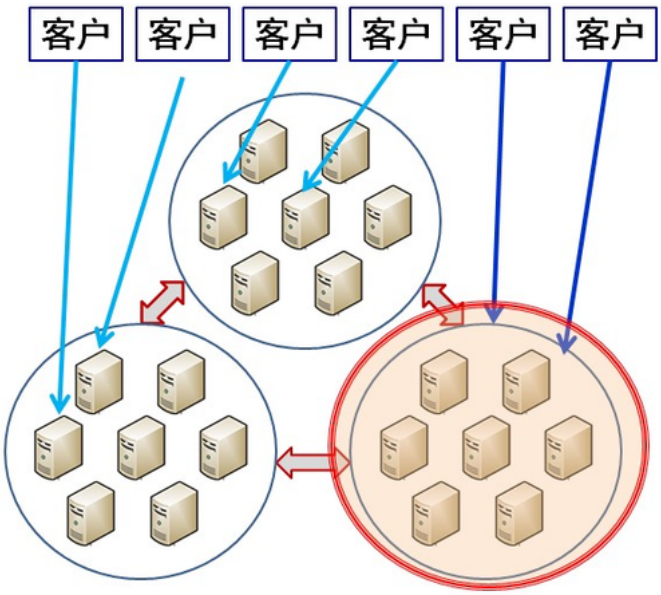
对于数据库，与传统企业相比，互联网企业最大的不同之一是并发访问量非常大。传统商业企业、银行，用户需要通过收银台、银行终端、ATM柜员机、POS机等专用设备开展业务并访问数据库，几百和几千的数据库并发访问比较常见，几万以上的并发访问相当少见。在互联网上，每一个草根网民都可以发起购物交易并访问数据库，几十万的数据库并发访问时常可见，几百万甚至千万的并发访问都可以见到（例如双11下的淘宝、天猫和支付宝）。如此之大的并发访问下，商业数据库软件及其高可靠的数据库服务器和共享存储的成本成为了不可承担之重。

由于上述原因，OceanBase的一个基本假设就是硬件(服务器、存储、网络等)是不可靠的，另一个基本假设是单机(数据库服务器及共享存储)无法满足互联网业务的需求。因此，OceanBase必须是一个多机(分布式)系统，并且必须保证任何时刻出现的少量硬件(服务器、存储、网络等)异常不影响业务。

为此，OceanBase引入了Paxos协议，每一笔事务，主库执行完成后，要同步到半数以上库(包括主库自身)，例如3个库中的2个库，或者5个库中的3个库，事务才成功。这样，少数库(例如3个库中的1个库，或者5个库中的2个库)异常后业务并不受影响：



OceanBase则是“多活”设计，即多个库（3个，5个等）每个都可以有部分读写流量，升级时先把要升级的库的读写流量切走，升级后先进行数据对比，正常后逐步引入读写流量(白名单，1%，5%，10%.....)，一切正常并运行一段时间后再升级其他的库：



基于硬件不可靠的假设并且能够容忍少量服务器的故障，OceanBase使用了相对廉价的PC服务器代替高可靠服务器并且不再使用昂贵的共享存储，从而不仅提供了比使用高可靠服务器和共享存储低得多的成本，容忍少数服务器乃至少数机群故障意味着比传统数据库更高的可靠性。通过灰度升级，OceanBase避免了传统数据库的“一锤子买卖”的升级，极大地降低了数据库维护升级的风险。

参考资料

- http://blog.csdn.net/liuxiaochen123/article/details/47981111
- http://docs.jinkan.org/docs/celery/index.html
- https://github.com/0rpc/zerorpc-python/blob/master/doc/protocol.md
- http://msgpack.org/
- http://tech.it168.com/a2011/0701/1212/000001212291_all.shtml
- http://www.chinacloud.cn/show.aspx?id=19024&cid=17