讨论课二

1352837 倪雨婷

分布式

CAP理论是由Eric Brewer提出的分布式系统中最为重要的理论之一：

1 Consistency：[强]一致性，事务保障，ACID模型。

2 Availiablity：[高]可用性，冗余以避免单点，至少做到柔性可用（服务降级）。

3 Partition tolerance：[高]可扩展性（分区容忍性）：一般要求系统能够自动按需扩展，比如HBase。

CAP原理告诉我们，这三个因素最多只能满足两个，不可能三者兼顾。对于分布式系统来说，分区容错是基本要求，所以必然要放弃一致性。对于大型网站来说，分区容错和可用性的要求更高，所以一般都会选择适当放弃一致性。

在CAP三者中，“可扩展性”是分布式系统的特有性质。分布式系统的设计初衷就是利用集群多机的能力处理单机无法解决的问题。当需要扩展系统性能时，一种做法是优化系统的性能或者升级硬件(scale up)，一种做法就是“简单”的增加机器来扩展系统的规模(scale out)。好的分布式系统总在追求”线性扩展性”，即性能可以随集群数量增长而线性增长。

可用性和可扩展性一般是相关联的，可扩展性好的系统，其可用性一般会比较高，因为有多个服务(数据)节点，不是整体的单点。所以分布式系统的所有问题，基本都是在一致性与可用性和可扩展性这两者之间的一个协调和平衡。对于没有状态的系统，不存在一致性问题，根据CAP原理，它们的可用性和分区容忍性都是很高，简单的添加机器就可以实现线性扩展。而对于有状态的系统，则需要根据业务需求和特性在CAP三者中牺牲其中的一者。一般来说，交易系统类的业务对一致性的要求比较高，一般会采用ACID模型来保证数据的强一致性，所以其可用性和扩展性就比较差。而其他大多数业务系统一般不需要保证强一致性，只要最终一致就可以了，它们一般采用BASE模型，用最终一致性的思想来设计分布式系统，从而使得系统可以达到很高的可用性和扩展性。

存储

数据库分区

分区技术主要目的是为了在特定的SQL操作中减少数据读写的总量以缩减响应时间。分区主要有两种形式：

1.水平分区

这种形式分区是对表的行进行分区，通过这样的方式不同分组里面的物理列分割的数据集得以组合，从而进行个体分割（单分区）或集体分割（1个或多个分区）。所有在表中定义的列在每个数据集中都能找到，所以表的特性依然得以保持。

2.垂直分区

这种分区方式一般来说是通过对表的垂直划分来减少目标表的宽度，使某些特定的列被划分到特定的分区，每个分区都包含了其中的列所对应的行。

举个简单例子：一个包含了大text和BLOB列的表，这些text和BLOB列又不经常被访问，这时候就要把这些不经常使用的text和BLOB了划分到另一个分区，在保证它们数据相关性的同时还能提高访问速度。

在数据库供应商开始在他们的数据库引擎中建立分区（主要是水平分区）时，DBA和建模者必须设计好表的物理分区结构，不要保存冗余的数据（不同表中同时都包含父表中的数据）或相互联结成一个逻辑父对象（通常是视图）。这种做法会使水平分区的大部分功能失效，有时候也会对垂直分区产生影响。

分布式存储

分布式存储系统，是将数据分散存储在多台独立的设备上。传统的网络存储系统采用集中的存储服务器存放所有数据，存储服务器成为系统性能的瓶颈，也是可靠性和安全性的焦点，不能满足大规模存储应用的需要。分布式网络存储系统采用可扩展的系统结构，利用多台存储服务器分担存储负荷，利用位置服务器定位存储信息，它不但提高了系统的可靠性、可用性和存取效率，还易于扩展。 通过数据分布，复制以及容错等机制，能够将分布式存储系统部署到成千上万台服务器。可扩展性的实现手段很多，如通过增加副本个数或者缓存提高读取能力，将数据分片使得每个分片可以被分配到不同的工作节点以实现分布式处理，把数据复制到多个数据中心，等等。主流的分布式存储系统大多带有总控节点，且能够支持成千上万台的集群规模。

负载均衡

负载均衡 建立在现有网络结构之上，它提供了一种廉价有效透明的方法扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性，指分摊到多个操作单元上进行执行，例如Web服务器、FTP服务器、企业关键应用服务器和其它关键任务服务器等，从而共同完成工作任务。

负载均衡可以帮我们解决两个方面的问题，第一个即提高可用性。这里面的可用性主要是从WEB服务器，的角度来讲的，如果说我们只有一台Web服务器，而它遇到了某种未知的错误导致IIS无法启动，那么我们的网站就无法访问了，这就是一种比较低的可用性。那么利用负载均衡，放在我们Web服务器的前面，由它来收集所有的请求，然后转发给我们的Web服务器， 这时候我们就可以添加两台Web服务器，如果其中有一台坏了，至少还有另一台在工作，也不至于导致我们的网络无法访问。

除了可用性以外，负载均衡还可以帮助我们提高可扩展性，当然这个可扩展性同样是指的Web服务器层面。从网站性能的角度来讲，好几个程序员花上好几天的时间做了一些优化所带来的效果有时候可能还没有直接加一根内存条来的快。内存加完了没什么影响，我们还可以换更好的CPU，CPU换完了，我们还可以用固态硬盘，甚至很多公司已经开始直接把数据放到内存中了。 如果这些都不可以再加了呢？那就再加机器吧，一台服务器可以处理1000个并发，那么两台理论上是2000了，所以这就有了我们的横向扩展。

可扩展性就是系统通过规模的扩展来提高系统的承载能力，毕竟服务器自身的垂直扩展很快就会受到制约，单机很快便不能满足我们的需求，因此这种能力往往通过增加物理服务器或集群节点等方面来实现，这种能力越强，承载能力可提升的空间也越大。而web站点的水平扩展，负载均衡是一种最常见的手段。例如：HTTP重定向。例如我们请求某个页面时，被转向登录页，登录页面之后又被转到了某个页面。大致

来说就是，浏览器请求某个URL后，服务器通过HTTP响应头信息中的Location标记返回一个新的URL，这样浏览器主会继续请求这个新的URL，完成自动跳转。也正是因为HTTP重定向有请求转移和自动跳转的能力，所以我们就可以用它来实现负载均衡以实现WEB扩展。

常用的负载均衡方法

很多的网站一开始并不需要太大的规模，但是做为网站设计者从一开始就必须考虑到扩展，做一个可扩展性强的架构。所谓可扩展性就是系统通过规模的扩展来提高系统的承载能力，毕竟服务器自身的垂直扩展很快就会受到制约，单机很快便不能满足我们的需求，因此这种能力往往通过增加物理服务器或集群节点等方面来实现，这种能力越强，承载能力可提升的空间也越大。而web站点的水平扩展，负载均衡是一种最常见的手段。下面介绍一下几种实现负载均衡的方式。

HTTP重定向

HTTP重定向，相信对于所有web程序员都不陌生，例如我们请求某个页面时，被转向登录页，登录页面之后又被转到了某个页面。大致来说就是，浏览器请求某个URL后，服务器通过HTTP响应头信息中的Location标记返回一个新的URL，这样浏览器主会继续请求这个新的URL，完成自动跳转。也正是因为HTTP重定向有请求转移和自动跳转的能力，所以我们就可以用它来实现负载均衡以实现WEB扩展。

DNS负载均衡

我们知道DNS是负责解析域名的，当我们用域名访问站点时，实际上都会经过DNS服务器来获取域名指向的IP，实际上DNS服务器完成了域名到IP的映射，同样这个映射可以是一对多的，也就是DNS可以把对域名的请求按照一定的策略分配到不同的服务器上，这样我们就可以依此来实现负载均衡。貌似和HTTP重定向很样，但是实现的机制却是完全不同的。（在window下可以用nslookup命令查询域名对应的IP地址列表，这个命令会返回离你最近的DNS服务器缓存的记录并不一定是全部）

反向代理负载均衡

我们之前的缓存介绍中，介绍过反向代理服务器，它同样可以作为调度器来实现负载均衡系统。反向代理服务器核心工作是转发HTTP请求，它工作在HTTP层面，也就是TCP七层中的第七层应用层，因此基于反射代理的负载均衡系统也称为七层负载均衡。目前几乎所有主流WEB服务器都支持基于反向代理的负载均衡，因此实现它并不困难。

IP负载均衡

基于IP负载均衡的系统工作在传输层，会对数据包中的IP地址和端口信息进行修改，所以也称为四层负载均衡。它会在数据到达应用层之前，已完成转发，因些这工作都是由系统内核来完成，应用程序对此无能为力，当然性能来说也会很大的提升。

直接路由

不同于IP负载均衡，直接路由负载均衡高度器工作在数据链路层，它通过修改数据包的目标MAC地址，将数据包转发到实际服务器，不同的是，这些处理的结果直接发送给用户，不再经过调度器。这时我们的实际服务器必须直接连接到外网，并且不在以调度服务器为默认网关。

IP隧道

简单的说，就是调度器将收到的数据包封装到一个新的IP数据包中，转交给实现服务器，然后实际服务器可以处理数据包直接响应客户端。

ID分配

取模或分段的分布式分配

将整个id空间按取模或分段等分为若干个独立的id子空间，每个id子空间由一个独立的分配器负责。

优点：简单，各个id分配器无需协作，即使发生网络划分时，也可保证可用性和id的不冲突。

如果在国际化环境的多IDC里进行部署，需要预先将id空间划分为N份，每个国家里部署若干份。每个IDC内应用只连本IDC的id分配服务。

在均衡性上的不足：在同一个IDC内，均衡性可以在接入层均衡算法保证，但是在多个IDC里，ID分配器个数的比例和id增长的服务往往是不吻合的，因此在多个IDC内，id是无法保证均衡增长的。

均衡性上的改进

将id分配分为两层：

· 上层的“id分配器”对应用暴露，提供一次申请一个id的接口，一般本IDC的应用只连本IDC的id分配器。

·下层的“段分配器”对“id分配器”提供服务。id分配器“知晓”所有IDC的所有段分配器的存在，使用均衡策略向段分配器申请一个id段，当所持有的id段快耗尽时，再请求下一个段。

唯一性：全局中，根据分片规则，每个段分配器会持有不同的id段。例如下表中，每个段的大小是100，段分配器A持有分片0和分片1。对于每个分片而言，是一个个跳跃的id段。特殊的，当段大小为1时，段分配器就是改进前的id分配器。

均衡策略：均衡策略在id分配器来实现，简单的讲，是一个轮询策略。每个id分配器会轮询下游段分配器的状态，并选中id段的最小的那个，然后发起id段申请。由于不会加锁，当多个id分配器同时竞争时，可能会出现获取的id段不是全局最小的，可以附加一些策略来调优，比如再多获取一次，并本地排序。从整体上而言，id还是比较均衡的，可满足需求。

可用性：当发生网络划分时，本IDC的id分配器可以只连接本IDC的段分配器，成功的申请到id段。整个系统可容忍一定时间内不可协作，长时间不可协作的唯一危害是id增长不均衡，此时，就退化为改进前的方案。

多IDC环境的适应性：id分配器需要和所有IDC的段分配器交互，但是交互频率很低，同时和提供id分配服务是两个独立的阶段，不会受到多IDC网络环境的干扰。

参考：

http://www.360doc.com/content/14/0821/09/1073512_403507606.shtml / / 分布式

http://baike.baidu.com/link?url=2bHLwEltRpMbdfwVuUUGVUwoA2ZRJyroi8BJAgBlyoqjPjcfzYWE2oSCpMmqhip88frx8F1OpjypLratAHQuBa / / 负载均衡

http://server.zzidc.com/fwqjs/796.html / / 负载均衡

http://wenku.baidu.com/link?url=5HDH0VaJY0ELsrBk6lI7PN44AaNMVXnfxXa4BFSKJHuUc-xmjeCrQflY2oN28dXH8-2fdJHnzDiLuw6Bx0x_GgYE6-1mDPgHiMRDJSE2ZUa / / 分布式存储

http://www.searchsoa.com.cn/showcontent_37453.htm / / 缓存

http://www.jdon.com/37625 / / ACID&BASE

http://baidutech.blog.51cto.com/4114344/1033677/ / / ID分配