讨论课解决方案2

冯夏令 1352920

分布式系统比单机系统有着更高的可用性和更高的吞吐量，但是也带来了一系列挑战。比如在分布式系统中，会遇到很多在单机系统中不会遇到的问题，如一致性问题等。在参考了业界的方法之后，我提出了如下的解决方案。

一致性问题：

· 协议：

在分布式系统中，节点间数据同步问题是一个很难以解决的问题，典型的解决方法有两阶段递交协议、三阶段递交协议、Paxos协议、Totem协议、Gossip协议等。其中Gossip协议相对简单，通信性能也较好，分布式非关系型数据库Cassandra就是通过Gossip协议达到数据一致性的。

在Cassandra对Gossip的实现中，Cassandra通过Gossip协议维护集群的状态。通过Gossip，每个节点都能知道集群中包含哪些节点，以及这些节点的状态，这使得Cassandra集群中的任何一个节点都可以完成任意key的路由，任意一个节点不可用都不会造成灾难性的后果，即使有节点宕机或者有新节点的加入，在经过一段时间后，这些节点的状态也会与其他节点达成一致，所以说Gossip天然具有分布式容错的优点。

Gossip协议模拟了人类中传播谣言的行为而来。谣言的传播首先要传播谣言就要有种子节点。种子节点每秒都会随机向其他节点发送自己所拥有的节点列表，以及需要传播的消息。任何新加入的节点，就在这种传播方式下很快地被全网所知道。这个协议的神奇就在于它从设计开始就没想到信息一定要传递给所有的节点，但是随着时间的增长，在最终的某一时刻，全网会得到相同的信息。

Cassandra内部Gossip实现的原理如下：

1、随机取一个当前活着的节点，并向它发送同步请求

2、向随机一台不可达的机器发送同步请求

3、如果第一步中所选择的节点不是seed，或者当前活着的节点数少于seed数，则向随意一台seed发送同步请求

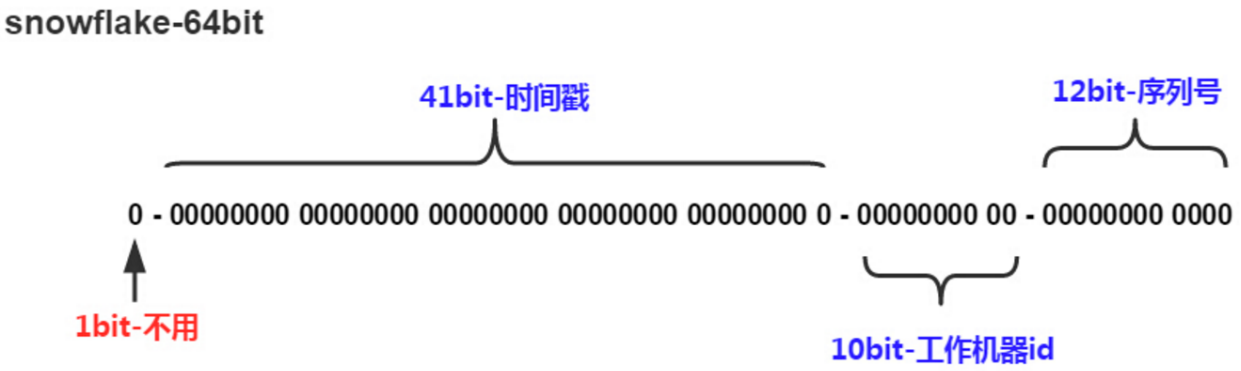
通过以上的步骤，集群中的各个节点的信息会达到最终一致性。

可靠性和健壮性：

· ID分配：

在大型分布式系统中，为了满足每秒上千上万条消息的请求，每条消息都必须分配一个唯一的ID，因此分布式系统的每一个节点产生的ID必须不同。在此问题上我们可以参考Twitter的解决方案。

Twitter使用的是snowflake算法，该算法的核心是将时间戳，工作机器id和序列号组合在一起，如下图：



其中工作机器id可以用使用机器级的MAC地址进行标识，也可以使用进程级别的IP+PATH区分工作进程。

· 消息队列：

消息队列中间件是分布式系统中重要的组件，主要解决应用耦合，异步消息，流量削锋等问题。常见的消息队列有ActiveMQ，RabbitMQ等。

性能：

· 负载均衡：

负载均衡，是将负载（工作任务，访问请求）进行平衡、分摊到多个节点上进行执行。

负载均衡解决了一下问题：

1.解决并发压力，提高应用处理性能（增加吞吐量，加强网络处理能力）；

2.提供故障转移，实现高可用；

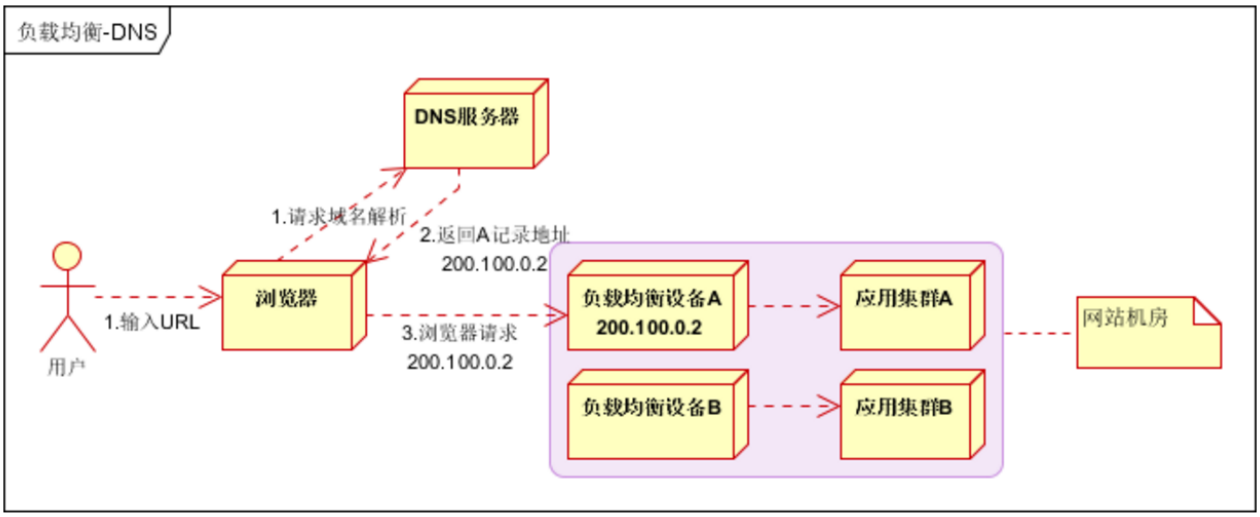
3.通过添加或减少服务器数量，提供网站伸缩性（扩展性）；

4.安全防护；（负载均衡设备上做一些过滤，黑白名单等处理）

负载均衡的技术主要有DNS负载均衡，IP负载均衡，链路层负载均衡等。

DNS负载均衡：

最早的负载均衡技术，利用域名解析实现负载均衡，在DNS服务器，配置多个A记录，这些A记录对应的服务器构成集群。但是该技术解析DNS的时间较长、控制权掌控在域名商里，可扩展性差、并且维护性差。现在一般DNS作为第一级负载均衡，从DNS服务器请求回来的记录对应着内部负载均衡的IP地址，通过内部负载均衡将请求分发到真实的Web服务器上，如下图：



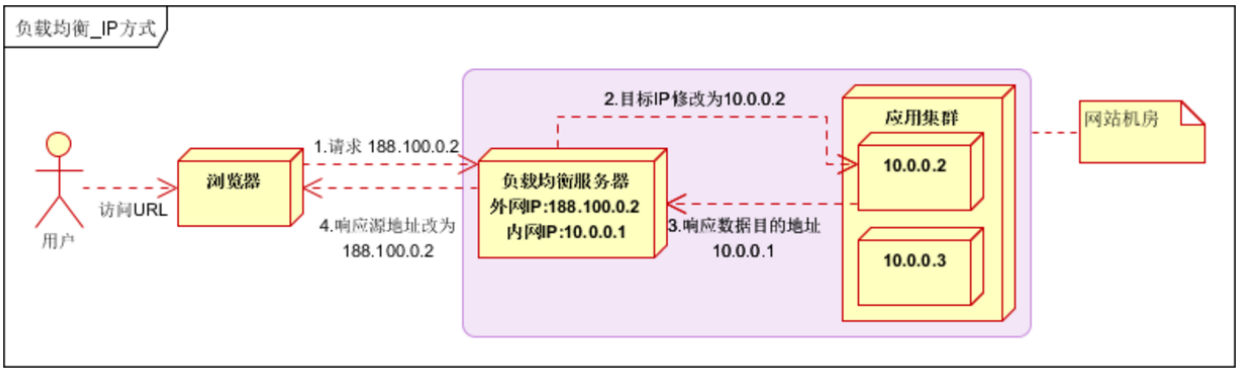
IP负载均衡：

IP负载均衡技术是在网络层通过修改请求目标地址进行负载均衡。

用户请求数据包，到达负载均衡服务器后，负载均衡服务器在操作系统内核进程获取网络数据包，根据负载均衡算法得到一台真实服务器地址，然后将请求目的地址修改为，获得的真实ip地址，不需要经过用户进程处理。

真实服务器处理完成后，响应数据包回到负载均衡服务器，负载均衡服务器，再将数据包源地址修改为自身的ip地址，发送给用户浏览器。

IP负载均衡的过程如下图：

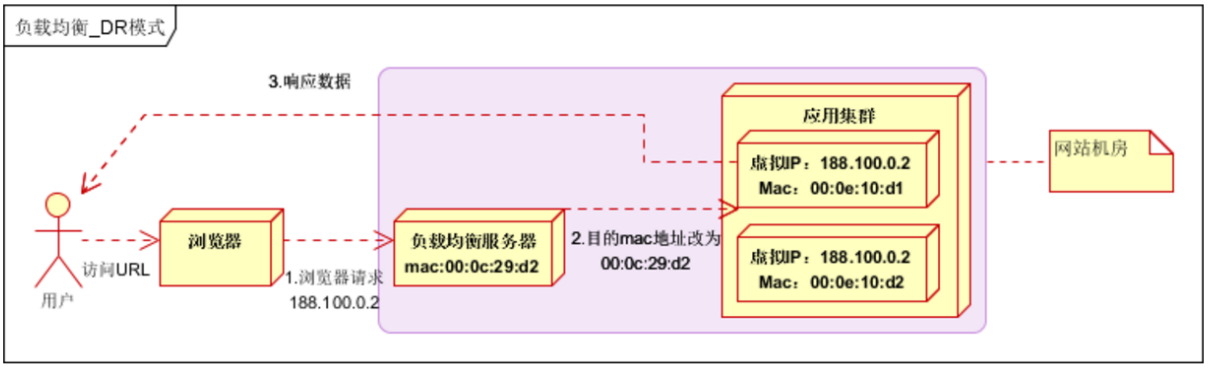


链路层负载均衡：

在通信协议的数据链路层修改mac地址，进行负载均衡。

数据分发时，不修改ip地址，指修改目标mac地址，配置真实物理服务器集群所有机器虚拟ip和负载均衡服务器ip地址一致，达到不修改数据包的源地址和目标地址，进行数据分发的目的。

实际处理服务器ip和数据请求目的ip一致，不需要经过负载均衡服务器进行地址转换，可将响应数据包直接返回给用户浏览器，避免负载均衡服务器网卡带宽成为瓶颈。



链路负载均衡的工作过程如下图：

安全：

在消息传输过程中，为了保证消息的安全性，我们需要对消息进行加密，常用的信息加密算法有DES加密算法、AES加密算法、RSA加密算法。

DES加密算法

DES加密算法是一种分组密码，以64位为分组对数据加密，它的密钥长度是56位，加密解密用同一算法。DES加密算法是对密钥进行保密，而公开算法，包括加密和解密算法。这样，只有掌握了和发送方相同密钥的人才能解读由DES加密算法加密的密文数据。因此，破译DES加密算法实际上就是搜索密钥的编码。对于56位长度的密钥来说，如果用穷举法来进行搜索的话，其运算次数为256。

随着计算机系统能力的不断发展，DES的安全性比它刚出现时会弱得多，然而从非关键性质的实际出发，仍可以认为它是足够的。不过，DES现在仅用于旧系统的鉴定，而更多地选择新的加密标准。

AES加密算法

AES加密算法是密码学中的高级加密标准，该加密算法采用对称分组密码体制，密钥长度的最少支持为128、192、256，分组长度128位，算法应易于各种硬件和软件实现。这种加密算法是美国联邦政府采用的区块加密标准，这个标准用来替代原先的DES，已经被多方分析且广为全世界所使用。

AES加密算法被设计为支持128／192／256位（/32=nb)数据块大小（即分组长度）；支持128／192／256位（/32=nk)密码长度，，在10进制里，对应34×1038、62×1057、1.1×1077个密钥。

RSA加密算法

RSA加密算法是目前最有影响力的公钥加密算法，并且被普遍认为是目前最优秀的公钥方案之一。RSA是第一个能同时用于加密和数宇签名的算法，它能够抵抗到目前为止已知的所有密码攻击，已被ISO推荐为公钥数据加密标准。RSA加密算法基于一个十分简单的数论事实：将两个大素数相乘十分容易，但那时想要，但那时想要对其乘积进行因式分解却极其困难，因此可以将乘积公开作为加密密钥。

参考资料：

http://darktea.github.io/notes/2013/12/08/Unique-ID

https://en.wikipedia.org/wiki/Gossip_protocol

http://blog.arganzheng.me/posts/thinking-in-distributed-systems.html

http://blog.csdn.net/cloudresearch/article/details/23127985

http://hellosure.github.io/%E5%88%86%E5%B8%83%E5%BC%8F/2015/03/28/distributed-system/

http://www.infoq.com/cn/articles/distributed-data-stores-for-mere-mortals

https://yq.aliyun.com/articles/7534

https://www.ustack.com/blog/distributedsystems_networksdivision/

http://itsoul.iteye.com/blog/777212

http://blog.jobbole.com/97957/