Spring MVC

第一章 网站架构及其演变过程

人

客户端

服务器程序

数据库

服务端

CS结构的程序已经可以完成网络通信了，不过使用起来还是比较麻烦的，首先软件提供商需要同时开发客户端和服务端两套软件；

网站程序

数据库

服务器程序

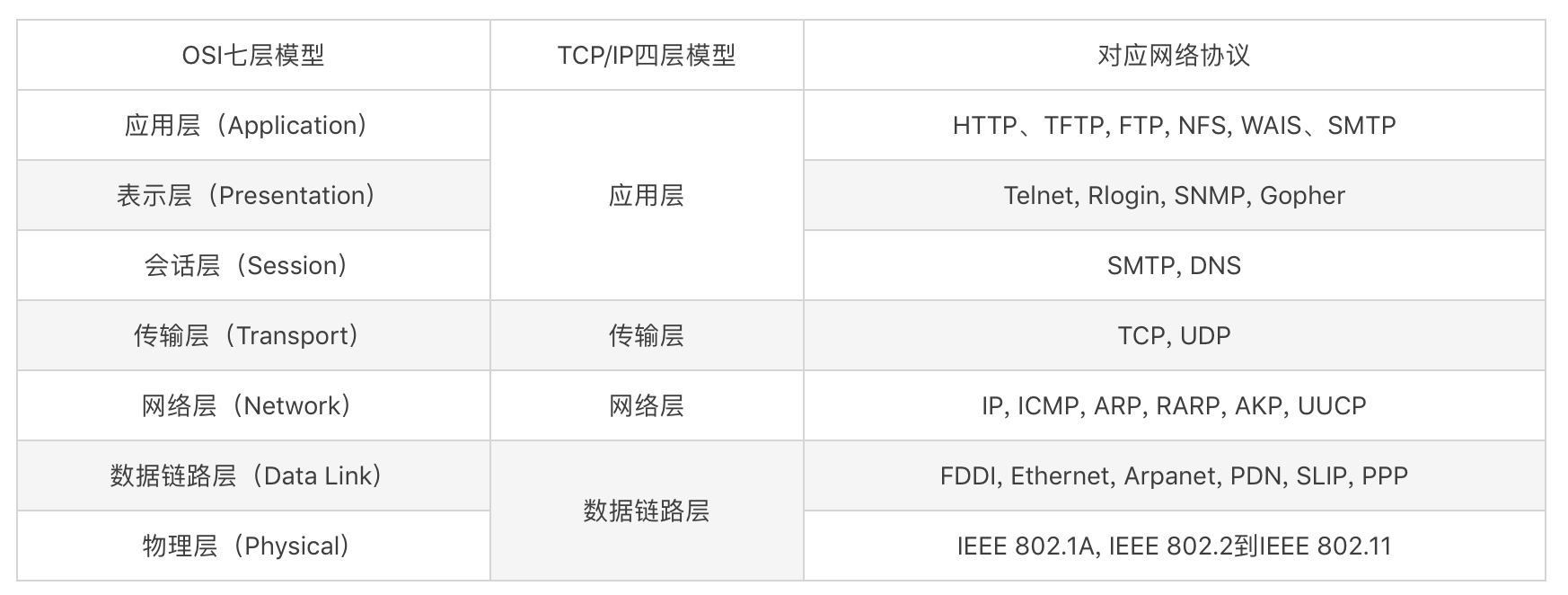
服务端

浏览器

BS结构除了提供同意的客户端，还根据相应协议和标准提供通用的服务器程序，服务器程序统一处理数据连接，封装和解析等工作。

软件的三大类型：单机类型，CS类型和BS类型。

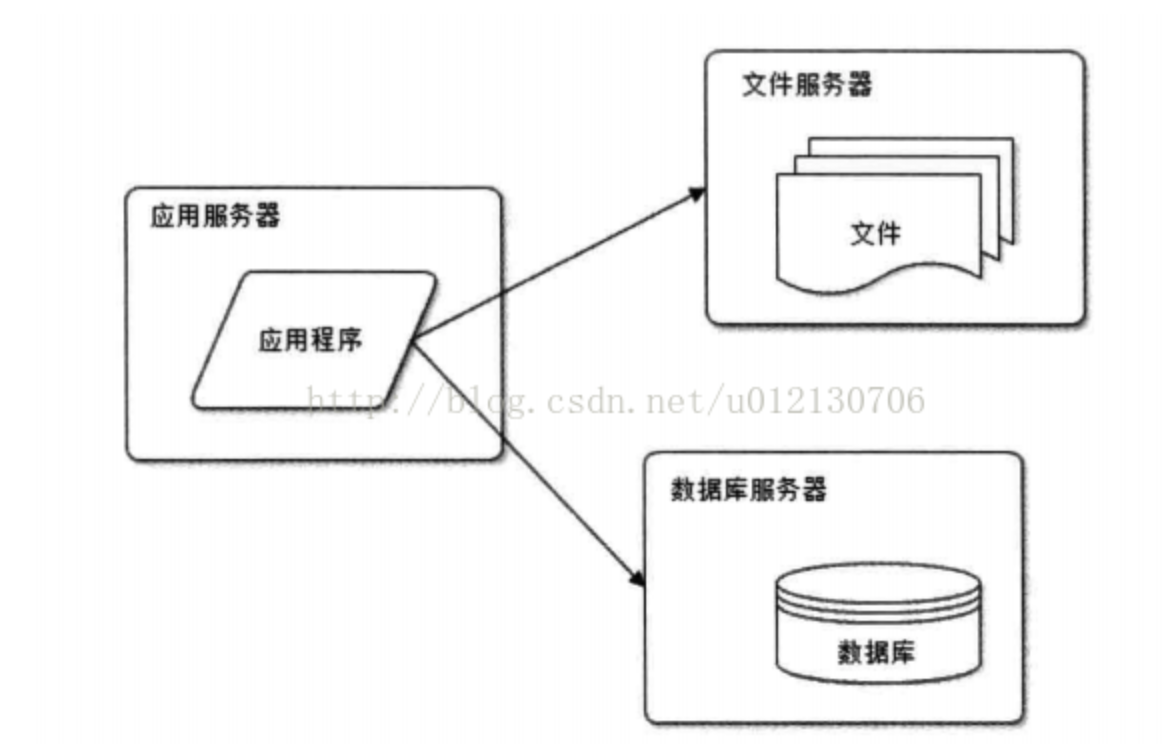
BS结构网络传输的分解方式有两种：一种是标准的OSI参考类型，另一种是TCP/IP参考类型。

网络接入层：将需要相互连接的节点接入网络中，从而为数据传输提供条件。

网络层：找到要传输数据的目标节点。IP

传输层：实际传输数据。TCP

应用层：使用接收到的数据。HTTP



应用和数据分离结构图

缓存和页面静态化

数据量大这个问题最直接的解决方案就是使用缓存，缓存就是将从数据库中获取的结果暂时保存起来，在下次使用的时候无须重新到数据库中获取，这样可以大大降低数据库的压力。

缓存的使用方式可以分为通过程序直接保存到内存中和使用缓存框架两种方式。程序直接操作主要是使用Map，尤其是ConcurrentHashMapp，而常用的缓存框架有Ehcache，Memcache和Redis等。缓存使用过程中最重要的问题是什么时候创建缓存和缓存的失效机制。缓存可以在第一次获取的时候创建也可以在程序启动和缓存失效之后创建，缓存的失效可以定期失效，也可以在数据发生变化的时候失效，如果按数据发生变化让缓存失效，还可以分粗类度失效和细类度失效。、

页面静态化，它的原理上跟缓存非常相似，缓存是将从数据库中获取到的数据(当然也可以是别的任何可以序列化的东西)保存起来，而页面静态化是将程序最后生成的页面保存起来，使用页面静态化后就不需要每次调用都重新生成页面了。这样不但不需要查询数据库，而且连应用程序处理都省了，所以页面静态化同时对数据量大和并发量高两大问题都有好处。

页面静态化可以在程序中使用模版技术生成，如常用的Freemarker和Velocity都可以根据模版生成静态页面。

数据库优化

表结构优化，SQL语句优化，分区和分表，索引优化，使用存储过程代替直接操作等。另外有时候合理的使用zhui余也能获得非常好的效果。

1）表结构优化：数据库中最基础也是最重要的，如果表结构优化得不合理，就可能导致严重的性能问题。

2）SQl语句优化：基础的SQL优化是语法层面的优化，不过最重要的是处理逻辑的优化，这也需要根据实际情况具体处理，而且要和索引缓存等配合使用。不过SQL优化有一个通用的做法就是：首先要将涉及大数据的业务SQL语句执行时间详细记录下来，其次通过仔细分析日志找出需要优化的语句和其中的问题，然后再有的放矢地优化，而不是不分重点地对每条语句都花同样的时间和精力优化。

3）分区：分区就是将一张表中的数据按照一定的规则分到不同的区来保存，这样在查询数据时如果数据的范围在同一个区内那么可以只对一个区的数据进行操作，这样操作的数据更少，数据更快，而且这种方法对程序是透明的，程序不需要做任何改动。

4）分表：是将一个表中的不同类型的字段分到不同的表中保存，这么做最直接的好处就是增删改数据的时候锁定的范围减少了，没被锁定的表中的数据不受影响，如果一个表的操作频率很高，在增删改其中一部分字段数据的同时另一部分字段也可能被操作，而且用不到被增删改的字段，那么就可以把不同类型的字段分别保存到不同的表中，这样可以减少操作时锁定数据的范围。不过这样分表之后，如果需要查询完整的数据就的使用多表操作了。

5）索引优化：索引的大致原理是在数据变化的时候就预先按指定字段的顺序排列后保存到一个类似表的结构中，这样在查找索引字段为条件的记录时就可以很快地从索引中找到对应记录的指针并从表中获取到记录，这样速度就快多了。不过索引也是一把双刃剑，它在提高查询速度的同时也降低了增删改的速度，因为每次数据的变化都需要更新相应的索引。

6）使用存储过程代替直接操作：因为存储过程只需要编译一次，而且可以在一个存储过程里面做一些复杂的操作。

分离活跃数据

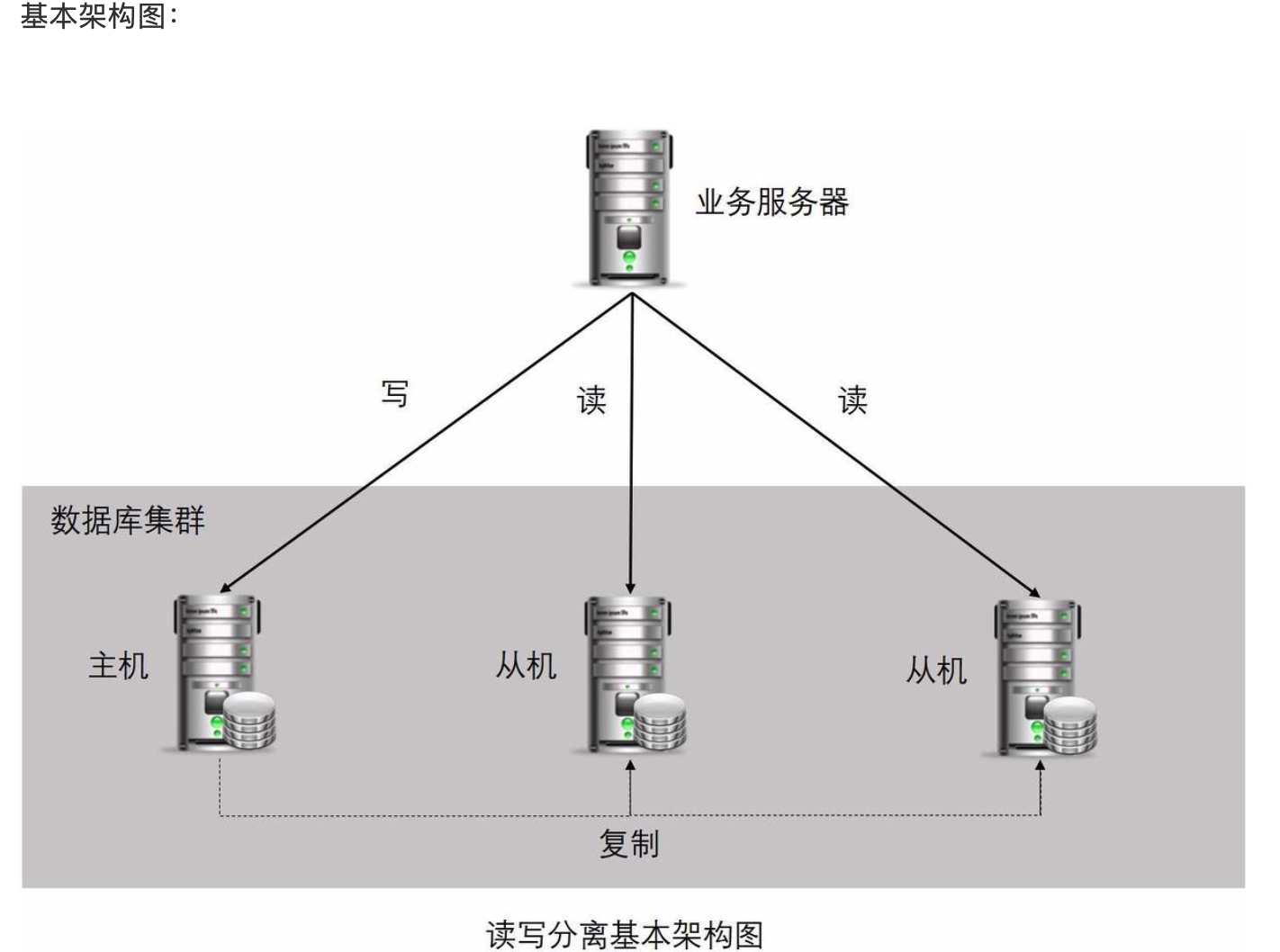
可以将活跃数据单独保存起来从而提高处理效率。

批量读取和延迟修改

批量读取和延迟修改的原理是通过减少操作的次数来提高效率，如果使用得恰当，效率将会呈数量级提升。延迟修改主要针对高并发且频繁修改的数据，如一些统计数据。这种情况可以先将需要修改的数据暂时保存到缓存中，然后定时将缓存中的数据保存到数据库中，

读写分离

对数据库进行集群，这样可以在高并发的情况下将数据库的操作分配到多个数据库服务器去处理从而降低单台服务器的压力，不过由于数据库的特殊性——每台服务器所保存的数据都需要一致，所以数据同步就成了数据库集群中最核心的问题。如果多台服务器都可以写数据那么数据同步将变得非常复杂，所以一般情况下是将写操作交给专门的一台服务器处理，这台专门负责写的服务器叫做主服务器。当主服务器写入数据后从底层同步到别的服务器，读数据的时候到从服务器读取，从服务器可以有多台，这样就可以实现读写分离，并且将读请求分配到多个服务器处理。主服务器向从服务器同步数据时，如果从服务器数量多，那么可以让主服务器先向其中一部分从服务器同步数据，第一部分从服务器接收到数据后再向另外一部分同步，



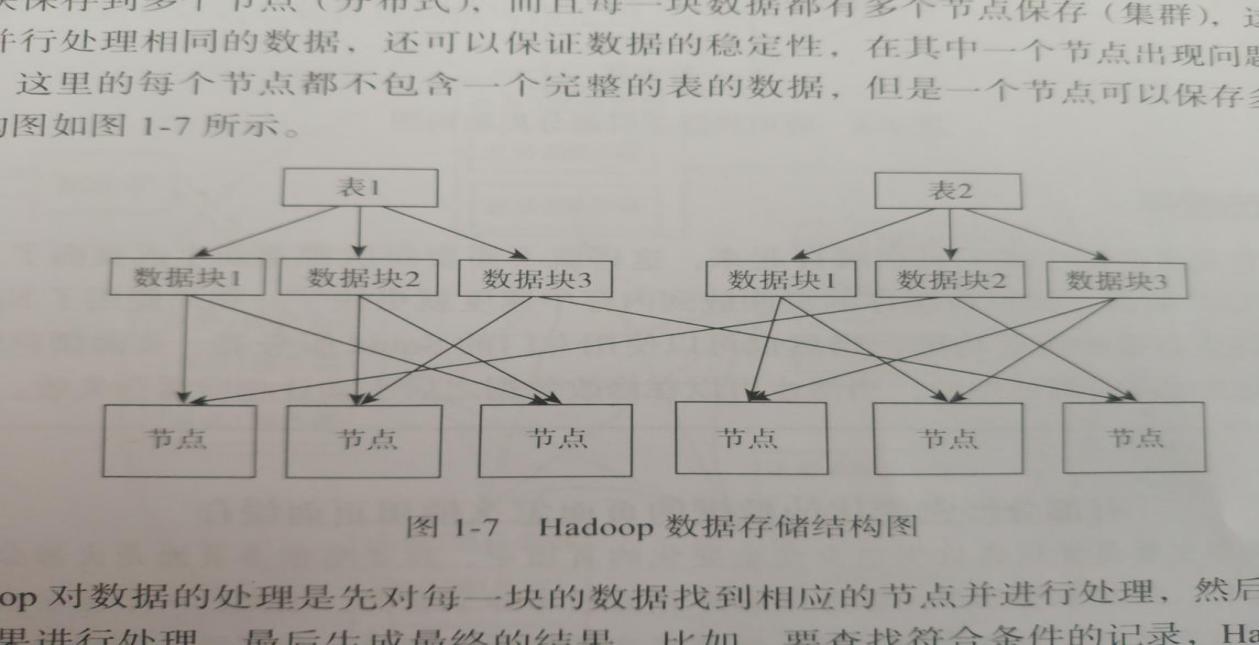
简单的数据同步方式可以采用数据库的热备份功能，不过读取到的数据可能会存在一定的滞后性，高级的方式需要使用专门的软硬件配合。另外既然是集群就涉及负载均衡问题，负载均衡和读写分离的操作一般采用专门程序处理，而且对应系统来说是透明的。

分布式数据库

分布式数据库是将不同的表存放到不同的数据库中然后再放到不同的服务器。这样在处理请求时，如果需要调用多个表，则可以让多台服务器同时处理，从而提高处理速度。

NoSQL和Hadoop

一般使用的数据库(SQL数据库)都是需要现将表的结构定义出来，一个表有几个字段，每个字段各是什么类型，然后才能往里面按照相应的类型保存数据，而且按照数据库范式的规定，一个字段只能保存单一的信息，不可以包括多层内容，这就对使用的灵活性带来了很大的限制，NoSQL就是突破这些跳跳框框，可以非常灵活的进行操作，另外NoSQL通过多个块存储数据的特点，其操作大数据的速度非常快。Hadoop是专门针对大数据处理的一套框架，Hadoop是将同一个表中的数据分成多块保存到多个节点，而且每一块数据都有多个节点保存，这里集群除了可以并行处理相同的数据，还可以保证数据的稳定性，其中一个节点出现问题后数据不会丢失，这里的每个节点都不包含一个完整的表的数据，但是一个节点可以保存多个表的数据。



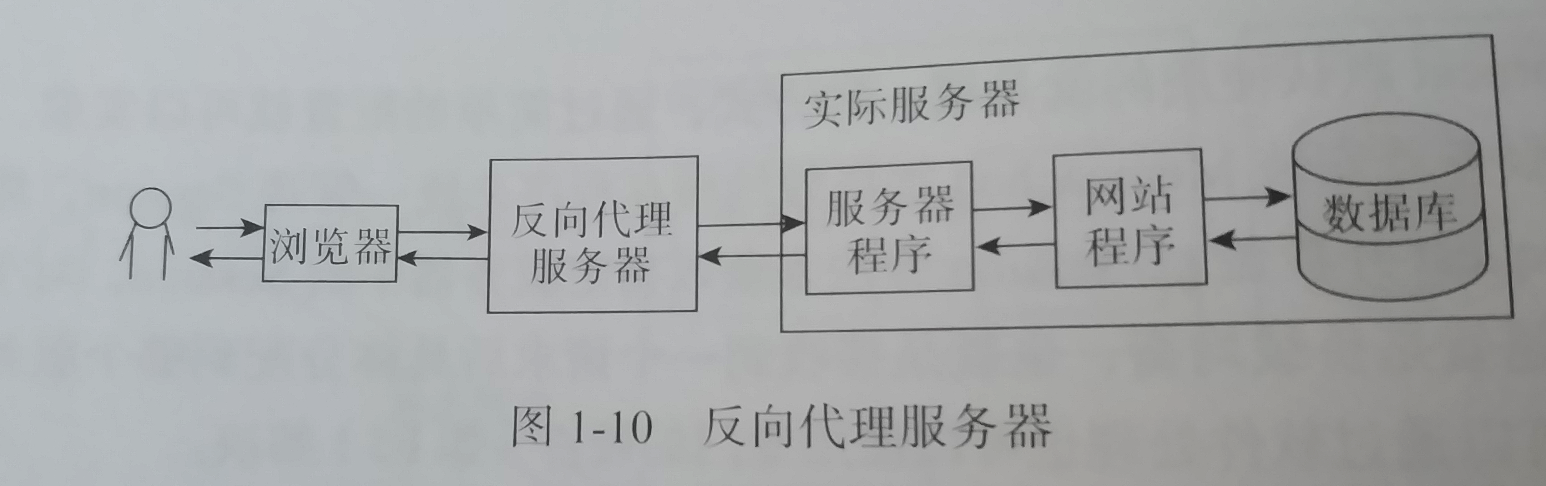
Hadoop对数据的处理是先对每一块的数据找到相应的节点并进行处理，然后在对每一个处理的结果进行处理，最后生成最终的结果。比如，要查找符合条件的记录，Hadoop的处理方式是先找到每一块符合条件的记录，然后再将所有获取到的结果合并到一起，这样就可以将同一个查询分到多个服务器处理，处理的速度也很快。

页面缓存：将应用生成的页面缓存起来，对于这个问题可以先生成静态页面然后使用Ajax来读取并修改相应的数据。

集群与分布式：集群和分布式处理都是使用多台服务器进行处理的，集群是每台服务器都具有相同的功能，处理请求时调用哪台服务器都可以，主要起分流的作用，分布式是将不同的业务放到不同的服务器中，处理一个请求可能需要用到多台服务器，这样就可以提高一个请求的处理速度，而且集群和分布式也可以同时使用。

集群有两个方式：一种是静态资源集群。另一种是应用程序集群，静态资源集群比较简单，而应用程序集群就有点复杂。因为应用程序在处理过程中可能会使用到一些缓存的数据，如果集群就需要同步这些数据，其中最重要的就是Session，Session同步也是应用程序集群中非常核心的一个问题。Session同步有两种处理方式，一种是在Session发生变化后自动同步到其他服务器，另外一种方式是用一个程序统一管理Session。所有集群的服务都使用相同的一个Session。

反向代理：指的是客户端直接访问的服务器并不真正提供服务，它从别的服务器获取资源然后将结果返回给用户的。



反向代理服务器可以和实际处理请求的服务器在同一台主机上，而且一台反向代理服务器也可以访问多台实际处理请求的服务器。反向代理服务器主要有三个作用：1）可以作为前端服务器跟实际处理请求的服务器集成。 2）可以用作负载均衡； 3）转发请求，比如，可以将不同类型的资源请求转发到不同的服务器去处理，可以将动态资源转发到Tomcat，Php等动态程序而将图片等静态资源的请求转发到静态资源的服务器，另外也可以在url地址结构发生变化后将新地址转发到原来的旧地址上。

CDN其实是一种特殊的集群页面缓存服务器，它和普通集群的多台页面缓存服务器比主要是它存放的位置和分配请求的方式有点特殊。CDN的服务器是分布在全国各地的，当接收到用户的请求后会将请求分配到最合适的CDN服务器节点获取数据。CDN的每个节点其实就是一个页面缓存服务器，如果没有请求资源的缓存就会从主服务器获取，否则直接返回缓存的页面。