buffer cache

buffer: 一般 用于写操作， 写缓冲

cache: 一般 用于读操作， 读缓存。提高数据读取速度

CPU： L1 L2 L3 ： CPU cache 位于内存与CPU之间。 CPU把内存中的数据读取到cache上，然后下次从cache读取，这样比读内存更快。

buffer 缓冲区，用于提高速度不同传输速度。 cpu把数据写到内存的磁盘缓存区 然后系统启动一个进程（pd flash）把内存的数据写到硬盘。

都是解决速度不一致的问题。 有的太快，有的太慢。他们之前进行IO操作时，需要用buffer cache来提高IO操作，首先cache来读缓存，buffer写缓冲，写到离目的地最近的地方，然后再写到目的地。

（DMA, NUMA）

服务器的内存不能乱插，是有关系的。

cache的重点：

存放位置： 客户端（浏览器缓存）

内存（本机内存，远程服务器内存）

硬盘（本地硬盘（ssd, sas, stat,ide），远程服务器硬盘） 西部数据4T stat红盘1299

过期时间： TTL：

强制过期： 提前过期

cache命中率：

云同步，云备份，是否需要cdn。不需要。

alidns.com

缓存协商： 浏览器和web服务器进行缓存的一个对话。

缓存方法有：1: last\_Modified 最后修改时间，web服务器通过stat系统调用 来获取系统中静态文件在硬盘上的最后修改时间，然后在返回静态文件的时候会在http响应头部中把最后修改时间附带上。

about:cache 火狐查看缓存

http请求，返回的是304，然后使用本地缓存 （得是有些文件内容不变，最后修改时间经常变，就得重新再接收。 多个web服务器最后修改时间不一样，当第二次访问的是另一台web服务器的时候）

2: Etag: MD5

3: expires: 过期时间

链接输入回车后： 所有没有过期的直接使用缓存，所有缓存机制均生效

f5刷新后： expires 不受影响，但是别的都会受影响，会发送缓存协商，

ctrl + f5 ： 所有缓存都不生效，重新重新请求

cache-control （秒）相对本地时间最大缓存时间。

js 缓存1小时，但期间修改了后可以通过改名重新缓存，也可以在后面加参数。

CDN：

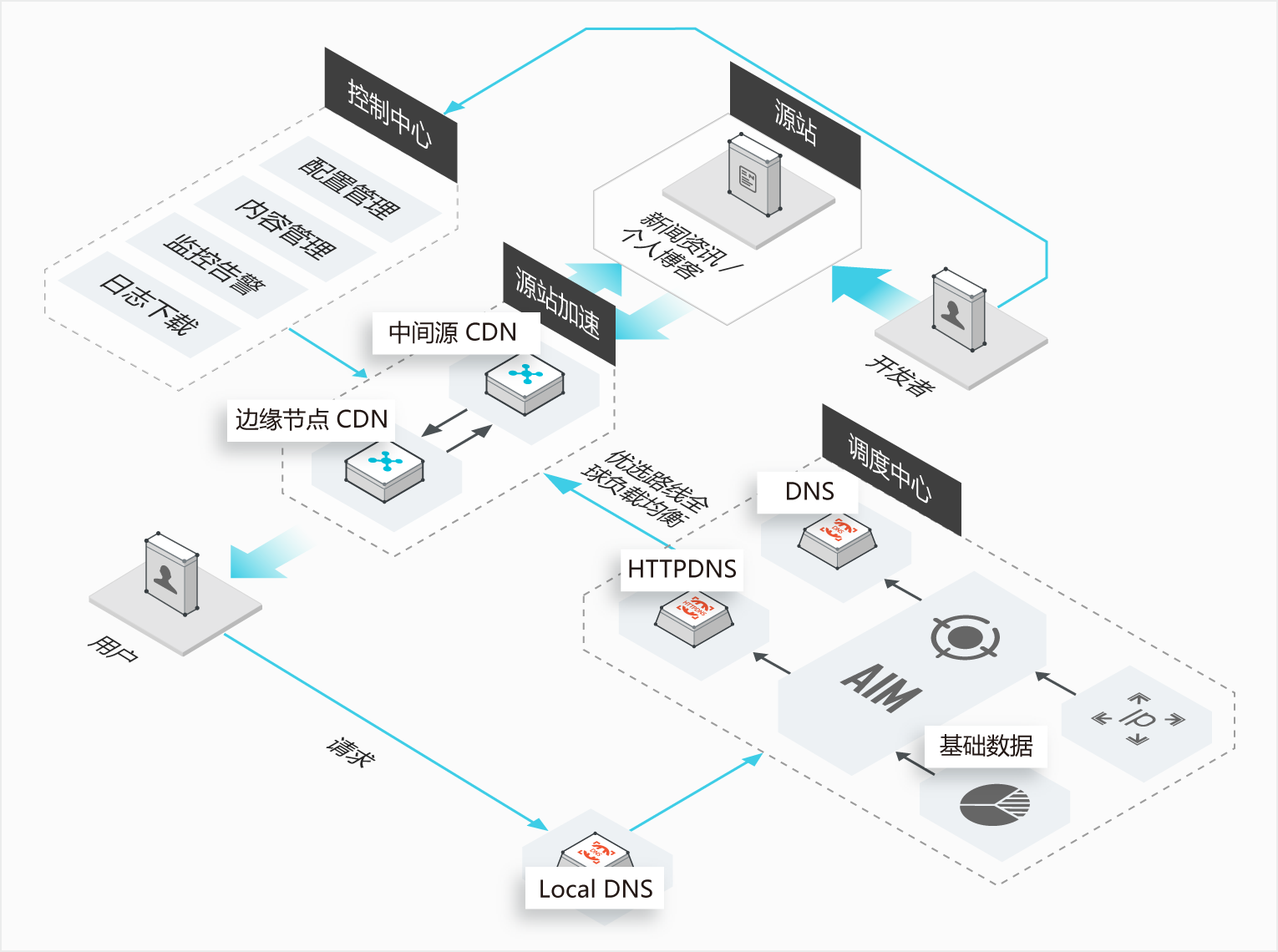
内容分发网络， 将源站内容发布到最接近用户的边缘节点，使用户可以就近取得所需内容，提高用户访问的响应速度和成功率，解决因分布，带宽，服务器能力的问题

目前服务器面临访问慢的问题

1.不同运营商之间互联的问题。BGP（贵）

2. 用户到服务器之间网络传输长，节点多，延迟大

3. 现在服务器带宽有限



边缘节点：stat 盘 4T 插满

CDN关键技术： 智能DNS。 GSLB + DDNS（通过local dns的IP来判断）+LB SLB 全局检测系统

调度系统： 智能DNS

监测系统： 监测各个节点的链路状态和健康情况

内容分发： 预缓存

过期刷新： 缓存刷新

数据储存： 文件的存储

用户配置平台： saltstack, ansible

计费系统

数据可视化，日志分析，日志下载。

防攻击， WAF

开放API

nginx 缓存

proxy\_temp\_path /data/cdn\_cache/proxy\_temp\_dir;

proxy\_cache\_path /data/cdn/cache/proxy\_cache\_dir level=1:2 keys\_zone=cache\_one:50m inactive=1d max\_size=1g;

proxy\_connect\_timeout 5;

proxy\_read\_timeout 60;

proxy\_send\_timeout 5;

proxy\_buffer\_size 16k;

proxy\_buffers 4 64k;

proxy\_busy\_buffers\_size 128k;

proxy\_temp\_file\_write\_size 128k;

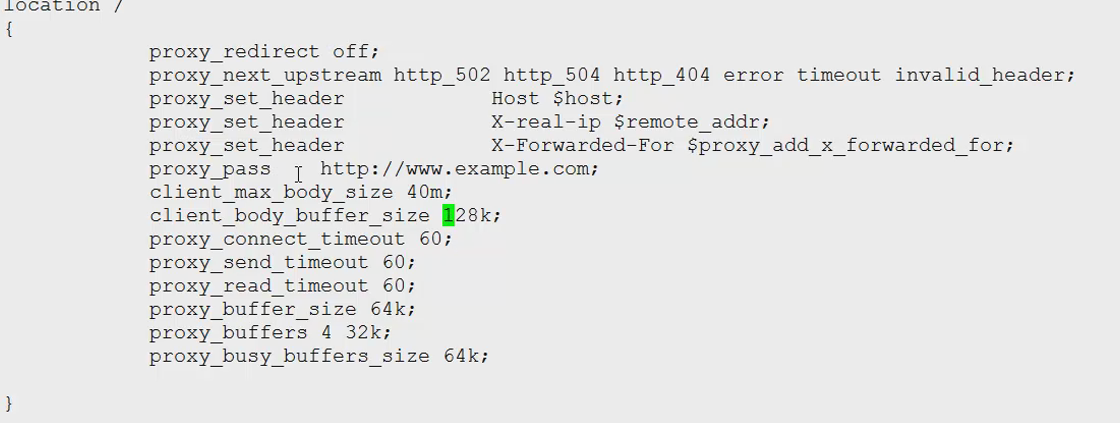
proxy\_next\_upstream error timeout invalid\_header http\_500 http\_502 http\_503 http\_404;

上面配置可以include 到nginx http配置里面。

level=1:2 keys\_zone=cache\_one:50m inactive=1d max\_size=1g;

缓存级别， 内存缓存50m 自动清空1天没访问的文件，最大硬盘缓存文件1g





jps 命令，可以查看tomcat进程

缓存刷新，其实就是删掉缓存