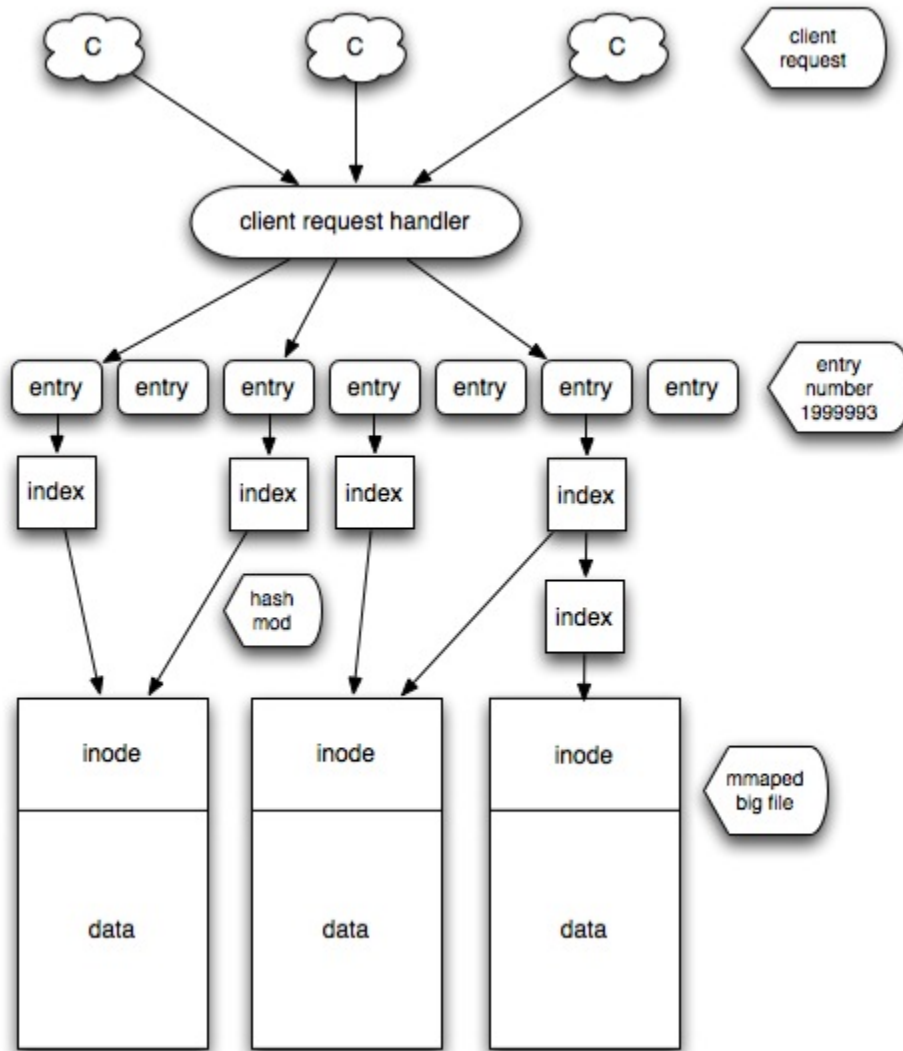


NCache 3.X 技术架构

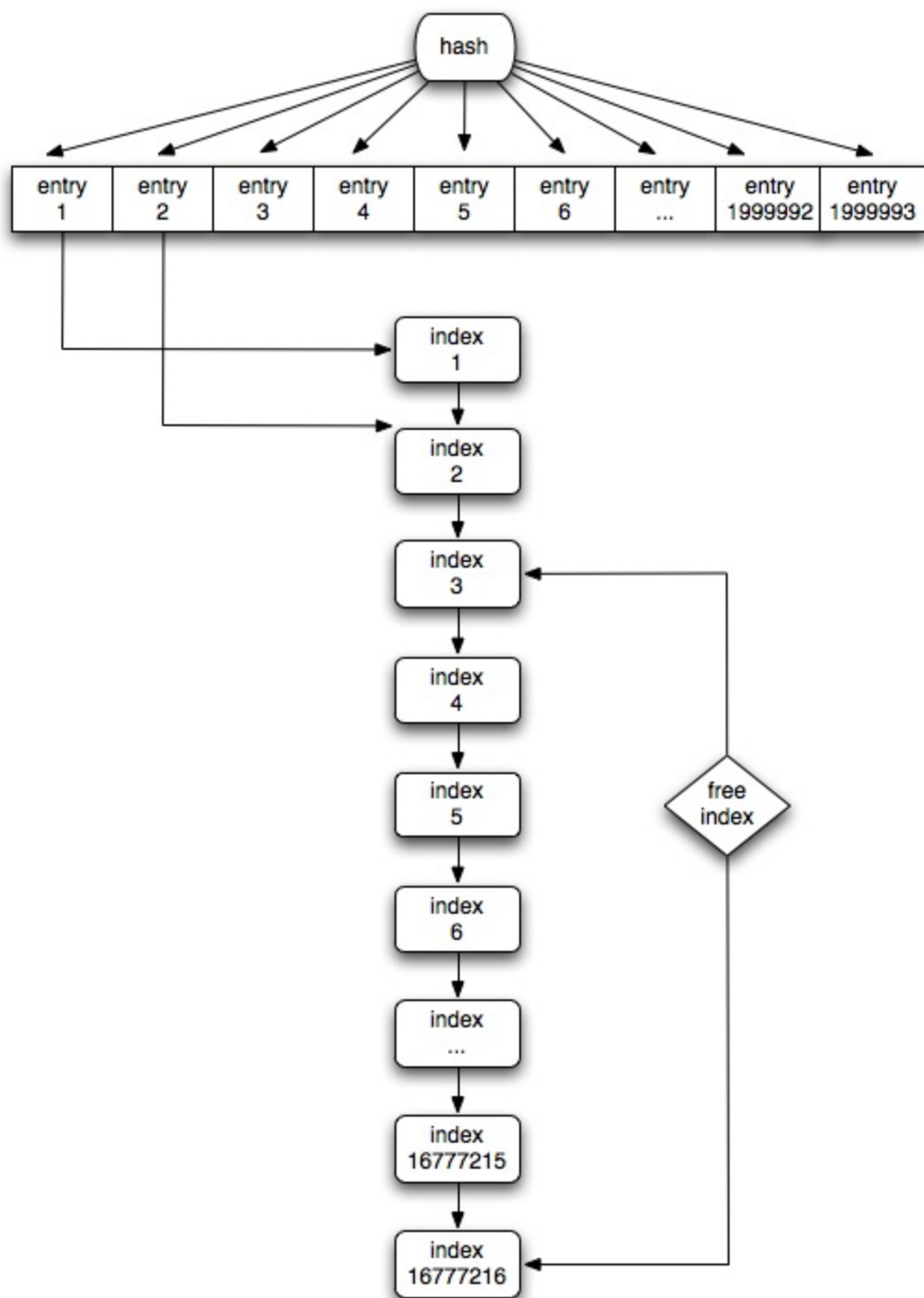
作者:庞帆 付根西

版权:新浪网技术中国有限公司

1.总体架构



2.HASH索引结构



1.所有的数据通过HASH计算出应该均匀的分布在哪个ENTRY上

2.每个ENTRY实际只存储一个INDEX的下标值和一个互斥锁,这样可以压缩HASH表第一层所占用的空间,避免HASH表的空间浪费,现在全部索引约占用700MB空间

3.每个索引中存储了与缓存数据相关的信息:何时过期(缓存的存储时间 + MAX-AGE的值,每次缓存命中时都会用当前时间与这个时间进行比对), 数据长度,是否为压缩数据,是否为UTF-8格式 等等

4.FREE INDEX 负责指向INDEX链表中 空闲的第一个索引块(用于分配) 和 INDEX链表中 空闲的最后一个索引块(用于回收) 这样可以将 分配与回收 使用不同的锁来控制 减少冲突 提升效率

3.MMAP存储模式

1.在64位系统上采用MMAP挂载一个大文件的形式用来作为数据的存储容器,这样做可以将数据的冷热切换,内存的冷热交换完全交给操作系统来负责,不需要再写利用LRU等内存交换算法进行数据的冷热置换

2.一个INODE对应一个DATA块(4096KB),存储DATA块的下标地址及DATA块内容的长度和一个指向NEXT INODE的值

3.当缓存命中需要使用SENDFILE吐出数据的时候,可以根据INODE中的信息来判断是否需要多次不连续的发送DATA块 或者是 一次连续的发送(因为每个DATA块的内容不一定能存满4096,由INODE中的长度来判断是否连续,如果连续就只发送一次,否则发送多次)

4.需要再进一步优化数据的发送,分配和回收环节,需要一套高效的算法,优化甚至完全避免存储和读取INODE信息,并可以将数据连续存储连续发送

5.FREE INODE 做法与 FREE INDEX 相同

4.多进程互斥锁

1.fcntl 文件记录锁

2.semaphores 信号量锁

3.pthread_mutex_t PROCESS_SHARED 可在进程间共享的互斥锁

尝试了多种锁机制,互斥锁性能最好,但在FREEBSD上没有实现进程间共享,信号量锁性能比互斥锁稍差,在FREEBSD中也没有实现进程间共享,记录锁性能最差,但在各种操作系统间的兼容性最好,在NCACHE LINUX 64位版本中使用互斥锁,放置在索引的ENTRY层,每个ENTRY一个锁,锁住各自的INDEX下拉链表,可以尽量避免冲突

5.分配与回收

1.INODE 与 DATA块的分配与回收:

根据后端传来的HTTP头中的 **CONTENT-LENGTH** 的大小计算出需要分配多少个块, 4096 对齐, 先分配**INODE**, 然后将数据写入相应**DATA**块中, 回收时只需要回收**INODE**到空闲链表即可

2.ENTRY 与 INDEX 的分配与回收:

ENTRY是一个固定大小的数组, 通过**HASH**进行定位, 下面链接的**INDEX**链表 是从一个**INDEX**空闲链表上进行分配和回收

6.数据的清理

1.数据冷热系数: 每一个缓存数据都可以根据热度系数的计算公式计算出某一时刻的热度

$(\text{数据被访问次数} / \text{数据被缓存的时长}) * 1000$

2.根据配置文件中设定的时间,启动清理进程,删除过期数据,删除被**PURGE**的数据,删除热度系数低于一个阈值的数据

3.**PURGE**删除在执行的时候只是简单的将被**PURGE**数据设置为过期,而不会主动删除磁盘中的数据,只有当数据被再次访问的时候,才会引起磁盘数据的回收和再分配,否则将被清理进程定期清除,这样可以节约不必要的磁盘消耗,提升**PURGE**删除性能

7.NGINX PROXY模块开发

http handler

header filter

body filter

proxy handler 参考 proxy module 和 memcache module

create_request, reinit_request, process_header, abort_request, finalize_request, rewrite_redirect

config 文件

NGINX 模块的config 文件实际上就是一个小的**SHELL**变量添加程序, 在执行**NGINX**的 configure --add-moudle=... 的时候 会被加入到真正的**NGINX**编译文件配置中

`ngx_addon_name=ngx_http_clean_module` 这个是模块的名称比如与模块中的 `module` 名称相同

`HTTP_FILTER_MODULES="$HTTP_FILTER_MODULES ngx_http_clean_module"` 这里是选择模块的属性,可以是`HTTP_MODULE` 也可以是`FILTER_MODULE` 区别就是加载和执行的顺序不一样,详细的顺序可以查看编译后的 `obj`目录下的 `ngx_module.c`文件

`NGX_ADDON_SRCS="$NGX_ADDON_SRCS $ngx_addon_dir/ngx_http_clean_module.c"` 这里制定源文件的路径

`CORE_LIBS="$CORE_LIBS -lrt -lm -lcrypto"` 这里添加编译模块时需要的库