# vfs dcache hash分析

宝德技术研究院 李磊

在kernel中维护着一张hash表, 用来记录活动的dentry, 直接从高速缓存中读取dentry, 避免了从慢速的磁盘中读取, 此hash表在dcache.c中定义

static struct hlist\_head \*dentry\_hashtable \_\_read\_mostly;

\_\_read\_mostly宏用来提示系统此处会被经常读取，系统加载时需要将其读入cache中, dentry\_hashtable其实可以认为是一个数组,该数组的每个成员都是一个list\_head结构, list\_head串起hash值相同的dentry, hash表如图1



图1

相关数据定义如下

struct hlist\_head {

struct hlist\_node \*first;

};

struct hlist\_node {

struct hlist\_node \*next, \*\*pprev;

};

dentry\_hashtable中保存了一个指针, 指向下一个链表元素, 而链表元素具有两个指针, next及pprev, 其中next指向下一个链表元素, pprev指向前一个链表元素的next指针.

在kernel起来后就会调用dcache\_init将dentry\_hashtabl*e*初始化.

static void \_\_init dcache\_init(unsigned long mempages)

{

int loop;

/\*

\* A constructor could be added for stable state like the lists,

\* but it is probably not worth it because of the cache nature

\* of the dcache.

\*/

dentry\_cache = kmem\_cache\_create("dentry\_cache",

sizeof(struct dentry),

0,

(SLAB\_RECLAIM\_ACCOUNT|SLAB\_PANIC|

SLAB\_MEM\_SPREAD),

NULL, NULL);

set\_shrinker(DEFAULT\_SEEKS, shrink\_dcache\_memory);

/\* Hash may have been set up in dcache\_init\_early \*/

if (!hashdist)

return;

dentry\_hashtable =

alloc\_large\_system\_hash("Dentry cache",

sizeof(struct hlist\_head),

dhash\_entries,

13,

0,

&d\_hash\_shift,

&d\_hash\_mask,

0);

for (loop = 0; loop < (1 << d\_hash\_shift); loop++)

INIT\_HLIST\_HEAD(&dentry\_hashtable[loop]);

}

在查找一个dentry时通过函数d\_hash找到对应hash值的那个链表头

static inline struct hlist\_head \*d\_hash(struct dentry \*parent,

unsigned long hash)

{

hash += ((unsigned long) parent ^ GOLDEN\_RATIO\_PRIME) / L1\_CACHE\_BYTES;

hash = hash ^ ((hash ^ GOLDEN\_RATIO\_PRIME) >> D\_HASHBITS);

return dentry\_hashtable + (hash & D\_HASHMASK);

}