Subject: [RFC][PATCH 2/2] record cpu hint for future fput()
Posted by Dave Hansen on Wed, 21 Feb 2007 02:03:53 GMT
View Forum Message <> Reply to Message

Most mnt_want/drop_write() pairs are really close in
the code; they aren't held for very long. So, in
practice is hard to get bounced between cpus between
when you mnt_want_write() and mnt_drop_write().

The exception to this is the pair in may_open() and
__fput(). Between those two it is pretty common to
move between cpus. During a kernel compile of around
900 files on a 4-way, I saw it happen ~400 times.

This patch assumes that the cpu doing the allocating of
the 'struct file' is also the one doing the
mnt_want_write(). It is OK that it is wrong sometimes,
it just means that we regress back to the
spinlock-protected search of all of the cpus' counts.

My kernel compile from before went from 400 misses
during a compile to just 20 with this patch.

It might also be helpful to do the writer count per-node
which would _greatly_ decrease the number of migrations
that we see.

```
 lxc-dave/fs/file_table.c       |    2 +-
 lxc-dave/fs/namespace.c        |   17 +++++++++--------
 lxc-dave/fs/open.c             |    4 ++++
 lxc-dave/include/linux/fs.h    |    1 +
 lxc-dave/include/linux/mount.h |    1 +
 5 files changed, 16 insertions(+), 9 deletions(-)

diff -puN fs/file_table.c~fput-cpu fs/file_table.c
--- lxc/fs/file_table.c~fput-cpu 2007-02-20 17:59:48.000000000 -0800
+++ lxc-dave/fs/file_table.c 2007-02-20 17:59:49.000000000 -0800
@@ -215,7 +215,7 @@ void fastcall __fput(struct file *file)
  if (file->f_mode & FMODE_WRITE) {
   put_write_access(inode);
   if(!special_file(inode->i_mode))
-   mnt_drop_write(mnt);
+   __mnt_drop_write(mnt, file->f_write_cpu);
  }
  put_pid(file->f_owner.pid);
```

```
  file_kill(file);
diff -puN fs/namespace.c~fput-cpu fs/namespace.c
--- lxc/fs/namespace.c~fput-cpu 2007-02-20 17:59:48.000000000 -0800
+++ lxc-dave/fs/namespace.c 2007-02-20 18:00:27.000000000 -0800
@@ -89,8 +89,8 @@ struct vfsmount *alloc_vfsmnt(const char
 int mnt_want_write(struct vfsmount *mnt)
 {
  int ret = 0;
- atomic_t *cpu_writecount;
  int cpu = get_cpu();
+ atomic_t *cpu_writecount;
 retry:
  /*
   * Not strictly required, but quick and cheap
@@ -122,22 +122,17 @@ out:
  put_cpu();
  return ret;
 }
-EXPORT_SYMBOL_GPL(mnt_want_write);

-void mnt_drop_write(struct vfsmount *mnt)
+void __mnt_drop_write(struct vfsmount *mnt, int cpu)
 {
  static int miss = 0;
  atomic_t *cpu_writecount;
- int cpu;
  int borrowed = 0;
  int retries = 0;
 retry:
- cpu = get_cpu();
  cpu_writecount = per_cpu_ptr(mnt->writers, cpu);
- if (atomic_add_unless(cpu_writecount, -1, 0)) {
-  put_cpu();
+ if (atomic_add_unless(cpu_writecount, -1, 0))
   return;
- }
  spin_lock(&vfsmount_lock);
  /*
   * Holding the spinlock, and only checking cpus that
@@ -167,6 +162,12 @@ retry:
  if (!borrowed)
   goto retry;
 }
+void mnt_drop_write(struct vfsmount *mnt)
+{
+ int cpu = get_cpu();
+ __mnt_drop_write(mnt, cpu);
+ put_cpu();
```

```
+}
 EXPORT_SYMBOL_GPL(mnt_drop_write);

 /*
diff -puN fs/open.c~fput-cpu fs/open.c
--- lxc/fs/open.c~fput-cpu 2007-02-20 17:59:48.000000000 -0800
+++ lxc-dave/fs/open.c 2007-02-20 17:59:49.000000000 -0800
@@ -715,6 +715,10 @@ static struct file *__dentry_open(struct
  f->f_path.mnt = mnt;
  f->f_pos = 0;
  f->f_op = fops_get(inode->i_fop);
+ /*
+  * This is OK to race because it is just a hint
+  */
+ f->f_write_cpu = smp_processor_id();
  file_move(f, &inode->i_sb->s_files);

  if (!open && f->f_op)
diff -puN include/linux/fs.h~fput-cpu include/linux/fs.h
--- lxc/include/linux/fs.h~fput-cpu 2007-02-20 17:59:48.000000000 -0800
+++ lxc-dave/include/linux/fs.h 2007-02-20 17:59:49.000000000 -0800
@@ -766,6 +766,7 @@ struct file {
  struct fown_struct f_owner;
  unsigned int  f_uid, f_gid;
  struct file_ra_state f_ra;
+ int f_write_cpu;

  unsigned long  f_version;
 #ifdef CONFIG_SECURITY
diff -puN include/linux/mount.h~fput-cpu include/linux/mount.h
--- lxc/include/linux/mount.h~fput-cpu 2007-02-20 17:59:49.000000000 -0800
+++ lxc-dave/include/linux/mount.h 2007-02-20 17:59:49.000000000 -0800
@@ -94,6 +94,7 @@ static inline int __mnt_is_readonly(stru

 extern int mnt_want_write(struct vfsmount *mnt);
 extern void mnt_drop_write(struct vfsmount *mnt);
+extern void __mnt_drop_write(struct vfsmount *mnt, int cpu);
 extern void mntput_no_expire(struct vfsmount *mnt);
 extern void mnt_pin(struct vfsmount *mnt);
 extern void mnt_unpin(struct vfsmount *mnt);

_
```