This patch adds recall_lock hold to nfsd_forget_delegations() to protect
nfsd_process_n_delegations() call.
Also, looks like it would be better to collect delegations to some local
on-stack list, and then unhash collected list. This split allows to simplify
locking, because delegation traversing is protected by recall_lock, when
delegation unhash is protected by client_mutex.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
fs/nfsd/nfs4state.c |   32 ++++++++++++++++++++++++--------
1 files changed, 24 insertions(+), 8 deletions(-)
diff --git a/fs/nfsd/nfs4state.c b/fs/nfsd/nfs4state.c
index 21266c7..f004e61 100644
--- a/fs/nfsd/nfs4state.c
+++ b/fs/nfsd/nfs4state.c
@@ -2597,7 +2597,7 @@ out:
return ret;
}

-static void nfsd_break_one_deleg(struct nfs4_delegation *dp)
+static void nfsd_break_one_deleg(struct nfs4_delegation *dp, void *data)
{  
/* We're assuming the state code never drops its reference
 * without first removing the lease. Since we're in this lease
@@ -2633,7 +2633,7 @@ static void nfsd_break_deleg_cb(struct file_lock *fl)
 spin_lock(&recall_lock);
 fp->fi_had_conflict = true;
 list_for_each_entry(dp, &fp->fi_delegations, dl_profile)
-  nfsd_break_one_deleg(dp);
+  nfsd_break_one_deleg(dp, NULL);
 spin_unlock(&recall_lock);
 }

@@ -4694,7 +4694,7 @@ void nfsd_forget_openowners(u64 num)
 printk(KERN_INFO "NFSD: Forgot %d open owners", count);
 }

-int nfsd_process_n_delegations(u64 num, void (*deleg_func)(struct nfs4_delegation *))
+int nfsd_process_n_delegations(u64 num, void (*deleg_func)(struct nfs4_delegation *, void *)
, void *data)
{  
int i, count = 0;
struct nfs4_file *fp, *fnext;
for (i = 0; i < FILE_HASH_SIZE; i++) {
    list_for_each_entry_safe(fp, fnext, &file_hashtbl[i], fi_hash) {
        list_for_each_entry_safe(dp, dnext, &fp->fi_delegations, dl_perfile) {
            deleg_func(dp);
        }
    }
}

int nfsd_process_n_delegations(u64 num, void (*deleg_func)(struct
nfs4_delegation *dp)) {
    unsigned int count;
    struct nfs4_delegation *dp, *dnext;
    LIST_HEAD(unhash_list);
    nfs4_lock_state();
    count = nfsd_process_n_delegations(num, unhash_delegation);
    nfs4_unlock_state();
    spin_lock(&recall_lock);
    count = nfsd_process_n_delegations(num, collect_delegation, &unhash_list);
    spin_unlock(&recall_lock);
    printk(KERN_INFO "NFSD: Forgot %d delegations", count);
    nfs4_lock_state();
    list_for_each_entry_safe(dp, dnext, &unhash_list, dl_perfile)
    unhash_delegation(dp);
    nfs4_unlock_state();
}

void nfsd_recall_delegations(u64 num)
{
    nfs4_lock_state();
    count = nfsd_process_n_delegations(num, collect_delegation, &unhash_list);
    nfs4_unlock_state();
    spin_lock(&recall_lock);
    count = nfsd_process_n_delegations(num, unhash_delegation);
    spin_unlock(&recall_lock);
    printk(KERN_INFO "NFSD: Forgot %d delegations", count);
    nfs4_lock_state();
    list_for_each_entry_safe(dp, dnext, &unhash_list, dl_perfile)
    unhash_delegation(dp);
    nfs4_unlock_state();
}
nfs4_lock_state();
spin_lock(&recall_lock);
-count = nfsd_process_n_delegations(num, nfsd_break_one_deleg);
+count = nfsd_process_n_delegations(num, nfsd_break_one_deleg, NULL);
spin_unlock(&recall_lock);
nfs4_unlock_state();

Subject: Re: [PATCH] NFSd: fix locking in nfsd_forget_delegations()
Posted by bfields on Wed, 23 May 2012 21:31:58 GMT

On Tue, May 22, 2012 at 02:25:14PM +0400, Stanislav Kinsbursky wrote:
> This patch adds recall_lock hold to nfsd_forget_delegations() to protect
> nfsd_process_n_delegations() call.
> Also, looks like it would be better to collect delegations to some local
> on-stack list, and then unhash collected list. This split allows to simplify
> locking, because delegation traversing is protected by recall_lock, when
> delegation unhash is protected by client_mutex.

All this indirection is getting a little much.

How about replacing nfsd_process_n_delegations by something that always
does the list-move?:

void nfsd_forget_delegations(u64 num)
{
    unsigned int count;
    list_head victims;

    nfs4_lock_state();
    count = nfsd_get_n_delegations(num, &victims);
    list_for_each_entry_safe(..., &victims, ...)
        unhash_delegation();
    unlock_state();
}

ditto for recall_delegations, and take the recall_lock inside
nfsd_get_n_delegations?

Or something like that.

--b.

> Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
> ---
> fs/nfsd/nfs4state.c | 32 ++++++++++++++++++++++++--------
static void nfsd_break_one_deleg(struct nfs4_delegation *dp)
{
    /* We're assuming the state code never drops its reference
     * without first removing the lease. Since we're in this lease
    */
    static void nfsd_break_deleg_cb(struct file_lock *fl)
    { spin_lock(&recall_lock);
        fp->fi_had_conflict = true;
        list_for_each_entry(dp, &fp->fi_delegations, dl_perfile)
            nfsd_break_one_deleg(dp);
        spin_unlock(&recall_lock);
    }

    int i, count = 0;
    struct nfs4_file *fp, *fnext;
    int nfsd_process_n_delegations(u64 num, void (*deleg_func)(struct nfs4_delegation *))
    { for (i = 0; i < FILE_HASH_SIZE; i++) {
        list_for_each_entry_safe(dp, dnext, &fp->fi_delegations, dl_perfile) {
            deleg_func(dp);
        }
        if (++count == num)
            return count;
    }
    return count;
}
static void collect_delegation(struct nfs4_delegation *dp, void *data)
{
    struct list_head *list = data;
    list_move(&dp->dl_perfile, list);
}

void nfsd_forget_delegations(u64 num)
{
    unsigned int count;
    struct nfs4_delegation *dp, *dnext;
    LIST_HEAD(unhash_list);

    nfs4_lock_state();
    count = nfsd_process_n_delegations(num, unhash_delegation);
    nfs4_unlock_state();
    spin_lock(&recall_lock);
    count = nfsd_process_n_delegations(num, collect_delegation, &unhash_list);
    spin_unlock(&recall_lock);

    printk(KERN_INFO "NFSD: Forgot %d delegations", count);
}

void nfsd_recall_delegations(u64 num)
{
    nfs4_lock_state();
    spin_lock(&recall_lock);
    count = nfsd_process_n_delegations(num, nfsd_break_one_deleg);
    spin_unlock(&recall_lock);
    nfs4_unlock_state();
}

Subject: Re: [PATCH] NFSd: fix locking in nfsd_forget_delegations()
Posted by Stanislav Kinsbursky on Thu, 24 May 2012 04:41:35 GMT
View Forum Message <> Reply to Message
24.05.2012 01:31, J. Bruce Fields
> On Tue, May 22, 2012 at 02:25:14PM +0400, Stanislav Kinsbursky wrote:
>> This patch adds recall_lock hold to nfsd_forget_delegations() to protect
>> nfsd_process_n_delegations() call.
>> Also, looks like it would be better to collect delegations to some local
>> on-stack list, and then unhash collected list. This split allows to simplify
>> locking, because delegation traversing is protected by recall_lock, when
>> delegation unhash is protected by client_mutex.
>> All this indirection is getting a little much.
>
>> How about replacing nfsd_process_n_delegations by something that always
>> does the list-move?:

Is it correct?
List move is suitable for unhash delegations since we anyway remove
delegation from fi_delegations list.
But seems we don't do this for delegations recall...

> void nfsd_forget_delegations(u64 num)
> {
>   unsigned int count;
>   list_head victims;
>   
>   nfs4_lock_state();
>   count = nfsd_get_n_delegations(num,&victims);
>   list_for_each_entry_safe(...,&victims, ...)
>     unhash_delegation();
>   unlock_state();
> }
>
> ditto for recall_delegations, and take the recall_lock inside
> nfsd_get_n_delegations?
>
> Or something like that.
>
> --b.
>
>> Signed-off-by: Stanislav Kinsbursky<skinsbursky@parallels.com>
>> ---
>> fs/nfsd/nfs4state.c | 32 ++++++++++++++++++++++++--------
>> 1 files changed, 24 insertions(+), 8 deletions(-)
>> diff --git a/fs/nfsd/nfs4state.c b/fs/nfsd/nfs4state.c
>> index 21266c7..f004e61 100644
>> --- a/fs/nfsd/nfs4state.c
>> +++ b/fs/nfsd/nfs4state.c
>> @@ -2597,7 +2597,7 @@ out:
return ret;
}

- static void nfsd_break_one_deleg(struct nfs4_delegation *dp)
+ static void nfsd_break_one_deleg(struct nfs4_delegation *dp, void *data)
{
/* We're assuming the state code never drops its reference
 * without first removing the lease. Since we're in this lease
@@ -2633,7 +2633,7 @@ static void nfsd_break_deleg_cb(struct file_lock *fl)
   spin_lock(&recall_lock);
   fp->fi_had_conflict = true;
   list_for_each_entry(dp,&fp->fi_delegations, dl_perfile)
-  nfsd_break_one_deleg(dp);
+  nfsd_break_one_deleg(dp, NULL);
   spin_unlock(&recall_lock);
}

@@ -4694,7 +4694,7 @@ void nfsd_forget_openowners(u64 num)
   printk(KERN_INFO "NFSD: Forgot %d open owners", count);
  }

-int nfsd_process_n_delegations(u64 num, void (*deleg_func)(struct nfs4_delegation *))
+int nfsd_process_n_delegations(u64 num, void (*deleg_func)(struct nfs4_delegation *, void *), void *data)
{
  int i, count = 0;
  struct nfs4_file *fp, *fnext;
@@ -4703,7 +4703,7 @@ int nfsd_process_n_delegations(u64 num, void (*deleg_func)(struct nfs4_delegation *)
   for (i = 0; i< FILE_HASH_SIZE; i++) {
     list_for_each_entry_safe(dp, dnext,&fp->fi_delegations, dl_perfile) {
-      deleg_func(dp);
+      deleg_func(dp, data);
      if (++count == num)
        return count;
     }
@@ -4713,15 +4713,31 @@ int nfsd_process_n_delegations(u64 num, void (*deleg_func)(struct nfs4_delegation *)
   return count;
  }

+/* Called under the recall_lock spinlock. */
+ static void
+collect_delegation(struct nfs4_delegation *dp, void *data)
+{
+  struct list_head *list = data;
+  

Subject: Re: [PATCH] NFSd: fix locking in nfsd_forget_delegations()
Posted by bfields on Thu, 24 May 2012 10:56:48 GMT

On Thu, May 24, 2012 at 08:41:35AM +0400, Stanislav Kinsbursky wrote:
> 24.05.2012 01:31, J. Bruce Fields
> > On Tue, May 22, 2012 at 02:25:14PM +0400, Stanislav Kinsbursky wrote:
> > >> This patch adds recall_lock hold to nfsd_forget_delegations() to protect
> > >> nfsd_process_n_delegations() call.
> > >> Also, looks like it would be better to collect delegations to some local
> > >> on-stack list, and then unhash collected list. This split allows to simplify

```c
void nfsd_forget_delegations(u64 num)
{
    unsigned int count;
    struct nfs4_delegation *dp, *dnext;
    LIST_HEAD(unhash_list);

    nfs4_lock_state();
    count = nfsd_process_n_delegations(num, unhash_delegation);
    nfs4_unlock_state();
    spin_lock(&recall_lock);
    count = nfsd_process_n_delegations(num, collect_delegation,&unhash_list);
    spin_unlock(&recall_lock);

    printk(KERN_INFO "NFSD: Forgot %d delegations", count);
    nfs4_lock_state();
    list_for_each_entry_safe(dp, dnext,&unhash_list, dl_perfile)
        unhash_delegation(dp);
    nfs4_unlock_state();
}

void nfsd_recall_delegations(u64 num)
@@ -4730,7 +4746,7 @@ void nfsd_recall_delegations(u64 num)
    nfs4_lock_state();
    spin_lock(&recall_lock);
    count = nfsd_process_n_delegations(num, nfsd_break_one_deleg);
    spin_unlock(&recall_lock);
    nfs4_unlock_state();
}
```

---

Page 8 of 15 ---- Generated from OpenVZ Forum
locking, because delegation traversing is protected by recall_lock, when
delegation unhash is protected by client_mutex.
All this indirection is getting a little much.

How about replacing nfsd_process_n_delegations by something that always
does the list-move?:

Is it correct?
List move is suitable for unhash delegations since we anyway remove
delegation from fi_delegations list.
But seems we don't do this for delegations recall...

Oh, blah, you're right of course.

Still, this seems a little tangled, and I'd prefer not to have to add
the useless extra parameter to break_one_deleg().

--b.

void nfsd_forget_delegations(u64 num)
{
    unsigned int count;
    list_head victims;

    nfs4_lock_state();
    count = nfsd_get_n_delegations(num,&victims);
    list_for_each_entry_safe(...,&victims, ...)
        unhash_delegation();
    unlock_state();
}
ditto for recall_delegations, and take the recall_lock inside
nfsd_get_n_delegations?

Or something like that.

--b.

Signed-off-by: Stanislav Kinsbursky<skinsbursky@parallels.com>
static void nfsd_break_one_deleg(struct nfs4_delegation *dp, void *data) {
  /* We're assuming the state code never drops its reference
   * without first removing the lease. Since we're in this lease
   */
  spin_lock(&recall_lock);
  fp->fi_had_conflict = true;
  list_for_each_entry(dp,&fp->fi_delegations, dl_perfile)
  nfsd_break_one_deleg(dp);
  spin_unlock(&recall_lock);
}

int nfsd_process_n_delegations(u64 num, void (*deleg_func)(struct nfs4_delegation *), void *data) {
  int i, count = 0;
  struct nfs4_file *fp, *fnext;
  for (i = 0; i < FILE_HASH_SIZE; i++) {
    list_for_each_entry_safe(fp,fnext,&file_hashtbl[i], fi_hash) {
      list_for_each_entry_safe(dp, dnext,&fp->fi_delegations, dl_perfile) {
        deleg_func(dp, data);
        if (++count == num)
          return count;
      }
    }
  }
  nfsd_process_n_delegations(num, void (*deleg_func)(struct nfs4_delegation *))
  nfsd_process_n_delegations(num, void (*deleg_func)(struct nfs4_delegation *, void *), void *data)
    nfsd_process_n_delegations(num, void (*deleg_func)(struct nfs4_delegation *, void *), void *data)
  return count;
}

/* Called under the recall_lock spinlock. */
static void collect_delegation(struct nfs4_delegation *dp, void *data) {
  /* Called under the recall_lock spinlock. */
}
Subject: Re: [PATCH] NFSd: fix locking in nfsd_forget_delegations()
Posted by Stanislav Kinsbursky on Thu, 24 May 2012 11:09:48 GMT

View Forum Message <> Reply to Message

On 24.05.2012 14:56, J. Bruce Fields wrote:
> On Thu, May 24, 2012 at 08:41:35AM +0400, Stanislav Kinsbursky wrote:
> >>> 24.05.2012 01:31, J. Bruce Fields
> >>> On Tue, May 22, 2012 at 02:25:14PM +0400, Stanislav Kinsbursky wrote:
This patch adds recall_lock hold to nfsd_forget_delegations() to protect 
nfsd_process_n_delegations() call.
Also, looks like it would be better to collect delegations to some local 
on-stack list, and then unhash collected list. This split allows to simplify 
locking, because delegation traversing is protected by recall_lock, when 
delegation unhash is protected by client_mutex.
All this indirection is getting a little much.

How about replacing nfsd_process_n_delegations by something that always 
does the list-move?:

Is it correct?
List move is suitable for unhash delegations since we anyway remove 
delegation from fi_delegations list.
But seems we don't do this for delegations recall...
Oh, blah, you're right of course.
Still, this seems a little tangled, and I'd prefer not to have to add
the useless extra parameter to break_one_deleg().

Ok, I'll try to handle it somehow...

void nfsd_forget_delegations(u64 num)
{
unsigned int count;
list_head victims;

nfs4_lock_state();
count = nfsd_get_n_delegations(num,&victims);
list_for_each_entry_safe(...,&victims, ...)
unhash_delegation();
unlock_state();
}
ditto for recall_delegations, and take the recall_lock inside
nfsd_get_n_delegations?
Or something like that.
diff --git a/fs/nfsd/nfs4state.c b/fs/nfsd/nfs4state.c
index 21266c7..f004e61 100644
--- a/fs/nfsd/nfs4state.c
+++ b/fs/nfsd/nfs4state.c
@@ -2597,7 +2597,7 @@ out:
   	return ret;
   }

-static void nfsd_break_one_deleg(struct nfs4_delegation *dp)
+static void nfsd_break_one_deleg(struct nfs4_delegation *dp, void *data)
 {
   /* We're assuming the state code never drops its reference
   * without first removing the lease. Since we're in this lease
@@ -2633,7 +2633,7 @@ static void nfsd_break_deleg_cb(struct file_lock *fl)
   	spin_lock(&recall_lock);
   fp->fi_had_conflict = true;
   list_for_each_entry(dp,&fp->fi_delegations, dl_perfile)
-    nfsd_break_one_deleg(dp);
+    nfsd_break_one_deleg(dp, NULL);
   spin_unlock(&recall_lock);
 }

@@ -4694,7 +4694,7 @@ void nfsd_forget_openowners(u64 num)
   	printk(KERN_INFO "NFSD: Forgot %d open owners", count);
 }

-int nfsd_process_n_delegations(u64 num, void (*deleg_func)(struct nfs4_delegation *))
+int nfsd_process_n_delegations(u64 num, void (*deleg_func)(struct nfs4_delegation *, void *), void *data)
 {
   int i, count = 0;
   struct nfs4_file *fp, *fnext;
@@ -4703,7 +4703,7 @@ int nfsd_process_n_delegations(u64 num, void (*deleg_func)(struct nfs4_delegation *, void *), void *data)
   for (i = 0; i< FILE_HASH_SIZE; i++) {
   list_for_each_entry_safe(fp, fnext,&file_hashtbl[i], fi_hash) {
   list_for_each_entry_safe(dp, dnext,&fp->fi_delegations, dl_perfile) {
     -deleg_func(dp);
+deleg_func(dp, data);
     if (++count == num)
       return count;
   }
```
int nfsd_process_n_delegations(u64 num, void (*deleg_func)(struct nfs4_delegation *), void *data)
{
    ...return count;
}

+/* Called under the recall_lock spinlock. */
+static void
+collect_delegation(struct nfs4_delegation *dp, void *data)
+
+struct list_head *list = data;
+
+list_move(&dp->dl_perfile, list);
+
+void nfsd_forget_delegations(u64 num)
+{
+    unsigned int count;
+    struct nfs4_delegation *dp, *dnext;
+    LIST_HEAD(unhash_list);
+
+    -nfs4_lock_state();
+    -count = nfsd_process_n_delegations(num, unhash_delegation);
+    -nfs4_unlock_state();
+    +spin_lock(&recall_lock);
+    +count = nfsd_process_n_delegations(num, collect_delegation,&unhash_list);
+    +spin_unlock(&recall_lock);
+    
+    printk(KERN_INFO "NFSD: Forgot %d delegations", count);
+
+    +
+    +nfs4_lock_state();
+    +list_for_each_entry_safe(dp, dnext,&unhash_list, dl_perfile)
+    +unhash_delegation(dp);
+    +nfs4_unlock_state();
+}

void nfsd_recall_delegations(u64 num)
{
    ...nfs4_lock_state();
    ...spin_lock(&recall_lock);
    ...count = nfsd_process_n_delegations(num, nfsd_break_one_deleg);
    ...spin_unlock(&recall_lock);
    ...nfs4_unlock_state();
}
```
Best regards,
Stanislav Kinsbursky