Subject: [PATCH v2 5/6] SUNRPC: cleanup GSS pipes usage
Posted by Stanislav Kinsbursky on Mon, 26 Dec 2011 11:45:28 GMT
View Forum Message <> Reply to Message

Currently gss auth holds RPC inode pointer which is now redundant since it
requires only pipes operations which takes private pipe data as an argument.
Thus this code can be cleaned and all references to RPC inode can be replaced
with privtae pipe data references.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

---
 net/sunrpc/auth_gss/auth_gss.c | 76 ++++++++++++++++++++----------------
 1 files changed, 38 insertions(+), 38 deletions(-)

diff --git a/net/sunrpc/auth_gss/auth_gss.c b/net/sunrpc/auth_gss/auth_gss.c
index 5ba678d..f99fde5 100644
--- a/net/sunrpc/auth_gss/auth_gss.c
+++ b/net/sunrpc/auth_gss/auth_gss.c
@@ -112,7 +112,7 @@ gss_put_ctx(struct gss_cl_ctx *ctx)
 /* gss_cred_set_ctx:
  * called by gss_upcall_callback and gss_create_upcall in order
  * to set the gss context. The actual exchange of an old context
- * and a new one is protected by the rpci->pipe->lock.
+ * and a new one is protected by the pipe->lock.
  */
 static void
 gss_cred_set_ctx(struct rpc_cred *cred, struct gss_cl_ctx *ctx)
@@ -251,7 +251,7 @@ struct gss_upcall_msg {
  struct rpc_pipe_msg msg;
  struct list_head list;
  struct gss_auth *auth;
- struct rpc_inode *inode;
+ struct rpc_pipe *pipe;
  struct rpc_wait_queue rpc_waitqueue;
  wait_queue_head_t waitqueue;
  struct gss_cl_ctx *ctx;
@@ -294,10 +294,10 @@ gss_release_msg(struct gss_upcall_msg *gss_msg)
 }

 static struct gss_upcall_msg *
-__gss_find_upcall(struct rpc_inode *rpci, uid_t uid)
+__gss_find_upcall(struct rpc_pipe *pipe, uid_t uid)
 {
  struct gss_upcall_msg *pos;
- list_for_each_entry(pos, &rpci->pipe->in_downcall, list) {
+ list_for_each_entry(pos, &pipe->in_downcall, list) {
  if (pos->uid != uid)

```
    continue;
   atomic_inc(&pos->count);
@@ -315,17 +315,17 @@ __gss_find_upcall(struct rpc_inode *rpci, uid_t uid)
 static inline struct gss_upcall_msg *
 gss_add_msg(struct gss_upcall_msg *gss_msg)
 {
- struct rpc_inode *rpci = gss_msg->inode;
+ struct rpc_pipe *pipe = gss_msg->pipe;
  struct gss_upcall_msg *old;

- spin_lock(&rpci->pipe->lock);
- old = __gss_find_upcall(rpci, gss_msg->uid);
+ spin_lock(&pipe->lock);
+ old = __gss_find_upcall(pipe, gss_msg->uid);
  if (old == NULL) {
   atomic_inc(&gss_msg->count);
-  list_add(&gss_msg->list, &rpci->pipe->in_downcall);
+  list_add(&gss_msg->list, &pipe->in_downcall);
  } else
   gss_msg = old;
- spin_unlock(&rpci->pipe->lock);
+ spin_unlock(&pipe->lock);
  return gss_msg;
 }

@@ -341,14 +341,14 @@ __gss_unhash_msg(struct gss_upcall_msg *gss_msg)
 static void
 gss_unhash_msg(struct gss_upcall_msg *gss_msg)
 {
- struct rpc_inode *rpci = gss_msg->inode;
+ struct rpc_pipe *pipe = gss_msg->pipe;

  if (list_empty(&gss_msg->list))
   return;
- spin_lock(&rpci->pipe->lock);
+ spin_lock(&pipe->lock);
  if (!list_empty(&gss_msg->list))
   __gss_unhash_msg(gss_msg);
- spin_unlock(&rpci->pipe->lock);
+ spin_unlock(&pipe->lock);
 }

 static void
@@ -375,11 +375,11 @@ gss_upcall_callback(struct rpc_task *task)
  struct gss_cred *gss_cred = container_of(task->tk_rqstp->rq_cred,
    struct gss_cred, gc_base);
  struct gss_upcall_msg *gss_msg = gss_cred->gc_upcall;
- struct rpc_inode *rpci = gss_msg->inode;
```

```
+ struct rpc_pipe *pipe = gss_msg->pipe;

- spin_lock(&rpci->pipe->lock);
+ spin_lock(&pipe->lock);
  gss_handle_downcall_result(gss_cred, gss_msg);
- spin_unlock(&rpci->pipe->lock);
+ spin_unlock(&pipe->lock);
  task->tk_status = gss_msg->msg.errno;
  gss_release_msg(gss_msg);
 }
@@ -451,7 +451,7 @@ gss_alloc_msg(struct gss_auth *gss_auth, uid_t uid, struct rpc_clnt *clnt,
   kfree(gss_msg);
   return ERR_PTR(vers);
  }
- gss_msg->inode = RPC_I(gss_auth->dentry[vers]->d_inode);
+ gss_msg->pipe = RPC_I(gss_auth->dentry[vers]->d_inode)->pipe;
  INIT_LIST_HEAD(&gss_msg->list);
  rpc_init_wait_queue(&gss_msg->rpc_waitqueue, "RPCSEC_GSS upcall waitq");
  init_waitqueue_head(&gss_msg->waitqueue);
@@ -475,7 +475,7 @@ gss_setup_upcall(struct rpc_clnt *clnt, struct gss_auth *gss_auth, struct
rpc_cr
   return gss_new;
  gss_msg = gss_add_msg(gss_new);
  if (gss_msg == gss_new) {
-  int res = rpc_queue_upcall(gss_new->inode->pipe, &gss_new->msg);
+  int res = rpc_queue_upcall(gss_new->pipe, &gss_new->msg);
   if (res) {
    gss_unhash_msg(gss_new);
    gss_msg = ERR_PTR(res);
@@ -506,7 +506,7 @@ gss_refresh_upcall(struct rpc_task *task)
  struct gss_cred *gss_cred = container_of(cred,
    struct gss_cred, gc_base);
  struct gss_upcall_msg *gss_msg;
- struct rpc_inode *rpci;
+ struct rpc_pipe *pipe;
  int err = 0;

  dprintk("RPC: %5u gss_refresh_upcall for uid %u\n", task->tk_pid,
@@ -524,8 +524,8 @@ gss_refresh_upcall(struct rpc_task *task)
   err = PTR_ERR(gss_msg);
   goto out;
  }
- rpci = gss_msg->inode;
- spin_lock(&rpci->pipe->lock);
+ pipe = gss_msg->pipe;
+ spin_lock(&pipe->lock);
  if (gss_cred->gc_upcall != NULL)
   rpc_sleep_on(&gss_cred->gc_upcall->rpc_waitqueue, task, NULL);
```

```
  else if (gss_msg->ctx == NULL && gss_msg->msg.errno >= 0) {
@@ -538,7 +538,7 @@ gss_refresh_upcall(struct rpc_task *task)
    gss_handle_downcall_result(gss_cred, gss_msg);
    err = gss_msg->msg.errno;
   }
- spin_unlock(&rpci->pipe->lock);
+ spin_unlock(&pipe->lock);
   gss_release_msg(gss_msg);
  out:
   dprintk("RPC: %5u gss_refresh_upcall for uid %u result %d\n",
@@ -549,7 +549,7 @@ out:
  static inline int
  gss_create_upcall(struct gss_auth *gss_auth, struct gss_cred *gss_cred)
  {
- struct rpc_inode *rpci;
+ struct rpc_pipe *pipe;
   struct rpc_cred *cred = &gss_cred->gc_base;
   struct gss_upcall_msg *gss_msg;
   DEFINE_WAIT(wait);
@@ -573,14 +573,14 @@ retry:
    err = PTR_ERR(gss_msg);
    goto out;
   }
- rpci = gss_msg->inode;
+ pipe = gss_msg->pipe;
   for (;;) {
    prepare_to_wait(&gss_msg->waitqueue, &wait, TASK_KILLABLE);
-   spin_lock(&rpci->pipe->lock);
+   spin_lock(&pipe->lock);
    if (gss_msg->ctx != NULL || gss_msg->msg.errno < 0) {
     break;
    }
-   spin_unlock(&rpci->pipe->lock);
+   spin_unlock(&pipe->lock);
    if (fatal_signal_pending(current)) {
     err = -ERESTARTSYS;
     goto out_intr;
@@ -591,7 +591,7 @@ retry:
    gss_cred_set_ctx(cred, gss_msg->ctx);
   else
    err = gss_msg->msg.errno;
- spin_unlock(&rpci->pipe->lock);
+ spin_unlock(&pipe->lock);
  out_intr:
   finish_wait(&gss_msg->waitqueue, &wait);
   gss_release_msg(gss_msg);
@@ -609,7 +609,7 @@ gss_pipe_downcall(struct file *filp, const char __user *src, size_t mlen)
   const void *p, *end;
```

```
  void *buf;
  struct gss_upcall_msg *gss_msg;
- struct rpc_inode *rpci = RPC_I(filp->f_dentry->d_inode);
+ struct rpc_pipe *pipe = RPC_I(filp->f_dentry->d_inode)->pipe;
  struct gss_cl_ctx *ctx;
  uid_t uid;
  ssize_t err = -EFBIG;
@@ -639,14 +639,14 @@ gss_pipe_downcall(struct file *filp, const char __user *src, size_t mlen)

  err = -ENOENT;
  /* Find a matching upcall */
- spin_lock(&rpci->pipe->lock);
- gss_msg = __gss_find_upcall(rpci, uid);
+ spin_lock(&pipe->lock);
+ gss_msg = __gss_find_upcall(pipe, uid);
  if (gss_msg == NULL) {
-  spin_unlock(&rpci->pipe->lock);
+  spin_unlock(&pipe->lock);
   goto err_put_ctx;
  }
  list_del_init(&gss_msg->list);
- spin_unlock(&rpci->pipe->lock);
+ spin_unlock(&pipe->lock);

  p = gss_fill_context(p, end, ctx, gss_msg->auth->mech);
  if (IS_ERR(p)) {
@@ -674,9 +674,9 @@ gss_pipe_downcall(struct file *filp, const char __user *src, size_t mlen)
  err = mlen;

 err_release_msg:
- spin_lock(&rpci->pipe->lock);
+ spin_lock(&pipe->lock);
  __gss_unhash_msg(gss_msg);
- spin_unlock(&rpci->pipe->lock);
+ spin_unlock(&pipe->lock);
  gss_release_msg(gss_msg);
 err_put_ctx:
  gss_put_ctx(ctx);
@@ -722,23 +722,23 @@ static int gss_pipe_open_v1(struct inode *inode)
 static void
 gss_pipe_release(struct inode *inode)
 {
- struct rpc_inode *rpci = RPC_I(inode);
+ struct rpc_pipe *pipe = RPC_I(inode)->pipe;
  struct gss_upcall_msg *gss_msg;

 restart:
- spin_lock(&rpci->pipe->lock);
```

```
- list_for_each_entry(gss_msg, &rpci->pipe->in_downcall, list) {
+ spin_lock(&pipe->lock);
+ list_for_each_entry(gss_msg, &pipe->in_downcall, list) {

   if (!list_empty(&gss_msg->msg.list))
    continue;
   gss_msg->msg.errno = -EPIPE;
   atomic_inc(&gss_msg->count);
   __gss_unhash_msg(gss_msg);
-  spin_unlock(&rpci->pipe->lock);
+  spin_unlock(&pipe->lock);
   gss_release_msg(gss_msg);
   goto restart;
  }
- spin_unlock(&rpci->pipe->lock);
+ spin_unlock(&pipe->lock);

  put_pipe_version();
 }
```