Subject: [RFC PATCH 15/13] NFSd: make boot_time variable per network namespace
Posted by Stanislav Kinsbursky on Mon, 14 May 2012 14:00:19 GMT
View Forum Message <> Reply to Message

NFSd's boot_time represents grace period start point in time.

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

```
---
 fs/nfsd/netns.h     |    1 +
 fs/nfsd/nfs4state.c |   47 +++++++++++++++++++++++++++++-------------------
 fs/nfsd/state.h     |    4 +++-
 3 files changed, 32 insertions(+), 20 deletions(-)

diff --git a/fs/nfsd/netns.h b/fs/nfsd/netns.h
index b6deebd..65c2431 100644
--- a/fs/nfsd/netns.h
+++ b/fs/nfsd/netns.h
@@ -37,6 +37,7 @@ struct nfsd_net {

  struct lock_manager nfsd4_manager;
  bool grace_ended;
+ time_t boot_time;
 };

 extern int nfsd_net_id;
diff --git a/fs/nfsd/nfs4state.c b/fs/nfsd/nfs4state.c
index 7c88ee8..e858ce7 100644
--- a/fs/nfsd/nfs4state.c
+++ b/fs/nfsd/nfs4state.c
@@ -51,7 +51,6 @@
 /* Globals */
 time_t nfsd4_lease = 90;     /* default lease time */
 time_t nfsd4_grace = 90;
-static time_t boot_time;

 #define all_ones {{~0,~0},~0}
 static const stateid_t one_stateid = {
@@ -1011,12 +1010,12 @@ renew_client(struct nfs4_client *clp)

 /* SETCLIENTID and SETCLIENTID_CONFIRM Helper functions */
 static int
-STALE_CLIENTID(clientid_t *clid)
+STALE_CLIENTID(clientid_t *clid, struct nfsd_net *nn)
 {
- if (clid->cl_boot == boot_time)
+ if (clid->cl_boot == nn->boot_time)
   return 0;
```

```
  dprintk("NFSD stale clientid (%08x/%08x) boot_time %08lx\n",
-  clid->cl_boot, clid->cl_id, boot_time);
+  clid->cl_boot, clid->cl_id, nn->boot_time);
  return 1;
 }

@@ -1170,8 +1169,9 @@ same_creds(struct svc_cred *cr1, struct svc_cred *cr2)
 static void gen_clid(struct nfs4_client *clp)
 {
  static u32 current_clientid = 1;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

- clp->cl_clientid.cl_boot = boot_time;
+ clp->cl_clientid.cl_boot = nn->boot_time;
  clp->cl_clientid.cl_id = current_clientid++;
 }

@@ -2221,8 +2221,9 @@ nfsd4_setclientid_confirm(struct svc_rqst *rqstp,
  nfs4_verifier confirm = setclientid_confirm->sc_confirm;
  clientid_t * clid = &setclientid_confirm->sc_clientid;
  __be32 status;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

- if (STALE_CLIENTID(clid))
+ if (STALE_CLIENTID(clid, nn))
   return nfserr_stale_clientid;
  /*
   * XXX The Duplicate Request Cache (DRC) has been checked (??)
@@ -2621,8 +2622,9 @@ nfsd4_process_open1(struct nfsd4_compound_state *cstate,
  unsigned int strhashval;
  struct nfs4_openowner *oo = NULL;
  __be32 status;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

- if (STALE_CLIENTID(&open->op_clientid))
+ if (STALE_CLIENTID(&open->op_clientid, nn))
   return nfserr_stale_clientid;
  /*
   * In case we need it later, after we've already created the
@@ -3164,12 +3166,13 @@ nfsd4_renew(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
 {
  struct nfs4_client *clp;
  __be32 status;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

  nfs4_lock_state();
  dprintk("process_renew(%08x/%08x): starting\n",
```

```
    clid->cl_boot, clid->cl_id);
  status = nfserr_stale_clientid;
- if (STALE_CLIENTID(clid))
+ if (STALE_CLIENTID(clid, nn))
    goto out;
  clp = find_confirmed_client(clid);
  status = nfserr_expired;
@@ -3199,7 +3202,7 @@ nfsd4_end_grace(struct net *net)

  dprintk("NFSD: end of grace period\n");
  nn->grace_ended = true;
- nfsd4_record_grace_done(net, boot_time);
+ nfsd4_record_grace_done(net, nn->boot_time);
  locks_end_grace(&nn->nfsd4_manager);
  /*
   * Now that every NFSv4 client has had the chance to recover and
@@ -3305,9 +3308,9 @@ static inline __be32 nfs4_check_fh(struct svc_fh *fhp, struct
nfs4_ol_stateid *s
 }

 static int
-STALE_STATEID(stateid_t *stateid)
+STALE_STATEID(stateid_t *stateid, struct nfsd_net *nn)
 {
- if (stateid->si_opaque.so_clid.cl_boot == boot_time)
+ if (stateid->si_opaque.so_clid.cl_boot == nn->boot_time)
    return 0;
  dprintk("NFSD: stale stateid " STATEID_FMT "!\n",
    STATEID_VAL(stateid));
@@ -3407,13 +3410,14 @@ static __be32 check_stateid_generation(stateid_t *in, stateid_t *ref,
bool has_s
  return nfserr_old_stateid;
 }

-__be32 nfs4_validate_stateid(struct nfs4_client *cl, stateid_t *stateid)
+__be32 nfs4_validate_stateid(struct nfs4_client *cl, stateid_t *stateid,
+        struct nfsd_net *nn)
 {
  struct nfs4_stid *s;
  struct nfs4_ol_stateid *ols;
  __be32 status;

- if (STALE_STATEID(stateid))
+ if (STALE_STATEID(stateid, nn))
    return nfserr_stale_stateid;

  s = find_stateid(cl, stateid);
@@ -3434,10 +3438,11 @@ __be32 nfs4_validate_stateid(struct nfs4_client *cl, stateid_t
```

```
*stateid)
 static __be32 nfsd4_lookup_stateid(stateid_t *stateid, unsigned char typemask, struct nfs4_stid
**s)
 {
  struct nfs4_client *cl;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

  if (ZERO_STATEID(stateid) || ONE_STATEID(stateid))
   return nfserr_bad_stateid;
- if (STALE_STATEID(stateid))
+ if (STALE_STATEID(stateid, nn))
   return nfserr_stale_stateid;
  cl = find_confirmed_client(&stateid->si_opaque.so_clid);
  if (!cl)
@@ -3534,10 +3539,11 @@ nfsd4_test_stateid(struct svc_rqst *rqstp, struct
nfsd4_compound_state *cstate,
 {
  struct nfsd4_test_stateid_id *stateid;
  struct nfs4_client *cl = cstate->session->se_client;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

  nfs4_lock_state();
  list_for_each_entry(stateid, &test_stateid->ts_stateid_list, ts_id_list)
-  stateid->ts_id_status = nfs4_validate_stateid(cl, &stateid->ts_id_stateid);
+  stateid->ts_id_status = nfs4_validate_stateid(cl, &stateid->ts_id_stateid, nn);
  nfs4_unlock_state();

  return nfs_ok;
@@ -4101,6 +4107,7 @@ nfsd4_lock(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
  bool new_state = false;
  int lkflg;
  int err;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

  dprintk("NFSD: nfsd4_lock: start=%Ld length=%Ld\n",
   (long long) lock->lk_offset,
@@ -4132,7 +4139,7 @@ nfsd4_lock(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
     sizeof(clientid_t));

   status = nfserr_stale_clientid;
-  if (STALE_CLIENTID(&lock->lk_new_clientid))
+  if (STALE_CLIENTID(&lock->lk_new_clientid, nn))
    goto out;

   /* validate and update open stateid and open seqid */
@@ -4273,6 +4280,7 @@ nfsd4_lockt(struct svc_rqst *rqstp, struct nfsd4_compound_state
```

```
*cstate,
  struct nfs4_lockowner *lo;
  int error;
  __be32 status;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

  if (locks_in_grace(SVC_NET(rqstp)))
   return nfserr_grace;
@@ -4283,7 +4291,7 @@ nfsd4_lockt(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
  nfs4_lock_state();

  status = nfserr_stale_clientid;
- if (!nfsd4_has_session(cstate) && STALE_CLIENTID(&lockt->lt_clientid))
+ if (!nfsd4_has_session(cstate) && STALE_CLIENTID(&lockt->lt_clientid, nn))
   goto out;

  if ((status = fh_verify(rqstp, &cstate->current_fh, S_IFREG, 0)))
@@ -4434,6 +4442,7 @@ nfsd4_release_lockowner(struct svc_rqst *rqstp,
  struct list_head matches;
  unsigned int hashval = ownerstr_hashval(clid->cl_id, owner);
  __be32 status;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

  dprintk("nfsd4_release_lockowner clientid: (%08x/%08x):\n",
   clid->cl_boot, clid->cl_id);
@@ -4441,7 +4450,7 @@ nfsd4_release_lockowner(struct svc_rqst *rqstp,
  /* XXX check for lease expiration */

  status = nfserr_stale_clientid;
- if (STALE_CLIENTID(clid))
+ if (STALE_CLIENTID(clid, nn))
   return status;

  nfs4_lock_state();
@@ -4757,7 +4766,7 @@ nfs4_state_start(void)
   */
  get_net(net);
  nfsd4_client_tracking_init(net);
- boot_time = get_seconds();
+ nn->boot_time = get_seconds();
  locks_start_grace(net, &nn->nfsd4_manager);
  nn->grace_ended = false;
  printk(KERN_INFO "NFSD: starting %ld-second grace period\n",
diff --git a/fs/nfsd/state.h b/fs/nfsd/state.h
index 29d9f0b..5946ffd 100644
--- a/fs/nfsd/state.h
+++ b/fs/nfsd/state.h
```

```
@@ -451,6 +451,7 @@ static inline struct nfs4_ol_stateid *openlockstateid(struct nfs4_stid *s)
 #define WR_STATE         0x00000020

 struct nfsd4_compound_state;
+struct nfsd_net;

 extern __be32 nfs4_preprocess_stateid_op(struct net *net,
   struct nfsd4_compound_state *cstate,
@@ -477,7 +478,8 @@ extern __be32 nfs4_make_rec_clidname(char *clidname, struct
xdr_netobj *clname);
 extern int nfs4_client_to_reclaim(const char *name);
 extern int nfs4_has_reclaimed_state(const char *name, bool use_exchange_id);
 extern void release_session_client(struct nfsd4_session *);
-extern __be32 nfs4_validate_stateid(struct nfs4_client *, stateid_t *);
+extern __be32 nfs4_validate_stateid(struct nfs4_client *, stateid_t *,
+        struct nfsd_net *);
 extern void nfsd4_purge_closed_stateid(struct nfs4_stateowner *);

 /* nfs4recover operations */
```