lustre_msg v2

Lai Siyao

Jan-17-06

1 Functional Specification

1.1 new data types

The old msg format is renamed to lustre_msg_v1:

```
struct lustre_msg_v1 {
        struct lustre_handle lm_handle;
        __u32 lm_magic;
        __u32 lm_type;
        __u32 lm_version;
        __u32 lm_opc;
        __u64 lm_last_xid;
        __u64 lm_last_committed;
        __u64 lm_transno;
        __u32 lm_status;
        __u32 lm_flags;
        __u32 lm_conn_cnt;
        __u32 lm_bufcount;
        __u32 lm_buflens[0];
};
```

While the new format is as follows:

```
struct lustre_msg_v2 {
        __u32 lm_bufcount;   /* buf count */
        __u32 lm_secflvr;    /* sec flavor */
        __u32 lm_magic;      /* msg magic */
        __u32 lm_buflens[0]; /* buf lengths array */
};
struct ptlrpc_body { /* RPC-specific data */
        struct lustre_handle pb_handle;
        __u32 pb_type;
        __u32 pb_version;
        __u32 pb_opc;
```

```
        __u64 pb_last_xid;
        __u64 pb_last_committed;
        __u64 pb_transno;
        __u32 pb_status;
        __u32 pb_flags;
        __u32 pb_conn_cnt;
        __u32 pb_paddings[4];
};
```

Struct lustre_msg_v2 is new version of lustre_msg, and it only contains message related fields:

bufcount pending buffers count.

secflvr security flavor, will be used by gss/krb5 code.

magic act as both message magic and format version, and its offset is kept the same as the old format on purpose, which will help interoperate with old code.

buflens length array of pending buffers.

The RPC_specific fields are moved from old lustre_msg structure to struct ptlrpc_body, which will be packed into lustre_msg as a seperate message body. And 4 padding words are added for future use.

2 Use Case

2.1 mdc getattr

mdc prepare request, if lustre_msg_v1, pack with v1 packing functions, and don't pack ptlrpc_body in pending buffers, otherwise pack with v2 packing functions. Then send this request to mds.

mds receive this request, if using lustre_msg_v2, unpack ptlrpc_body in ptlrpc level.

mds handle getattr request. Then pack the reply according to request message magic, which is similar to the request message packing. Finally reply back.

mdc receive the reply, unpack it according the message magic in reply.

2.2 mdc connect

mdc set flag MSG_CONNECT_NEXT_VER in message. Then send connect request.

mds — old server ignore this flag, and reply back. But new server will check it, if set, also set this flag in reply message, otherwise set message magic for the according reverse import to lustre_msg_v1.

mdc — check the flags in reply message, if MSG_CONNECT_NEXT_VER is not set, set message magic for this import to lustre_msg_v1. And the following requests will pack message according to this magic.

3 Logical Specification

3.1 message pack/unpack and etc functions

To keep protocol compatible, two versions of lustre_msg should be supported. The 'magic' field is used as both magic and version, and its offset in two versions should be the same. And there should be two sets of message operation functions, and they are used according to the message magic field.

3.2 hide RPC-specific data

This falls in two parts:

- To support lustre_msg_v2, ptlrpc_body will be counted as a pending buffer, but for lustre_msg_v1 it should be ignored. So all the message pack/unpack and other function wrappers should adjust the arguments for lustre_msg_v1 (the arguments are for lustre_msg_v2 by default).
- add wrappers for all access to these data.

3.3 interoperation between old and new format

To achieve this, a new message flag MSG_CONNECT_NEXT_VER is introduced: new client will exchange format info with server, while old server will ignore this flag, and new server will set message format to use for the following requests, and reply this flag back.

3.4 enable RPC version for V2 message format

The RPC version code has been implemented, but it's disabled to keep protocol compatibility. Enable it for V2 message format.

4 Recovery

N/A

5 Test

Focus on interoperation between old and new format.