LNET Support for IPv6 is Long Overdue

Isaac Huang, Xyratex
We've been talking about IPv6 for a long time:

- IPv6 support might become a requirement in some contracts.
  - Already a requirement in some aspects, e.g. external access.
- Lustre over WAN, as IPv6 picking up steam.

Today we're going to talk about why it's hard and more importantly a possible solution to the problems.
Brief overview of the Lustre networking stacks

File system services
- LNet selftest
- PTLRPC
- Others

Generic LNet

LNet API

Internal LND API

Network specific API
- Other Networks
- TCP/IP
- InfiniBand
• Only 32 bits in a Lustre network address (the Inet_nid_t) for IP addresses
• The Inet_nid_t is a fundamental data structure
  – Used in the code, transferred over the wire, and even saved on the disk.
  – Current development could dig us deeper in the hole.
• Backward compatibility must be maintained.
The problem: the address

The LNet address: 32 bit address-within LNET + 32 bit LNET number = 64 bits / 8 bytes total

- Hedge a little by reserving an additional 32 bits for something we've not thought of yet and keeping the total a multiple of 64 bits to simplify alignment. 192 bits / 24 bytes total.
- Hedge a lot more. 256 bits / 32 bytes total.
The problem: LNDs

Lustre Network Drivers:
• The TCP LND needs to use sockets in address family AF_INET6
• The IB LND:
  – doesn't use IP protocol for data, but address resolution could work with IPv6 addresses.
  – reduced # of fragments supported, use map_on_demand
• Other LNDs need to handle the new bigger LNet addresses
  – Could cause problems to alignment sensitive networks.
The problem: PTLRPC and upper layers

- PTLRPC and RPC services: on wire protocol must all change if it includes LNet address
- On the disk:
  - Strings: in mountdata, and UUIDs in llogs. No disk format change.
  - __u64 in struct lustre_cfg::lcfg_nid. May need change.
The biggest problem

Backward compatibility:

- LNet, PTLRPC, and FS services must be able to handle both addresses.
- Routing adds more complexity:
  - LND level version negotiation is not end to end.
  - LNet protocol is connection less.
A solution: fight or flight?

• New network types (and new LNDs) for affected LNDs: a copy, plus IPv6 support
  – For example: @o2ib0 -> @ib0, @tcp0 -> @tcpng0

• Pros:
  – Essentially avoids version compatibility by adding new network types.
  – A good chance to clean up old features/protocol from new LNDs.
  – Simplify (though duplicate) LND code: each LND handles one address format.
  – Isolate changes: no IPv6, no need to run any new code.

• Cons:
  – More changes propagated to upper layers.
  – More code (though largely duplicated) to maintain and test.
answer(questions);
thank_you();
exit(0);