Portals4 LND Overview
Grégoire Pichon - Atos
05-10-2017
1 New Lustre Network Driver
LNet

Lustre Network layer

▶ communication infrastructure
  – between Lustre clients and servers
▶ supports many commonly-used network types
  – Infiniband, TCP/IP networks
  – allows routing between networks
▶ key features
  – high availability
  – recovery
▶ Lustre Network Driver
  – provides support for a particular network type
  – implements LNet-LND api
Portals 4 LND

New Lustre Network Driver

- **ptl4lnd**
  - based on Sandia National Laboratories Portals 4 Network Programming Interface
  - **LU-8932** inet: define new network driver *ptl4lnd*
    - kernel module: *kptl4lnd.ko*
    - network name: *ptlf*
    - network adapter identified by device number
      
      ```
      networks=ptlf0(0),ptlf1(1)
      ```

- **hardware**
  - **Bull eXascale Interconnect**
Portals 4

API for high-performance networking

- **Common semantic** for MPI and PGAS
  - Target memory descriptors: “persistent” (one-sided ops.) or “use once” (two-sided ops.)
  - Rich operations library: Put, Get, Swap, FetchAtomic

- **Full hardware offloading** from the host CPU
  - Performance is not impacted by heavy load on the host CPU
  - Triggered operations for protocol offloading (collectives, etc.)

- **Non-connected reliable protocol**
  - No connection establishment time
  - Constant memory footprint whatever the number of communicators

- **Unexpected messages aggregation**
  - N to 1 communications optimization – single buffer for multiple messages
Portals 4
Communication Model

MSG

request sent
nid / pid
portals idx
matchbits

Network Interface

Portals Table Entry

Portals Table

matchbits
ma

matchbits
mb

matchbits
mz

Match Entry

Memory region
Ptl4Ind internals
Small data transfer (from node A to node B)

Node A

`Ind_send()`
build ptl4Ind imm. msg with LNet hdr
copy LNet payload into msg

Node B

`PtlPut()`
IMMEDIATE message with fixed matchbits

persistent ME pre-posted with fixed matchbits

match a persistent ME
parse LNet hdr

`Ind_recv()`
copy payload from msg to LNet destination memory

`lnd_send()`
`lnd_recv()`
`PtlPut()`
`MSG`
Ptl4Ind internals

RDMA data transfer (from node A to node B)

Node A

Ind_send()
- get a unique matchbits
- prepare a use-once ME
- build ptl4Ind rdma msg
- LNet hdr & matchbits

match use-once ME
remove it from list

Node B

PtlPut()
- RDV message
- with fixed matchbits

match a persistent ME
parse LNet hdr

Ind_recv()
- prepare a Memory Descriptor

PtlGet()
- BULK message
- with unique matchbits

persistent ME pre-posted
with fixed matchbits

remove it from list

ME
MSG
MD

unique matchbits
**Ptl4Ind internals**

**resource management**

- **Transmission (TX) resources**
  - TX Event Queue
  - PTL4_PORTAL
    - tx descriptor
    - IMM. message or RDV message

- **Reception (RX) resources**
  - RX Event Queue
  - TX Event Queue

**Portals Table**
- start
- length
- fixed
- matchbits
- persistent

**Memory region**
- start
- length
- unique
- matchbits
- use-once
- iovec

**TX Event Queue**
- transmission (TX) resources
- reception (RX) resources

**Memory Descriptor**
- tx descriptor
- BULK message
Ptl4Ind parameters and tuning

- credits (32) and peercredits (8)
  - number of concurrent sends to all and to one peer
  - need uniform setting between cluster nodes
  - higher value allows higher bandwidth but consumes more resources

- checksum (off)
  - check integrity of non-bulk messages

- nscheds (2)
  - number of scheduler threads that handle RX buffer, TX finalization, ...
Ptl4lnd debug and statistics

- slots in event queues
  - number of reserved slots
  - total number of slots

- peer status
  - nid-pid, state, credits, alive time

- statistics
  - number of TX of different types

- dump TX and RX states
Ptl4Ind performance achievements

LNet performance

LNet Selftest – brw

Lustre IEEU 2.7.21
BXI 1.2 NIC – SWITCH - NIC
Ptl4lnd performance achievements

Lustre performance

Lustre IEEL 2.7.21
BXI 1.2 NIC – SWITCH – NIC

WORK IN PROGRESS
PtI4Ind development

Next steps

- Finalize LND code
  - perform some optimizations
  - handle remaining corner cases

- Ensure compatibility with recent LNet changes
  - currently runs on Lustre IEEL 2.7.21
  - build and test on Lustre master

- Push code to the Lustre community
2 Development Challenges
LNet – LND interface

looks simple …

- register to LNet
  - Inet_register_Lnd()
  - Inet_unregister_Lnd()

- provide a Inet_Lnd structure
  - Lnd id (SOCKLND, O2IBLND, LOLND, GNILND, PTL4LND)
  - startup/shutdown network communication on the network interface
  - send/receive LNet message on the network interface
  - notification /query on peer health / aliveness
  - control commands

- use LNet callbacks
  - Inet_parse(), Inet_finalize(), Inet_set-reply_msg_len(), ...

struct lnet_lnd ptfLnd = {
  .lnd_type = PTL4LND,
  .lnd_startup = ptf_startup,
  .lnd_shutdown = ptf_shutdown,
  .lnd_ctl = ptf_ctl,
  .lnd_send = ptf_send,
  .lnd_recv = ptf_recv,
  .lnd_query = ptf_query,
};
LNet – LND interface

... but still some interrogations

- lack of documentation
  - routine semantic
    - when should a LND call lnet_notify()?
    - what LNet callbacks should/shall a LND use?
- parameters description
- call context

```c
/* Start receiving 'mlen' bytes of payload data, skipping the following
 * 'rlen' - 'mlen' bytes. 'private' is the 'private' passed to
 * lnet_parse(). Return non-zero for immediate failure, otherwise
 * complete later with lnet_finalize(). This also gives back a receive
 * credit if the LND does flow control. */
int (*lnd_recv)(struct lnet_ni *ni, void *private, struct lnet_msg *msg,
               int delayed, unsigned int niov,
               struct kvec *iov, lnet_kiov_t *kiov,
               unsigned int offset, unsigned int mlen, unsigned int rlen);
```
Tests and bringup

- **LNet selftest is not convenient** for LNet debugging or LND bringup
  - test infrastructure is setup with ... LNet messages
  - not designed to perform 1 LNetPut() or LNetGet() operation
  - cannot test
    - specific (exotic) memory region transfers
    - error cases (non matching ME, short MD)

- **development of a LNet test kernel module**
  - define a pseudo device for ioctl command
    - post LNet ME-MD for reception
    - launch LNet data transfer
  - also post permanent LNet ME-MD
Opportunities offered by modern Interconnects
Network advanced features (1)

... that Lustre could benefit

- **Atomic operations**
  - atomically read and update data located in remote memory regions
  - host bypass, low latency
  - operations
    - min, max, sum, prod, or, and, swap, conditional swap, ...
  - usage
    - distributed lock
    - object sequence number
  - LNetAtomic(), LNetFetchAtomic(), LNetSwap()
Network advanced features (2)

... that Lustre could benefit

► Triggered operations
  – setup operations triggered by incoming messages
  – host bypass, low latency
  – usage:
    • tree based reduction
    • recovery synchronization
  – LNetTriggeredPut(), LNetTriggeredAtomic()
Thanks
gregoire.pichon@atos.net