Multi-Rail LNet Configuration Guide

Partner Guide

High Performance Data Division

June 27, 2016

World Wide Web: http://www.intel.com

Disclaimer and legal information

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel® Corporation or its suppliers or licensors. Title to the Material remains with Intel® Corporation or its suppliers and licensors. The Material contains trade secrets and proprietary and confidential information of Intel® or its suppliers and licensors. The Material is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel® in writing.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL® ASSUMES NO LIABILITY WHATSOEVER AND INTEL® DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel® Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL® AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL® OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL® PRODUCT OR ANY OF ITS PARTS.

Intel® may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel® reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Before using any third party software referenced herein, please refer to the third party software provider's website for more information, including without limitation, information regarding the mitigation of potential security vulnerabilities in the third party software.

Contact your local Intel® sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel® literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm.

Intel® and the Intel® logo are trademarks of Intel® Corporation in the U.S. and/or other countries.

* Other names and brands may be claimed as the property of others.

"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)

Contents

About this Document

Document Purpose

This document describes how to <to be completed when bulk of document is completed.>

Intended Audience

This guide is intended for systems integrators with a strong technical background in Linux system administration as well as Lustre file system deployment and management.

It is expected that readers have:

- Experience administering file systems and storage infrastructure, and familiarity with storage concepts such as RAID, SAN, and LVM.
- System management experience sufficient to install and configure a storage platform compatible with the requirements as defined in this guide.
- Proficiency in setting up, administering, and maintaining computer networks, including Ethernet, TCP/IP and InfiniBand where necessary. Some knowledge of Lustre networking (LNET) is required.
- Some experience with installing and managing Lustre file systems.

This document is not intended for end-users or application developers.

Conventions Used

<To be further completed when bulk of document is completed.>

Conventions used in this document include:

- # preceding a command indicates the command is to be entered as root
- $ indicates a command is to be entered as a user
- *<variable_name>* indicates the placeholder text that appears between the angle brackets is to be replaced with an appropriate value

Related Documentation

- *Intel® Enterprise Edition for Lustre* Software, Version 3.0.0.0 Release Notes*
- *Intel® Manager for Lustre* Software User Guide*
- *Hierarchical Storage Management Configuration Guide*
- *Configuring LNet Routers for File Systems based on Intel® EE for Lustre* Software*
- *Installing Hadoop, the Hadoop Adapter for Intel® EE for Lustre*, and the Job Scheduler Integration*

- *Creating an HBase Cluster and Integrating Hive on an Intel® EE for Lustre® File System*
- *Creating a High-Availability Lustre* Storage Solution over a ZFS File System*
- *Upgrading a Lustre file system to Intel® Enterprise Edition for Lustre* software (Lustre only)*
- *Creating a Scalable File Service for Windows Networks using Intel® EE for Lustre* Software*
- *Intel® EE for Lustre* Hierarchical Storage Management Framework White Paper*
- *Architecting a High-Performance Storage System White Paper*

What is Multi-Rail?

LNet currently supports one NID (network interface device) per node. This can represent a bandwidth bottleneck for Lustre nodes that are equipped with multiple CPUs and place heavy demands on LNet. Generically, multi-rail is a computer networking arrangement in which two or more network interfaces to a single network on a computer node are employed, to achieve increased throughput and redundancy. Multi-rail can also be where a node has one or more interfaces to multiple, even different kinds of networks, such as Ethernet, Infiniband, and Intel® Omni-Path. For Lustre clients, Multi-rail generally presents the combined network capabilities as a single LNet network. Peer nodes that are multi-rail capable are established during configuration, as are user-defined interface-section policies.

Configuring Multi-Rail

Every node using multi-rail networking needs to be properly configured. Multi-rail uses lnetctl and DLC for configuration. Configuring multi-rail for a given node involves two tasks:

1. Configuring multiple network interfaces present on the local node.
2. Adding remote peers that are multi-rail capable (are connected to one or more common networks with at least two interfaces).

Configure Multiple Interfaces on the Local Node

The `lnetctl` command is normally used to configure LNet interfaces. Following are `lnetctl` command parameters that are used to configure multi-rail interfaces for the local node.

http://wiki.lustre.org/images/b/bb/Multi-Rail_High-Level_Design_20150119.pdf

https://cug.org/proceedings/cug2016_proceedings/includes/files/pap153.pdf

From the lnetctl command we have these parameters:

```
net add: add a network
        --net: net ID (e.g. tcp0)
        --if: physical interface (e.g. eth0)
        --ip2net: specify networks based on IP address patterns
        --peer-timeout: time to wait before declaring a peer dead
        --peer-credits: define the max number of inflight messages
        --peer-buffer-credits: the number of buffer credits per peer
        --credits: Network Interface credits
        --cpt: CPU Partitions configured net uses (e.g. [0,1])
```

With multi-rail,

- `--net` - specifies the network type and number. Specifically, `tcp` specifies Ethernet, `o2ib` specifies Infiniband, and `gni` specifies? Note that this *no longer needs to be unique*, because multiple interfaces can be added to the same network.
- `--if` - the same interface per network can be added only once, however more than one interface can be specified (separated by a comma) for this node. For example: `eth0,eth1,eth2`

Following is the syntax for the `lnetctl` command to create a network interface with configuration parameters:

```
lnetctl > net add -h
Usage: net add --net <network> --if <interface> [--peer-timeout <seconds>]
        [--ip2nets <pattern>]
        [--peer-credits <credits>] [--peer-buffer-credits <credits>]
        [--credits <credits>] [--cpt <partition list>]
```

From YAML

```
net:
  - net type: <network>
   local NI(s):
     - nid: <nid>
       tunables:
        peer_timeout:  <Int. Timeout before consider a peer dead>
        peer_credits: <Int. Transmit credits for a peer>
        peer_buffer_credits: <Int. Credits available for receiving msgs>
        credits: <Integer. Network Interface credits>
        tcp bonding: <0 - use Multi-Rail. 1 - Use existing TCP bonding>
           CPT: "[<comma separated CPT>]"
```

Example of YAML net show

```
lnetctl net show -v

net:
    - net type: lo
      local NI(s):
        - nid: 0@lo
          status: up
          dev cpt: 0
          statistics:
              send_count: 0
              recv_count: 0
              drop_count: 0
```

```
        tunables:
            peer_timeout: 0
            peer_credits: 0
            peer_buffer_credits: 0
            credits: 0
            tcp bonding: 0
            CPT: "[0,1,2,3]"
- net type: tcp
  local NI(s):
    - nid: 192.168.122.199@tcp
      status: up
      dev cpt: -1
      statistics:
          send_count: 7845
          recv_count: 7047
          drop_count: 0
      tunables:
          peer_timeout: 180
          peer_credits: 8
          peer_buffer_credits: 0
          credits: 256
          tcp bonding: 0
          CPT: "[0,1,2,3]"
    - nid: 192.168.122.199@tcp
      status: up
      interfaces:
          0: eth0
      dev cpt: -1
      statistics:
          send_count: 4327
          recv_count: 2036
          drop_count: 0
      tunables:
          peer_timeout: 180
          peer_credits: 8
          peer_buffer_credits: 0
          credits: 256
          tcp bonding: 0
          CPT: "[0,1,2,3]"
```

Deleting Network Interfaces

To remain backward compatible, two forms of the delete command command have been implemented. The first delete command deletes the entire network and all network interfaces under it. The second delete command deletes a single network interface:

```
lnetctl > net del -h
net del: delete a network
Usage: net del --net <network> [--if <interface>]
```

where:

```
 --net: net name (e.g. tcp0)
 --if: interface name. (e.g. eth0)
```

The `--if` option can specify a comma separated list of interfaces to be removed. If the `--if` is omitted then the entire network is deleted.

Example

```
lnetctl net del --net tcp --if eth0
```

This can be achieved through YAML by the exact same YAML block shown above.

Adding Remote Peers that are Multi-Rail Capable

When configuring peers, use the `–key_nid` option to specify the key or primary nid of the peer node. Then follow that with the `--nid` option to specify a set of comma separated NIDs.

The `--key-nid` (primary nid for the peer node) can go unspecified. In this case, the first listed NID in the `--nid` option becomes the primary nid of the peer.

```
lnetctl > peer add -h
Usage: peer add --key_nid <nid> --nid <nid[, nid, ...]>
```

where:

```
peer add: add a peer
        --key_nid: Key or primary NID of the peer
        --nid: comma separated list of peer nids (e.g. 10.1.1.2@tcp0)
```

Example

```
lnetctl peer add --key_nid 10.10.10.2@tcp --nid
10.10.3.3@tcp1,10.4.4.5@tcp2
```

```
lnetctl peer_add --nid 10.10.10.2@tcp,10.10.3.3@tcp1,10.4.4.5@tcp2
```

Using YAML to Configure Peers

```
peer:
    - primary nid: <key or primary nid>
      Multi-Rail: True
      peer ni:
```

```
        - nid: <nid 1>
        - nid: <nid 2>
        - nid: <nid n>
```

As with all other commands, the result of the show command can be used to configure or delete the peer:

```
lnetctl peer show -v
```

Example

```
peer:
    - primary nid: 192.168.122.218@tcp
      Multi-Rail: True
      peer ni:
        - nid: 192.168.122.218@tcp
          state: NA
          max_ni_tx_credits: 8
          available_tx_credits: 8
          available_rtr_credits: 8
          min_rtr_credits: -1
          tx_q_num_of_buf: 0
          send_count: 6819
          recv_count: 6264
          drop_count: 0
          refcount: 1
        - nid: 192.168.122.78@tcp
          state: NA
          max_ni_tx_credits: 8
          available_tx_credits: 8
          available_rtr_credits: 8
          min_rtr_credits: -1
          tx_q_num_of_buf: 0
          send_count: 7061
          recv_count: 6273
          drop_count: 0
          refcount: 1
        - nid: 192.168.122.96@tcp
          state: NA
          max_ni_tx_credits: 8
          available_tx_credits: 8
          available_rtr_credits: 8
          min_rtr_credits: -1
          tx_q_num_of_buf: 0
          send_count: 6939
          recv_count: 6286
          drop_count: 0
          refcount: 1
```

Deleting a Peer

Use the following command to delete a peer:

```
lnetctl > peer del –h
```

where:

`key_nid` should always be specified. The `key_nid` identifies the peer. If the `key_nid` is the only one specified, then the entire peer is deleted.

```
peer del: delete a peer
   --key_nid: key_nid of the peer
   --nid: comma separated list of peer nids (e.g. 10.1.1.2@tcp0)
```

Examples

To delete `10.10.10.3@tcp`:

```
lnetctl peer del --key_nid 10.10.10.2@tcp --nid 10.10.10.3@tcp
```

To delete the entire peer:

```
lnetctl peer del --key_nid 10.10.10.2@tcp
```

Notes on Network and peer interaction

A peer can have some of its interfaces on a non-local network. These network interfaces are identified as not configured and will not be used.

After a network/interface is configured, it becomes available to the Multi-rail pool of network interfaces. If a network is deleted, all peer network interfaces present on that peer are removed.

ip2nets

Multi-rail deprecates the kernel parsing of ip2nets. ip2nets patterns are matched in user space and translated into Network interfaces to be added into the system.

The first interface that matches the IP pattern will be used when adding a network interface.

If an interface is explicity specified as well as a pattern, the interface matched using the IP pattern will be sanitized against the explicitly-defined interface.

For example, `tcp(eth0) 192.168.*.3` and there exists in the system `eth0 == 192.158.19.3` and `eth1 == 192.168.3.3`, then the configuration will fail, because the pattern contradicts the interface specified.

A clear warning will be displayed if inconsistent configuration is encountered.

To configure ip2nets

You could use the following command to configure ip2nets:

```
lnetctl import < ip2nets.yaml
```

Example

```
ip2nets:
  - net-spec: tcp1
    interfaces:
         0: eth0
         1: eth1
    ip-range:
         0: 192.168.*.19
         1: 192.168.100.105
  - net-spec: tcp2
    interfaces:
         0: eth2
    ip-range:
         0: 192.168.*.*
```