Highly Scalable Networking Configuration Management For Highly Scalable Systems

Nicholas P. Cardo
Lawrence Berkeley National Laboratory
National Energy Research Scientific Computing Center
cardo@nersc.gov

ABSTRACT: *Today's systems have large numbers of specialized nodes each requiring unique network configurations. On the XT4 at the National Energy Research Scientific Computing Center, there are 56 such nodes each requiring unique network addresses and network routes. Normal network management would be to specialize files to be unique to each node. A simpler mechanism with increased flexibility was needed. With very little initial set-up, all network addresses and routes can be maintained through two common files, one for routes and one for addresses. Changes to the configuration now only require simple edits to these two files. The details of configuring this environment and simplicity of its management will be discussed in this paper.*

KEYWORDS: XT4, Networking

1 Introduction

The system management of a single Unix system is relatively straightforward. However, when the system is made up of many individual Unix systems that when combined form a more complicated and powerful system, the overall management increases in difficulty.

Networking strategies are among the complicated components of today's high performance computer systems. Large-scale computers no longer have single network interfaces but rather a complicated array of interfaces spread out across numerous nodes within the computer. To further complicate matters these interfaces could be of a variety of types including Gigabit Ethernet, 10 Gigabit Ethernet and SeaStar.

As computer systems grow in scale, so does the overhead in configuring and managing the system. System Administrators of today are constantly finding new ways to simplify the pieces of overall computer system management.

2 The Problem

The problem is a combination of standard Linux networking, system scale and the unique environment of the Cray XT4.

2.1 Linux Networking

The directory `/etc/sysconfig/network` is the root level directory for SuSE network interface configuration files. Suppose the device `eth0` needs to be configured. Changes would be made to the file `ifcfg-eth0` to supply the necessary configuration information. This includes such configuration details as interface address and netmask.

One configuration file exists for each interface on the host. It is necessary to edit one file per interface to be configured. Suppose the host has four interfaces, then four files require configuration edits.

2.2 The Scaling Issue

Configuring network interfaces is a relatively trivial task. Simply edit the configuration file for the interface and it is done. Just repeat the process for each interface on that node in the cluster. Then, move on to the next node and repeat the process, continuing until all nodes

with interfaces have been configured. Suppose the cluster contains 50 nodes to be configured with network interfaces. Suppose it takes five minutes per node to make and verify the changes. Scaling this to 50 nodes would take 250 minutes or 4 hours and 10 minutes.

Each time a change is required in the future, the same manual effort is required.

2.3 The XT4 Complication

To further complicate matters, the Cray XT4 has a unique shared root environment where changes must be made. Suppose the XT4 system had nodes 5 through 10 each with an eth0 device to configure. From the boot node, it would be necessary to run `xtopview –n <nid>`, then edit the files, then commit the changes. This process is required for each node to be configured. If the file is not specialized to that node, then an additional step is required to complete the specialization.

The NERSC XT4 has 56 service nodes requiring configuration. It is hoped that once the initial configuration is in place, it will not change. However, the general rule in system management is that if it can be changed, it will be changed.

2.4 To Sum it Up

While each piece is relatively trivial, the combination makes this process unacceptably complicated and tedious. As systems grow in scale, and the demand on networking increases, so the impact of configuring and maintaining network interfaces. As the number of changes increase, so does the change of error.

3 The Solution

A configuration methodology is needed that can work as simple as editing a file while scaling up to thousands of nodes. The configuration specifications need also be flexible enough to adapt to more complicated configurations.

3.1 Goals

To measure the success of this project, the following objectives were established:

- Single points of data entry for routing and interface configurations.
- Scalable to support all service nodes.
- Flexible to handle complex routing if necessary.
- Minimal specialization.

3.2 Concept

The basic premise behind the solution is to utilize common files that can provide the functionality across all service nodes. Utilize a common directory that contains all the network configuration files for the entire system. All nodes should point to the files in this common location.

All network interfaces for the entire system should be defined in a common text file. One file would contain all the interfaces for all the nodes. Startup scripts would need to be modified to support such a file.

All network routes for the entire system should be defined in a common tet file. One file would contain all the routes for all the nodes. Startup files would need to be modified to support such a file. Processing of routes needs to be automatic and therefore the need for post processing script once and interface has been configured up.

4 Implementation

Although one objective was to minimize the per node specialization of files, some set-up changes are required. The extent of these changes is only to point key system start-up files to a common file across all nodes.

4.1 Configuration Requirements

All of Franklin's service nodes require publicly addressable SeaStar interface addresses. This is accomplished by adding a second address to the interface. The primary address is private and assigned as part of the system's build and boot. Both addresses will function on the same interface. The interface device designation is "`ss`".

Each of Franklin's login nodes contains a dual port Gigabit Ethernet card. These interfaces are designated by the devices "`eth0`" and "`eth1`".

Furthermore, each node may have unique network traffic routing rules. However, these rules cannot be specialized to individual nodes.

For performance reasons, the internal addresses need to be pre-loaded into the ARP table at boot time.

Meanwhile, any custom routes need to be set as well as any special network parameters. This can all be done by the `POST_UP_SCRIPT` for the device being configured.

4.2 Initial Set-up

As mentioned earlier, each interface requires an initialization file in order to configure the device at system boot. Additionally, pre-loading of the ARP table needs to occur.

Basically the four files required to accomplish this function need to be the same four files across the system. This could be accomplished by modifying the specialized

file. However, subsequent changes over time would again require the need to use `xtopview` on the boot node. The solution chosen was to turn these files into symbolic links to a common area that holds the real start-up files. This has the added benefit of consolidated all network configuration files for the entire system into one directory visible on all service nodes.

The first step is to create a directory for network configuration files. On Franklin, the directory `/etc/NERSC` was created for this purpose.

As already identified, there are three devices to configure: `ss`, `eth0`, and `eth1`. Their respective configuration files would be located in /etc/sysconfig/network and are named `ifcfg-ss`, `ifcfg-eth0`, and `eth1`. To begin create symbolic links for the device configuration files to files of the same name located in `/etc/NERSC`.

```
# ln –s /etc/NERSC/ifcfg-ss ifcfg-ss
# ln –s /etc/NERSC/ifcfg-eth0 ifcfg-eth0
# ln –s /etc/NERSC/ifcfg-eth1 ifcfg-eth1
```

The next step is to do the same for the files required to preload the ARP table and set any special routes for that node. This script is called `set-route-arp` and is located in `/etc/sysconfig/network/scripts`. A symbolic link is again created for this file.

```
# ln –s /etc/NERSC/set-route-arp set-route-arp
```

4.3 Address and Route Configuration Files

Two configuration files are used to define the full network configuration of the system.

The first file, hosts-external, represents all the interface configurations. It contains all the necessary information to configure any of the network devices on all of the service nodes. The file contains 7 fields:

1. Hostname
2. Physical address
3. Node identifying name
4. Interface device (`ss`, `eth0`, `eth1`)
5. Network address
6. Netmask
7. Device MTU

These fields provide sufficient information to configure the interfaces. Additional fields could easily be added as needed. The following example shows how to configure one of Franklin's login nodes which requires both the `ss` and `eth0` devices to be configured. There is one line per device to be configured.

```
nid04100 c1-0c0s1n0 login01 eth0 128.55.81.34  255.255.248.0   9000
nid04100 c1-0c0s1n0 login01 ss   128.55.42.134 255.255.255.192 NA
```

The second file contains any special route specifications. This file contains 9 fields:

1. Hostname
2. Physical address
3. Node identifying name
4. Interface device (`ss`, `eth0`, `eth1`)
5. Destination address
6. Gateway
7. Netmask
8. Route MTU
9. IPforward

While these fields provide sufficient information to configure routing, additional fields could easily be added. The login nodes for Franklin contain two routes, a default route and a network specific route. The entries look like:

```
nid04100 c1-0c0s1n0 login01 eth0 default 128.55.80.1 NA  NA   0
nid04100 c1-0c0s1n0 login01 ss 128.55.32.0 128.55.42.130 255.255.224.0
9000 0
```

4.4 Interface Configuration

The device configuration files need to be modified to extract their required information from the `hosts-external` file. The `ifcfg-eth0` file looks like:

```
NID=`cat /proc/cray_xt/nid | \
     awk '{printf("nid%5.5d\n",$1)}'`

DEVICE="eth0"
BOOTPROTO='static'
IPADDR=`grep $NID /etc/NERSC/hosts-external | \
        grep -v "^#" | grep $DEVICE | \
        awk '{ print $5 }'`

#
#  If no address, then stop
#
if [ "$IPADDR" = "" ]
then
        exit 0
fi

GATEWAY=
NETMASK=`grep $NID /etc/NERSC/hosts-external | \
         grep -v "^#" | grep $DEVICE | \
         awk '{ print $6 }'`
NETWORK=
MTU=`grep $NID /etc/NERSC/hosts-external | \
     grep -v "^#" | grep $DEVICE | \
     awk '{ print $7 }'`
STARTMODE='onboot'
POST_UP_SCRIPT="set-route-arp"
```

Other device files are identical with the exception of the DEVICE field. This is adjusted to represent the device to be configured.

4.5 Post Device Configuration

```
#
#  Identify this node
#
NID=`cat /proc/cray_xt/nid | \
     awk '{printf("nid%5.5d\n",$1)}'`
#
#  The file containing all the routes
#
ROUTESEXTERNAL=/etc/NERSC/routes-external
#
#  Populate the arp table
#
/sbin/arp -f /etc/NERSC/arp-ss -i ss
#
#  Loop through all route entries for this nid
#
for ln in `fgrep $NID $ROUTESEXTERNAL | \
           egrep -v "^#" | \
       awk '{print $4":"$5":"$6":"$7":"$8":"$9}'
do
        #
        #  Parse the fields
        #
        DEV=`echo $ln | cut -f 1 -d :`
        ADR=`echo $ln | cut -f 2 -d :`
        GWY=`echo $ln | cut -f 3 -d :`
        MSK=`echo $ln | cut -f 4 -d :`
        MTU=`echo $ln | cut -f 5 -d :`
        FOR=`echo $ln | cut -f 6 -d :`
        #
        #  Get the host addr from the full addr
        #
        HST=`echo $ADR | cut -f 4 -d \.`
        #
        # is this is a host or net route
        #
        if [ "$ADR" = "default" ]
        then
                TYP=""
        elif [ "$HST" = "0" ]
        then
                TYP="-net"
        else
                TYP="-host"
        fi
        #
        #  Set the netmask if applicable
        #
        if [ "$MSK" = "NA" ]
        then
                L_MSK=""
        else
                L_MSK="netmask $MSK"
        fi
        #
        #  Set the MTU if applicable
        #
        if [ "$MTU" = "NA" ]
        then
                L_MTU=""
        else
                L_MTU="mtu $MTU"
        fi
        #
        #  Now create the route
        #
        /sbin/route add $TYP $ADR gw $GWY \
                        $L_MSK dev $DEV $L_MTU
        #
        #  Set IP Forwarding appropriately
        #
        echo $FOR >/proc/sys/net/ipv4/ip_forward
        #
        #  Set arp flag for ss interface #
        if [ \( "$DEV" = "ss" \) -a $FOR -eq 1 ]
        then
                /sbin/ip link set $DEV arp on
        fi
done
```

5 Summary

Once the initial set-up is complete, changes only need to be made to the hosts-external and routes-external file. This has greatly simplified the number of modifications required each time a network configuration change is required.

6 Acknowledgments

This work was supported by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract number DE-AC02-05CH11231.

This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy.

This work is an adaptation of the methodology developed at Sandia National Laboratory.

7 About the Author

Nicholas P. Cardo is the Project Lead and Lead System Administrator of Franklin. He is a senior member of the Computational Systems Group at NERSC. He can be reached at Lawrence Berkeley National Laboratory, National Energy Research Scientific Computing Center, 1 Cyclotron Rd, bldg 943r0256, Berkeley, CA 94720 USA, E-mail: cardo@nersc.gov.