Management Server Configuration Guide

- Rev 0.1
- 2015-09-11
- NEC Corporation

Overview

This document describes how to setup a management server for the Mercury cluster.

Features

The management server provides the following features:

- dhcp server
- tftp server
- http server
- pxeboot server
- kickstart server
- nfs server
- ntp server
- yum repository
- http proxy server
- build environment of rpm packages
- build environment of RAMCloud and DPDK

Installation and Configuration

OS

Install CentOS 7.1 x86_64 with minimum packages.

Basic configuration

- setup `admin` account
  - enable to `sudo` to `root` without password
  - generate ssh keypair without passphrase
- disable SELinux
- mount the OS image `CentOS-7-x86_64-Minimal-1503-01.iso` to `/var/dist/centos/7`

Routing and Firewall

Disable routing between the Internet and the Mercury.

Configure firewalld as follows:

- enable `firewalld`
- bind the interface connected to the Internet to the `public` zone
- bind all the interfaces connected to the Mercury to `trusted` zone

Network configuration

```
internet -- [1] -- mgmt server --+-- [2] -- Mercury Control
                                 +-- [3] -- Mercury Data
                                 +-- [4] -- Mercury BMC
```

- `[1]`: external interface
- `[2]`: control lan interface
- `[3]`: data lan interface
- `[4]`: bmc lan interface

Proxy client

Configure http proxy client environment if required.

- set `http_proxy` environment variable
- confiure sudo to preserve the `http_proxy` environment variable.
- install and configure `corkscrew` to access external git repositories with ssh over firewall

yum configuration

- disable to update the kernel related packages
  - add `exclude=kernel*` to `/etc/yum.conf`
- add proxy config to `/etc/yum.conf` if required

NTP client

install and setup ntp client (chronyd) as follows:

- install `chrony` package
- configure chronyd to run as a ntp client

NTP server

install and setup ntp server (chronyd) as follows:

- install and enable `chrony` package
- configure chronyd to run as a ntp server
- configure chronyd to allow query from the Mercury

dhcp server

install and setup dhcp server as follows:

- install following packages:
    - `dhcp`
- assngin fixed address to all the atom host on the Mercury.

tftp server

install and setup tftp server as follows:

- install following packages:
    - `xinetd`
    - `tftp-server`
- enable `xinetd` service
- enable `tftp` service in the xinetd configuration
- configure server root to `/var/lib/tftproot`

http server

install and setup http server as follows:

- install following packages:
    - `httpd`
- enable `httpd` service

server configuration is described in later section.

pxeboot server

configure pxeboot server as follows:

- install following packages:
    - `syslinux`
- configure dhcp server to deliver the pxeboot information
    - next-server: address of the management server
    - filename: `/pxelinux.0`
- deploy the contents of the tftp server root (`/var/lib/tftpboot/`):

    - copy {initrd.img,vmlinuz} from `/var/dist/centos/7/images/pxeboot/`
    - copy pxelinux.0 from `/usr/share/syslinux/`
    - create `pxelinux.cfg/default`; see Appendix for details.

directory structure of the pxeboot environment is as follows:

```
/var/lib/tftpboot/
  initrd.img
  vmlinuz
  pxelinux.0
  pxelinux.cfg/default.cfg
```

kickstart server

configure kickstart server as follows:

- create directories for the kickstart server tree
    - /var/www/kickstart
    - /var/dist
- make sure the OS image is mounted on `/var/dist/centos/7`
- configure http server
    - map the url `/ks` to the path `/var/www/kickstart`
    - map the url `/dist` to the path `/var/dist`
- deploy kickstart files
    - create `/var/dist/kickstart/kickstart.cfg`. see Appendix for the contents of `kickstart.cfg` file.
    - copy the admin user's public key to `/var/dist/kickstart/admin.pub`
- make sure all the files and directories are readable by httpd
- configure httpd to permit access from all the hosts on the Mercury

directory structure of the kickstart environment is as follows:

```
/var/www/kickstart/
  kickstart.cfg
  admin.pub
  hosts
  ssh_known_hosts

/var/dist/
  CentOS-7-x86_64-Minimal-1503-01.iso
  md5sum.txt

/var/dist/centos/7/   # mount /var/dist/CentOS-7-x86_64-Minimal-1503-01.iso
  CentOS_BuildTag
  ...
```

HTTP proxy server

setup http forward proxy server on the management server so that the atom hosts on the Mercury can access yum repositories on the Internet.

configure `proxy_module` of the apache http server as follows:

- enable forward proxy
- configure the upstream proxy server
- listen on 8080 port
- allow access from the Mercury hosts
- deny access from the Internet
- set noproxy to localhost and local network

refer to http://www.apache.org/ for details.

NFS Server

Management server exports the `/home` partition over NFS to the Mercury hosts.

setup nfs server as follows:

- install following packages:
  - `nfs-utils`
- configure `/etc/exports` to export `/home` to the Mercury hosts
- enable following services:
  - `rpcbind`
  - `nfs-server`

IPMI

- install following packages:
  - `ipmitool`
- create the account files used by ipmitool to access BMCs on the Mercury hosts
  - `$HOME/.atom/ipmi_user.txt`
  - `$HOME/.atom/ipmi_password.txt`
- set permission of those files to 600

refer to the Mercury Documentation (?) for the account information to access BMCs.

local yum repisitory

create and configure local yum repository to distribute the dpdk package to the Mercury hosts.

- install following packages:

    - `createrepo`
- create repository tree
    - `/var/rpms/local/centos/7/noarch`
    - `/var/rpms/local/centos/7/x86_64`
- initialize repository using `createrepo`
    - `createrepo /var/rpms/local/centos/7`
- create the repository configuration file
    - see Appendix for details
- deploy the configuration file to `/etc/yum.repos.d/local.repo`
- deploy the configuration file to `/var/rpms/local.repo`
- configure http server to serve the repository
    - map url `/repo` to path `/var/rpms`
    - allow access to `/repo` from the Mercury hosts
- enable and restart the http service
- make sure the http server can read the repository files

directory structure of the local yum repository is as follows:

```
/var/rpms/
  local.repo
/var/rpms/local/centos/7/
  noarch/
  repodata/                # generated by createrepo
    ...
  x86_64/
    dpdk-2.0.0-xxxxxxxx.x86_64.rpm
    dpdk-devel-2.0.0-xxxxxxxx.x86_64.rpm
    dpdk-debuginfo-2.0.0-xxxxxxxx.x86_64.rpm
```

other packages

install the following packages:

- `git`
- `libselinux-utils`
- ... [TBD]

DPDK build environment

install required packages to build DPDK.

First, install the kernel related packages from the `base` repository.

- `kernel-devel`
- `kernel-headers`

To install them, temporary disables the `exclude` directive in `/etc/yum.conf` and disable all repository except for the `base`.

```
sudo yum install -y --disablerepo=\* --enablerepo=base \
  --disableexcludes=all install kernel-devel kernel-headers
```

Next, install the required packages to build dpdk.

- `Development Tools`
- `epel-release`
- `glibc.i686`
- `glibc-devel.i686`
- `libgcc.i686`
- `libpcap-devel`
- `doxygen`
- `pexpect`
- `wget`
- `curl`
- `fuse-devel`
- `libvirt-devel`

refer to the official site of the DPDK project http://dpdk.org/ for details.

RAMCloud build environment

install required packages to build RAMCloud.

- `git`
- `protobuf-devel`
- `protobuf-c-devel`
- `boost-devel`
- `pcre-devel`
- `openssl-devel`

`zookeeper` package is required to build RAMCloud but the rpm package of it does not seem to be provided. you can build and install the zookeeper from source as follows:

```
mkdir $tmpdir; cd $tmpdir
zookeeper=zookeeper-3.4.6
curl -s -L -o $zookeeper.tar.gz http://ftp.riken.jp/net/apache/zookeeper/$zookeeper/$zookeeper
tar xf $zookeeper.tar.gz
cd $zookeeper/src/c
CC=gcc44 ./configure
sudo make install
echo "/usr/local/lib" | sudo tee /etc/ld.so.conf.d/userlocal.conf
sudo ldconfig
```

add `--proxy <proxy_url>` to curl command line if the management server is behind the firewall.

refer to the official site of the RAMCloud project
http://ramcloud.atlassian.net/wiki/display/RAM/RAMCloud

Compatibility

default compiler on CentOS-7 is gcc-4.8, and RAMCloud can not build with gcc-4.8.

as a workaround, install gcc-4.4 to CentOS-7 and build RAMCloud with it. install the following packages from the epel repository and configure environment variables CC and CXX to refer the 4.4 compiler.

- `epel-release`
- `compat-gcc-44`
- `compat-gcc-44-c++`

rpm build environment

install required packages to build rpm packages and setup the rpm build environment.

- install the following packages:
    - `epel-release`
    - `rpmdevtools`
    - `rpm-build`
    - `rpm-devel`
    - `yum-utils`
- setup the rpm build directories under `~/rpmbuild`
    - run `rpmdev-setuptree` command

build DPDK and create rpm packages

refer to the official site of the DPDK project http://dpdk.org/ for details.

deploy to the local yum repository

- copy the rpm packages to `/var/rpms/local/centos/7/x86_64`
- update metadata of repository with `createrepo` command
    - `createrepo -v --update /var/rpms/local/centos/7/x86_64`

build RAMCloud

refer to the official site of the RAMCloud project

mercury specific scripts

[TBD]

```
git clone [repo_url]
cd [repo_dir]
sudo make install
```

Mercury configuration

refer to the sg document for more details.

collect the configuration info

First, scan all the Mercury hosts and collect the configuration information as follows:

- mac addresses
  - control lan interface
  - data lan interface
  - bmc lan interface
- chassis and slot number

this scan is performed the following sequence:

- setup temporary dhcp server for bmc lan with following option
  - `set vendor-string = option vendor-encapsulated-options;`
- power on the Mercury
- detect the CMM master based on the returned info from all clients
- query the bmc addresse of Mercury hosts to CMM
- query the config info to the bmc of Mercury hosts

create the dhcpd.conf

based on the configuration info collected by above sequence, create dhcpd.conf that assigns a static ip address and predefined hostname to the bmc and the control interface of each Mercury hosts.

replace the dhcp configuration to the newly generated one and restart dhcp server.

create the /etc/hosts

create /etc/hosts based on the config info, the same as dhcpd.conf.

/etc/hosts should be the following format:

```
192.168.3.1<tab>atom001<tab>#74:d4:35:1e:88:62
192.168.4.1<tab>atom001a<tab>#74:d4:35:1e:88:63
192.168.5.1<tab>atom001m<tab>#74:d4:35:1e:0c:72
...
```

deploy the OS to the Mercury hosts

boot the host with pxeboot mode by sending ipmi command. after boot up, the OS is installed to the host automatically by the kickstart server.

ssh_known_hosts

gather the host keys of the cluster hosts with `ssh-keyscan` and save them to `/etc/ssh/ssh_known_hosts`

---

configure the cluster hosts

after installation, the following configurations are required on each cluster host. refer to Appendix for details.

OS environment

- disable selinux
- disable firewalld
- get hostname from dhcp server
- disable requiretty
- configure sudoers policy
- fixup modes of sudoers config files
- install /etc/hosts
- configure yum
- configure the local yum repository
- fetch metadata of the local repo
- install required packages
- install kernel related packages from base repo.
- configure nfs client
- configure chrony daemon
- configure sshd
- enable latency-performance configuration
- restart tuned service
- disable cpupower service
- enable acpid for ipmi power soft
- disable irqbalance

- enable/restart chrony daemon
- enable/restart sshd service
- mount /home of management server over nfs

DPDK runtime

- install dpdk package from the local repository
- configure dpdk
- add group dpdk to admin users groups
- enable and start dpdk

dpdk specific environment is configured in the init script of the dpdk package.

RAMCloud runtime

- create ramcloud group
- create ramcloud user
- add group ramcloud to admin users groups
- set the soft limit of memlock to unlimited
- set the hard limit of memlock to unlimited
- create ramcloud runtime directories
- install packages required to run ramcloud
- create ramcloud group
- create ramcloud user

Users guide

refer to the mmatom document for details.

- RAMCloud in a Box - ATOM based Micro Modular Server 'mmatom'

mmuser

create a user on all the Mercury hosts based on a user attributes on the management server.

```
mmuser [username]
```

[username] should be exists on the management server.

mmres

usage of mmres is almost the same as rcres. refer to the documentation of rcres for details.

Allcheck.sh

list the configuration informations on the cluster hosts.

```
$ AllCheck.sh
SlotNo   IPAddress        [MAC Address]     F/W   BIOS                GUID
CMM M  mercury1cmm  [74:d4:35:1e:84:34]  1.17  ----  80e8131c-7882-e311-0180-74d4351e8435  74:
CMM S  192.168.5.194   [74:d4:35:1e:83:cc]  1.17  ----  00d08e20-7d82-e311-0180-74d4351e83cd
Slot01 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot02 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot03 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot04 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot05 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot06 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot07 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot08 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot09 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot10 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot11 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot12 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot13 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot14 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot15 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot16 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot17 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot18 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot19 192.168.5.1     [74:d4:35:1e:0c:72]  1.17  0007  00c1460c-1d77-e311-0180-74d4351e8862
Slot20 192.168.5.2     [74:d4:35:1e:0c:ad]  1.17  0007  8012719f-8cd6-e311-0180-74d4351e8a38
Slot21 192.168.5.3     [74:d4:35:1e:0b:32]  1.17  0007  0081cbe2-8cd6-e311-0180-74d4351e89a2
Slot22 192.168.5.4     [74:d4:35:1e:08:20]  1.17  0007  00f1031c-8dd6-e311-0180-74d4351e85e2
Slot23 192.168.5.5     [74:d4:35:1e:0d:2a]  1.17  0007  80c16f91-8dd6-e311-0180-74d4351e89c2
Slot24 192.168.5.6     [74:d4:35:1e:0c:a2]  1.17  0007  009ecbb5-8dd6-e311-0180-74d4351e8a22
Slot25 192.168.5.7     [74:d4:35:1e:07:32]  1.17  0007  80b7e617-1d77-e311-0180-74d4351e8826
Slot26 192.168.5.8     [74:d4:35:1e:0b:0e]  1.17  0007  00d667d2-8dd6-e311-0180-74d4351e87b6
Slot27 192.168.5.9     [74:d4:35:1e:0d:0d]  1.17  0007  00ae646d-96d6-e311-0180-74d4351e84ba
Slot28 192.168.5.10    [74:d4:35:1e:0b:7e]  1.17  0007  003436d6-4e6d-e311-0180-74d4351e89d6
Slot29 192.168.5.11    [74:d4:35:1e:08:12]  1.17  0007  80374b24-4f6d-e311-0180-74d4351e867e
Slot30 192.168.5.12    [74:d4:35:1e:0c:80]  1.17  0007  002d3627-4bd6-e311-0180-74d4351e87d4
Slot31 192.168.5.13    [74:d4:35:1e:0a:d7]  1.17  0007  008626b0-96d6-e311-0180-74d4351e8924
Slot32 192.168.5.14    [74:d4:35:1e:0c:b8]  1.17  0007  005e26f7-2c77-e311-0180-74d4351e8754
Slot33 192.168.5.15    [74:d4:35:1e:0c:0e]  1.17  0007  00904a3c-2d77-e311-0180-74d4351e857e
Slot34 192.168.5.16    [74:d4:35:1e:0b:cf]  1.17  0007  809e27c4-1b77-e311-0180-74d4351e8a7c
Slot35 192.168.5.17    [74:d4:35:1e:0c:c2]  1.17  0007  80e2dd14-8fd6-e311-0180-74d4351e85fc
Slot36 192.168.5.18    [74:d4:35:1e:07:26]  1.17  0007  801f6d75-8fd6-e311-0180-74d4351e8618
Slot37 192.168.5.19    [74:d4:35:1e:0b:50]  1.17  0007  8017be45-8fd6-e311-0180-74d4351e8504
Slot38 192.168.5.20    [74:d4:35:1e:08:14]  1.17  0007  801f6d75-8fd6-e311-0180-74d4351e86ee
Slot39 192.168.5.21    [74:d4:35:1e:08:13]  1.17  0007  80b91ac4-8fd6-e311-0180-74d4351e86fa
Slot40 192.168.5.22    [74:d4:35:1e:08:23]  1.17  0007  80fda68f-8fd6-e311-0180-74d4351e86c8
```

```
Slot41 192.168.5.23    [74:d4:35:1e:0c:89]  1.17  0007  003f9ef7-7baa-e411-0180-74d4351e88d0
Slot42 192.168.5.24    [74:d4:35:1e:0c:0c]  1.17  0007  0056a515-386d-e311-0180-74d4351e857c
Slot43 192.168.5.25    [74:d4:35:1e:0b:aa]  1.17  0007  80a80d4c-1c77-e311-0180-74d4351e8a74
Slot44 192.168.5.26    [74:d4:35:1e:0c:0b]  1.17  0007  802c1957-306e-e311-0180-74d4351e880c
Slot45 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
Slot46 -.-.-.-         [--:--:--:--:--:--]  ----  ----  ------------------------------------
```

ipmiaw2

send a ipmi command to the cluster hosts.

```
$ ipmiaw2 atom001m-atom003m power status
RUNNING [ipmitool -I lanplus -U Administrator -f $HOME/.atom/ipmi_password.txt -H atom001m  po
Chassis Power is on
RUNNING [ipmitool -I lanplus -U Administrator -f $HOME/.atom/ipmi_password.txt -H atom002m  po
Chassis Power is on
RUNNING [ipmitool -I lanplus -U Administrator -f $HOME/.atom/ipmi_password.txt -H atom003m  po
Chassis Power is on
```

PWOFFALL.sh, PWONALL.sh

PWOFFALL.sh and PWONALL.sh control the power status of the mercury.

```
$ PWONALL.sh
192.168.5.192
RUNNING [ipmitool -I lanplus -U Administrator -f $HOME/.atom/ipmi_password.txt -H 192.168.5.19
 ff
```

note that these commands does not control the power status of each cluster host. to power on a cluster host, send ipmi power on command to the host using ipmiaw2.

mmzap

mmzap arranges pxeboot config to cluster hosts to install the OS.

```
$ mmzap atom001m-atom003m
```

note that after pxe installation is completed, you should configure the runtime environment of the DPDK and RAMCloud.

Appendix

pxelinux.cfg/default

sample configuration

```
:url:     http://192.168.3.240/pxeboot/kickstart.cfg
:device:  eth0
:console: tty1
:label:   centos7
:default: centos7
```

erb template of pxelinux.cfg/default

```
default <%= @default %>
label <%= @label %>
kernel vmlinuz
append ks=<%= @url %> ksdevice=<%= @device %> load initrd=initrd.img devfs=nomount console=<%=
```

kickstart.cfg

sample configuration

```
:url:           http://192.168.3.240/ks
:dist_url:      http://192.168.3.240/dist/centos/7
:device:        eth0
:root_pw:       root
:admin_user:    admin
:admin_pw:      admin
:admin_uid:     900
:admin_gid:     900
:admin_groups:  wheel
:gateway:       192.168.3.240
:use_proxy:     true
:http_proxy:    http://proxy.somewhere.com:8080/
:hosts:     http://192.168.3.240/ks/hosts
:admin_key: http://192.168.3.240/ks/admin.pub
```

erb template of kickstart.cfg

```
install
url --url=<%= @dist_url %>

text
lang en_US.UTF-8
keyboard us
timezone --utc Asia/Tokyo
```

```
authconfig --enableshadow --passalgo=sha512
rootpw --plaintext <%= @root_pw %>
# group --name=<%= @admin_group %> --gid=<%= @admin_gid %>
user --name=<%= @admin_user %> --uid=<%= @admin_uid %> --gid=<%= @admin_gid %> --groups=<%= @a

network --device <%= @device %> --onboot yes --bootproto dhcp --noipv6
firewall --disable
selinux --disable

bootloader --location=mbr
clearpart --all --initlabel
part / --fstype=xfs --grow --size=1

poweroff

# additional yum repos
# repo --name=<%= @repo_name %> --baseurl=<%= @repo_url %> --cost=100

%packages --nobase --ignoremissing --excludedocs
@core --nodefaults
%end

%pre --log=/root/ks-pre.log
%end

%post --log=/root/ks-post.log

<% if @use_proxy -%>
cat << EOF > /etc/proxy.conf
export http_proxy=<%= @http_proxy %>
EOF
cat << EOF >> /etc/yum.conf
proxy=<%= @http_proxy %>
EOF
<% end -%>

cat << EOF >> /etc/yum.conf
exclude=kernel*
EOF

cat << EOF >> /etc/yum/pluginconf.d/fastestmirror.conf
prefer=ftp.riken.jp
include_only=.jp
EOF

# disable NetworkManager. otherwise, following gateway and hostname
# configs don't work. i don't know why.
systemctl disable NetworkManager
```

```
# configure default router
echo "GATEWAY=<%= @gateway %>" > /etc/sysconfig/network

# Get hostname from dhcp server
echo "request host-name;" >> /etc/dhcp/dhclient-<%= @device %>.conf

<% if @hosts -%>
curl -s -o /etc/hosts <%= @hosts %>
<% end -%>

<% if @admin_key -%>
mkdir -m 700 /home/<%= @admin_user %>/.ssh
curl -s -o /home/<%= @admin_user %>/.ssh/authorized_keys <%= @admin_key %>
chmod 600  /home/<%= @admin_user %>/.ssh/authorized_keys
chown -R <%= @admin_user %> /home/<%= @admin_user %>/.ssh
<% end -%>

echo '%wheel ALL=(ALL) NOPASSWD: ALL' > /etc/sudoers.d/wheel-nopasswd
chmod 440 /etc/sudoers.d/wheel-nopasswd

cat << EOF > /etc/sudoers.d/proxy-env
Defaults env_keep += "http_proxy https_proxy ftp_proxy no_proxy"
Defaults env_keep += "HTTP_PROXY HTTPS_PROXY FTP_PROXY NO_PROXY"
EOF
chmod 440 /etc/sudoers.d/proxy-env

%end
```

local.repo

sample configuration

```
---
:name:  local
:host:  192.168.3.240
:url:   /repo
```

erb template of kickstart.cfg

```
[<%= @name %>]
name=<%= @name %>
baseurl=http://<%= @host %><%= @url %>/<%= @name %>/centos/$releasever
enabled=0
gpgcheck=0
priority=1
```

```
proxy=_none_
mirrorlist=file:///dev/null
```

dpdk.spec

[TBD]

dpdk.init

```
#!/bin/sh
#
# dpdk - configure the intel dpdk runtime environment
#
# chkconfig:   - 85 15
# description: configure the Intel DPDK runtime environment
# processname:
# config:      /etc/sysconfig/dpdk

# Source function library.
. /etc/rc.d/init.d/functions

prog="dpdk"
sysconfig="/etc/sysconfig/$prog"
lockfile="/var/lock/subsys/$prog"
confdir="/var/run/${prog}"

# NUMA
# sysfs_hugepages="/sys/devices/system/node/node*/hugepages/hugepages-2048kB"
# Non-NUMA (sum of hugepages of all nodes)
sysfs_hugepages="/sys/kernel/mm/hugepages/hugepages-2048kB"
sysfs_pages="$sysfs_hugepages/nr_hugepages"
sysfs_free="$sysfs_hugepages/free_hugepages"
sysfs_uio="/sys/class/uio/uio*/device/config"
sysfs_vfio="/sys/class/vfio/*" ## currently not used

sysfs_pci_dev="/sys/bus/pci/devices"
resource_conf="/etc/security/limits.d/90-memlock.conf"

vfio_mod="vfio-pci"
vfio_mod_dep="vfio_pci vfio_iommu_type1 vfio"
vfio_dir="/dev/vfio"
vfio_dev="$vfio_dir/vfio"
vfio_dev_list="$vfio_dev $vfio_dir/[0-9]*"

uio_dev="/dev/uio[0-9]*"
uio_mod="igb_uio"
uio_mod_args=""
```

```
bind_cmd="/usr/sbin/dpdk_nic_bind"

numvfs_iface="max_vfs"
# $sysfs_pci_dev/$dev/max_vfs
# igb_uio, ixgbe-3?(obsolete) : max_vfs
# ixgbe-4 : sriov_numvfs

# kni
kni_mod="rte_kni"
sysfs_kni_params="/sys/module/$kni_mod/parameters"

# =begin
# dpdk.conf

# enable dpdk runtime enviroment at startup or not. [true|false]
dpdk_enable="true"

# Number of huge pages to allocate
# It is not always possible to allocate the requested amount of pages.
# To reserve hugepages at boot time, add the following kernel parameters:
#   default_hugepagesz=2M hugepagesz=2M hugepages=1024
nr_hugepages="1024"

# mount point of hugetlbfs
hugetlbfs_dir="/dev/hugepages"

# use vfio or igb_uio
use_vfio="true"

kern_mod=""
kern_mod_args=""
#kern_mod="ixgbe"
#kern_mod_args="allow_unsupported_sfp=1"

# VF
# num_vfs="0" to disable vf
num_vfs="2"

bind_if=""
#bind_if="ens2f0"

# PF devices
bind_dev="
0000:20:00.0
"

pci_extended_tag=""           ## ""|no|yes
pci_max_read_request_size=""  ## ""|0|128|256|512|1024|2048
```

```
uid="root"
gid="dpdk"
mode="ug+rw"

# kni
kni_enable="false"
# kthread mode: single|multiple
kni_kthread_mode="single"
# loopback mode: none|fifo|fifo_skb
kni_lo_mode="none"
# kni kthread reschedule interval (usec). default 5us
kni_resched_interval=""

# end of dpdk.conf
# =end

[ -f $sysconfig ]   && . $sysconfig
[ -n "$DPDK_CONF" ] && . $DPDK_CONF

dpdk_enabled () {
    case $dpdk_enable in
        true|yes|y) return 0 ;;
    esac
    return 1
}
dpdk_set_enable () {
    dpdk_enable="true"
}

dpdk_configured () {
    [ -f $lockfile ] && return 0
    return 1
}
dpdk_set_configured () {
    touch $lockfile
}
dpdk_clear_configured () {
    rm -f $lockfile
}

show_default_config () {
    local in_section=0 section_begin="# =begin" section_end="# =end"
    local IFS=''
    exec < $0    # redirect this file to the current shell
    while read line; do
        case $line in
            $section_begin) : $((in_section++)); continue ;;
            $section_end)   : $((in_section--)); continue ;;
        esac
        (( $in_section > 0 )) || { in_section=0;  continue; }
        echo $line
    done
}

setup_hugepages () {
    # allocate huge pages
    pages=$(cat $sysfs_pages)
    if [ $pages != "0" ]; then
        echo "hugepages: alreay allocated $pages pages. (2MB/page)"
    else
        echo $nr_hugepages > $sysfs_pages
        echo "hugepages: requested = $nr_hugepages pages (2MB/page)"
        echo "hugepages: allocated = $(cat $sysfs_free) pages (2MB/page)"
    fi

    # mount hugetlbfs
    [ -z "$hugetlbfs_dir" ] && {
        echo "skip to mount hugetlbfs"
        return 0
    }

    grep -q hugetlbfs /proc/mounts || {
        [ -d $hugetlbfs_dir ] || mkdir -p $hugetlbfs_dir
        mount -t hugetlbfs nodev $hugetlbfs_dir
        echo "hugetlbfs mounted on $hugetlbfs_dir"
    }
}

teardown_hugepages () {
    echo "unmount hugetlbfs"
    grep -q hugetlbfs /proc/mounts && umount $hugetlbfs_dir

    echo "free hupgepages"
    echo 0 > $sysfs_hugepages/nr_hugepages
}

# N.B. always unbind a driver from $dev
set_max_vfs () {
    local dev=$1 numvfs=$2
    local vfif nvfs

    [ -z "$dev" ]     && return
    [ -z "$numvfs" ]  && return
    [ $numvfs -le 0 ] && return

    # temporary bind the $uio_mod to the $dev to access max_vfs.
    # vfio-pci does'nt have max_vfs interface.
    $bind_cmd --bind=$uio_mod $dev
    vfif="$sysfs_pci_dev/$dev/max_vfs"
    nvfs=$(cat $vfif)
    # invalid argument error happens if write the same value to $vfif
    if [ "$nvfs" = "$num_vfs" ]; then
        echo "$dev: $num_vfs VFs are already configured."
    else
        echo "setting $vfif to $numvfs"
        echo $numvfs > $vfif
    fi

    # unbind the $uio_mod from the $dev
    $bind_cmd --unbind $dev
}

# todo: implement num of VFs for each port. ex: num_vfs=2,4,4,2
setup_vfs () {
    local dev
    [ -z "$num_vfs" ] && return
    [ $num_vfs -le 0 ] && return

    for dev in $bind_dev; do
        set_max_vfs $dev $num_vfs
    done
}
teardown_vfs () {
    local dev
    for dev in $bind_dev; do
        set_max_vfs 0
    done
}

load_uio_mod () {
    echo "loading $uio_mod $uio_mod_args"
    modprobe uio
    modprobe $uio_mod $uio_mod_args
}
unload_uio_mod () {
    echo "unloading $uio_mod"
    modprobe -r $uio_mod
}
load_vfio_mod () {
    echo "loading $vfio_mod"
    modprobe $vfio_mod
    if [ ! -e "$vfio_dev" ]; then
        echo "Error: failed to load $vfio_mod"
        return 1
    fi
}
```

```
unload_vfio_mod () {
    echo "unloading $vfio_mod"
    modprobe -r $vfio_mod_dep
}
check_vfio () {
    local mod="/lib/modules/$(uname -r)/kernel/drivers/vfio"
    [ "$use_vfio" = "true" ] || return
    [ -e "$mod" ] || {
        echo "vfio modules does not exist. disabling vfio"
        use_vfio="false"
    }
}

setup_dpdk () {
    local iface dev bind_mod vf vfdev

    check_vfio

    # de-activate the active ifaces
    for iface in $bind_if; do
        ifconfig $iface > /dev/null 2>&1 && ifconfig $iface down
    done

    # load the uio/vfio modules
    # uio_mod always exists but vfio exists on linux-3.6 and later
    unload_uio_mod
    load_uio_mod
    if [ "$use_vfio" = "true" ]; then
        unload_vfio_mod
        load_vfio_mod
    fi

    # initialize max_vfs for each PFs
    setup_vfs

    # bind the uio/vfio to all PF/VFs
    bind_mod=$uio_mod
    [ "$use_vfio" = "true" ] && bind_mod=$vfio_mod
    for dev in $bind_dev; do
        # bind the uio/vfio to the PF
        echo "binding $bind_mod to PF $dev"
        $bind_cmd --bind=$bind_mod $dev

        # bind the uio/vfio to all VFs
        for vf in $sysfs_pci_dev/$dev/virtfn[0-9]*; do
            [ -e $vf ] || break
            vfdev=$(basename $(readlink $vf))
            echo "binding $bind_mod to VF $vfdev on PF $dev"
            $bind_cmd --bind=$bind_mod $vfdev
        done
    done
}

teardown_dpdk () {
    local dev bind_mod vf vfdev

    check_vfio

    # unbind the uio/vfio from all PF/VFs
    bind_mod=$uio_mod
    [ "$use_vfio" = "true" ] && bind_mod=$vfio_mod
    for dev in $bind_dev; do
        # bind the uio/vfio to all VFs
        for vf in $sysfs_pci_dev/$dev/virtfn[0-9]*; do
            [ -e $vf ] || break
            vfdev=$(basename $(readlink $vf))
            echo "unbinding the driver from VF $vfdev on PF $dev"
            $bind_cmd --unbind $vfdev
        done

        echo "unbinding the driver from PF $dev"
        $bind_cmd --unbind $dev
    done

    # set max_vfs to 0 for each PFs
    teardown_vfs

    # unload the uio/vfio modules
    unload_uio_mod
    [ "$use_vfio" = "true" ] && unload_vfio_mod


    # bind the $kern_mod to the $bind_dev
    [ -z "$kern_mod" ] && return

    # workaround:
    # binding with ixgbe fails if an unsupported SFP+ or QSFP module type
    # is detected. the module parameter allow_unsupported_sfp=1 will not
    # affect when binding to a pci device via sysfs pci interface.
    case "$kern_mod" in
        ixgbe)
            for dev in $bind_dev; do
                $bind_cmd --unbind $dev
            done
            modprobe -r $kern_mod
            modprobe $kern_mod $kern_mod_args
            return 0 ;;
    esac
    modprobe $kern_mod $kern_mod_args

    # bind the $kern_mod to the $bind_dev
    for dev in $bind_dev; do
       $bind_cmd --bind=$kern_mod $dev
    done
}

setup_pci_config () {
    local dev
    for dev in $bind_dev; do
        setup_pci_config_dev $dev
    done
}
setup_pci_config_dev () {
    local dev=$1
    local pci="$sysfs_pci_dev/$dev"
    case $pci_extended_tag in
        "") ;;
        yes|no)
            echo $pci_extended_tag  > $pci/extended_tag ;;
        *)  echo "invalid pci_extended_tag: $pci_extended_tag" ;;
    esac
    case $pci_max_read_request_size in
        0|"") ;;
        128|256|512|1024|2048)
            echo $pci_max_read_request_size  > $pci/max_read_request_size ;;
        *) echo "invalid pci_max_read_request_size: $pci_max_read_request_size" ;;
    esac
}

fixup_env () {
    local sysfs_pci_dev_files=""
    local devlist=$bind_dev
    local dev vf vfdev

    for dev in $bind_dev; do
        for vf in $sysfs_pci_dev/$dev/virtfn[0-9]*; do
            [ -e $vf ] || break
            vfdev=$(basename $(readlink $vf))
            devlist="$devlist $vfdev"
        done
    done

    for dev in $devlist; do
        sysfs_pci_dev_files="$sysfs_pci_dev_files
$sysfs_pci_dev/$dev/resource[0-9]*
$sysfs_pci_dev/$dev/extended_tag
$sysfs_pci_dev/$dev/max_read_request_size
```

```
"
    done
    files="$hugetlbfs_dir $uio_dev $vfio_dev_list $sysfs_uio $sysfs_pci_dev_files"
    echo "fixing permission:"
    for file in $(eval echo $files); do
        [ -e $file ] && {
            echo $file
            chown $uid:$gid $file
            chmod $mode     $file
        }
    done

    [ -d "$confdir" ] || mkdir -p $confdir
    chown -R $uid:$gid $confdir
    chmod -R $mode     $confdir

    if [ "$use_vfio" = "true" ]; then
        chmod a+rx $vfio_dir
    fi

    echo "creating $resource_conf"
    echo "@$gid - memlock unlimited" > $resource_conf
}

setup_kni () {
    local kparam lparam iparam param
    [ "$kni_enable" = "true" ] || return 0
    case $kni_kthread_mode in
        multiple) kparam="kthread_mode=multiple" ;;
        *)        kparam="kthread_mode=single"   ;;
    esac
    case $kni_lo_mode in
        fifo_skb) lparam="lo_mode=lo_mode_fifo_skb" ;;
        fifo)     lparam="lo_mode=lo_mode_fifo"     ;;
        *)        lparam="lo_mode=lo_mode_none"     ;;
    esac
    case "$kni_resched_interval" in
        "")  iparam="" ;;
        *)   iparam="resched_interval=$kni_resched_interval" ;;
    esac
    ## first rmmod the module and then insmod it in case of updating prams.
    grep -q $kni_mod /proc/modules && modprobe -r $kni_mod
    modprobe $kni_mod $kparam $lparam $iparam
    echo "$kni_mod loaded."
    sleep 1  ## wait until sysfs entries are populated.
    for param in $sysfs_kni_params/*; do
        [ -r $param ] && echo "kni $(basename $param): $(cat $param)"
    done
    return 0
```

```
}
teardown_kni() {
    grep -qs $kni_mod /proc/modules && modprobe -r $kni_mod
    return 0
}


start() {
    dpdk_enabled     || { dpdk_clear_configured; return 0; }
    echo "Starting $prog"
    dpdk_configured  && return 0
    setup_hugepages  || return 6
    setup_dpdk       || return 7
    setup_pci_config || return 8
    fixup_env        || return 9
    setup_kni        || return 10

    dpdk_set_configured
    return 0
}
force_start () {
    dpdk_clear_configured
    dpdk_set_enable
    start
}

stop() {
    # do not check dpdk_enabled here. always teardown the environment.
    dpdk_configured || return 0;
    echo "Stopping $prog"
    teardown_kni       || return 5
    teardown_dpdk      || return 6
    teardown_hugepages || return 7
    dpdk_clear_configured
    return 0
}

restart() {
    stop  || return 6
    sleep 1
    start || return 7
    return 0
}
reload() { restart; }

status () {
    dpdk_configured || {
        dpdk_enabled || {
            echo "dpdk: not enabled"
            return 1
        }
        echo "dpdk: not configured"
        return 2
    }
    echo "dpdk: configured"
    return 0
}

show () {
    echo "hugepages:"
    grep -i huge /proc/meminfo
    echo "hugetlbfs: "
    grep hugetlbfs /proc/mounts || echo "not mounted"

    $bind_cmd --status
}

cmd="$1"
case $cmd in
    start|force_start|stop|restart|reload|status|show)
        $cmd
        ;;
    show_default_config)
        $cmd ;;
    *)
        echo $"Usage: $0 {start|force_start|stop|restart|reload|status|show}"
        exit 1
        ;;
esac
```

base.yaml

example ansible script to setup basic os environment on cluster hosts.

```
---
- hosts: cluster
  remote_user: admin
  sudo: yes
  gather_facts: no
  vars_files:
    - ../config/{{ config }}/base.yaml

  tasks:

# basic os configurations.
# these configurations are done by kickstart script, but configure again
# in case of configs are changed after kickstart install.
```

```
    - name: disable selinux
      selinux: state=disabled conf=/etc/selinux/config
    - name: disable firewalld
      service: name=firewalld state=stopped enabled=no
    - name: get hostname from dhcp server
      lineinfile: dest=/etc/dhcp/dhclient-eth0.conf state=present create=yes
                  line="request host-name"
#    - name: hostname
#      hostname: name="{{ inventory_hostname }}"
    - name: disable requiretty
      copy: content='Defaults !requiretty\n'
            dest=/etc/sudoers.d/disable-requiretty owner=root mode=0440 force=yes
            validate='visudo -cf %s'
    - name: configure sudoers policy
      copy: "content='%wheel ALL=(ALL) NOPASSWD: ALL\n'
            dest=/etc/sudoers.d/wheel-nopasswd owner=root mode=0440 force=yes
            validate='visudo -cf %s'"
    - name: fixup modes of sudoers configs
      file: path=/etc/sudoers.d/{{ item }} owner=root group=root mode=0440
      with_items:
        - disable-requiretty
        - proxy-env
        - wheel-nopasswd
    - name: install /etc/hosts
      get_url: url={{ hosts_url }} dest=/etc/hosts mode=0444 force=yes use_proxy=no

# yum configurations
    - name: configure yum
      lineinfile: dest=/etc/yum.conf regexp="^{{ item.param }}="
                  line="{{ item.param }}={{ item.value }}"
      with_items: yum_conf

    - name: configure the local yum repository
      get_url: url={{ local_repo.conf }} dest=/etc/yum.repos.d/local.repo
               mode=0444 force=yes use_proxy=no
    - name: fetch metadata of the local repo
      command: yum --disablerepo='*' --enablerepo={{ local_repo.name }} makecache

# install required packages
    - name: install required packages
      yum: name={{ item }} state=latest update_cache=yes
      with_items: packages

    - name: install kernel related packages from base repo.
      command: yum -y --disablerepo='*' --enablerepo=base --disableexcludes=all
               install {{ item }}
      with_items: kernel_packages
```

```
   - name: configure nfs client
     lineinfile: dest=/etc/fstab state=present
                 regexp="^.*/home\s.*" line="{{ fstab }}"

   - name: configure chrony daemon
     lineinfile: dest=/etc/chrony.conf state=present
                 regexp="^server" line="server {{ ntp.server }}"

   - name: configure sshd
     lineinfile: dest=/etc/ssh/sshd_config state=present
                 regexp="^{{ item.param }} "
                 line="{{ item.param }} {{ item.value }}"
     with_items: sshd_conf

# tune for low latency
   - name: enable latency-performance configuration
     shell: tuned-adm profile latency-performance
   - name: restart tuned service
     service: name=tuned enabled=yes state=restarted

# cpupower service conflicts tuned service.
# use tuned for the moment, but benchmarking required.
   - name: disable cpupower service
     service: name=cpupower enabled=no state=stopped

   - name: enable acpid for ipmi power soft
     service: name=acpid enabled=yes state=started

   - name: disable irqbalance
     service: name=irqbalance state=stopped enabled=no

# handlers:

   - name: enable/restart chrony daemon
     service: name=chronyd enabled=yes state=restarted

   - name: enable/restart sshd service
     service: name=sshd enabled=yes state=restarted

   - shell: grep -q '/home nfs' /proc/mounts
     register: mount_status
     ignore_errors: True
   - name: mount /home of management server over nfs
     shell: mount /home
     when: mount_status|failed
   ## mount of /home should be the last recipe of playbook,
   ## or change the ansible remote tmp path in in ansiglbe.cfg
   ## to /tmp or somewhere not under the /home
```

```
    - name: finish basic configuration
      debug: msg="reboot the target system now"
```

dpdk.yaml

example ansible script to install and configure dpdk runtime on cluster hosts.

```
---
- hosts: cluster
  remote_user: admin
  sudo: yes
  gather_facts: no
  vars_files:
    - ../config/{{ config }}/dpdk.yaml

  tasks:

   # dpdk runtime directory /var/run/dpdk is created by the dpdk
   # init script (/etc/init.d/dpdk).

   # stop/uninstall dpdk
   # in case of the pkg is updated without update the revision.
   - name: stop dpdk if it is active
     service: name='dpdk' enabled=no state=stopped
   - name: uninstall the dpdk packages
     yum: name='dpdk' state=absent
   - name: clean up the yum cache
     command: yum --disablerepo='*' --enablerepo={{ repo }} clean all
   - name: install dpdk package from local repository
     yum: name='dpdk' state=latest
          disablerepo=* enablerepo={{ repo }} update_cache=yes

   - name: configure dpdk
     lineinfile: dest='/etc/sysconfig/dpdk' state=present
                 regexp='^{{ item.key }}='
                 line='{{ item.key }}=\"{{ item.val }}\"'
     with_items: config

   # dpdk user/group are created by the rpm installation script.
   - name: add group dpdk to admin users groups
     user: name='admin' groups='dpdk' append=yes

   - name: enable and start dpdk
     service: name=dpdk enabled=yes state=started
```

ramcloud.yaml

example ansible script to setup ramcloud runtime environment on cluster hosts.

```
---
- hosts: cluster
  remote_user: admin
  sudo: yes
  gather_facts: no
  vars_files:
    - ../config/{{ config }}/ramcloud.yaml

  tasks:

   - name: create ramcloud group
     group: name={{ group }} gid={{ gid }} state=present

   - name: create ramcloud user
     user: name={{ user }} uid={{ uid }} group={{ group }} state=present
           groups={{ grouplist | join(",") }} createhome=no
           password={{ passwd }} comment="{{ comment }}"

   - name: add group ramcloud to admin users groups
     user: name='admin' groups='ramcloud' append=yes

   ## resource limit configuration
   - name: set the soft limit of memlock to unlimited
     lineinfile: dest=/etc/security/limits.conf state=present
                 line="@{{ group }} soft memlock unlimited"
   - name: set the hard limit of memlock to unlimited
     lineinfile: dest=/etc/security/limits.conf state=present
                 line="@{{ group }} hard memlock unlimited"

   - name: create ramcloud runtime directories
     file: dest={{ item }} state=directory owner=root group={{ group }} mode=0775
     with_items:
       - /var/ramcloud/backup

   - name: install packages required to run ramcloud
     yum: name={{ item }} state=latest
     with_items: packages

   - name: install getmac script
     copy: dest="/usr/local/bin/getmac" mode=0755 owner=root group=root
           src=../../sg/tools/getmac

- hosts: mgmt
  remote_user: admin
  sudo: yes
  gather_facts: no
```

```
vars_files:
  - ../config/{{ config }}/ramcloud.yaml

tasks:


 - name: create ramcloud group
   group: name={{ group }} gid={{ gid }} state=present

 - name: create ramcloud user
   user: name={{ user }} uid={{ uid }} group={{ group }} state=present
         groups={{ grouplist | join(",") }}
         password={{ passwd }} comment="{{ comment }}" home={{ home }}
         generate_ssh_key=yes ssh_key_comment="ramcloud@cluster"
```