DX2000 from NEC lets you put big data to work

with quick analysis in a robust, space-efficient, and scalable solution, powered by Intel®

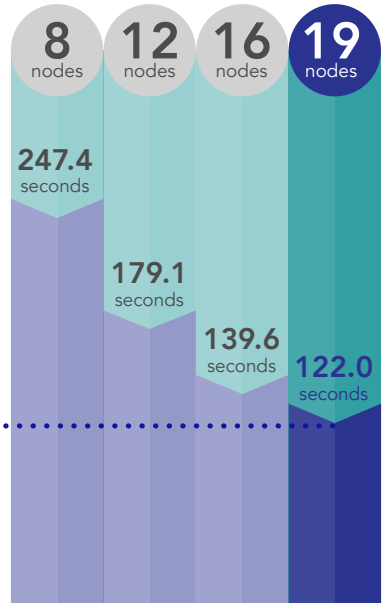
Modern businesses generate and collect data at an incredible rate. The data sets are sometimes too large for traditional processing, and the emergence of big data fuels the need to process and analyze data differently than we have in the past. Analyzing and understanding big data is vital to the success of most businesses as it can lead to improvements in:

- Customer experience
- New products or services
- Revenue generation
- Customer base growth
- Market reach
- Customization of existing products or services

Just collecting data is not enough; to get these improvements you have to identify data patterns through processing and analysis, which requires a speedy, scalable, and flexible hardware and software solution.

We set up a new Scalable Modular Server DX2000 from NEC, powered by the Intel Xeon® processor D product family. We then configured a Red Hat® Enterprise Linux® OpenStack® cloud environment with Apache Spark™, an industry-leading big-data analytics engine, to put the DX2000 through its paces. The solution analyzed a big data sample set not only quickly and efficiently, but most importantly – in a predictable, scalable fashion. When we added more server nodes, it processed the big data more quickly.

Cut analysis time in half by adding server nodes

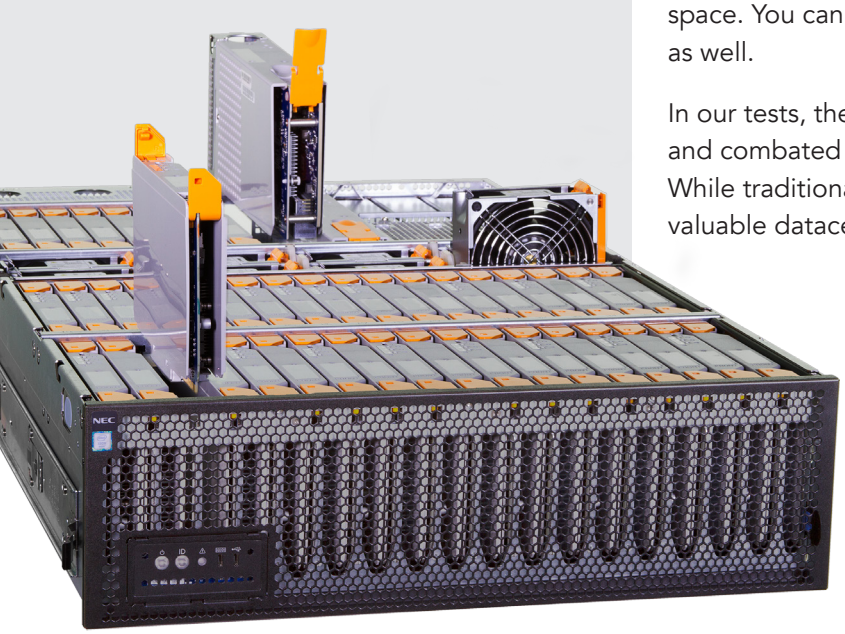


The Scalable Modular Server DX2000 from NEC has 44 customizable slots available to create a number of configurations. Each DX2000 could hold up to the following:

- 44 server nodes featuring Intel Xeon D processors for compute density, 2.75TB DDR4 memory (64GB per server node), and 22TB of flash-based storage (512GB per server node)
- 22 dual 10Gb Ethernet links for additional networking (up to 22 slots for server nodes)
- 8 PCIe card modules for expanding resources available to specific server nodes (up to 36 slots for server nodes)

Clustering nodes of necessary resources in a small space makes sense for big data initiatives, particularly because big data requires so much processing power. Each of the 22 server nodes in our DX2000 enclosure featured the following:

- 8-core Intel Xeon processor D-1541
- 64GB memory
- 256GB flash-based storage
- Dual 10Gb Ethernet links



Drive your business on facts, not speculation

Money-making facts live in your business data, but it's up to you to find them efficiently. Whether you want to run a targeted email advertising campaign to drive sales or analyze customer feedback to improve product quality, big data can help you spot patterns that predict probability based on key data points. Taking the guesswork out of vital decision-making processes can have a positive impact on your bottom line.

Processing large volumes of data in a timely manner requires compute power. The DX2000 brings that power to your big data initiatives by providing up to 44 single-processor server modules per 3U enclosure.

Apache Spark, the analytics tool we used in testing, uses in-memory processing to give you data analysis results as fast as possible. Combining Apache Spark with the memory resources of DX2000 server nodes allows you to get results more quickly than traditional disk-based approaches in many applications.

Combat datacenter sprawl and meet big data challenges

Just because your data is big doesn't mean your hardware infrastructure has to be. While managing a big data environment can be challenging, as over 90% of respondents to a recent survey agree,[1] your business can help contain datacenter sprawl by selecting a dense, scalable modular server platform.

A scalable modular server platform, like the DX2000 by NEC, is a rack-mounted chassis holding multiple server units, often called server nodes, which commonly work together as a cluster. These compact server nodes pack compute, storage, and networking resources into a small space. You can configure the nodes to provide ample memory resources as well.

In our tests, the DX2000 from NEC quickly analyzed our sample data set and combated datacenter sprawl by doing it in only 3U of rackspace. While traditional servers can run big data analytics, they take up a lot of valuable datacenter space.

How can big data analysis help boost business?

When a business can analyze big data quickly, multiple departments have the ability to reap business benefits. Consider the following scenarios:

Serve customers better

Sandy is a marketing department head for a retail-clothing manufacturer. Keeping track of customer purchases – for example, the type of jackets that customers of a certain age or region have bought in the past week – can inform targeted, predictive advertisements and communications for these customers in the future. The faster the company's servers analyze this data, the sooner Sandy gets the reports she needs in order to send out customized emails telling customers about other products that may interest them. She'll also use those reports to send offers to buyers that fit those demographics, but have not made purchases yet. This successful campaign will ultimately lead to more business from new and existing customers. Using the Scalable Modular Server DX2000 from NEC and Apache Spark for her data analysis, Sandy can quickly get the valuable information that helps her make informed business decisions.

Speed big data processing with Apache Spark

Apache Spark is an industry-standard framework that processes big data in-memory. In-memory processing keeps data in RAM to shorten response times. Companies like Amazon, eBay, NASA Jet Propulsion Laboratories, TripAdvisor, and Yahoo use this tool to help them get real-time bid optimization, machine learning-based user data modeling, forecasting, and other predictive analytics.[2]

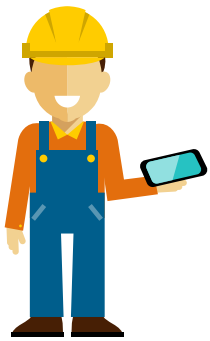
Your business can use robust hardware platforms like the DX2000 from NEC in conjunction with machine learning functionality – among other valuable kinds of data processing in Apache Spark – to make better use of your growing data and achieve fast analysis turnaround times.

Find what you're looking for

In our datacenter, we used the clustering algorithm *k*-means. *k*-means takes large data sets and identifies patterns in them. Companies often use it in predictive analysis for cost modeling, market research, price forecasting, and customer-retention applications.

Improve product quality

Devon, an inventory manager for a distribution center, oversees the returns department. Devon and his team must track why customers return items, and they rely on rapid big data analysis to identify patterns in the returned items. They need to ensure that product quality is good and address customer satisfaction issues with specific items. His team works closely with the product development team and manufacturing leaders to guarantee that their products live up to customer needs and expectations. The Scalable Modular Server DX2000 from NEC and Apache Spark can help Devon and his team quickly identify quality control issues and provide the necessary feedback to the leaders responsible for correcting the problem.



Grow your revenue

Chris, a regional sales manager for a retail chain, relies on sales, marketing, and advertising data analysis from the retailer's IT department to drive decisions on in-store sales and marketing campaigns. He needs fast and reliable data to help his team identify which of his advertising drives are most effective, whether the time of year affects sales numbers, and what incentives are likely to be most effective on first-time customers. Apache Spark and the Scalable Modular Server DX2000 from NEC provide a powerful platform for valuable big data analysis that can help Chris bring in new customers and grow his company's revenue.

Facts from our datacenter

Find patterns

We put the DX2000 solution through its paces with a *k*-means data cluster analysis test from the HiBench benchmark suite. The results from this performance evaluation show you the benefits of combining the Apache Spark in-memory engine with the Scalable Modular Server DX2000 from NEC. When all available compute nodes in our configuration were running the *k*-means analysis, the NEC solution took just over 2 minutes, or 122 seconds, to process 100GB of data. Even though the data set we used is small by some big data standards, a few hundred GB is representative of some publicly available applications and use cases, and it could reflect scenarios that you can relate to. For example, 100GB is large enough to store demographic data for the world's population in 2009[3] or three months of Wikipedia's page traffic statistics.[4]

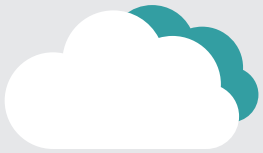
Cut analysis time in half

We also compared throughput, a measure of how quickly a solution can process data, at several different server node counts. We did this to demonstrate how increasing the number of server nodes can improve the throughput of the DX2000, thus cutting down on processing time. We began our scaling comparison with eight server nodes. We tested fewer nodes than eight, but using so few nodes for a data set of this size yielded less than optimal results. This was due to the application and data set footprint exceeding the memory resources available to Apache Spark in those low node counts. As we added additional server nodes to our DX2000, the solution scaled nearly linearly in throughput. Ultimately, 19 nodes delivered over twice the throughput of eight, cutting analysis time of our 100GB data set in half compared to our initial eight-server-node count. At our maximum 22-server configuration, we used 19 server nodes for data processing and three nodes for management services. Based on the results we got from eight to 19 nodes, we expect your big data throughput would scale up as your business grows and you continue to populate your DX2000 enclosure with additional server nodes.

We chose a sample data set big enough to store the demographic data for 6.75 billion people.[3]

Big data analytics with private cloud flexibility

We deployed and tested a Red Hat OpenStack Platform environment in the Principled Technologies datacenter. In this OpenStack environment, we used two OpenStack controller servers for high availability, one server for management tasks and Red Hat OpenStack Platform Director services, and the DX2000 from NEC with 22 server nodes. This deployment provided the flexible private cloud environment we needed for our Apache Spark VMs.



Red Hat Enterprise Linux OpenStack Platform 7

is an Infrastructure-as-a-Service (IaaS) private cloud solution. It allows you to build a cloud platform on your hardware and use resources efficiently.

When you use Red Hat Enterprise Linux OpenStack Platform 7 in conjunction with Apache Spark and the DX2000 environment, your team can dedicate as much or as little space to big data analysis as you need. The flexibility of this platform means you can spin up application instances to meet changing IT demands.

Cloud platforms such as Red Hat Enterprise Linux OpenStack Platform help increase your flexibility by adapting your hardware environment without having to rebuild your infrastructure from the ground up. For more information, please visit Red Hat's website.[5]

Conclusion

The data that your business collects is constantly growing, making it increasingly difficult for traditional systems to keep up with resource demands. Understanding your big data can help you serve your customers better, improve product quality, and grow your revenue, but you need a platform that can handle the strain.

In hands-on tests in our datacenter, the Scalable Modular Server DX2000 from NEC processed big data quickly and scaled nearly linearly as we added server nodes. In our *k*-means data cluster analysis test, a DX2000 solution running Apache Spark and Red Hat Enterprise Linux OpenStack Platform processed 100GB in approximately 2 minutes. We also saw that as we doubled the number of server nodes, the DX2000 solution cut analysis time in half when processing the same amount of data, producing excellent scalability.

The Scalable Modular Server DX2000 by NEC is a good choice when you're ready to put big data to work for you.

---

1 http://www.ca.com/us/~/media/Files/infographics/bright-lights-big-data.PDF

2 https://cwiki.apache.org/confluence/display/SPARK/Powered+By+Spark

3 Basic demographic data refers to age, sex, income, ethnicity, language, religion, housing status, and location. Read a more detailed explanation and review at http://dl.acm.org/citation.cfm?id=1536632.

4 https://aws.amazon.com/datasets/wikipedia-page-traffic-statistic-v3/

5 https://access.redhat.com/documentation/en/red-hat-enterprise-linux-openstack-platform/version-7/red-hat-enterprise-linux-openstack-platform-7-architecture-guide/preface

Appendix A – Inside the server we tested

Figure 1 provides detailed configuration information for each of the 22 server nodes we tested.

Server configuration information	NEC Scalable Modular Server DX2000
BIOS name and version	MM60-B30, BIOS 5.0.0005
Non-default BIOS settings	Changed from UEFI to Legacy BIOS
Operating system name and version/build number	Red Hat Enterprise Linux 7.2 3.10.0-327.10.1.el7.x86_64
Date of last OS updates/patches applied	02/12/2016
Power management policy	Default
Processor	
Number of processors	1
Vendor and model	Intel Xeon CPU D-1541
Core count (per processor)	8
Core frequency (GHz)	2.10
Stepping	V2
Memory module(s)	
Total memory in system (GB)	64
Number of memory modules	4
Vendor and model	Hynix HMA82GS7MFR8N-TF
Size (GB)	16
Type	PC4-17000
Speed (MHz)	2133
Speed running in the server (MHz)	2133
Storage controller	
Vendor and model	Intel 82801JI
Cache size	N/A
Driver version	2.13
Local storage	
Number of drives	1
Drive vendor and model	Toshiba THNSNJ256G8NU
Drive size (GB)	256
Drive information (speed, interface, type)	6Gb/s, M.2, SSD
Network adapter	
Vendor and model	Intel X552
Number and type of ports	2 x 10GbE
Driver version	4.3.13

Figure 1: Detailed configuration information for each of the 22 server nodes we tested.

Figure 2 provides detailed configuration information for the server enclosure we used in our tests.

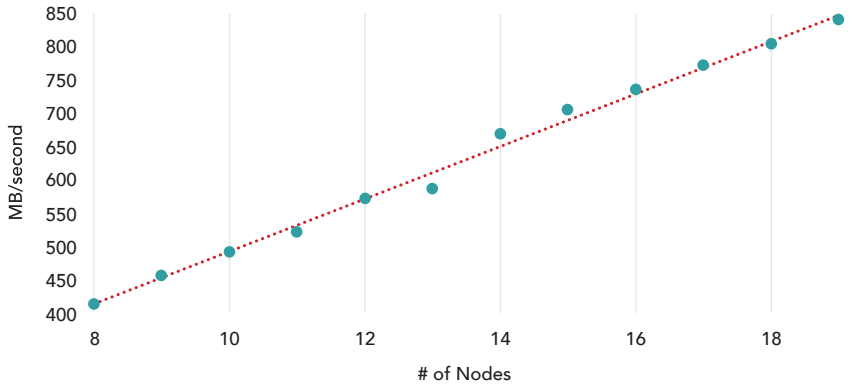
Server enclosure configuration information	NEC Scalable Modular Server DX2000
Number of management modules	2
Management module firmware revision	00.19
I/O module	
Vendor and model number	NEC Micro Modular Server DX2000 LAN Switch
I/O module firmware revision	ZebOS-XP version 1.2.0.5
Number of modules	2
Occupied bay(s)	1, 2
Power supplies	
Vendor and model number	Delta Electronics DPS-1600FB
Number of power supplies	3
Wattage of each (W)	1600
Cooling fans	
Dimensions in millimeters	80x80x30
Number of fans	8

Figure 2: Configuration information for the server enclosure.

Appendix B – Inside our testing

Detailed information on the results of our testing

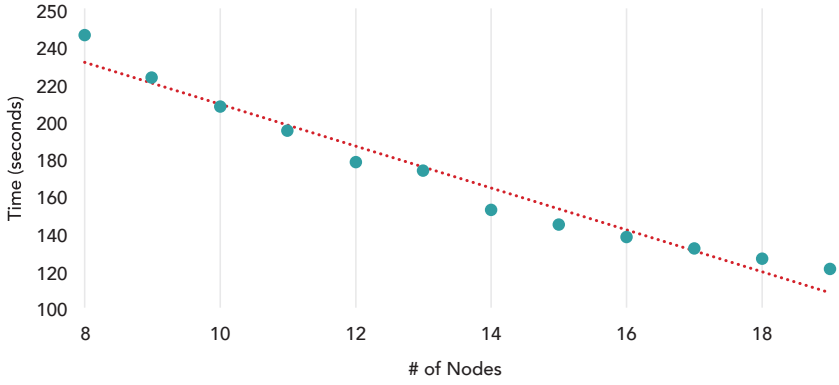
Figure 3 shows our median throughput results from eight server nodes to 19. The red dotted linear regression trend line illustrates the even scaling of throughput as we added server nodes, with the green dots demonstrating the median throughput at each server node count.



Server nodes	Median throughput (MB/s)
19	840.0
18	803.8
17	770.7
16	734.1
15	703.3
14	667.0
13	586.5
12	572.0
11	522.3
10	490.5
9	456.3
8	414.2

Figure 3: *k*-means median throughput.

Figure 4 shows the median times to process the sample data set from eight server nodes to 19. The red dotted linear regression trend line illustrates how evenly adding server nodes decreased time to process the sample data set, and the green dots are the actual median times at each server node count.



Server nodes	Median time (seconds)
19	122.0
18	127.5
17	132.9
16	139.6
15	145.7
14	153.6
13	174.7
12	179.1
11	196.2
10	208.9
9	224.6
8	247.4

Figure 4: *k*-means median time.

Detailed information on how we tested the DX2000

Complete the steps in the following link for UEFI deployment: http://docs.openstack.org/developer/ironic/deploy/install-guide.html#ipxe-setup

Configuring Red Hat Enterprise Linux OpenStack Platform Director

1. Create a Director user:

```
useradd stack
passwd stack

echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack
chmod 0440 /etc/sudoers.d/stack

su - stack
```

2. Create directories for templates and images:

```
mkdir ~/images
mkdir ~/templates
```

3. Set the hostname of the system:

```
sudo hostnamectl set-hostname director.test.lan
sudo hostnamectl set-hostname --transient director.test.lan

sudo vi /etc/hosts
```

An example of the content in the host file:

```
127.0.0.1   director.test.lan director localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         director.test.lan director localhost localhost.localdomain localhost6 localhost6.localdomain6
```

4. Register the system:

```
sudo subscription-manager register

sudo subscription-manager list --available --all
```

5. Locate the OpenStack pool ID in output, and replace it with the following:

```
sudo subscription-manager attach --pool=8a85f9814ff0134a014ff710053466b7
sudo subscription-manager repos --disable=*
sudo subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-7-server-optional-rpms --en-
able=rhel-7-server-extras-rpms --enable=rhel-7-server-openstack-7.0-rpms --enable=rhel-7-server-openstack-
7.0-director-rpms
sudo yum install -y yum-utils yum-plugin-priorities
sudo yum-config-manager --enable rhel-7-server-openstack-7.0-rpms --setopt="rhel-7-server-openstack-7.0-
rpms.priority=1"
sudo yum-config-manager --enable rhel-7-server-rpms --setopt="rhel-7-server-rpms.priority=1"
sudo yum-config-manager --enable rhel-7-server-optional-rpms --setopt="rhel-7-server-optional-rpms.priori-
ty=1"
sudo yum-config-manager --enable rhel-7-server-extras-rpms --setopt="rhel-7-server-extras-rpms.priority=1"
sudo yum-config-manager --enable rhel-7-server-openstack-7.0-director-rpms --setopt="rhel-7-server-open-
stack-7.0-director-rpms.priority=1"
sudo yum update -y
sudo reboot
```

6. Install the Director packages and additional widgets:

```
su - stack
sudo yum install -y python-rdomanager-oscplugin
sudo reboot

yum install -y wget vim sysstat
```

7. Complete SSL certificate configuration:

```
su - stack
cp /etc/pki/tls/openssl.cnf .

vim ~/openssl.cnf
```

a. Modify the following lines in the SSL certificate:

```
[ req_distinguished_name ]
countryName_default             = US
stateOrProvinceName_default     = Default State
localityName_default            = Default City
organizationalUnitName_default  = TestOrg
commonName_default              = 192.0.2.2

[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = IP:192.0.2.2

openssl genrsa -out privkey.pem 2048
openssl req -new -x509 -key privkey.pem -out cacert.pem -days 365 -config ~/openssl.cnf
cat cacert.pem privkey.pem > undercloud.pem

sudo mkdir /etc/pki/instack-certs
sudo cp ~/undercloud.pem /etc/pki/instack-certs/.
sudo semanage fcontext -a -t etc_t "/etc/pki/instack-certs(/.*)?"
sudo restorecon -R /etc/pki/instack-certs
```

8. To configure the Director file, modify the `dchp_end` and `discovery_iprange` as indicated below, and leave the remaining default options:

```
cp /usr/share/instack-undercloud/undercloud.conf.sample ~/undercloud.conf
vim ~/undercloud.conf
[DEFAULT]
image_path = .
local_ip = 192.0.2.1/24
undercloud_public_vip = 192.0.2.2
undercloud_admin_vip = 192.0.2.3
undercloud_service_certificate =
local_interface = eth1
masquerade_network = 192.0.2.0/24
dhcp_start = 192.0.2.5
dhcp_end = 192.0.2.99
network_cidr = 192.0.2.0/24
network_gateway = 192.0.2.1
discovery_interface = br-ctlplane
discovery_iprange = 192.0.2.100,192.0.2.199
discovery_runbench = false
undercloud_debug = true

[auth]
```

9. Complete installation of the Undercloud:

```
openstack undercloud install

source ~/stackrc
```

10. Log in to the Red Hat Network, and get the most recent URLs from the Red Hat OpenStack documentation for images of the Overcloud nodes:

```
cd ~/images
wget 'https://access.cdn.redhat.com/…/overcloud-full-7.3.0-56.tar'
wget 'https://access.cdn.redhat.com/…/deploy-ramdisk-ironic-7.3.0-39.tar'
wget 'https://access.cdn.redhat.com/…/discovery-ramdisk-7.3.0-56.tar'
for tarfile in *.tar*; do tar -xf $tarfile; done
openstack overcloud image upload --image-path /home/stack/images/
openstack image list
```

Sample output:

```
+--------------------------------------+------------------------+
| ID                                   | Name                   |
+--------------------------------------+------------------------+
| b10a15d7-d558-4d39-89a1-824e2e39c5f3 | bm-deploy-kernel       |
| 214f9cbf-a935-4d40-84fe-22e1d3764a51 | bm-deploy-ramdisk      |
| 7529fd44-84d4-4db2-8d82-36997d570a0e | overcloud-full         |
| b429415d-15d3-4911-a326-73c2cdf1c16d | overcloud-full-initrd  |
| dced3b92-fbae-4bd6-a0bb-795971b7ce77 | overcloud-full-vmlinuz |
+--------------------------------------+------------------------+

ls /httpboot –l
```

Sample output:

```
total 155504
-rw-r--r--. 1 ironic ironic       240 Feb 16 02:48 discoverd.ipxe
-rwxr-xr-x. 1 root   root     5152928 Feb 16 03:00 discovery.kernel
-rw-r--r--. 1 root   root   154075101 Feb 16 03:00 discovery.ramdisk
```

11. Enter a nameserver for the Overcloud:

```
neutron subnet-list
```

Sample output:

```
+--------------------------------------+------+--------------+--------------------------------------------+
| id                                   | name | cidr         | allocation_pools                           |
+--------------------------------------+------+--------------+--------------------------------------------+
| 58cb5657-53a6-45c9-aedc-5f04a6bd6793 |      | 192.0.2.0/24 | {“start”: “192.0.2.5”, “end”: “192.0.2.99”} |
+--------------------------------------+------+--------------+--------------------------------------------+

neutron subnet-update 58cb5657-53a6-45c9-aedc-5f04a6bd6793 --dns-nameserver 192.0.2.254
Updated subnet: 58cb5657-53a6-45c9-aedc-5f04a6bd6793

neutron subnet-show 58cb5657-53a6-45c9-aedc-5f04a6bd6793
+-------------------+-----------------------------------------------------------------+
| Field             | Value                                                           |
+-------------------+-----------------------------------------------------------------+
| allocation_pools  | {“start”: “192.0.2.5”, “end”: “192.0.2.99”}                     |
| cidr              | 192.0.2.0/24                                                    |
| dns_nameservers   | 192.0.2.254                                                     |
| enable_dhcp       | True                                                            |
| gateway_ip        | 192.0.2.1                                                       |
| host_routes       | {“destination”: “169.254.169.254/32”, “nexthop”: “192.0.2.1”}   |
| id                | 58cb5657-53a6-45c9-aedc-5f04a6bd6793                            |
| ip_version        | 4                                                               |
| ipv6_address_mode |                                                                 |
| ipv6_ra_mode      |                                                                 |
| name              |                                                                 |
| network_id        | 660174c5-5300-4efd-a6ca-227effbd7b2b                            |
| subnetpool_id     |                                                                 |
| tenant_id         | 9a658b9fea5641c38a5a052e4e0d5a3d                                |
+-------------------+-----------------------------------------------------------------+
```

12. Install the Overcloud:

```
source ~/stackrc

openstack baremetal import --json ~/chassis1.json
openstack baremetal configure boot
openstack baremetal list
```

Sample output:

```
+--------------------------------------+------+---------------+-------------+-----------------+-------------+
| UUID                                 | Name | Instance UUID | Power State | Provision State | Maintenance |
+--------------------------------------+------+---------------+-------------+-----------------+-------------+
| 28eb4296-a646-4162-8fce-4d619c7e37ae | None | None          | power off   | available       | False       |
| 5ecd4490-5c46-4016-8a5d-864725169915 | None | None          | power off   | available       | False       |
| 486900d0-5e16-4dc6-8f3c-22875624d1fa | None | None          | power off   | available       | False       |
| a46b6781-ddcb-4da8-87b2-818b761c489e | None | None          | power off   | available       | False       |
| adc866a1-6c3c-4918-aaf9-bcdd1a853bdc | None | None          | power off   | available       | False       |
| 7a14db26-9a70-45ec-b382-f0db0a20a11b | None | None          | power off   | available       | False       |
| 2c7480ab-afbf-460d-8cc3-472029401e42 | None | None          | power off   | available       | False       |
| 695a0c67-cb6c-4dc7-a868-c2ce8b337f09 | None | None          | power off   | available       | False       |
| d76790c2-5037-432f-a022-b5286f140f3c | None | None          | power off   | available       | False       |
| 3d38354e-f44f-47ae-9e39-dd6d656b2f95 | None | None          | power off   | available       | False       |
| b57b78a3-4315-4249-bc4f-f31d8f39d8e1 | None | None          | power off   | available       | False       |
| d99e9cd7-7386-4fc4-a7dc-4a83d4083064 | None | None          | power off   | available       | False       |
| fd0cf747-3362-4a8d-a9f4-ee3fcb5f9b15 | None | None          | power off   | available       | False       |
| 42cb3836-f295-4ceb-bfb5-a47f5bc0cc61 | None | None          | power off   | available       | False       |
| 427dd24d-74e1-4ca5-930a-3eb37803d282 | None | None          | power off   | available       | False       |
| fdc6bc25-848f-453c-a1ff-a948132909b7 | None | None          | power off   | available       | False       |
| b1b4af72-d1d7-4dda-90a8-910bbaefbed0 | None | None          | power off   | available       | False       |
| fbd82a87-fd4c-4e27-97ed-a30aad2b2ea5 | None | None          | power off   | available       | False       |
| c93bedd4-f51f-453b-b7e4-4935fc1d9925 | None | None          | power off   | available       | False       |
| 2da31b69-83a7-4a76-8283-23fd3a83a958 | None | None          | power off   | available       | False       |
| ce8e6efc-2f4b-413a-b018-1ec9e704bb1b | None | None          | power off   | available       | False       |
| 8a1475f5-df75-44fa-ace6-07175a6e205c | None | None          | power off   | available       | False       |
+--------------------------------------+------+---------------+-------------+-----------------+-------------+
```

```
ironic chassis-create -d “Chassis1 - 22 server modules @ 8 cores”
```

Sample output:

```
+-------------+--------------------------------------+
| Property    | Value                                |
+-------------+--------------------------------------+
| uuid        | 7a041d99-36f3-47f1-a5d7-879a83f2dc5b |
| description | Chassis1 - 22 server modules @ 8 cores |
| extra       | {}                                   |
+-------------+--------------------------------------+
```

```
openstack baremetal list | awk ‘/power off/{print $2}’ > ~/chassis1.txt
for i in `cat ~/chassis1.txt`; do ironic node-update $i replace chassis_uuid=7a041d99-36f3-47f1-a5d7-879a83f-
2dc5b ; done

openstack baremetal import --json ~/controller.json
openstack baremetal configure boot
openstack baremetal list
```

Sample output:

```
+--------------------------------------+------+---------------+-------------+-----------------+-------------+
| UUID                                 | Name | Instance UUID | Power State | Provision State | Maintenance |
+--------------------------------------+------+---------------+-------------+-----------------+-------------+
| 28eb4296-a646-4162-8fce-4d619c7e37ae | None | None          | power off   | available       | False       |
| 5ecd4490-5c46-4016-8a5d-864725169915 | None | None          | power off   | available       | False       |
| 486900d0-5e16-4dc6-8f3c-22875624d1fa | None | None          | power off   | available       | False       |
| a46b6781-ddcb-4da8-87b2-818b761c489e | None | None          | power off   | available       | False       |
| adc866a1-6c3c-4918-aaf9-bcdd1a853bdc | None | None          | power off   | available       | False       |
| 7a14db26-9a70-45ec-b382-f0db0a20a11b | None | None          | power off   | available       | False       |
| 2c7480ab-afbf-460d-8cc3-472029401e42 | None | None          | power off   | available       | False       |
| 695a0c67-cb6c-4dc7-a868-c2ce8b337f09 | None | None          | power off   | available       | False       |
| d76790c2-5037-432f-a022-b5286f140f3c | None | None          | power off   | available       | False       |
| 3d38354e-f44f-47ae-9e39-dd6d656b2f95 | None | None          | power off   | available       | False       |
| b57b78a3-4315-4249-bc4f-f31d8f39d8e1 | None | None          | power off   | available       | False       |
| d99e9cd7-7386-4fc4-a7dc-4a83d4083064 | None | None          | power off   | available       | False       |
| fd0cf747-3362-4a8d-a9f4-ee3fcb5f9b15 | None | None          | power off   | available       | False       |
| 42cb3836-f295-4ceb-bfb5-a47f5bc0cc61 | None | None          | power off   | available       | False       |
| 427dd24d-74e1-4ca5-930a-3eb37803d282 | None | None          | power off   | available       | False       |
| fdc6bc25-848f-453c-a1ff-a948132909b7 | None | None          | power off   | available       | False       |
| b1b4af72-d1d7-4dda-90a8-910bbaefbed0 | None | None          | power off   | available       | False       |
| fbd82a87-fd4c-4e27-97ed-a30aad2b2ea5 | None | None          | power off   | available       | False       |
| c93bedd4-f51f-453b-b7e4-4935fc1d9925 | None | None          | power off   | available       | False       |
| 2da31b69-83a7-4a76-8283-23fd3a83a958 | None | None          | power off   | available       | False       |
| ce8e6efc-2f4b-413a-b018-1ec9e704bb1b | None | None          | power off   | available       | False       |
| 8a1475f5-df75-44fa-ace6-07175a6e205c | None | None          | power off   | available       | False       |
| 1c8e5593-c504-487e-b407-5e29e0e3899e | None | None          | power off   | available       | False       |
+--------------------------------------+------+---------------+-------------+-----------------+-------------+
```

```
openstack baremetal introspection bulk start

sudo journalctl -l -u openstack-ironic-discoverd -u openstack-ironic-discoverd-dnsmasq -u openstack-iron-
ic-conductor -f

watch -n 60 -d 'openstack baremetal list | grep -v COMPLETE'

openstack baremetal list | awk '/None/{print "ironic node-update "$2" add properties/capabilities='\''pro-
file:compute,boot_option:local'\''"}' > baremetal_assign.sh
```

13. Modify the last two lines in `baremetal_assign.sh`:

```
vim baremetal_assign.sh
```

Sample output:

```
ironic node-update 28eb4296-a646-4162-8fce-4d619c7e37ae add properties/capabilities='profile:compute,boot_op-
tion:local'
ironic node-update 5ecd4490-5c46-4016-8a5d-864725169915 add properties/capabilities='profile:compute,boot_op-
tion:local'
ironic node-update 486900d0-5e16-4dc6-8f3c-22875624d1fa add properties/capabilities='profile:compute,boot_op-
tion:local'
ironic node-update a46b6781-ddcb-4da8-87b2-818b761c489e add properties/capabilities='profile:compute,boot_op-
tion:local'
ironic node-update adc866a1-6c3c-4918-aaf9-bcdd1a853bdc add properties/capabilities='profile:compute,boot_op-
tion:local'
ironic node-update 7a14db26-9a70-45ec-b382-f0db0a20a11b add properties/capabilities='profile:compute,boot_op-
tion:local'
ironic node-update 2c7480ab-afbf-460d-8cc3-472029401e42 add properties/capabilities='profile:compute,boot_op-
tion:local'
ironic node-update 695a0c67-cb6c-4dc7-a868-c2ce8b337f09 add properties/capabilities='profile:compute,boot_op-
tion:local'
ironic node-update d76790c2-5037-432f-a022-b5286f140f3c add properties/capabilities='profile:compute,boot_op-
tion:local'
ironic node-update 3d38354e-f44f-47ae-9e39-dd6d656b2f95 add properties/capabilities='profile:compute,boot_op-
tion:local'
ironic node-update b57b78a3-4315-4249-bc4f-f31d8f39d8e1 add properties/capabilities='profile:compute,boot_op-
tion:local'
```

```
ironic node-update d99e9cd7-7386-4fc4-a7dc-4a83d4083064 add properties/capabilities='profile:compute,boot_
option:local'
ironic node-update fd0cf747-3362-4a8d-a9f4-ee3fcb5f9b15 add properties/capabilities='profile:compute,boot_
option:local'
ironic node-update 42cb3836-f295-4ceb-bfb5-a47f5bc0cc61 add properties/capabilities='profile:compute,boot_
option:local'
ironic node-update 427dd24d-74e1-4ca5-930a-3eb37803d282 add properties/capabilities='profile:compute,boot_
option:local'
ironic node-update fdc6bc25-848f-453c-a1ff-a948132909b7 add properties/capabilities='profile:compute,boot_op-
tion:local'
ironic node-update b1b4af72-d1d7-4dda-90a8-910bbaefbed0 add properties/capabilities='profile:compute,boot_
option:local'
ironic node-update fbd82a87-fd4c-4e27-97ed-a30aad2b2ea5 add properties/capabilities='profile:compute,boot_
option:local'
ironic node-update c93bedd4-f51f-453b-b7e4-4935fc1d9925 add properties/capabilities='profile:compute,boot_
option:local'
ironic node-update 2da31b69-83a7-4a76-8283-23fd3a83a958 add properties/capabilities='profile:compute,boot_
option:local'
ironic node-update ce8e6efc-2f4b-413a-b018-1ec9e704bb1b add properties/capabilities='profile:compute,boot_
option:local'
ironic node-update 8a1475f5-df75-44fa-ace6-07175a6e205c add properties/capabilities='profile:compute,boot_
option:local'
```

Modify the following lines:

```
ironic node-update 778afb64-972c-4eff-b817-b4231c004599 add properties/capabilities='profile:control,boot_op-
tion:local'
ironic node-update 6f0537ff-1d42-45a8-af72-13147fab2d33 add properties/capabilities='profile:control,boot_op-
tion:local'

openstack flavor create --id auto --ram 98304 --disk 54 --vcpus 24 control
openstack flavor create --id auto --ram 65536 --disk 237 --vcpus 16 compute
openstack flavor create --id auto --ram 4096 --disk 40 --vcpus 1 baremetal


cp -r /usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans ~/templates/nic-configs
```

14. Edit control-scale and compute-scale to match the environment:

```
openstack overcloud deploy --templates -e /usr/share/openstack-tripleo-heat-templates/environments/net-
work-isolation.yaml -e ~/templates/network-environment.yaml -e ~/templates/storage-environment.yaml --con-
trol-scale 2 --compute-scale 22 --control-flavor control --compute-flavor compute --ntp-server 192.0.2.254
--neutron-network-type vxlan --neutron-tunnel-types vxlan

heat stack-list --show-nested | grep -v COMPLETE
watch -n 60 -d 'heat stack-list --show-nested | grep -v COMPLETE'
```

15. Build the most recent Intel NIC drivers:

```
rpmbuild -tb ixgbe-4.3.13.tar.gz
cp ~/rpmbuild/RPMS/x86_64/*.rpm ~
scp *.rpm heat-admin@192.0.2.XXX:/home/heat-admin/
ssh heat-admin@192.0.2.XX "sudo yum localinstall -y i*.rpm ; sudo dracut --force"
```

16. Reboot the nodes:

```
nova list
nova reboot NODE_INSTANCE_ID
```

17. Use the CLI to create an advanced Overcloud with Ceph nodes:

```
ssh heat-admin@192.0.2.XX
```

Replace XX with the IP address of a controller node Note: `ipaddr` is the IP address of your controller servers' IPMI interface.

```
sudo pcs stonith create my-ipmilan-for-controller01 fence_ipmilan pcmk_host_list=overcloud-controller-0 ip-
addr=192.168.0.251 login=Administrator passwd=Administrator lanplus=1 cipher=1 op monitor interval=60s
sudo pcs constraint location my-ipmilan-for-controller01 avoids overcloud-controller-0
```

```
sudo pcs stonith create my-ipmilan-for-controller02 fence_ipmilan pcmk_host_list=overcloud-controller-1 ip-
addr=192.168.0.252 login=Administrator passwd=Administrator lanplus=1 cipher=1 op monitor interval=60s
sudo pcs constraint location my-ipmilan-for-controller02 avoids overcloud-controller-1

sudo pcs stonith show
sudo pcs property set stonith-enabled=true
sudo pcs property show
sudo pcs status
```

18. Create the Overcloud tenant network:

```
source ~/overcloudrc
neutron net-create default --shared
neutron subnet-create --name default --gateway 172.20.1.1 default 172.20.0.0/16
neutron net-list
```

19. Create the Overcloud external network (using a non-native VLAN):

```
source ~/overcloudrc
neutron net-create nova --router:external --provider:network_type vlan --provider:physical_network datacen-
tre --provider:segmentation_id 1200
neutron subnet-create --name nova --enable_dhcp=False --allocation-pool=start=10.1.1.51,end=10.1.1.250
--gateway=10.1.1.1 nova 10.1.1.0/24
```

20. Validate your Overcloud:

```
openstack role list
keystone role-create --name heat_stack_owner
mkdir ~/tempest
cd ~/tempest
/usr/share/openstack-tempest-kilo/tools/configure-tempest-directory
tools/config_tempest.py --deployer-input ~/tempest-deployer-input.conf --debug --create identity.uri $OS_
AUTH_URL identity.admin_password $OS_PASSWORD
```

21. Configure the router:

```
neutron router-create default-router
neutron router-interface-add default-router default
neutron router-gateway-set default-router nova
```

22. Run the following commands on both controller nodes to complete the DHCP/DNSMASQ fix for DSN forwarding:

```
sudo openstack-config --set /etc/neutron/dhcp_agent.ini DEFAULT dnsmasq_dns_servers 10.1.1.1
sudo systemctl restart neutron-dhcp-agent
```

Configuring Red Hat Enterprise Linux OpenStack Platform Manager

1. Install Red Hat Enterprise Linux 7.2 Server with GUI, DNS Server, and all virtualization groups:

```
setenforce 0
sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
firewall-cmd --permanent --direct --add-rule ipv4 nat POSTROUTING 0 -o enp3s0f0 -j MASQUERADE
firewall-cmd --permanent --direct --add-rule ipv4 filter FORWARD 0 -i br1 -o enp3s0f0 -j ACCEPT
firewall-cmd --permanent --direct --add-rule ipv4 filter FORWARD 0 -i enp3s0f0 -o br1 -m state --state RELAT-
ED,ESTABLISHED -j ACCEPT
firewall-cmd --permanent --direct --add-rule ipv4 filter FORWARD 0 -i br2 -o enp3s0f0 -j ACCEPT
firewall-cmd --permanent --direct --add-rule ipv4 filter FORWARD 0 -i enp3s0f0 -o br2 -m state --state RELAT-
ED,ESTABLISHED -j ACCEPT
firewall-cmd --permanent --direct --add-rule ipv4 filter FORWARD 0 -i br3 -o enp3s0f0 -j ACCEPT
firewall-cmd --permanent --direct --add-rule ipv4 filter FORWARD 0 -i enp3s0f0 -o br3 -m state --state RELAT-
ED,ESTABLISHED -j ACCEPT
firewall-cmd --reload
hostnamectl set-hostname manager.test.lan
hostnamectl set-hostname --transient manager.test.lan

sudo subscription-manager register
sudo subscription-manager list --available –all
```

2. Locate the OpenStack `pool_id` in output, and replace it with the following ID in the next command:

```
sudo subscription-manager attach --pool=<pool_id>
sudo subscription-manager repos --disable=*
sudo subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-7-server-optional-rpms --en-
able=rhel-7-server-extras-rpms

yum update -y
reboot
```

3. Install Tiger VNC server:

```
yum install -y tigervnc-server

cp /usr/lib/systemd/system/vncserver@.service /etc/systemd/system/vncserver@.service
vim /etc/systemd/system/vncserver@.service
```

4. Modify the following lines from USER to root:

```
ExecStart=/usr/sbin/runuser -l root -c "/usr/bin/vncserver %i"
PIDFile=/root/.vnc/%H%i.pid

systemctl daemon-reload
su - root
vncpasswd

firewall-cmd --permanent --add-port=5901/tcp
firewall-cmd --reload

systemctl start vncserver@:1.service
systemctl enable vncserver@:1.service
```

5. Use Virtual Machine Manager to create two bridge interfaces:

```
br1: enp3s0f1: 192.168.0.1/24
br2: ens1: 192.0.2.254/24
```

6. Configure the DHCP server:

```
yum install -y dhcp
vim /etc/dhcp/dhcpd.conf
subnet 192.168.0.0 netmask 255.255.255.0 {
        option routers                  192.168.0.1;
        option subnet-mask              255.255.255.0;
        option domain-search              "test.lan";
        option domain-name-servers       192.168.0.1;

        option time-offset             -18000;     # Eastern Standard Time
        range 192.168.0.51 192.168.0.99;

        include "/etc/dhcp/mms-static.conf";
}

echo > /etc/dhcp/mms-static.conf
systemctl enable dhcpd
systemctl start dhcpd
```

7. Configure DNS:

```
yum install -y bind
firewall-cmd --permanent --add-service=dns
firewall-cmd --reload
vim /etc/named.conf
```

   a. Modify the following entries:

```
listen-on port 53 { 127.0.0.1; };
```

```
        allow-query           { localhost; };
    dnssec-validation yes;


        listen-on port 53 { any; };
        allow-query           { any; };
    dnssec-validation no;
```

b. Append these lines to the end of the file:

```
zone "test.lan" {
   type master;
   file "test.lan.zone";
   allow-update { none; };
};

zone "0.168.192.in-addr.arpa" {
   type master;
   file "external.zone";
   allow-update { none; };
};

zone "2.0.192.in-addr.arpa" {
   type master;
   file "deployment.zone";
   allow-update { none; };
};
```

8. Configure the NTP time server:

```
yum install -y ntp
sed -i '/^server [^ ]* iburst/d' /etc/ntp.conf
echo "server 10.41.0.5 iburst" >> /etc/ntp.conf
systemctl start ntpd
systemctl enable ntpd
```

9. Configure the DX2000 Management tool:

```
yum install -y ipmitool OpenIPMI
systemctl enable ipmi
systemctl enable ipmievd
systemctl start ipmi
systemctl start ipmievd

cd /opt/mng/
./mng_util
```

Sample output:

```
mng_util version 01.03

 > search 192.168.0.51-192.168.0.99
```

Sample output:

```
Chassis serial : GFH9PA312A0006
Board        ManagementLAN MAC IP              DataLAN1 MAC      DataLAN2 MAC
-----------  ----------------- --------------- ----------------- -----------------
CSC          40:8d:5c:17:3c:71 192.168.0.51
LAN-SW1      40:8d:5c:57:94:a0 192.168.0.53
LAN-SW2      40:8d:5c:57:a2:10 192.168.0.52
CPU Board1   40:8d:5c:5e:ad:9a 192.168.0.69    40:8d:5c:5e:ad:98 40:8d:5c:5e:ad:99
CPU Board3   40:8d:5c:5e:ae:0c 192.168.0.63    40:8d:5c:5e:ae:0a 40:8d:5c:5e:ae:0b
CPU Board5   40:8d:5c:5e:ae:cc 192.168.0.72    40:8d:5c:5e:ae:ca 40:8d:5c:5e:ae:cb
CPU Board7   40:8d:5c:5e:af:4a 192.168.0.66    40:8d:5c:5e:af:48 40:8d:5c:5e:af:49
CPU Board9   40:8d:5c:5e:ac:ce 192.168.0.73    40:8d:5c:5e:ac:cc 40:8d:5c:5e:ac:cd
CPU Board11  40:8d:5c:5e:ab:e1 192.168.0.64    40:8d:5c:5e:ab:df 40:8d:5c:5e:ab:e0
CPU Board13  40:8d:5c:5e:ae:77 192.168.0.68    40:8d:5c:5e:ae:75 40:8d:5c:5e:ae:76
CPU Board15  40:8d:5c:5e:ad:b5 192.168.0.61    40:8d:5c:5e:ad:b3 40:8d:5c:5e:ad:b4
CPU Board17  40:8d:5c:5e:ab:b4 192.168.0.55    40:8d:5c:5e:ab:b2 40:8d:5c:5e:ab:b3
```

```
CPU Board19  40:8d:5c:5e:af:4d 192.168.0.74    40:8d:5c:5e:af:4b 40:8d:5c:5e:af:4c
CPU Board20  40:8d:5c:5e:ac:c5 192.168.0.62    40:8d:5c:5e:ac:c3 40:8d:5c:5e:ac:c4
CPU Board21  40:8d:5c:5e:ac:26 192.168.0.57    40:8d:5c:5e:ac:24 40:8d:5c:5e:ac:25
CPU Board22  40:8d:5c:5e:ab:78 192.168.0.59    40:8d:5c:5e:ab:76 40:8d:5c:5e:ab:77
CPU Board23  40:8d:5c:5e:ad:c1 192.168.0.75    40:8d:5c:5e:ad:bf 40:8d:5c:5e:ad:c0
CPU Board24  40:8d:5c:5e:ab:fc 192.168.0.60    40:8d:5c:5e:ab:fa 40:8d:5c:5e:ab:fb
CPU Board25  40:8d:5c:5e:ad:a0 192.168.0.56    40:8d:5c:5e:ad:9e 40:8d:5c:5e:ad:9f
CPU Board26  40:8d:5c:5e:ae:3f 192.168.0.67    40:8d:5c:5e:ae:3d 40:8d:5c:5e:ae:3e
CPU Board27  40:8d:5c:5e:ac:a4 192.168.0.58    40:8d:5c:5e:ac:a2 40:8d:5c:5e:ac:a3
CPU Board29  40:8d:5c:5e:ac:7a 192.168.0.71    40:8d:5c:5e:ac:78 40:8d:5c:5e:ac:79
CPU Board31  40:8d:5c:5e:af:41 192.168.0.70    40:8d:5c:5e:af:3f 40:8d:5c:5e:af:40
CPU Board33  40:8d:5c:5e:ab:d8 192.168.0.65    40:8d:5c:5e:ab:d6 40:8d:5c:5e:ab:d7
CPU Board35  40:8d:5c:5e:ae:3c 192.168.0.54    40:8d:5c:5e:ae:3a 40:8d:5c:5e:ae:3b

 > savelist -I all -f /root/maclist.csv
 > quit

cat /root/maclist.csv | awk -F',' '/CSC|LAN-|CPU/{print $2","$3","$5","$6}' | sed -e 's/CPU Board/srv/' -e
's/LAN-SW/switch/' -e 's/CSC/csc/' > /root/mms1.csv
MMS=1; cat /root/mms${MMS}.csv | awk -F',' '{printf "host mms-%s { hardware ethernet %s; fixed-address
192.168.0.%d; }\n",$1,$2,$1}' | sed -e "s/mms-/mms${MMS}-/" -e "s/\.csc/.${MMS}0/" -e "s/\.switch/.${MMS}/"
-e "s/\.srv/.${MMS}/" > /etc/dhcp/mms-static.conf
```

Note: If the scrips fail to execute correctly, edit the /etc/dhcp/mms-static.conf file to match the following:

```
vim /etc/dhcp/mms-static.conf
host controller-ipmi { hardware ethernet 78:e7:d1:91:30:4e; fixed-address 192.168.0.252; }
host mms1-csc { hardware ethernet 40:8d:5c:17:3c:71; fixed-address 192.168.0.10; }
host mms1-switch1 { hardware ethernet 40:8d:5c:57:94:a0; fixed-address 192.168.0.11; }
host mms1-switch2 { hardware ethernet 40:8d:5c:57:a2:10; fixed-address 192.168.0.12; }
host mms1-srv1 { hardware ethernet 40:8d:5c:5e:ad:9a; fixed-address 192.168.0.101; }
host mms1-srv3 { hardware ethernet 40:8d:5c:5e:ae:0c; fixed-address 192.168.0.103; }
host mms1-srv5 { hardware ethernet 40:8d:5c:5e:ae:cc; fixed-address 192.168.0.105; }
host mms1-srv7 { hardware ethernet 40:8d:5c:5e:af:4a; fixed-address 192.168.0.107; }
host mms1-srv9 { hardware ethernet 40:8d:5c:5e:ac:ce; fixed-address 192.168.0.109; }
host mms1-srv11 { hardware ethernet 40:8d:5c:5e:ab:e1; fixed-address 192.168.0.111; }
host mms1-srv13 { hardware ethernet 40:8d:5c:5e:ae:77; fixed-address 192.168.0.113; }
host mms1-srv15 { hardware ethernet 40:8d:5c:5e:ad:b5; fixed-address 192.168.0.115; }
host mms1-srv17 { hardware ethernet 40:8d:5c:5e:ab:b4; fixed-address 192.168.0.117; }
host mms1-srv19 { hardware ethernet 40:8d:5c:5e:af:4d; fixed-address 192.168.0.119; }
host mms1-srv20 { hardware ethernet 40:8d:5c:5e:ac:c5; fixed-address 192.168.0.120; }
host mms1-srv21 { hardware ethernet 40:8d:5c:5e:ac:26; fixed-address 192.168.0.121; }
host mms1-srv22 { hardware ethernet 40:8d:5c:5e:ab:78; fixed-address 192.168.0.122; }
host mms1-srv23 { hardware ethernet 40:8d:5c:5e:ad:c1; fixed-address 192.168.0.123; }
host mms1-srv24 { hardware ethernet 40:8d:5c:5e:ab:fc; fixed-address 192.168.0.124; }
host mms1-srv25 { hardware ethernet 40:8d:5c:5e:ad:a0; fixed-address 192.168.0.125; }
host mms1-srv26 { hardware ethernet 40:8d:5c:5e:ae:3f; fixed-address 192.168.0.126; }
host mms1-srv27 { hardware ethernet 40:8d:5c:5e:ac:a4; fixed-address 192.168.0.127; }
host mms1-srv29 { hardware ethernet 40:8d:5c:5e:ac:7a; fixed-address 192.168.0.129; }
host mms1-srv31 { hardware ethernet 40:8d:5c:5e:af:41; fixed-address 192.168.0.131; }
host mms1-srv33 { hardware ethernet 40:8d:5c:5e:ab:d8; fixed-address 192.168.0.133; }
host mms1-srv35 { hardware ethernet 40:8d:5c:5e:ae:3c; fixed-address 192.168.0.135; }
```

10. Enable the NFS server:

```
mkdir -p /export/cinder
mkdir -p /export/glance
chmod 777 /export/*
vim /etc/exports
```

a. Append the following to the file:

```
/export/cinder 172.18.0.0/24(rw,no_root_squash)
/export/glance 172.18.0.0/24(rw,no_root_squash)

firewall-cmd --permanent --zone public --add-service mountd
firewall-cmd --permanent --zone public --add-service rpc-bind
firewall-cmd --permanent --zone public --add-service nfs
firewall-cmd --permanent --zone public --add-service ntp
firewall-cmd --reload
```

```
systemctl enable rpcbind
systemctl enable nfs-server
systemctl enable nfs-lock
systemctl enable nfs-idmap
systemctl restart rpcbind
systemctl restart nfs-server
systemctl restart nfs-lock
systemctl restart nfs-idmap
```

11. Configure Red Hat Enterprise Linux 7 and the HDP 2.4 mirror:

```
yum install -y yum-utils createrepo httpd
systemctl enable httpd
systemctl restart httpd
firewall-cmd --permanent --zone public --add-service http
firewall-cmd --reload

wget -nv http://public-repo-1.hortonworks.com/ambari/centos7/2.x/updates/2.2.1.0/ambari.repo -O /etc/yum.
repos.d/ambari.repo
wget -nv http://public-repo-1.hortonworks.com/HDP/centos7/2.x/updates/2.4.0.0/hdp.repo -O /etc/yum.repos.d/
hdp.repo

mkdir -p /var/www/html/repos
cd /var/www/html/repos
reposync -l
for repo in `ls`; do createrepo $repo ; done
wget http://public-repo-1.hortonworks.com/ambari/centos6/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins
```

12. Edit the Red Hat Enterprise Linux guest KVM image.

```
cd /var/lib/libvirt/images
mkdir /mnt/guest
guestmount --rw -i -a rhel-guest-image-7.2-20160301.0.x86_64_hdp.img /mnt/guest
cd /mnt/guest
```

a. Disable SELinux in the guest KVM image:

```
sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /mnt/guest/etc/selinux/config
```

b. Update the repository in guest image to point to the local repository:

```
vi /mnt/guest/etc/yum.repos.d/ambari.repo

#VERSION_NUMBER=2.2.1.1-70
[Updates-ambari-2.2.1.1]
name=ambari-2.2.1.1 - Updates
baseurl=http://10.1.1.1/repos/Updates-ambari-2.2.1.1
gpgcheck=1
gpgkey=http://10.1.1.1/repos/RPM-GPG-KEY-Jenkins
enabled=1
priority=1

vi /mnt/guest/etc/yum.repos.d/hdp.repo

#VERSION_NUMBER=2.4.0.0-169
[HDP-2.4.0.0]
name=HDP Version - HDP-2.4.0.0
baseurl=http://10.1.1.1/repos/HDP-2.4.0.0
gpgcheck=1
gpgkey=http://10.1.1.1/repos/HDP-2.4.0.0/RPM-GPG-KEY-Jenkins
enabled=1
priority=1

[HDP-UTILS-1.1.0.20]
name=HDP Utils Version - HDP-UTILS-1.1.0.20
baseurl=http://10.1.1.1/repos/HDP-UTILS-1.1.0.20
gpgcheck=1
gpgkey=http://10.1.1.1/repos/HDP-2.4.0.0/RPM-GPG-KEY-Jenkins
enabled=1
priority=1
```

```
vi /mnt/guest/etc/yum.repos.d/rh.repo

[rhel-7-server-rpms]
baseurl = http://10.1.1.1/repos/rhel-7-server-rpms
ui_repoid_vars = releasever basearch
name = Red Hat Enterprise Linux 7 Server (RPMs)
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
enabled = 1
gpgcheck = 1

[rhel-7-server-extras-rpms]
baseurl = http://10.1.1.1/repos/rhel-7-server-extras-rpms
ui_repoid_vars = basearch
name = Red Hat Enterprise Linux 7 Server - Extras (RPMs)
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
enabled = 1
gpgcheck = 1

[rhel-7-server-optional-rpms]
baseurl = http://10.1.1.1/repos/rhel-7-server-optional-rpms
ui_repoid_vars = releasever basearch
name = Red Hat Enterprise Linux 7 Server - Optional (RPMs)
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
enabled = 1
gpgcheck = 1

[rhel-7-server-rh-common-rpms]
baseurl = http://10.1.1.1/repos/rhel-7-server-rh-common-rpms
ui_repoid_vars = releasever basearch
name = Red Hat Enterprise Linux 7 Server - RH Common (RPMs)
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
enabled = 1
gpgcheck = 1
```

13. Install the priorities plugin on the image and enable it:

```
yum --installroot=/mnt/guest install -y yum-plugin-priorities
vi /mnt/guest/etc/yum/pluginconf.d/priorities.conf
[main]
enabled = 1
gpgcheck = 0


vi /etc/yum/pluginconf.d/priorities.conf
[main]
enabled = 1
gpgcheck = 0
```

14. Install updates:

```
yum --installroot=/mnt/guest update -y
```

15. Install Ambari required packages and remove `chrony`:

```
yum --installroot=/mnt/guest remove -y chrony
yum --installroot=/mnt/guest install -y openssh-clients curl unzip tar wget openssl python ntp ja-
va-1.8.0-openjdk-devel postgresql-jdbc postgresql-odbc sysstat numpy
```

16. Clean up installers in the guest image:

```
yum --installroot=/mnt/guest clean all
```

17. Enable NTP in guest:

```
ln -s /usr/lib/systemd/system/ntpd.service /mnt/guest/etc/systemd/system/multi-user.target.wants/ntpd.ser-
vice
sed -i '/^server [^ ]* iburst/d' /mnt/guest/etc/ntp.conf
echo "server 10.1.1.1 iburst" >> /mnt/guest/etc/ntp.conf
```

18. Zero fill the guest image, convert the file, and compress it:

```
dd if=/dev/zero of=/mnt/guest/tmp.bin bs=1M ; sync ; sleep 1 ; sync ; rm -f /mnt/guest/tmp.bin ; sync
cd /var/lib/libvirt/images
umount /mnt/guest
qemu-img convert -c -O qcow2 rhel-guest-image-7.2-20160301.0.x86_64_hdp.img rhel-guest-image-7.2-
20160301.0.x86_64_hdp.qcow2
```

19. Install Ambari server:

```
ssh -i hdpkey cloud-user@10.1.1.185
sudo su
yum install -y ambari-server

ambari-server setup --jdbc-db=postgres --jdbc-driver=/usr/share/java/postgresql-jdbc.jar --java-home=/usr/
lib/jvm/java-1.8.0-openjdk
```

Sample output:

```
Using python  /usr/bin/python
Setup ambari-server
Copying /usr/share/java/postgresql-jdbc.jar to /var/lib/ambari-server/resources
JDBC driver was successfully initialized.
Ambari Server 'setup' completed successfully.


ambari-server setup --java-home=/usr/lib/jvm/java-1.8.0-openjdk
```

Sample output:

```
Using python  /usr/bin/python
Setup ambari-server
Checking SELinux...
SELinux status is 'disabled'
Customize user account for ambari-server daemon [y/n] (n)?
Adjusting ambari-server permissions and ownership...
Checking firewall status...
Redirecting to /bin/systemctl status  iptables.service

Checking JDK...
WARNING: JAVA_HOME /usr/lib/jvm/java-1.8.0-openjdk must be valid on ALL hosts
WARNING: JCE Policy files are required for configuring Kerberos security. If you plan to use Kerberos,please
make sure JCE Unlimited Strength Jurisdiction Policy Files are valid on all hosts.
Completing setup...
Configuring database...
Enter advanced database configuration [y/n] (n)?
Configuring database...
Default properties detected. Using built-in database.
Configuring ambari database...
Checking PostgreSQL...
Running initdb: This may take upto a minute.
Initializing database ... OK


About to start PostgreSQL
Configuring local database...
Connecting to local database...done.
Configuring PostgreSQL...
Restarting PostgreSQL
Extracting system views...
ambari-admin-2.2.1.1.70.jar
......
Adjusting ambari-server permissions and ownership...
Ambari Server 'setup' completed successfully.


ambari-server start
```

Sample output:

```
Using python  /usr/bin/python
Starting ambari-server
Ambari Server running with administrator privileges.
Organizing resource files at /var/lib/ambari-server/resources...
Server PID at: /var/run/ambari-server/ambari-server.pid
Server out at: /var/log/ambari-server/ambari-server.out
Server log at: /var/log/ambari-server/ambari-server.log
Waiting for server start...................
Ambari Server 'start' completed successfully.
```

```
ambari-server status

Sample output:
http://10.1.1.185:8080
admin/admin
```

20. To complete Ambari web setup, open the URL from the server setup in step 19, log in with the appropriate credentials, and create a cluster:

    a. Type `cluster1` for the cluster name.

    b. Type `host-172-21-0-[66-88].openstacklocal` for the target host information. Browse to hdpkey SSH key, and type `cloud-user` for the SSH User Account.

    c. Uncheck the following options:

    Sqoop
    Oozie
    Falcon
    Flume
    Accumulo
    Atlas
    Knox
    Slider
    SmartSense

    d. Distribute all services across the first three nodes or your three master instances with the exception of Metric Collector, which should be assigned to a client.

    e. Type `Password1` for the Hive database password.

    f. Set a password on the Hive database: Password1

    g. Accept defaults and continue.

    h. Accept defaults and continue.

    i. Complete web setup.

Configuring the HiBench client instance

From the Ambari GUI, add another client instance, add the Apache Kafka broker role, and complete the following steps:

1. Set the maximum number of client connections to 60:

```
maxClientCnxns=60
```

2. Install HiBench.

    a. Add a floating IP to the client instance:

```
ssh -i hdpkey cloud-user@10.1.1.186
sudo su - hdfs
hdfs dfs -mkdir /HiBench
```

```
hdfs dfs -chown -R cloud-user:hadoop /HiBench
hdfs dfs -mkdir /home/cloud-user
hdfs dfs -chown cloud-user /user/cloud-user
exit
yum install -y maven git vim numpy blas64 lapack64
git clone https://github.com/intel-hadoop/HiBench.git
cd HiBench/src
```

b. Open the datagen pom XML file. Replace the following:

```
vim streambench/datagen/pom.xml

<dependency>
     <groupId>org.apache.kafka</groupId>
     <artifactId>kafka-clients</artifactId>
     <version>0.8.1</version>
     <scope>system</scope>
     <systemPath>${basedir}/lib/kafka-clients-0.8.1.jar</systemPath>
   </dependency>

<dependency>
     <groupId>org.apache.kafka</groupId>
     <artifactId>kafka-clients</artifactId>
     <version>0.8.1</version>
   </dependency>
```

c. Open the following XML file, and replace the following:

```
vim streambench/sparkbench/pom.xml

     <exclusion>
          <groupId>org.sonatype.sisu.inject</groupId>
          <artifactId>*</artifactId>
        </exclusion>
        <exclusion>
          <groupId>org.xerial.snappy</groupId>
          <artifactId>*</artifactId>
        </exclusion>

        <exclusion>
          <groupId>org.sonatype.sisu.inject</groupId>
          <artifactId>inject</artifactId>
        </exclusion>
        <exclusion>
          <groupId>org.xerial.snappy</groupId>
          <artifactId>snappy</artifactId>
        </exclusion>
```

d. Complete the HiBench installation:

```
mvn install:install-file -Dfile=streambench/datagen/lib/kafka-clients-0.8.1.jar -DgroupId=org.apache.kafka
-DartifactId=kafka-clients -Dversion=0.8.1 -Dpackaging=jar
mvn clean package -D spark1.6 -D MR2
cd ..

cp conf/99-user_defined_properties.conf.template conf/99-user_defined_properties.conf

grep -v "^#" conf/99-user_defined_properties.conf | grep -v "^$"
```

Sample output:

```
hibench.hadoop.home             /usr/hdp/current/hadoop-client
hibench.spark.home              /usr/hdp/current/spark-client
hibench.hadoop.mapreduce.home   /usr/hdp/current/hadoop-mapreduce-client
hibench.hdfs.master             hdfs://host-172-21-0-66.openstacklocal:8020
hibench.spark.master            yarn-client
hibench.hadoop.release          hdp
hibench.hadoop.version           hadoop2
hibench.spark.version          spark1.6
hibench.default.map.parallelism         76
hibench.default.shuffle.parallelism     76
```

```
hibench.yarn.executor.num 19
hibench.yarn.executor.cores      16
spark.executor.memory    50G
spark.driver.memory      8G
spark.rdd.compress            false
spark.shuffle.compress           false
spark.broadcast.compress  false
spark.io.compression.codec    org.apache.spark.io.SnappyCompressionCodec
spark.akka.frameSize          1000
spark.akka.timeout            600
spark.kryoserializer.buffer        2000mb
hibench.scale.profile                    census
hibench.compress.profile          disable
hibench.compress.codec.profile           snappy
hibench.streamingbench.benchname identity
hibench.streamingbench.scale.profile    ${hibench.scale.profile}
hibench.streamingbench.zookeeper.host   host-172-21-0-66.openstacklocal:2181
hibench.streamingbench.brokerList       host-172-21-0-89.openstacklocal:9021
hibench.streamingbench.storm.home       /usr/hdp/current/storm-client
hibench.streamingbench.kafka.home       /usr/hdp/current/kafka-broker
hibench.streamingbench.storm.nimbus     host-172-21-0-66.openstacklocal
hibench.streamingbench.partitions       1
```

This project was commissioned by NEC Corp.

Facts matter.®