Apache Hadoop on IBM PowerKVM

Hadoop configuration on IBM POWER8 processor-based systems running IBM PowerKVM

Pradipta Kumar Banerjee (bpradipt@in.ibm.com)
Ashish Kumar (ashishk4@in.ibm.com)
Poornima Nayak (Poornima.Nayak@in.ibm.com)
Yogananth Subramanian (anantyog@in.ibm.com)
Sudeesh John (sudeeshjohn@in.ibm.com)
Soumyojyoti Maitra (soumyojyotimaitra@in.ibm.com)

IBM Systems Group

May 2015

© Copyright IBM Corporation, 2015
# Table of contents

Abstract ..................................................................................................................................... 1  
Executive summary .................................................................................................................. 1  
Introduction .............................................................................................................................. 1  
Solution architecture ................................................................................................................ 2  
Why Hadoop on POWER8 ........................................................................................................ 4  
   (A) Simultaneous multithreading (SMT ) .......................................................................................... 4  
   (B) Large memory and I/O bandwidth .............................................................................................. 4  
   (C) Java on POWER8 tunings for long-running Hadoop jobs ................................................................. 4  
Total cost of ownership – Hadoop on POWER8 ........................................................................ 5  
Hardware configuration ........................................................................................................... 6  
System software ....................................................................................................................... 7  
High-level deployment steps with OpenStack Sahara ........................................................... 7  
High-level deployment steps without Sahara ......................................................................... 7  
Single-click cluster configuration using Sahara .................................................................... 8  
   Disk layout ............................................................................................................................................... 8  
   Compute and controller nodes........................................................................................................... 8  
   Nova availability zone for all the compute nodes .............................................................................. 8  
   Partition the local disk on compute node into smaller chunks ......................................................... 10  
   Configure cinder volume server ..................................................................................................... 11  
   Create cinder volumes .................................................................................................................... 12  
   Sahara and Disk Image Builder configuration ............................................................................... 13  
Manual Hadoop cluster deployment ..................................................................................... 14  
   Operating system configuration .................................................................................................. 15  
   Enable Hadoop environment ....................................................................................................... 17  
   Configuration of Hadoop configuration files ............................................................................... 18  
   Start Hadoop services ................................................................................................................ 21  
Performance benchmark ........................................................................................................ 24  
   Trigger workloads ........................................................................................................................ 24  
   Benchmark results ..................................................................................................................... 25  
Summary ................................................................................................................................. 27  
Acknowledgment .................................................................................................................... 28  
Resources ....................................................................................................................................... 29  
About the authors ..................................................................................................................... 30  
Trademarks and special notices .............................................................................................. 31
Abstract

This paper provides detailed information on the setup and configuration of an Apache Hadoop cluster on scale out IBM Power servers, using OpenStack and Sahara. Users looking for a single-click Hadoop deployment on scale out Power servers can benefit from the information provided in this paper.

Executive summary

IBM® PowerKVM™ provides an open virtualization choice for IBM scale-out Linux® systems based on the IBM POWER8™ technology. This provides an open extendable solution for running virtual machines (VMs) on Linux scale-out servers that enables cloud deployments, scale-out processing, and big data solutions reducing complexity and cost.

To accelerate adoption of Hadoop over OpenStack by providing one-click provisioning of Hadoop cluster and elasticity, Hortonworks, Mirantis and Red Hat partnered together to create the Sahara plug-in for OpenStack. Sahara makes Hadoop cluster management very easy and user friendly. OpenStack Sahara solution was primarily built for x86 platform.

The details provided in this paper can help to use OpenStack and Sahara to achieve the same level of operational flexibility on IBM scale-out Power Systems™ as well.

Additionally, the paper also presents the performance evaluations done for a sample workload.

The following key components are used for the experiments.

- Two IBM Power® System S822L servers as OpenStack compute node.
- One Intel® server acting as OpenStack controller node
- Apache Hadoop

Introduction

IBM Power System S822L is a Linux on Power server that provides the ideal foundation for scale-out data and cloud environment. It provides optimized workload solution for Hadoop, big data, and analytics.

IBM PowerKVM™ provides an open virtualization choice for IBM scale-out Linux systems based on the POWER8 technology, where VMs are managed in the same way as any other KVM hosts, leveraging OpenStack, libvirt, and open Linux tools.

Apache Hadoop is an open source software project that enables the distributed processing of large data sets across clusters of commodity servers. It is designed to scale up from a single server to thousands of systems, each offering local computation and storage. Hadoop’s distributed nature and the proliferation of big data applications in the enterprise have made running Hadoop workloads and provisioning of Hadoop environments for development or test purposes ubiquitous.

Hadoop on PowerKVM provides most of the capabilities of other enterprise platforms with significant performance improvements. You can make best use of the capabilities of the VMs on POWER8 to run as Hadoop nodes by referring to the steps mentioned in this paper.
The most popular Hadoop benchmark, TeraSort, has been run on the Hadoop cluster set up on the PowerKVM guests. With standard Hadoop performance tuning parameters, performance results are captured and depicted for user’s reference.

OpenStack Sahara plug-in provides a way to provision Hadoop clusters using templates in a single click and in an easily repeatable fashion.

**Solution architecture**

Installation, configuration, and running a Hadoop cluster is non-trivial. However, there are solutions available, which makes it very easy to deploy the Hadoop cluster. One such solution is the OpenStack Sahara plug-in. OpenStack Sahara enables one-click deployment of virtual Hadoop cluster. This paper describes using Hadoop cluster deployment on PowerKVM compute nodes using OpenStack Sahara.

The solution stack to deploy a Hadoop cluster on an IBM Power Systems server includes IBM POWER8 hardware, PowerKVM, and open source Hadoop 2.5.2, as major deployment components.

A virtual Hadoop cluster solution with OpenStack using local disks is shown in Figure 1: Hadoop solution architecture using local disks on PowerKVM.

As shown in Figure 1, the PowerKVM compute node performs an additional role of a storage node, serving local disks to the virtual machines.

In a PowerKVM host acting as both a compute and storage node, nova-compute and cinder-volume services need to be enabled. The cinder driver used is BlockDeviceDriver.

In addition to the OpenStack services, you must also ensure that cinder volumes are attached to the virtual machines as mentioned in the following points.

- The default cinder quota might not be sufficient for real usage, and therefore, you need to change it accordingly.
- Till OpenStack Juno release, OpenStack did not have a way to automatically ensure co-existence of instance and cinder volume on the same node. In other words, it might
happen that the cinder volume that is being attached to an instance is from a remote node that is from a node not running the instance. This is depicted in Scenario 1. For Hadoop, you might avoid Scenario 1, where a disk is served to an instance over iSCSI. Although, network bandwidth has improved significantly and in most of the cases, iSCSI should be fine. But for Hadoop workloads, this might get into potential scalability issues as the number of compute and data volume grows. Instead, you can have scenario-2.

- There is a way to achieve this manually, which is described in detail in the “Single-click cluster configuration using Sahara” section.

Figure 2: Local disk cinder volumes attached to VMs hosted on PowerKVM host – scenario 1

Figure 3: Local disk cinder volumes attached to VMs hosted on PowerKVM host – scenario 2
Why Hadoop on POWER8

The Hadoop on PowerKVM solution uses the following POWER8 capabilities.

(A) Simultaneous multithreading (SMT)

Simultaneous multithreading (SMT) allows the concurrent execution of multiple long running MapReduce jobs on the same processor core. IBM POWER8 processors offers four types of SMT: 1-way, 2-way, 4-way, and 8-way. With 8-way SMT, POWER8 enables Hadoop to use this capability by running more number of mappers and reducers. You can use below command to enable SMT8.

```
ppc64_cpu --smt=8
```

(B) Large memory and I/O bandwidth

The large-memory POWER8 node is warranted in cases where the analytics requires large buffers and windows. For example, processing large volume of data and aggregating or deduplicating data over long running periods. Long-running MapReduce jobs require high data throughput thereby using high memory and I/O bandwidth capabilities on POWER8.

(C) Java on POWER8 tunings for long-running Hadoop jobs

You need to apply the following Java™ tunings for Hadoop jobs.

- Long-running jobs fail frequently with OutOfMemory exceptions and garbage collection overhead limits exceed the error
  ```
  mapred.child.java.opts : Java opts for the task tracker child processes.
  ```
  (It defines the maximum Java heap size for Hadoop map and reduce tasks). General practice is that the standard value should be 600m for standard use cases and can go up to 70%-75% of the available memory on node.

- Large pages are best suited for long running applications with large memory requirements
  ```
  -Xlp option is used to select the 16 MB pages for the heap and code cache.
  ```

- Java prefetching is an important strategy in order to reduce memory latency and take full advantage of on-chip caches.
  ```
  -XtlhPrefetch option is specified to enable aggressive
Total cost of ownership – Hadoop on POWER8

Your customers can benefit the following advantages by deploying Hadoop solutions on IBM POWER8 scale-out servers.

Figure 4: Savings against Hadoop solution deployment on POWER8
The sizing on POWER8 is done against 1 PB of RAW data. Capacity solution on POWER8 indicates 12 data nodes and three management nodes.

**Hardware configuration**

IBM POWER8 processor-based system with two Power S822L servers each with the following configuration:

- Two sockets (core 24)
- 1 TB RAM
- 7.2 TB local storage (RAID level 0)
System software

PowerKVM: IBM PowerKVM Hypervisor-2.1.1. Refer ibm.com/systems/power/software/linux/powerkvm/

OpenStack Controller: Any OpenStack controller (for example, as provided by IBM Cloud Manager, devstack, RDO and so on) can be used.

Hadoop: Version 2.5.2

Sahara and Disk Image Builder: Upstream Sahara and Disk Image Builder are used for this exercise. Configuration is provided in the “Sahara and Disk Image Builder configuration” section

Virtual machines: RHEL 7

High-level deployment steps with OpenStack Sahara

This section outlines the high-level deployment steps with Sahara.

1. Set up the OpenStack controller with the Sahara plug-in.
2. Install the PowerKVM 2.1.1 release on two IBM Power S822L servers. Refer ibm.com/redbooks/abstracts/sg248231.html?open for more information.
3. Build the Hadoop PPC64 based images using diskimage-builder.
4. Add the Hadoop images to Glance.
5. Register the image with Sahara.
6. Create the cinder volumes.
7. Create the Sahara cluster templates.
8. Deploy the Hadoop cluster and workload using the templates.

High-level deployment steps without Sahara

This section outlines the high-level deployment steps without Sahara.

1. Set up the OpenStack controller.
2. Install the PowerKVM 2.1.1 release on two IBM Power S822L servers. Refer to ibm.com/redbooks/abstracts/sg248231.html
3. Build Hadoop PPC64 based images using diskimage-builder
4. Add the Hadoop images to Glance
5. Create the cinder volumes
6. Deploy the required instances using the Hadoop image
7. Manually configure the Hadoop cluster.

You can follow any one of the following two ways for deployment:

• Single-click cluster configuration using Sahara
• Manual configuration of the Hadoop cluster
**Single-click cluster configuration using Sahara**

This section describes cluster deployment with a single click using OpenStack Sahara plug-ins. The deployment tasks explained in this section are for single-click cluster configuration using Sahara.

**Disk layout**

There are two disk partitions created on PowerKVM host operating system with Nova directory and a local disk is used to host the OpenStack instance.

- `/dev/sda`: hosts the PowerKVM OS as well nova directory (`/var/lib/nova`)
- `/dev/sdb`: local disk to be provided to OpenStack instances.

![Figure 5: Nova configurations on hosts](image)

**Compute and controller nodes**

The lab setup for the Hadoop solution has been configured with the following names for the nodes:

- **OpenStack controller node** – "icmnode1"
- **OpenStack compute nodes (PowerKVM)** – "icmhost1" and "icmhost2"

**Nova availability zone for all the compute nodes**

You need to perform the following steps for Nova availability zone configurations.

1. Create a host aggregate and an availability zone in the controller node as shown in the following figure.
2. Add the compute node to the host aggregate.

3. Ensure that the availability zones are created or not by running the commands provided in the following figure.
Partition the local disk on compute node into smaller chunks

[root@icmshost1 ~]# gdisk /dev/sdb
Command (? for help): n
Partition number (1-128, default 1): 1
First sector (34-6694453214, default = 2048) or [+]-size(KMTP):
Last sector (2048-6694453214, default = 6694453214) or [+]-size(KMTP): +300G
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8390):
Changed type of partition to 'Linux filesystem'
Command (? for help): n
Partition number (2-128, default 2):
First sector (34-6694453214, default = 629147540) or [+]-size(KMTP):
Last sector (629147540-6694453214, default = 6694453214) or [+]-size(KMTP): +300G
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8390):
Changed type of partition to 'Linux filesystem'

Figure 8: Nova service configuration

Figure 9: Local disk partitioning
Figure 10: Local disk partitioning (continued)

Continue the steps depending on the number of chunks you want to create.

**Configure cinder volume server**

Ensure that the cinder availability zone matches with the nova availability zone for the node.

```
****<etc/cinder/cinder.conf>****
```

Figure 11: Cinder volume creation after local disk partitioning

The storage availability zone and the nova availability zone are the same – ‘icmhost1’.
Create cinder volumes

After the disk partitioning is complete, you need to create the cinder volumes.

```
[root@icmhost ~]# cinder create 100 --display-name icmhost2-vol1 --availability-zone icmHost2
```

![Image of cinder volumes created](image1.png)

**Figure 12: Cinder volumes created**

```
[root@icmhost ~]# cinder list
```

![Image of availability zone status](image2.png)

**Figure 13: Availability zone status**

Attach the cinder volume to the virtual machine.
Apache Hadoop on IBM PowerKVM

Figure 14: Cinder volumes being attached to VMs

Display the VM configuration after attaching the cinder volume.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS-DCF:diskConfig</td>
<td>MANUAL</td>
</tr>
<tr>
<td>OS-EXT-AZ:availability_zone</td>
<td>nova</td>
</tr>
<tr>
<td>OS-EXT-SRV-ATT:hostname</td>
<td></td>
</tr>
<tr>
<td>OS-EXT-SRV-ATT: hypervisor_hostname</td>
<td></td>
</tr>
<tr>
<td>OS-EXT-SRV-ATTR:instance_name</td>
<td>instance-00000015</td>
</tr>
<tr>
<td>OS-EXT-SIS:power_state</td>
<td>0</td>
</tr>
<tr>
<td>OS-EXT-STS:task_state</td>
<td>scheduling</td>
</tr>
<tr>
<td>OS-EXT-STS:vm_state</td>
<td>building</td>
</tr>
<tr>
<td>OS-SRV-USG:launched_at</td>
<td></td>
</tr>
<tr>
<td>OS- SRV-USG:terminated_at</td>
<td></td>
</tr>
<tr>
<td>accessIPv4</td>
<td>zkp6yh4q6cJ</td>
</tr>
<tr>
<td>accessIPv6</td>
<td></td>
</tr>
<tr>
<td>adminPass</td>
<td></td>
</tr>
<tr>
<td>config_drive</td>
<td>m1.small (2)</td>
</tr>
<tr>
<td>created</td>
<td>2014-11-03T04:37:44Z</td>
</tr>
<tr>
<td>flavor</td>
<td></td>
</tr>
<tr>
<td>hostId</td>
<td>e881d65f-92ae-4a51-9b35-1e4e0619b0e6</td>
</tr>
<tr>
<td>image</td>
<td>hadoop-hdfs (2c8bf73f-2f9d-472f-8c0d-8b67a5ffe71)</td>
</tr>
<tr>
<td>key_name</td>
<td></td>
</tr>
<tr>
<td>metadata</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>hadoop1</td>
</tr>
<tr>
<td>os-extended-volumes:volumes_attached</td>
<td>[&quot;id&quot;: &quot;e8df6f74-595b-4c2e-991b-cd2eb8204f85&quot;]</td>
</tr>
<tr>
<td>progress</td>
<td>0</td>
</tr>
<tr>
<td>security_groups</td>
<td>default</td>
</tr>
<tr>
<td>status</td>
<td>BUILD</td>
</tr>
<tr>
<td>tenant_id</td>
<td>ece648e685cc94c689e9be4c6e5b84bc</td>
</tr>
<tr>
<td>updated</td>
<td>2014-11-03T04:37:45Z</td>
</tr>
<tr>
<td>user_id</td>
<td>2571e7c4ee5541e98cf46f7b5c872fce</td>
</tr>
</tbody>
</table>

Figure 15: VM configuration after attaching cinder volume

**Sahara and Disk Image Builder configuration**

Upstream Sahara and Disk Image Builder are used in this experiment. Disk Image Builder that patches for ppc64 support has been accepted in upstream. If user is using an older version of Disk Image Builder, then user has to apply the following patches.

- [https://review.openstack.org/#/c/149045/](https://review.openstack.org/#/c/149045/)
- [https://review.openstack.org/#/c/149165/](https://review.openstack.org/#/c/149165/)
- [https://review.openstack.org/#/c/153404/](https://review.openstack.org/#/c/153404/)

Add IBM Java, Hadoop native libraries location to diskimage-create/diskimage-create.sh. Add IBM Hadoop download location to elements/hadoop/post-install.d/40-setup-hadoop and hive download location to elements/hive/post-install.d/60-hive and run the following command to create the image.
sahara-image-elements/diskimage-create/diskimage-create.sh -p vanilla -v 2.4 -i fedora

After creating the images, add these to OpenStack glance repository, and deploy from Sahara after making the necessary changes to the node-group and cluster template as shown in the following figure.

Figure 16: Sahara Plugin – Defining node group templates

Here are two Nova instances deployed through Hadoop flow.

Figure 17: Nova instances

User can go ahead and run any Hadoop workload on these clusters with a single click.

**Manual Hadoop cluster deployment**

This section provides details of manual configuration of Hadoop cluster on virtual machines, which are up and running on POWER8 processor-based systems. Hadoop cluster brought up using the following steps had three nodes and the nodes of the cluster were residing on two POWER8 processor-based servers (physical machines). Among the three nodes, one will be configured as name node, and two as data nodes. Virtual machines can be created manually or using any other management layers such as OpenStack.
For the evaluation, the test team created three Hadoop nodes with DNS names as below:

- bigdatahdfs01 as NameNode
- bigdatahdfs02 as DataNode 1
- bigdatahdfs03 as DataNode 2

**Operating system configuration**

Run the following steps in all the nodes of the cluster (that is, in this example, bigdatahdfs01, bigdatahdfs02, and bigdatahdfs03) to configure the OS.

- Ensure that SELinux is either disabled or set to permissive mode. You can check the current SELinux status by running the `sestatus` command.

```
[root@bigdatahdfs01 ~]# sestatus
SELinux status: enabled
SELinuxfs mount: /sys/fs/selinux
SELinux root directory: /etc/selinux
Loaded policy name: targeted
Current mode: permissive
Mode from config file: permissive
Policy MLS status: enabled
Policy deny_unknown status: allowed
Max kernel policy version: 29
```

To permanently disable SELinux or to setting it to permissive mode edit /etc/selinux/config and make the following changes.

```
SELINUX=disabled
or
SELINUX=permissive
```

Reboot the node for the changes to take effect.

- Disable IPTables in all the nodes.

```
[hadoop@bigdatahdfs01 ~]# /etc/init.d/iptables stop
Flushing firewall rules: [ OK ]
Setting chains to policy ACCEPT: nat mangle filter [ OK ]
Unloading iptables modules: [ OK ]
```

- Disable Ipv6.

```
Append following to /etc/sysctl.conf
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

- Update /etc/hosts with host names of each node.

```
[root@bigdatahdfs01 ~]# cat /etc/hosts
```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4 ::1 localhost localhost.localdomain localhost6 localhost6.localdomain6 192.168.122.40 bigdatahdfs01 192.168.122.213 bigdatahdfs02 192.168.122.176 bigdatahdfs03

- Append following code to /etc/profile for making the IBM Java as default Java: and source the file

```bash
JAVA_HOME=/usr/lib/jvm/java-1.6.0-ibm-1.6.0.16.2.ppc64/jre
PATH=$PATH:$JAVA_HOME/bin
export JAVA_HOME
export PATH
```

- Create a Hadoop user named `hadoop` and set a password for the user.

```bash
[root@bigdatahdfs01 ~]# useradd -m hadoop
[root@bigdatahdfs01 ~]# passwd hadoop
```

- Mount the local disks in datanode 1 and datanode 2 (do the same for all the data nodes).

```bash
[root@bigdatahdfs02 ~]# mkdir /mnt/disk1
[root@bigdatahdfs02 ~]# mkfs.ext4 /dev/vdb1
```

- Append `/mnt/disk1` in `/etc/fstab` and ensure that you get following output. Also change hadoop user as owner of `/mnt` and all the subdirectories in `/mnt`.

```bash
[root@bigdatahdfs02 ~]# grep /mnt /etc/fstab
/dev/vdb1 /mnt/disk1 ext4 rw 0 0
[root@bigdatahdfs02 ~]# chown hadoop:hadoop /mnt/ -R
```

- Enable and ensure a passwordless login for all the three nodes by running the following steps in all the three nodes as `hadoop` user.

```bash
[hadoop@bigdatahdfs01 ~]# run ssh-keygen -t rsa -P "" (Just press 'Enter' key for all the queries)
[hadoop@bigdatahdfs01 ~]# cat /home/hadoop/.ssh/id_rsa.pub >> /home/hadoop/.ssh/authorized_keys
[hadoop@bigdatahdfs01 ~]# chmod 700 ~/.ssh
[hadoop@bigdatahdfs01 ~]# chmod 600 ~/.ssh/authorized_keys
```

- Verify the password-less login within the node.

```bash
[hadoop@bigdatahdfs01 ~]# ssh localhost
```

*(you should be able to login without entering the password)*

- Configure password less login between each nodes by executing following steps in each node.
From datahdfs01:
[hadoop@bigdatahdfs01 ~]$ ssh-copy-id hadoop@bigdatahdfs03
[hadoop@bigdatahdfs01 ~]$ ssh-copy-id hadoop@bigdatahdfs02
From datahdfs02:
[hadoop@bigdatahdfs02 ~]$ ssh-copy-id hadoop@bigdatahdfs01
[hadoop@bigdatahdfs02 ~]$ ssh-copy-id hadoop@bigdatahdfs03
From datahdfs03:
[hadoop@bigdatahdfs03 ~]$ ssh-copy-id hadoop@bigdatahdfs01
[hadoop@bigdatahdfs03 ~]$ ssh-copy-id hadoop@bigdatahdfs02

- Verify the password-less login between nodes.

[hadoop@bigdatahdfs01 ~]$ ssh hadoop@bigdatahdfs03

Last login: Wed Nov 26 00:04:15 2014 from bigdatahdfs01

You should be able to log in between nodes without using a password.

Enable Hadoop environment

After the OS environment in the virtual machines are ready for Hadoop configuration, you need to run the following steps on all the nodes of Hadoop, unless specifically mentioned not to.

1. Download Hadoop from upstream (http://hadoop.apache.org/releases.html) and extract it in the /opt directory. The configuration of hadoop-2.5.2.tgz is referred in this activity.

[root@bigdatahdfs01 opt]# pwd
/opt
[root@bigdatahdfs01 opt]# tar -zxvf hadoop-2.5.2.tgz
[root@bigdatahdfs01 ~]# chown hadoop:hadoop /opt/hadoop-2.5.2 -R

2. Set up the Hadoop binary and library paths as the hadoop user. All the steps from here on should be run as the hadoop user. Update the hadoop environment variables in .bashrc and ensure that the contents of the .bashrc file is as shown in the following lines of code (snippet of the .bashrc file is shown).

[hadoop@bigdatahdfs01 ~]$ cat .bashrc
# User specific aliases and functions
export HADOOP_HOME=/opt/hadoop-2.5.2
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME
export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop
export PATH=$PATH:/opt/hadoop-2.5.2/bin:/opt/hadoop-2.5.2/sbin

3. At the beginning of the /opt/hadoop-2.5.2/libexec/hadoop-config.sh file, add the following line of code.
export JAVA_HOME=/usr/lib/jvm/java-1.6.0-ibm-1.6.0.16.2.ppc64/jre

Grep and ensure JAVA_HOME variable is set
[hadoop@bigdatahdfs01 libexec]$ grep JAVA_HOME hadoop-config.sh
export JAVA_HOME=/usr/lib/jvm/java-1.6.0-ibm-1.6.0.16.2.ppc64/jre
# Attempt to set JAVA_HOME if it is not set
if [[ -z $JAVA_HOME ]]; then
  export JAVA_HOME=/Library/Java/Home
  echo "Error: JAVA_HOME is not set and could not be found." 1>&2
  JAVA=$JAVA_HOME/bin/java
fi

4. Update the JAVA_HOME variable in /opt/hadoop-2.5.2/etc/hadoop/hadoop-env.sh:

[hadoop@bigdatahdfs01 opt]$ grep JAVA_HOME /opt/hadoop-2.5.2/etc/hadoop/hadoop-env.sh
# The only required environment variable is JAVA_HOME. All others are
# set JAVA_HOME in this file, so that it is correctly defined on
export JAVA_HOME=/usr/lib/jvm/java-1.6.0-ibm-1.6.0.16.2.ppc64/jre
#export JAVA_HOME=${JAVA_HOME}

5. Verify whether the configuration done so far is working fine by verifying the version of the required software.
[hadoop@bigdatahdfs01 opt]$ java -version
java version "1.6.0"
Java(TM) SE Runtime Environment (build pzp6460sr16fp2-20141026_01(SR16
FP2))
IBM J9 VM (build 2.4, JRE 1.6.0 IBM J9 2.4 Linux ppc64-64 jvmxp6460sr16-
20141010_216764 (JIT enabled, AOT enabled)
J9VM - 20141010_216764
JIT - r9_20140523_644691fx2
GC - GA24_Java6_SR16_20141010_1202_B216764)
JCL - 20141005_01

[hadoop@bigdatahdfs01 opt]$ hadoop version
Hadoop 2.5.2
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r
cc72e9b000545b86b75a61f4835eb86d57bfafc0
Compiled by jenkins on 2014-11-14T23:45Z
Compiled with protoc 2.5.0
From source with checksum df7537a4faa4658983d397abf4514320

This command was run using /opt/hadoop-2.5.2/share/hadoop/common/hadoop-common-2.5.2.jar.

Configuration of Hadoop configuration files

Perform the following steps to configure Hadoop configuration files.

1. Append following lines to /opt/hadoop-2.5.2/etc/hadoop/hadoop-env.sh for setting yarn class path.
CLASSPATH=${CLASSPATH}:${HADOOP_YARN_HOME}/${YARN_DIR}/*
CLASSPATH=${CLASSPATH}:${HADOOP_YARN_HOME}/${YARN_LIB_JARS_DIR}/*
CLASSPATH=${CLASSPATH}:${HADOOP_YARN_HOME}/share/hadoop/yarn/*
CLASSPATH=${CLASSPATH}:${HADOOP_YARN_HOME}/share/hadoop/yarn/lib/*
CLASSPATH=${CLASSPATH}:${HADOOP_YARN_HOME}/share/hadoop/mapreduce/*
CLASSPATH=${CLASSPATH}:${HADOOP_YARN_HOME}/etc/hadoop/*

2. Append following lines to both /opt/hadoop-2.5.2/etc/hadoop/hadoop-env.sh and /opt/hadoop-2.5.2/etc/hadoop/yarn-env.sh

```bash
export HADOOP_COMMON_LIB_NATIVE_DIR=${HADOOP_HOME}/lib/native
export HADOOP_OPTS="$HADOOP_OPTS -Djava.net.preferIPv4Stack=true -Djava.library.path=$HADOOP_HOME/lib"
```

3. Add the host and hadoop version information in the /opt/hadoop-2.5.2/etc/hadoop/core-site.xml file.

```xml
[.hadoop@bigdatahdfs03 ~]$ cat /opt/hadoop-2.5.2/etc/hadoop/core-site.xml
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://bigdatahdfs01:9000</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/opt/hadoop-2.5.2/tmp</value>
  </property>
</configuration>
```

4. Configure the hdfs for hadoop by adding required information in /opt/hadoop-2.5.2/etc/hadoop/hdfs-site.xml.

```xml
[.hadoop@bigdatahdfs03 ~]$ cat /opt/hadoop-2.5.2/etc/hadoop/hdfs-site.xml
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file://$(hadoop.tmp.dir)/dfs/name</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/mnt/disk1/data</value>
  </property>
  <property>
    <name>dfs.permissions</name>
    <value>false</value>
  </property>
  <property>
    <name>dfs.datanode.du.reserved</name>
    <value>1024</value>
  </property>
</configuration>
```
<value>1073741824</value>
<final>true</final>
</property>

<property>
  <name>dfs.block.size</name>
  <value>134217728</value>
</property>

</configuration>

The value "<value>file:/mnt/disk1</value> " should point to the location where the local disk is mounted.

5. Create the **mapred-site.xml** file with following content.

```bash
[hadoop@bigdatahdfs03 ~]$ cat /opt/hadoop-2.5.2/etc/hadoop/mapred-site.xml
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>mapred.child.java.opts</name>
    <value>-Xmx600m</value>
  </property>
  <property>
    <name>tasktracker.map.tasks.maximum</name>
    <value>45</value>
  </property>
  <property>
    <name>tasktracker.map.tasks.reduce</name>
    <value>25</value>
  </property>
  <property>
    <name>mapred.reduce.tasks</name>
    <value>2</value>
  </property>
</configuration>
```

6. Create **yarn-site.xml** with the following content.

```bash
[hadoop@bigdatahdfs03 ~]$ cat /opt/hadoop-2.5.2/etc/hadoop/yarn-site.xml
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
</configuration>
```
<value>bigdatahdfs01:8025</value>
</property>
<property>
  <name>yarn.resourcemanager.scheduler.address</name>
  <value>bigdatahdfs01:8030</value>
</property>
<property>
  <name>yarn.resourcemanager.address</name>
  <value>bigdatahdfs01:8040</value>
</property>
</configuration>

### Start Hadoop services

Among the three nodes, configure the nodes such that one node is a name node and the other two are data nodes.

1. Add the data node names in `/opt/hadoop-2.5.2/etc/hadoop/slaves`.

   ```bash
   [hadoop@bigdatahdfs01 ~]$ cat /opt/hadoop-2.5.2/etc/hadoop/slaves
   bigdatahdfs02
   bigdatahdfs03
   ```

2. Format the Hadoop Distributed File System (HDFS) from the name node and run the following command only in the name node.

   ```bash
   [hadoop@bigdatahdfs01 ~]$ hadoop namenode -format
   ```

3. Start the Hadoop services from name node by running `start-dfs.sh` and `start-yarn.sh` in the `/opt/hadoop/bin/` path. Ensure that you see the following output without any errors. If Hadoop services fail to start, check the contents of the log file. The path of the log file will be given in the output. These commands will bring up Hadoop's name nodes and data nodes. Also brings up resource/nodemanager in all the nodes of Hadoop.

   ```bash
   [hadoop@bigdatahdfs01 ~]$ start-dfs.sh
   14/12/10 07:20:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
   Starting namenodes on [bigdatahdfs01]
   bigdatahdfs01: starting namenode, logging to /opt/hadoop-2.5.2/logs/hadoop-hadoop-namenode-bigdatahdfs01.out
   bigdatahdfs03: starting datanode, logging to /opt/hadoop-2.5.2/logs/hadoop-hadoop-datanode-bigdatahdfs03.out
   ```

   ```bash
   bigdatahdfs02: starting datanode, logging to /opt/hadoop-2.5.2/logs/hadoop-hadoop-datanode-bigdatahdfs02.out
   Starting secondary namenodes [0.0.0.0]
   0.0.0.0: starting secondarynamenode, logging to /opt/hadoop-2.5.2/logs/hadoop-hadoop-secondarynamenode-bigdatahdfs01.out
   14/12/10 07:20:37 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
   ```

   ```bash
   [hadoop@bigdatahdfs01 ~]$ start-yarn.sh
   ```
starting yarn daemons
starting resourcemanager, logging to /opt/hadoop-2.5.2/logs/yarn-hadoop-resource-manager-bigdatahdfs01.out
bigdatahdfs03: starting nodemanager, logging to /opt/hadoop-2.5.2/logs/yarn-hadoop-nodemanager-bigdatahdfs03.out
bigdatahdfs02: starting nodemanager, logging to /opt/hadoop-2.5.2/logs/yarn-hadoop-nodemanager-bigdatahdfs02.out

4. Verify whether the required processes are running in the Hadoop nodes.
   - In the name node, verify that the process name node is running. In this example, because bigdatahdfs01 is a name node, you can check it in bigdatahdfs01.

   ```bash
   [hadoop@bigdatahdfs01 ~]$ ps -ef | grep namenode
   hadoop  4579     1  3 07:20 ?        00:00:08 /usr/lib/jvm/java-1.6.0-ibm-1.6.0.16.2.ppc64/jre/bin/java -Dproc_namenode -Xmx1000m
   -Djava.net.preferIPv4Stack=true -Djava.net.preferIPv4Stack=true -... -Dhadoop.security.logger=INFO,RFAS
   org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode
   hadoop  5250  1688  0 07:23 pts/0    00:00:00 grep namenode
   ```

   - In all the data nodes, ensure that the process data node process is running.

   ```bash
   [hadoop@bigdatahdfs02 logs]$ ps -ef | grep Data
   hadoop  1573     1  2 07:20 ?        00:00:05 /usr/lib/jvm/java-1.6.0-ibm-1.6.0.16.2.ppc64/jre/bin/java -... hadoop  1941  1659  0 07:23 pts/0    00:00:00 grep Data
   ```

   - Verify the Hadoop configurations by running the `hdfsadmin` command in the name nodes.

   ```bash
   [hadoop@bigdatahdfs02 bin]$ hdfs dfsadmin -report
   14/12/10 07:26:56 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
   Configured Capacity: 293141925888 (273.01 GB)
   Present Capacity: 279896375296 (260.67 GB)
   DFS Remaining: 68241096704 (63.55 GB)
   DFS Used: 211655278592 (197.12 GB)
   DFS Used%: 75.62%
   Under replicated blocks: 766
   Blocks with corrupt replicas: 0
   Missing blocks: 0
   ------------------------------
   Live datanodes (2):
   Name: 192.168.122.213:50010 (bigdatahdfs02)
   Hostname: bigdatahdfs02
   ```
Decommission Status : Normal
Configured Capacity: 157460312064 (146.65 GB)
DFS Used: 100787580928 (93.87 GB)
Non DFS Used: 715847936 (6.68 GB)
DFS Remaining: 49496883200 (46.10 GB)
DFS Used%: 64.01%
DFS Remaining%: 31.43%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Wed Dec 10 07:26:57 EST 2014

Name: 192.168.122.176:50010 (bigdatahdfs03)
Hostname: bigdatahdfs03
Decommission Status : Normal
Configured Capacity: 135681613824 (126.36 GB)
DFS Used: 110867697664 (103.25 GB)
Non DFS Used: 6069702656 (5.65 GB)
DFS Remaining: 18744213504 (17.46 GB)
DFS Used%: 81.71%
DFS Remaining%: 13.81%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Wed Dec 10 07:26:57 EST 2014

After completing these steps, you will be able to see the attached disks from each node. This means that the Hadoop nodes are now ready to run workloads, as shown in the following figure.
Summary

Security is off.
Safe_mode is off.
182 files and directories, 138 blocks = 420 total filesystem objects.
Heap Memory used 42.01 MB of 55.88 MB heap Memory. Max Heap Memory is 1800 MB.
Non Heap Memory used 31.15 MB of 45.45 MB committed Non Heap Memory. Max Non heap Memory is -1 B.

<table>
<thead>
<tr>
<th>Configured Capacity</th>
<th>588.50 GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFS Used</td>
<td>15.97 GB</td>
</tr>
<tr>
<td>Non DFS Used</td>
<td>28.37 GB</td>
</tr>
<tr>
<td>DFS Remaining</td>
<td>544.24 GB</td>
</tr>
<tr>
<td>DFS Used%</td>
<td>2.71%</td>
</tr>
<tr>
<td>DFS Remaining%</td>
<td>92.47%</td>
</tr>
<tr>
<td>Block Pool Used</td>
<td>15.97 GB</td>
</tr>
<tr>
<td>Block Pool Used%</td>
<td>2.71%</td>
</tr>
<tr>
<td>DataNodes usages% (Min/Median/Max/stdDev):</td>
<td>2.71% / 2.71% / 2.71% / 0.00%</td>
</tr>
<tr>
<td>Live Nodes</td>
<td>2 (Decommissioned: 0)</td>
</tr>
<tr>
<td>Dead Nodes</td>
<td>0 (Decommissioned: 0)</td>
</tr>
<tr>
<td>Decommissioning Nodes</td>
<td>0</td>
</tr>
<tr>
<td>Number of Under-Replicated blocks</td>
<td>349</td>
</tr>
<tr>
<td>Number of Blocks Pending Deletion</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 18: Hadoop services status

Performance benchmark

This section describes the benchmark details for a sample workload on this setup.

Trigger workloads

Run one of most popular Hadoop workload teragen on the Hadoop cluster setup. Workloads jar files of Hadoop are located in /opt/hadoop/share/hadoop/mapreduce. Run the following commands in the name node of the cluster in the same following sequence.

[hadoop@bigdatahdfs01 mapreduce]$ hadoop jar /usr/lib/gphd/hadoop-mapreduce/hadoop-mapreduce-examples.jar teragen 500000000 /teraInput
[hadoop@bigdatahdfs01 mapreduce]$ hadoop jar /usr/lib/gphd/hadoop-mapreduce/hadoop-mapreduce-examples.jar terasort /teraInput /teraOutput
[hadoop@bigdatahdfs01 mapreduce]$ hadoop jar /usr/lib/gphd/hadoop-mapreduce/hadoop-mapreduce-examples.jar teravalidate /teraoutput /terareport

After running the workloads on the Hadoop cluster, through web interface, you should be able to see the workloads that were run.
This section provides the performance measure of throughput obtained from running workload on a Hadoop cluster setup on IBM Power Systems. Terasort for 500 GB of workload took 7000 seconds on this environment with two data nodes and one name node on PowerKVM Hypervisor.
Figure 20: Terrasort benchmark of Hadoop on POWERKVM
Summary

Using the steps given in this paper, you can seamlessly deploy a Hadoop cluster on IBM POWER8 processor-based systems on which PowerKVM hypervisor is configured. Performance of Hadoop cluster was validated with configuration given in this paper on IBM Power S822L. Terasort for 500 GB of workload took 7000 seconds on this environment with two data nodes and one name node on PowerKVM Hypervisor. The processor utilization on Power was generally lesser than 50%. By further tuning the Hadoop configuration, you might get much better throughput.
Acknowledgment

This work is never possible without the guidance and involvement from a larger set of people. This team would like to thank the following people for their guidance and help in making this a success.

- Dipankar Sarma, Distinguished Engineer, Linux Technology Center, IBM Systems Group, for initiating and seeding this work
- Tarun Kalra, Senior Manager, Linux Technology Center, IBM Systems Group for his help in managing this entire work.
- Vaidyanathan Srinivasan, Linux Kernel Architect, Linux Technology Center, IBM Systems Group for his help with review and guidance on performance
- Anbazhagan Mani, Cloud System Software Architect, IBM Systems Group for his help with OpenStack controller and Sahara
- Amey P Gokhale, Advisory Software Engineer, Cloud System Software, IBM Systems Group for his help with OpenStack controller and Cinder
- Walesa Francis, Tester, Linux Technology Center, India for his help with the setup.
- Pradeep Surisetty, Linux Technology Center, India for his help with the setup
Resources
The following references provide constructive information that contained in this paper:

IBM Power Systems
- ibm.com/systems/in/power/?lnk=mhpr

IBM Power 8 Processor
- ibm.com/systems/in/power/hardware/linux.html

OpenStack
- ibm.com/developerworks/servicemanagement/cvm/sce/

Sahara
- http://docs.openstack.org/developer/sahara/userdoc/diskimagebuilder.html

Elastic Hadoop on scale out Power Systems (on Youtube)
- https://www.youtube.com/watch?v=JMprhJAF8FQ
About the authors

Ashish Kumar is a Technology Manager in IBM Systems ISV Enablement Organization. He has more than 14 years of total experience and specializes in Big Data Analytics Solutioning. You can reach Ashish at ashishk4@in.ibm.com.

Pradipta Banerjee leads the cloud and docker work for scale-out Power servers in Linux Technology Center at IBM. He is an open source enthusiast with 15 years of experience in the area of operating systems, distributed computing, virtualization and cloud computing. You can connect with him on twitter @pradipta_kr or through email at bpradipt@in.ibm.com. He maintains a personal blog at www.cloudgeekz.com.

Soumyojyoti Maitra is a SME for Linux on Power solutions and is responsibility for building Linux ecosystem in India / South Asia. He specializes in IT hardware and software consultation and business development. You can reach Soumyojyoti at soumyojyotimaitra@in.ibm.com.

Poornima Nayak is a Test Manager for Linux test projects in Linux Technology Center at IBM. She has more than 17 years of experience in IT industry with around 12 years of experience in developing test automation solutions for Linux and testing Linux on various IBM platforms. You can reach Poornima at Poornima.Nayak@in.ibm.com.

Sudeesh is Test Lead for a Linux test project in Linux Technology Center at IBM. He has more than 10 years of experience in IT industry. He is very experienced in debugging issues in Linux environment and developing tools for testing Linux on IBM platforms. You can reach Sudeesh at Sudeeshjohn@in.ibm.com.

Yogananth Subramanian is a development engineer in Linux Technology Center at IBM focusing on bare metal cloud support for scale-out Power servers. He has around 7 years of experience in IT industry. You can reach Yogananth at anantyog@in.ibm.com.

@IBMSystemsISVs
Trademarks and special notices

© Copyright IBM Corporation 2015.

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized re seller for the full text of the specific Statement of Direction.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multi - programming in the user’s job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.
Photographs shown are of engineering prototypes. Changes may be incorporated in production models. Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.