Solution Reference

PowerAI – Software for Deep Learning

Multiple Node Deployment for TensorFlow

For PowerAI Version 3.4

DRAFT

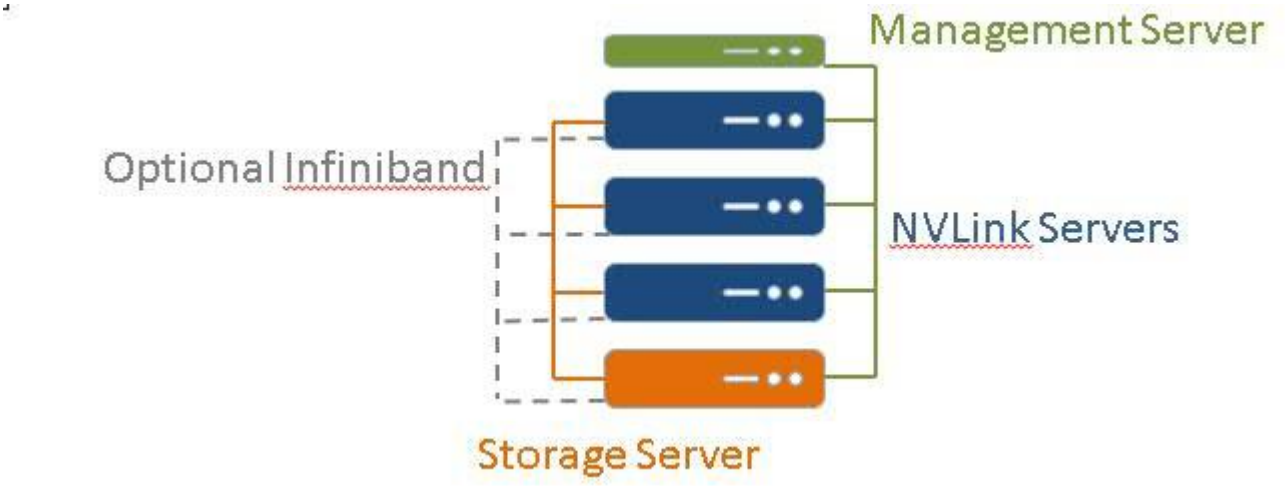
By IBM Systems Group

April, 2017

SOLUTION SUMMARY

This solution consists of this document, a prescribed Bill of Materials (BoM), and a deployment configuration file. The Bill of Materials document provides a description and representation of a PowerAI TensorFlow installation across multiple OpenPOWER servers. It provides information such as model numbers and feature codes to simplify the ordering process and it provides the racking and cabling rules for the preferred layout of the servers, switches and cables.

A high-level diagram representation of a minimal deployment of the solution is below:



The cluster consists of NLink capable Power8 servers with an additional server for shared storage and an additional server for cluster management. In the design, there are dedicated networks for storage (orange) and management (green). The solution includes an option for an additional high speed, low latency InfiniBand network for tensor communication.

The Management server is used to manage the configuration and, optionally, automatically deploy the entire cluster. This deployment automation is accomplished using a tool called **Cluster Genesis** which uses the configuration file as input to install the nodes in the cluster and subsequently install and configure the needed software for the solution.

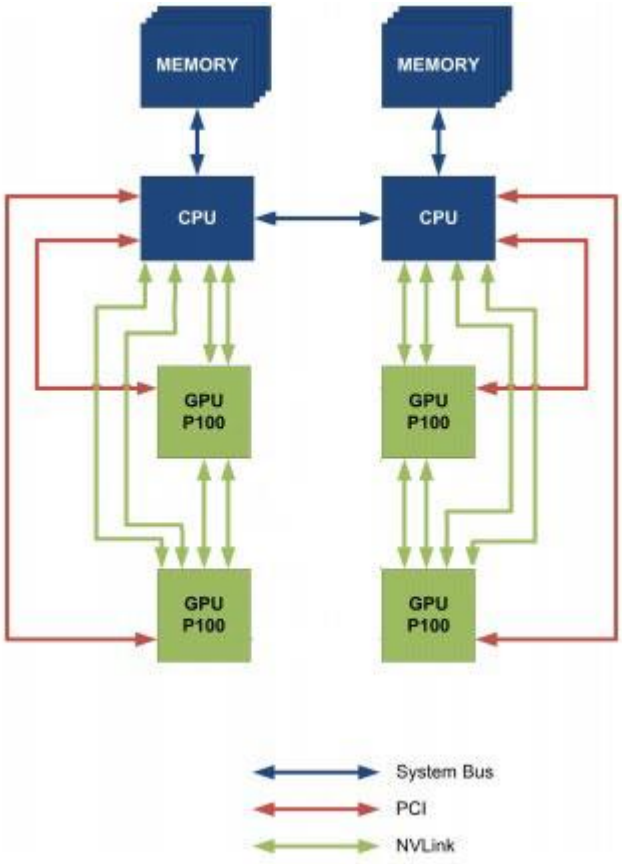
The included deployment configuration file provides a mapping of servers and switches to software for deployment. Software is mapped by assigning each server a software *role*. A TensorFlow training environment requires an entity to perform a parameter serving role for updating and sharing parameter data. In addition to this **Parameter Server** role, there are also the **Storage Server**, and **Training Compute** roles.

In this small-scale cluster, the Parameter Server role and the Storage Server role are assigned to the same physical server, the Storage Server. The NVLink servers are given the training compute role.

In the case that the automated tooling mentioned above is not or cannot be used, this document also includes a description for manual system installation, configuration and software installation.

Compute Node Architecture

The Power8 NVLink servers have a unique architecture that includes the high bandwidth NVLink bus between each Power8 CPU and the two GPUs each is attached to. NVLink is NVIDIA's high speed interconnect technology for GPU-accelerated computing. The NVLink bandwidth far exceeds what is capable with PCIe Gen3. Latency and code paths lengths are reduced as well. The high-level architecture design of the compute node servers is described in the diagram below.



The initialization of the GPUs is done through the PCIe interfaces shown above. The PCIe interfaces also contain side band communication for status, power management, and so on. Once the GPU is up and running, however, all data communications is done using the NVLink bus.

SOLUTION PREPARATION STEPS

1) Acquire the hardware, choose options
2) Rack and cable the hardware
3) Prepare the management node

Independent of the decision to use automation to install and configure the cluster nodes, these preparation steps need to be completed manually before proceeding.

1) PREP: ACQUIRE THE HARDWARE

Go here for the Bill of Materials list of required parts. If you do not already have the needed parts, go here to contact and IBM representative to help you.

At a high level, the system requirements are in the table below.

Power8 19-inch "Minsky", " Stratton", and "Briggs" Based Systems			
Config	**Management (x1)**	**Compute Worker (x6)**	**Compute Parameter/Storage (x1)**
Server	2s1u OpenPower P8 S812LC: "Stratton"	2s2u OpenPower P8 S822LC for HPC: "Minsky"	1s2u OpenPower P8 S822LC for Big Data: "Briggs"
CPU	2 x Power8 8 cores / 64 Threads @2.328 GHz	2 x Power8 8 cores / 64 Threads @3.259 GHz	1 x Power8 8 cores / 64 Threads @3.32 GHz
Memory	128GB DDR4 RAM(16 x 8GB)	256GB DDR4 RAM(16 x 16GB)	128GB DDR4 RAM(16 x 8GB)
Internal Drives	1 x 128GB Disk on Module SATA Drive 1 x 8TB SFF SATA Drive	2 x 1TB SSD SATA Drives *1 x 3.2TB PCIe NVMe	6 x 8TB SFF 7.2K SATA Drives
GPU		4 x Nvidia Tesla P100	
IMPI	IPMI 2.0	IPMI 2.0	IPMI 2.0
Network	1x Broadcom 10gE/1gE (2 port each)	1x Broadcom 10gE/1gE (2 port each)	1x Broadcom 10gE/1gE (2 port each) *1x MLNX CX4 dual port EDR IB

		*1x MLNX CX4 dual port EDR IB	
Power	Redundant	Redundant	Redundant
Warranty	3YR 9x5 Parts & Labor	3YR 9x5 Parts & Labor	3YR 9x5 Parts & Labor

*optional components

OPTIONAL SOLUTION COMPONENTS

NVMe local storage

Cache training data locally to the nodes and fast access storage

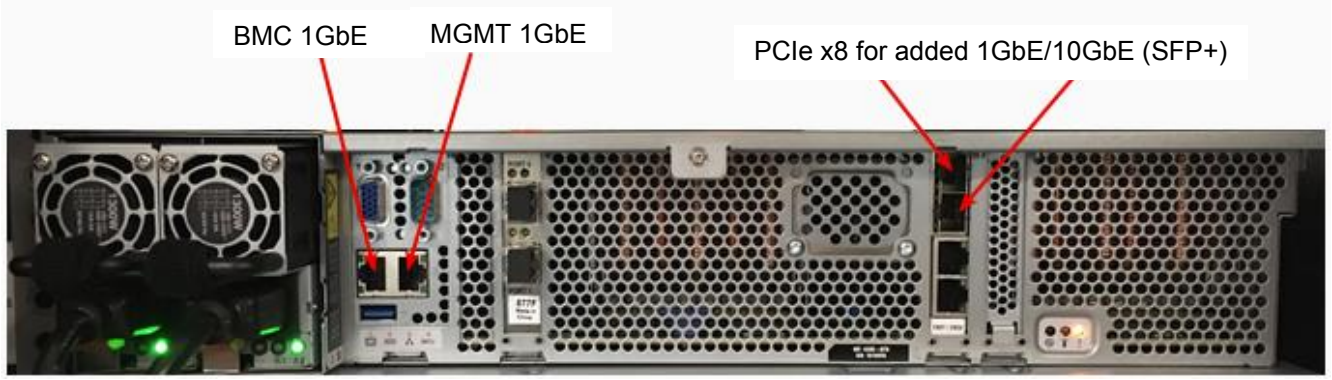
InfiniBand

Add an InfiniBand network to increase bandwidth and lower latency when using IPoIB. This option has an added benefit of separating the storage traffic from the compute traffic.

2) PREP: RACK AND CABLE THE CLUSTER

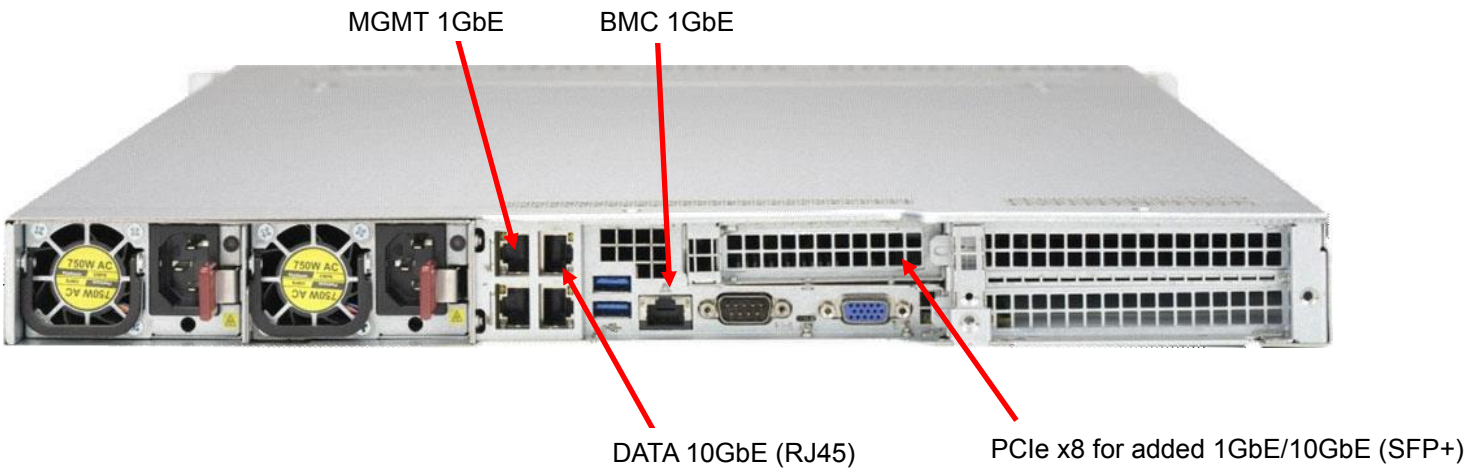
Compute Node

The illustration below shows the back of the compute nodes (“S822LC for HPC”) and its base networking. Note that while these servers can share the BMC service port (a multi-function port), the automation requires the port to be set up for BMC data only.



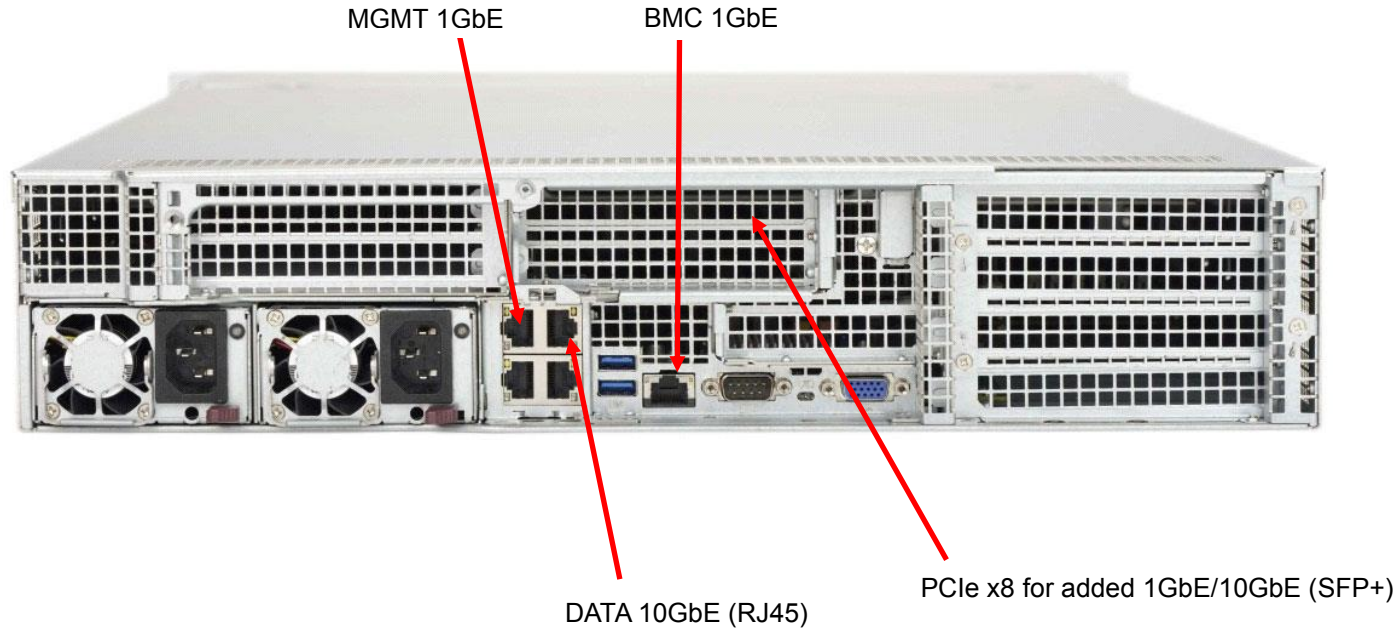
Management Node

The management node (S812LC) rear view is shown below. The four native Ethernet ports support 1GbE, 10GbE, or 100MbE and can be configured for separate speeds. They are RJ45 style ports. The solution also includes a 10GbE card in the PCI3 Slot3. This card has SFP+ connectors. Use the appropriate port based on the switch being used. The 10GbE connection from the management node is optional and can be used to transfer data into the cluster faster, if desired.



Storage Node

The storage node (S822LC) rear view is shown below. The four native Ethernet ports support 1GbE, 10GbE, or 100MbE and can be configured for separate speeds. They are RJ45 style ports. The solution also includes a 10GbE card in the PCI3 Slot3. This card has SFP+ connectors. Use the appropriate port based on the switch being used.



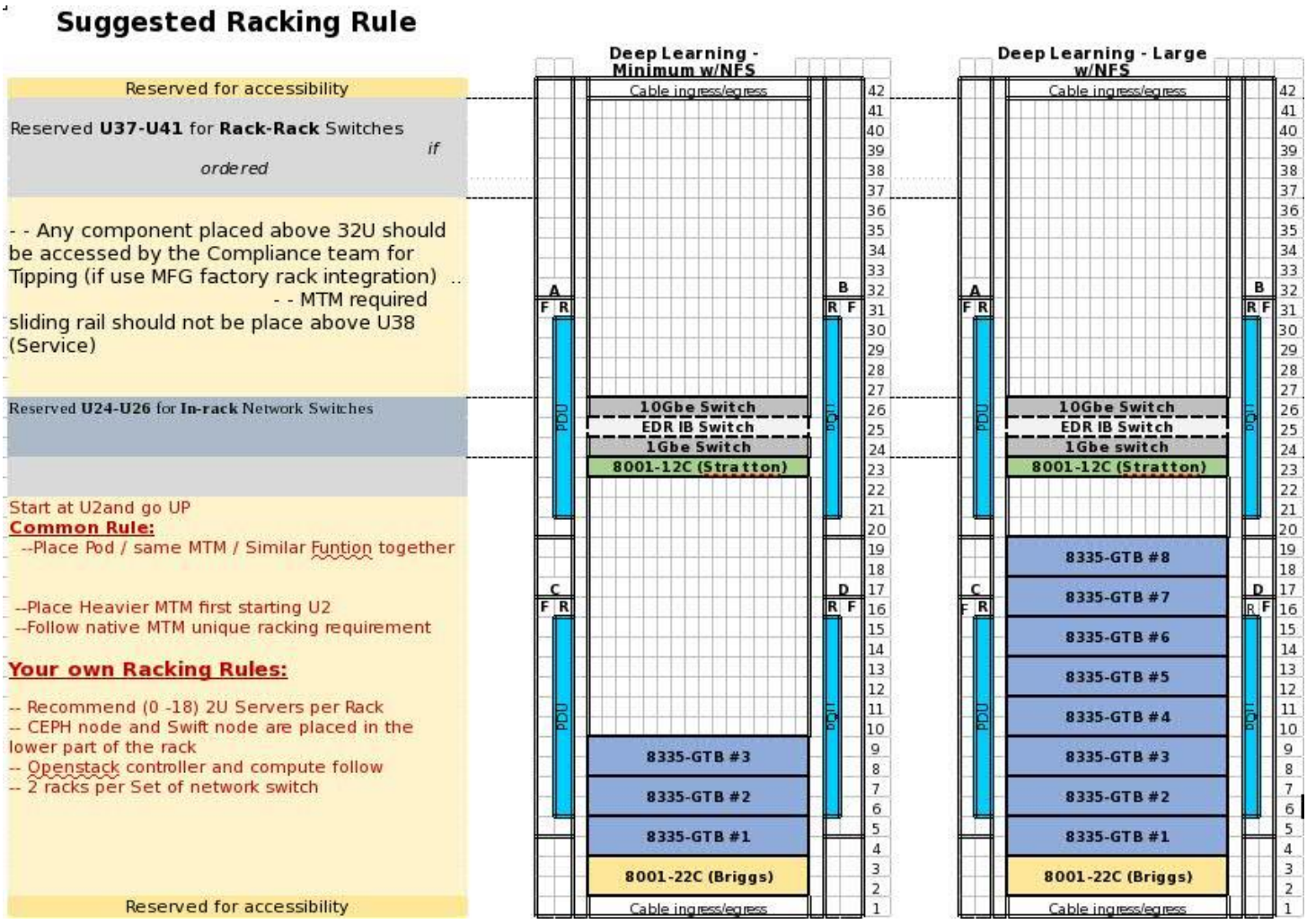
RACKING THE COMPONENTS

This section specifies racking rules that specify both where to place the servers and switches

And where to connect the cables.

These suggested racking rules focus on enabling:

- Modularity per rack
- Consistency
- Expandability
- Ease of servicing, repurposing, shipping, and cooling



Place the intra-rack (leaf) network switches in **U24-U26** as follows:

- 10G storage network switch in **U26**

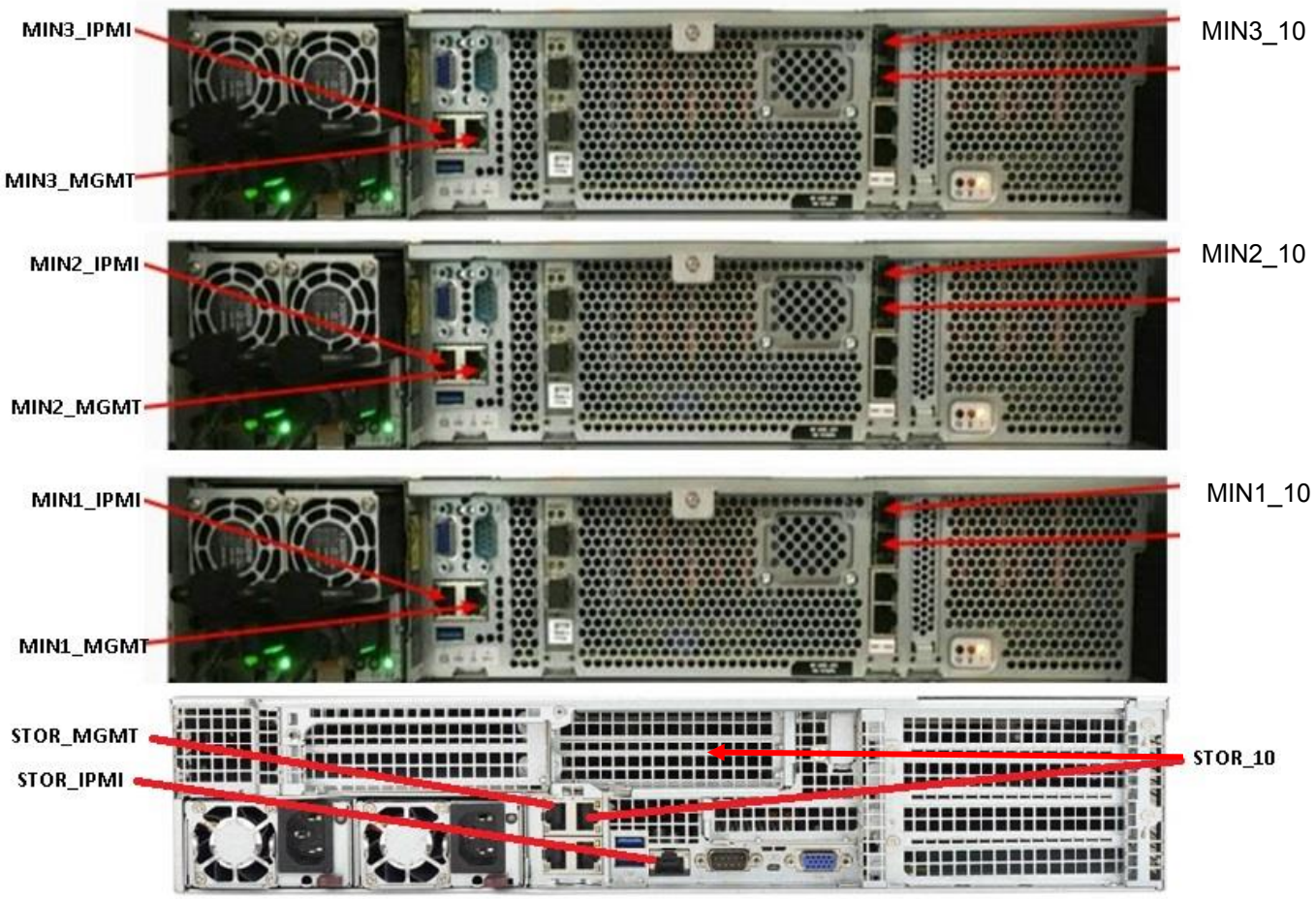
- EDR IB tensor network switch in **U25**
- 1G management network switch in **U24**

Place inter-rack (spine) switches in slots **U37-U41**

If more than 4 PDUs are needed, place 3 horizontal PDUs in 40U and 41U. Spine switches take priority over additional PDUs. If 40U and 41U are occupied by spine switches, place horizontal PDUs in next available slots.

CABLE TOGETHER ALL COMPONENTS

Rear server view with labels for three compute nodes names MIN1, MIN2, and MIN3 along with the storage node.

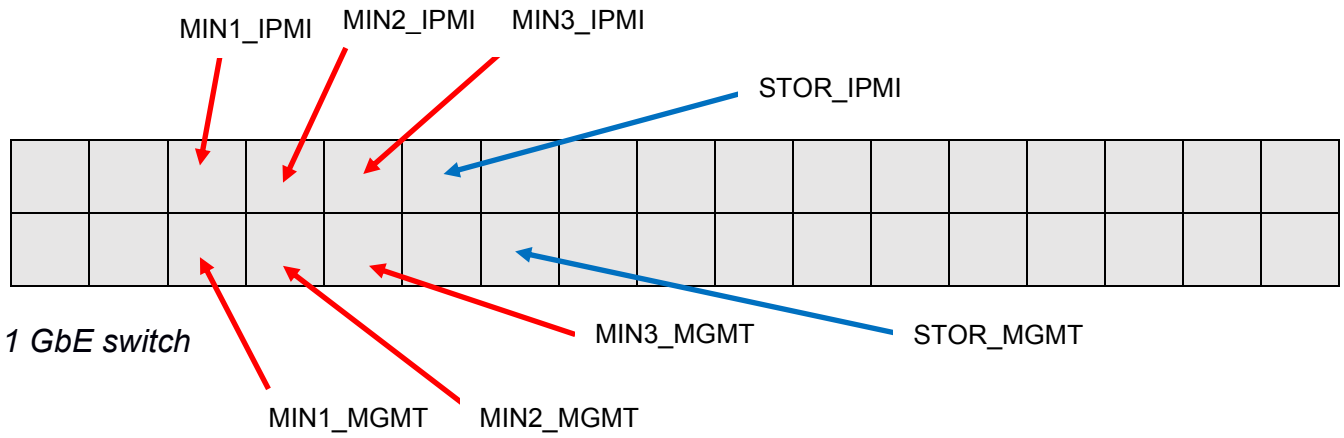


Network Switches

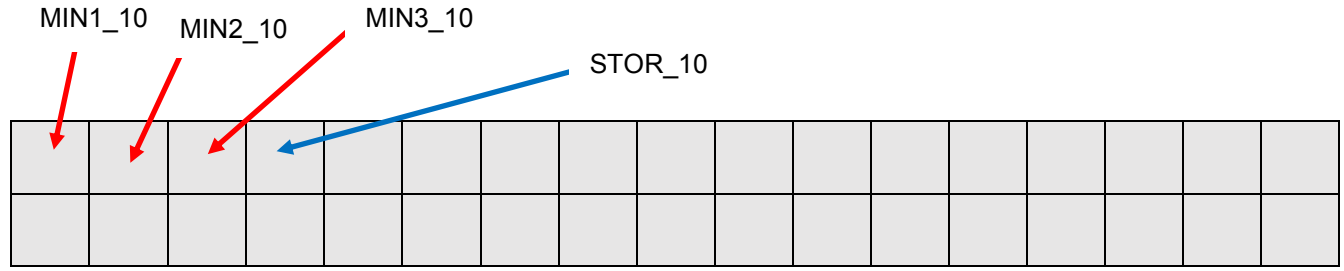
The network switch view is below. The 1GbE switch connects to both the IPMI and management ports on each server

The 10GbE switch creates the storage networks and connects to a 10GbE port on each

server.



1 GbE switch



10 GbE switch

3) PREPARE THE MANGEMENT NODE

The management server is the multi-homed head node of the cluster. It has access to any external networks as well as the internal cluster networks. Additionally, when using Cluster Genesis for automated deployment, the management node will automatically check out the latest software and deployment tools to install and populate the cluster. We've defined the management node as a 1U OpenPOWER server, but it can also be an x86 server with at least 32GB RAM.

Ubuntu 16.04 is needed on the management server. You can obtain it from the following

locations:

- Power8 servers: **https://www.ubuntu.com/download/server/power8**
- X86 servers: **https://releases.ubuntu.com/16.04.1/**

Check the "**Manual Cluster Deployment**" section for instructions for manually installing Ubuntu to OpenPOWER servers using the Ubuntu netboot resources., if needed.

AUTOMATED CLUSTER DEPLOYMENT

Introduction to Cluster Genesis

This guide includes Cluster Genesis configuration and deployment information which can be used to run an automated set up of the reference architecture described in this document. Cluster Genesis is an open source automation tool and simplifies configuration of clusters of bare metal OpenPOWER and x86 servers running Linux. It leverages widely used open source tools such as Cobbler, Ansible and Python. Genesis can configure simple flat networks for typical HPC environments or more advanced networks with VLANs and bridges for OpenStack environments. Genesis configures the switches in the cluster, and uses the port number and MAC address combination to identify each server in the cluster.

Deployment is done from the management server. Deployment installation and communication happen over the 1Gbe management network included in the cluster.

SOLUTION PREPARATION STEPS

This reference solution, including the documentation, Genesis configuration file and associated post-Genesis Ansible playbooks, is stored on `github.com` in the `open-power-ref-design` project. Some aspects of the solution may be updated over time.

1	Checkout the solution from Github
2	Choose your configuration parameters
3	Deploy the cluster (Genesis)
4	Complete any post Genesis configurations

1) CHECK OUT THE SOLUTION FROM GITHUB

Use the git clone command to check out the solution to the management server.

```
$ git clone https://www.github.com/open-power-ref-design/deep-learning
```

2) CHOOSE CONFIGURATION PARAMETERS

To facilitate the automation process of the solution, the parameters in the following table need to be collected. This data will be edited into the configuration file which will be used to automatically configure and deploy the entire solution.

Parameter	Description	Example
Domain Name		ibm.com
Upstream DNS Servers	While a DNS server is configured within the cluster, upstream DNS servers need to be defined for names that cannot otherwise be resolved.	4.4.4.4, 8.8.8.8 as default public upstream DNS servers
Management Node Hostname	What do you want to call your management node?	mgmtnode
Management network IP address	Management for the cluster happens on its own internal network.	192.168.3.3
Data network IP address	Labeled *interconnect* in config.yml	10.0.0.1/24
Management switch IP address	Needed for switch access. Labeled *ipaddr-mgmt-switch* in config.tml in example below	192.168.3.5
Data switch IP address	Needed for switch access. Labeled *ipaddr-data-switch* in config.tml in example below	10.0.3.178
Default login data	Both IDs and passwords	BMC network, OS mgmt network
Data node hostnames and IP addresses	Each node in the cluster needs a hostname and IP address for each of the management and data networks.	

3) DEPLOY THE CLUSTER (GENESIS)

This section covers the power on, initialization, configuration and installation of the cluster. This deployment kit provides an automated method to quickly and more predictably go from physical rack and stack to an operational state.

	GENESIS AUTOMATICALL INITIALIZES AND CONFIGURES THE HARDWARE BY…
A	Reading the configuration files with edited environment-specific changes
B	Driving the BMCs to populate the IP addresses to the nodes
C	Detecting and populating relevant config data to the management node
D	Deploying needed operating system images to the server nodes
E	Configuring the network switches (some manual steps are required)
F	Installing required software once the nodes in the cluster are up

OBTAIN THE DEFAULT CONFIGURATION FILE

The genesis automation uses a configuration file written in YAML to specify the target cluster configuration. The deployment uses this YAML text file to specify the IP address locations of the managed switches and the system nodes attached to the switches as well as other useful details for the deployment process.

The generic master copy of the configuration file, which assumes a 4-node cluster (3 compute, 1 storage/parameter) is located here:

```
https://www.github.com/open-power-ref-design/deep-
learning/config.yml.tfdist.4min
```

TAILOR THE CONFIGURATION FILE FOR YOUR ENVIRONMENT

The configuration file contains a lot of information. To enable a cluster tailored to a specific environment, edit the configuration file with the parameters that were collected in step 2.

Refer to the illustration below and replace the **RED** text with your data.

```
~~~~~~licensing comments and YAML ~~~~~~

ipaddr-mgmt-network: 192.168.3.0/24
ipaddr-mgmt-switch:
   rack1: 192.168.3.5
ipaddr-data-switch:
   rack1: 10.0.3.178

~~~~~~ more YAML and comments ~~~~~~

nfs:
   nfs_network: interconnect
   nfs_server: parameter
   nfs_clients: worker
   nfs_shares:
      - /data
      - /home

~~~~~~ more YAML and comments ~~~~~~


networks:
   interconnect:
      description: Private 10G Storage Network
      addr: 10.0.0.0/24
      broadcast: 10.0.0.255
      method: static
      eth-port: eth11
   infiniband:
      description: Private IPoIB Network
      addr: 10.0.1.0/24
      broadcast: 10.0.1.255
      method: static
      eth-port: ib0

~~~~~~ more YAML and comments ~~~~~~

node-templates:
   worker:
      hostname: worker
      userid-ipmi: ADMIN
      password-ipmi: admin
      powerai-frameworks:
      - tensorflow
```

ADDITIONAL NON-AUTOMATABLE STEPS

There are a few steps that are not automatable today. These include software dependencies that are current behind a required login and/or manual click-through license acceptance. Complete the steps below before running the cluster automation to manually download the required software dependencies.

NVIDIA CuDNN

Download NVIDIA's latest CuDNN packages from the NVIDIA website.

```
https://developer.nvidia.com/cudnn
```

Login or register for NVIDIA's Accelerated Computing Developer Program. Download the following deb files:

- cuDNN v5.1 Runtime Library for Ubuntu16.04 Power8 (Deb)
- cuDNN v5.1 Developer Library for Ubuntu16.04 Power8 (Deb)

Copy the .deb file to the management server and export the location as the CUDNN5 and CUDNN5_DEV environment variables. For example:

```
export CUDNN5=/tmp/libcudnn5_5.1.5-1+cuda8.0_ppc64el.deb
export CUDNN5_DEV=/tmp/libcudnn5-dev_5.1.5-1+cuda8.0+ppc64el.deb
```

Mellanox OFED

If using the **optional** InfiniBand network, download Mellanox OFED from the Mellanox website. Visit the link below to download the correct package. There is a click-through table at the bottom of the page in the *Downloads* tab. Download the .tgz version of the package.

```
http://www.mellanox.com/page/products_dyn?product_family=26&mtag
=linux_sw_drivers
```

Once downloaded, copy the .tgz file to the management server and export the location as the

MLNX_OFED environment variable. For example:

```
export MLNX_OFED=/tmp/MLNX_OFED_LINUX-4.0-2.0.0.1-ubuntu16.04-
ppc64le.tgz
```

Note: if using the InfiniBand option, it is recommended to enable the InfiniBand session manager on the IB switch. InfiniBand switch management is not currently included in Genesis.

PERFORM THE DEPLOYMENT

Run the install script

```
$ install.sh
```

The install.sh script checks out Cluster Genesis from its own github repository, applies patches, and downloads the various dependent packages needed for the install. These dependencies include cuda, PowerAI, and a few specific Ubuntu packages needed during the automated deployment.

Once the install.sh is run cleanly, start the automated Genesis deployment:

```
$ deploy.sh myconfig.yml
```

THE INVENTORY FILE

During the Genesis deploy, an entire inventory of the cluster is taken and written into a separate YAML file. This file is the de-facto database for the cluster. It consists of the network switches and server nodes. The data structure for the switches indicates the type of switch (management, data), their IP addresses and associated login credentials. Similarly, the server nodes entries also contain access information, descriptive information, and general details.

After Genesis is complete, this inventory file is located on the deployment node in `/var/oprc/inventory.yml`

4) POST GENESIS CONFIGURATIONS

TensorFlow run scripts

The Genesis automation will fully install the nodes with an operating system and needed software, including TensorFlow. It also will place a run script on each node at:

```
/opt/DL/tensorflow/samples/dist_driver.sh
```

The script is personalized for each node and can be used to run distributed models on the cluster. Note that not all TensorFlow models implement distributed training, they have to be explicitly written and tuned to take advantage of the distributed TensorFlow functions.

Shared Storage

A shared NFS shared storage disk is created by Genesis and the TensorFlow models library

is checked out there. The default mounted NFS share location for these models is:

```
/data/models
```

There is a distributed version of Inception V3 included in the model library. ImageNet is a common academic data set to use with the Inception V3 model and can be downloaded from:

```
http://www.image-net.org
```

MANUAL CLUSTER DEPLOYMENT

This section describes manual deployment of the reference cluster described above. Each step below needs to be completed for each node in the cluster.

ALL NODES: BMC CONFIGURATION

The BMC network needs to be cabled and attached to a network that can be used for system service. Follow one of the two options below to get the BMC active on the network.

Option 1) The BMC is set to DHCP by default, if a DHCP server with dynamic allocations is on the network, it will have received an address. If this is sufficient, no other work is needed

Option 2) Attach a VGA console and follow a procedure to set a static BMC network address.

- Power on the server with the Power button on the front of the system.
- Watch the VGA screen until the Petitboot boot loader screen appears. Arrow down to 'exit to shell' to drop to the BusyBox shell prompt.
- Set the network address of the BMC using `ipmitool`:

```
$ ipmitool lan print 1
$ ipmitool lan set 1 ipsrc static
$ ipmitool lan set 1 netmask 255.255.255.0
$ ipmitool lan set 1 defgw ipaddr x.x.x.1
```

  Type **`exit`** to get back the petitboot menu

- Configure the network address of the server firmware (petitboot)
  - Select “Configuration”
  - Select static IP and input a network address for the appropriate online network port (external network)

Once the BMC has an active network, it is recommended to use the IPMI Serial over LAN (SOL) console. The VGA can be unplugged and reserved for service scenarios where network connectivity to the service port is unavailable. Use the `ipmitool` command on your local laptop or service workstation to attach to the SOL console.

```
$ ipmitool -I lanplus -H [BMC_address] -U ADMIN sol activate
```

The default user and password for the BMC are:

Username	ADMIN
Password	Admin

ALL NODES: OPERATING SYSTEM INSTALLATION

This solution requires Ubuntu 16.04 Xenial as the Operating System on all nodes. There are many methods for operating system installation, however OpenPower servers can install Ubuntu directly from Canonical's online netboot repositories. This method is quick and efficient and documented below.

Ubuntu 16.04 network installation method

For this method, the host network needs to be cabled to an active network with a route (either direct of via proxy) to the Ubuntu online repositories.

Petitboot is the firmware resident bootloader on OpenPower Systems. It can boot from a large amount of network and internet targets. We'll use the published Ubuntu netboot images for Ubuntu 16.04 Xenial located here:

```
http://cdimage.ubuntu.com/netboot/16.04/
```

At the Petitboot menu, navigate down to “Configure System”. We'll need to enable Petitboot to access the network. Select the interface with an active up link and fill out the network information. Note this can be the same information that the OS itself will use once it's up. Proxy information can be added here as well if required. Navigate down to 'OK' to save the configuration.

Press 'n' to create a new boot target. The “kernel” and “initrd” fields are required. Install the “Xenial” version of Ubuntu with the **version 4.4 kernel “Xenial GA Kernel”**. You can navigate from the link above to get link for the kernel and initrd. Link copied below:

```
Kernel  http://ports.ubuntu.com/ubuntu-ports/dists/xenial-updates/main/installer-ppc64el/current/images/netboot/ubuntu-installer/ppc64el/vmlinux

Initrd  http://ports.ubuntu.com/ubuntu-ports/dists/xenial-updates/main/installer-ppc64el/current/images/netboot/ubuntu-installer/ppc64el/initrd.gz
```

Once created, you should see “`User Item 1`” in the boot list. Select that to boot into the Ubuntu installer.

Walk through the Ubuntu installer. The defaults are sufficient. Towards the end you will be able to select some additional software. Select the following:

```
Basic Ubuntu Server
OpenSSH Server
```

Configure Optional NVMe Storage

If using the optional NVMe drives, configure them for use. Format drive. Mount, add to
`/etc/fstab`

Configure the management network

The Ubuntu installer should have configured the network out of the box.

Configure the storage/tensor network

A high speed 10Gbe network is included in the design. This network will be used for tensor communications unless the InfiniBand option is being implemented. Each compute node needs an address on this network.

Optional: Configure tensor network over Infiniband (IPoIB)

If using the **optional** infiniband network, download Mellanox OFED from the Mellanox website. Visit the location below to download the correct package.

```
http://www.mellanox.com/page/products_dyn?product_family=26&mtag
=linux_sw_drivers
```

Mount the .iso file:

```
$ sudo mount -o loop MLNX_OFED_LINUX-3.4-2.0.0.0-ubuntu16.04-
ppc64le.iso /mnt
```

Install Mellanox OFED:

```
$ /mnt/mlnxofedinstall
```

After the install, the system will then have an interface called **ib0**. Setting an IP address on this interface is done the same way as any other interface. Add an new interface entry to the `/etc/network/interfaces` file:

```
auto ib0
Iface ib0 inet static
   address 10.0.3.102
   netmask 255.255.255.0
   network 10.0.3.0
```

STORAGE NODE: NFS CONFIGURATION

The stroage server will be provisioned to run NFS for a shared data store.

```
$ sudo apt-get update
$ sudo apt-get install nfs-kernel-server
```

Create a mount point shared directory

```
$ sudo mkdir /data
$ sudo chown nobody:nogroup /data
```

Edit `/etc/exports` to add the data directory

```
/data 10.0.0.*(rw, sync,no_root_squash,no_subtree_check)
```

Restart the NFS services

```
$ sudo systemctl restart nfs-kernel-server
```

Configure the firewall to allow NFS

```
$ sudo ufw allow from 10.0.0.0 to any port nfs
```

COMPUTE NODES: SOFTWARE AND DEPENDENCIES

All the cluster nodes (including the compute worker and parameter server nodes) need additional software.

The first step is to install some software components using apt.

```
$ sudo apt-get update
$ sudo apt-get install -y make build-essential dpkg-dev patch
gcc kmod initramfs-tools dkms
```

Ensure the `libc6` package is at least version `2.23-0ubuntu5`. Ubuntu 16.04.2 should come with this version by default. Check using the `dpkg` command:

```
$ sudo dpkg -s libc6
   Package: libc6
   Status: install ok installed
   Priority: required
   Section: libs
   Installed-Size: 24294
   Maintainer: Ubuntu Developers <ubuntu-devel-
   discuss@lists.ubuntu.com>
   Architecture: ppc64el
   Multi-Arch: same
   Source: glibc
   Version: 2.23-0ubuntu5
   Depends: libgcc1
   Suggests: glibc-doc, debconf | debconf-2.0, locales
   Breaks: hurd (<< 1:0.5.git20140203-1), libtirpc1 (<< 0.2.3),
   locales (<< 2.23), locales-all (<< 2.23), nscd (<< 2.23)
   Conffiles:
    /etc/ld.so.conf.d/powerpc64le-linux-gnu.conf
   5c7b2b585e0b8ab8a42d0f7c11b02bb8
   Description: GNU C Library: Shared libraries
    Contains the standard libraries that are used by nearly all
   programs on
    the system. This package includes shared versions of the
   standard C library
    and the standard math library, as well as many others.
   Homepage: http://www.gnu.org/software/libc/libc.html
   Original-Maintainer: GNU Libc Maintainers <debian-
   glibc@lists.debian.org>
```

Install Cuda-8 and cuDNN 5.1

The Deep Learning packages require NVIDIA CUDA 8.0 and cuDNN 5.1.

For Cuda, visit:

```
https://developer.nvidia.com/cuda-downloads
```

Select *Operating System:* Linux

- Select *Architecture:* **ppc64le**
- Select *Distribution* **Ubuntu**
- Select *Version* **16.04**
- For *Installer Type*, select deb(network)

Select the download link to retrieve the package. Install it:

```
$ sudo dpkg -i cuda-repo-ubuntu1604_8.0.61-1_ppc64el.deb
```

Now, install cuda from the repository

```
$ sudo apt-get install cuda
```

This should install **cuda version 8.0.61** or later as recommended.

For CuDNN Visit:

```
https://developer.nvidia.com/cudnn
```

Login or register for NVIDIA's Accelerated Computing Developer Program. Download the following deb files:

- cuDNN v5.1 Runtime Library for Ubuntu16.04 Power8 (Deb)
- cuDNN v5.1 Developer Library for Ubuntu16.04 Power8 (Deb)

Install cuDNN:

```
$ sudo dpkg -i libcudnn5*
```

Install PowerAI

PowerAI 3.4 is delivered as a local repository from which the various frameworks can be installed. Download the latest **`mldl-repo-local.deb`** file from the location below:

```
https://download.boulder.ibm.com/ibmdl/pub/software/server/mldl/
```

Install the repository package:

```
$ sudo dpkg -i mldl-repo-local*.deb
```

Update the package cache

```
$ sudo apt-get update
```

Installing TensorFlow

All the Deep Learning frameworks can be installed at once using the power-mldl meta-package:

```
$ sudo apt-get install tensorflow
```

Tuning recommendations

Recommended settings for optimal Deep Learning performance on the S822LC for High

Performance Computing are:

- Enable Performance Governor

```
$ sudo apt-get install linux-tools-common cpufrequtils lsb-release
$ sudo cpupower -c all frequency-set -g performance
```

- Enable GPU persistence mode

  Use `nvidia-persistenced` (http://docs.nvidia.com/deploy/driver-persistence/index.html) or

```
$ sudo nvidia-smi -pm ENABLED
```

- Set GPU memory and graphics clocks

```
$ sudo nvidia-smi -ac 715,1480
```

- For TensorFlow, set the SMT mode

```
$ sudo ppc64_cpu --smt=2
```

Getting started with TensorFlow in PowerAI

General setup

PowerAI provides a shell script to simplify the TensorFlow environment set up.

We recommend that users update their shell rc file (e.g. .`bashrc`) to source the desired setup scripts. Acitvate TensorFlow on each compute node:

```
$ Source /opt/DL/tensorflow/bin/tensorflow-activate
```

A test script is provided to verify basic function:

```
$ tensorflow-test
```

Getting started with Tensorflow

The TensorFlow homepage (https://www.tensorflow.org/) has a variety of information, including Tutorials, How Tos, and a Getting Started guide.

Additional tutorials and examples are available from the community, for example:

- https://github.com/nlintz/TensorFlow-Tutorials
- https://github.com/aymericdamien/TensorFlow-Examples

API changes and sample models

Note that the TensorFlow API is updated in version 1.0, so programs written for earlier versions of TensorFlow may need to be updated. The TensorFlow v1.0.0 release notes describe the changes and link to a conversion tool. See:

```
https://github.com/tensorflow/tensorflow/releases/tag/v1.0.0
```

The TensorFlow team provides example models on GitHub at

```
https://github.com/tensorflow/models
```

Note that some of the example models may not be updated for the new API.

For the `inception/imagenet_train` example:

- For Tensorflow 1.0.0

  Commit `ef84162c` from fork repo https://github.com/ibmsoe/tensorflow-models (i.e. branch inception-imagenet-1.0) should work.

- For TensorFlow 0.12.0

  Commit `91c7b91f` from upstream repo https://github.com/tensorflow/models should work. The example will print may API warnings as it starts up, but should run to completion.

Additional features

The PowerAI TensorFlow packages include TensorBoard. See:

```
https://www.tensorflow.org/get_started/summaries_and_tensorboard
```

The TensorFlow 1.0 package also includes NVIDIA NCCL support, and experimental suppport for XLA JIT compilation. See:

```
https://www.tensorflow.org/versions/master/experimental/xla/
```

TensorFlow versions

PowerAI 3.4 includes packages for both TensorFlow versions 0.12 and 1.0.1 The versions may behave differently (e.g. in performance or convergence) with different models.

TensorFlow version 1.0 will be installed by default. TensorFlow 0.12 can be installed as

follows:

- If the PowerAI packages are not yet installed, you can specify TensorFlow 0.12 during the initial install:

```
$ sudo apt-get install tensorflow=0.12.0-3ibm1 power-mldl
```

- If TensorFlow 1.0 is already installed, then we recommend to uninstall it first before installing 0.12:

```
$ sudo apt-get purge tensorflow
…
The following packages were automatically installed and are no
longer required:
Bazel caffe-bvlc caffe-ibm caffe-nv chainer digits libopenblas
python-engineio python-flask-socketio python-lmdb python-mako
python-scikit-frm python-socketio-server theano torch
…
The following packages will be REMOVED:
   power-mldl* tensorflow*
…
Do you want to continue? [Y/n]  y
…
Removing power-mldl (3.4.0) …
Removing tensorflow (1.0.1-3ibm1) …

$ sudo apt-get install tensorflow=0.12.0-3ibm1 power-mldl
```

Resources on distributed TensorFlow models

TODO