OPEN VERTEX

ADMINISTRATION AND USER GUIDE

Version: 1.0

10th Nov 2011

1 Table of Contents

1.0 INTRODUCTION

VERTEX is a hybrid HPC platform designed to run user's applications across a scalable and heterogeneous cluster. This cluster could be a mix of both commodity and specialized hardware. The word 'heterogeneous' is significant in the sense that VERTEX allows seamless execution of user's applications on different types of compute nodes e.g. x86, ppc, GPGPUs, etc. From a user's point of view these applications are running on control nodes called 'Vertex' nodes (logical super nodes) whereas the actual execution is occurring on Compute nodes. Each Vertex node is responsible for scheduling jobs and managing IO for a group of compute nodes. The complete VERTEX system consists of distributed Vertex nodes managing light weight Compute nodes on a scalable network fabric. VETTEX supports standard parallel programming models such as MPI, OpenMP, CUDA, etc. ***without*** requiring any modification to user's programs.

1.1 Software Components of VERTEX

1.1.1 Xcpu2

This is a 'Process Management System' (open source) which allows the users an illusion of cluster computation resources (Compute nodes) being available seamlessly as an extension to their workstation (Vertex node). XCPU2 allows programs running on the Compute nodes to use the libraries, tools, configuration files, data, etc. residing on the Vertex node.

1.1.2 Scheduler

The scheduler tracks all the resources (number of processors or cores) on Compute nodes available to all applications running on a Vertex node. The scheduler allocates and de-allocates resources based on applications' resource needs.

1.1.3 Monitor

The job of the Monitor is to continuously track the availability of Compute nodes such that Scheduler is aware of the Compute node's resources available to applications.

1.1.4 Loader

For executing applications seamlessly on VERTEX system, the Loader requests the required resources from the Scheduler and dispatches applications to the Compute nodes accordingly.

2.0 SETTING-UP VERTEX SYSTEM

2.1 Prerequisites

2.1.1 On Vertex nodes

Following is required on Vertex nodes (assuming x86):

1. CentOS 5.6, 64 bit for x86
2. Kernel version 2.6.18-238.12.1.el5
3. If using Infiniband, then install OFED software (refer section "Infiniband interconnect fabric" below).

4. If the vertex system needs to be integrated with Lustre file system, then install lustre client software on Vertex nodes (refer section "Lustre file system" below)

2.1.2 On Compute nodes

(a) x86 64 bit Compute nodes

Following is required on x86 64 bit based Compute nodes:

1. CentOS 5.6, 64 bit for x86
2. Kernel version 2.6.18-238.12.1.el5
3. If using Infiniband, then install OFED software (refer section "Infiniband interconnect fabric" below).

(b) ppc64 Compute nodes (PS3)

Following is required on PS3 Compute nodes (refer section "Installing customized kernel on PS3 compute nodes") :

1. Yellow Dog Linux 6.2 for ppc64
2. Kernel version 2.6.34ydl61.4.9p (9P enabled kernel)

2.1.3 Cloned SYSROOT file system from compute node

NOTE:

Before cloning the SYSROOT file system image from a compute node, any additional software that is required for running user applications should be installed on compute nodes e.g. Intel MPI, Open MPI, libraries required by specific compilers during run-time such as Intel compilers, CUDA software, etc. This is required before the SYSROOT file system from compute node is cloned (refer the process for cloning below).

Cloning Process:

Use the script provided at "/utils/clone_sysroot_image" in the vertex tar ball to clone the SYSROOT file system image from a compute node. Basically this script uses "rsync" to copy over the specified file system from the compute node onto SYSROOT file system image. Note that that cloning is required from only one compute node out of the many compute nodes. This SYSROOT image resides on the vertex node and is shared by all compute nodes attached to a vertex node.

NOTE: To save time in copying as well as disk space on vertex node, exclude all other sub directories under "/lib/modules/" which are not part of the installed kernel. Refer the above script for details.

2.1.4 Installing customized kernel on PS3 Compute nodes

1. Download kernel rpm from

   https://github.com/HPCLinks/Open-Vertex/downloads/kernel-2.6.34ydl61.4.9p1-1.ppc64.rpm

2. Download "initrd-2.6.34ydl61.4.9p1.img" from:

   https://github.com/HPCLinks/Open-Vertex/downloads/initrd-2.6.34ydl61.4.9p1.img

3. Copy the downloaded initrd file to /boot on PS3 by executing the following command:

   ```
   cp initrd-2.6.34ydl61.4.9p1.img /boot
   ```

4. Install the downloaded kernel rpm by executing the following command:

   ```
   rpm -ivh kernel-2.6.34ydl61.4.9p1.ppc64.rpm
   ```

5. Edit and add the following lines in /etc/yaboot.conf:

   ```
   image=/vmlinuz-2.6.34ydl61.4.9p1
   label=2.6.34ydl61.4.9p1
   read-only
   initrd=/initrd-2.6.34ydl61.4.9p1.img
   append="rhgb quiet root=LABEL=/"
   ```

6. Reboot PS3 into new kernel by executing command “reboot”

2.2 Installing Vertex

2.2.1 Installing Vertex software on Vertex nodes

Follow these steps after untar’ing the vertex software tar ball on a vertex node:

- `cd <source_code_subdir_location>`
  `-- e.g. “cd /root/vertex0.8”`
- `./configure --prefix=/opt/vertex --with-uid-method=local  --with-sysrootfile=/hpc/vertex/SYSROOT --with-configfile=/var/vertex`

  NOTE:
  1. The option “—with-configfile” assumes default value of “/var/vertex”. Therefore it is optional unless the user wants to use a different location for this variable.
  2. `“--with-uid-method” accepts one of two values: “yp” for Yellow Pages (i.e. NIS) and “local” for standard Linux user management system`
- `make`
- `make install`
- `If using multiple Vertex nodes but installing Vertex software at a common shared location for all Vertex nodes then execute following steps:`
  - `make prefix-install (to be executed once)`
    -- installs all the binaries at $PREFIX (shared location) e.g. /opt/vertex
  - `make other-install (to be repeated at all Vertex nodes)`
    -- installs /etc/init.d/vertex, /etc/sysconfig/vex, /etc/profile.d/vex.sh, etc.

The configuration files are located at:

1. `/var/vertex/etc/namespace.conf`
   -- add user defined mount points here.
   NOTE: “/home” and “/tmp” are required namespace therefore these should exist in “namespace.conf” file. For example:
   /home
   /tmp

/data
/lustre
/work

2. `/var/vertex/etc/vex.conf`
   -- contains compute node information (refer section "Adding or Deleting a compute node")

These files are NOT overwritten whenever user installs vertex software using the steps as described above. The corresponding sample files are located at $VEXVAR/doc which are installed during vertex software installation.

2.2.2 Installing vertex software on compute nodes

Following steps need to be repeated on each of the compute nodes:

- `Untar the provided vertex software tar ball`
- `cd <source_code_subdir_location>`
  `-- e.g. "cd /root/vertex0.1"`
- `./configure`
- `make`
- `make computenode-install`
  `-- installs two binaries xcpufs and xcpufs.static at /usr/local/vertex/sbin/`
  `-- installs vertex initialization script: /etc/init.d/vertex`
- `/etc/init.d/vertex start`
  `This will start xcpufs.static on compute node`

Additionally when a PS3 is used as compute node, execute the following command to mount /spu partition:

```
mkdir /mnt/sandbox/spu
```

2.3 Infiniband interconnect fabric

To be able to use Infiniband interconnect fabric on Vertex system, the software from OFED needs to be installed and configured. Also the Open MPI that ships with the OFED distribution can be installed and configured. This release of OpenVertex has been tested with OFED version 1.5.3.1 along with the Open MPI version 1.4.3 that is shipped with this version of OFED.

2.4 Lustre parallel file system

Integration of lustre file system with Vertex system requires installation of lustre client software on Vertex nodes. The current release of Open Vertex software has been tested with luster client software version 1.8.6 available from Wham Cloud site (www.whamcloud.com). Please install lustre-client RPMs after building these RPMs using the downloaded lustre client software from the above site.
NOTE: The lustre client software version 1.8.6 has been tested against the kernel version 2.6.18-238.12.1.el5 only.

3.0 USING VERTEX

3.1 Executing user applications on compute nodes

A user binary needs to be invoked as follows:

```
vrun <path_to_user_executable>
```

"vrun" is nothing but a symbolic link to "Vex loader".
Essentially "Vex loader" is a program which finds out the available node based on the resource requirement (currently the resource is set to one general purpose processor as default) and dispatches user binary for execution onto that node via xcpu.

3.2 Running MPI applications

3.2.1 Running OpenMPI applications

(a) *Setting environment variables*
To be able to run OpenMPI based applications on Vertex system, the user needs to define $PATH and $LD_LIBRARY_PATH in his .bash_profile. A typical method for adding these variables is given below:

```
PATH=/usr/mpi/gcc/openmpi-1.4.3/bin:$PATH
if [ -z "$LD_LIBRARY_PATH" ] ; then
   LD_LIBRARY_PATH=/usr/mpi/gcc/openmpi-1.4.3/lib64
else
   LD_LIBRARY_PATH=/usr/mpi/gcc/openmpi-1.4.3/lib64:$LD_LIBRARY_PATH
fi
export PATH
export LD_LIBRARY_PATH
```

(b) *Configuration file for defining MCA parameters*
The default values for MCA parameters have been specified in the parameters file "openmpi-mca-params.conf" which is located at two places on vertex system as given below:

1. `/usr/mpi/gcc/openmpi-1.4.3/etc/openmpi-mca-params.conf`
2. `$VEXSYSROOT/Linux/x86_64/usr/mpi/gcc/openmpi-1.4.3/etc/openmpi-mca-params.conf`

Note that the contents of the "openmpi-mca-params.conf" file should be same at both of these locations i.e. both files are identical.

(c) Executing OpenMPI applications
A few examples given below (assuming these commands are executed on v1). This is based on the assumption that $VEXVAR/etc/vex.conf on v1 contains only node n1 and $VEXVAR/etc/vex.conf on v2 contains only node n2.

- `mpirun –np 2 –H v1,v2 vrun <executable_name>`
  This will invoke two instances of the binary; one will execute on node n1 and second on node n2.
- `mpirun –np 2 vrun <executable_name>`
  This will invoke two instances of the binary and both will execute on node n1.

3.2.2 Running Intel MPI applications

(a) *Setting environment variables*

To be able to run Intel MPI based applications on Vertex system, the user needs to define $PATH and $LD_LIBRARY_PATH in his .bash_profile accordingly.

Additionally the env variable required for using the type of interconnect for message passing between processes has been already added to the vex_loader file. The env variable is:

```
I_MPI_FABRICS=ofa
```

(b) *Starting "mpd" daemon*

Running Intel MPI applications requires that "mpd" daemons should be running on both Vertex nodes v1 and v2. The list of nodes where "mpd" daemons run can be specified in a file called "mpd.hosts". By default if the mpd.hosts file exists in the PWD of the user, then it will be used for the list of nodes for running "mpd" daemon processes. Otherwise the user needs to specify the file location on the command line.

(c) *Executing Intel MPI applications*

A few examples given below (assuming these commands are executed on v1). This is based on the assumption that $VEXVAR/etc/vex.conf on v1 contains only node n1 and $VEXVAR/etc/vex.conf on v2 contains only node n2. Note that mpd.hosts file should list two nodes v1 and v2 such that when command "mpdboot" is executed, it will cause daemon process "mpd.py" to run on each Vertex node (i.e. v1 and v2).

- `mpiexec -rr -n 2 vrun <mpi_executable>`

  In the above command, the option "-rr" denotes round-robin scheduling which should be used otherwise it appears that Intel MPI schedules both processes on the same compute node. The above command will cause one MPI process to run on n1 and the second process on n2.

- `mpiexec -n 2 vrun <mpi_executable>`

  Above command will cause both processes to be invoked at Vertex node v1 which in turn will dispatch both processes to node n1 for execution.

3.3 Running CUDA applications

(a) *Setting environment variables*

First the user needs to define $PATH and $LD_LIBRARY_PATH in his .bash_profile as below:

```
export PATH=/usr/local/cuda/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
```

(b) *Executing CUDA applications*

```
vrun <cuda_executable>
```

3.4 Environment considerations

3.4.1 Userid/Authentication

Only requirement from the perspective of using Vertex system is that the user's home dir should have RSA keys (refer section "Adding or Deleting users"). This will allow all such users to be authenticated during start of "/etc/init.d/vertex" script.

4.0 ADMINISTRATION

4.1 Initializing Vertex

"vertex" init script can be started and stopped by executing following commands as root.

```
> /etc/init.d/vertex start
> /etc/init.d/vertex stop
> /etc/init.d/vertex restart
```

VERTEX init script will start the VERTEX Scheduler, VERTEX monitor, and the xcpufs (server). The log messages for VERTEX Scheduler and Monitor are redirected at /var/log/vexsched and /var/log/vexmon respectively.

4.2 Updating the exported Vertex namespace

Any new file partition aka "namespace" in xcpu terminology can be created as follows:
Let us take the example of creating a new file partition "/data" which needs to be exported to compute nodes. Following steps should be followed:

On vertex node:
1. Add a new line "/data" in $VEXVAR/etc/namespace.conf.
2. Need to restart vertex script as: "/etc/init.d/vertex restart"

This new namespace would have been created by now and ready for use by the applications executing on compute nodes.

4.3 Distribution of RSA keys to the compute nodes

Before user applications can be executed remotely on compute nodes, the ssh key should be generated on vertex node and distributed to each of the compute nodes.

Example:
On vertex node (as root):

```
 > ssh-keygen -t rsa -f admin_key
 > mkdir /etc/xcpu
 > cp admin_key* /etc/xcpu
```

Once the key pair is generated, copy the key pair to each compute node at the location /etc/xcpu/

4.4 Adding or Deleting a compute node

Compute nodes aggregated by a vertex node can be added or deleted by modifying the configuration file located at $VEXVAR/etc/vex.conf.
The notation for denoting a node and corresponding resources is as given below:
<node_name>:<number_of_gpp_cores>:<number_of_gpu_cores>

<u>Field description:</u>
node_name is the "network hostname" of the compute node
number_of_gpp_cores is the number of general purpose (e.g. x86) cores
number_of_gpu_cores is the number of GPU cores (e.g. number of cores in Nvidia processor)

Examples:
node0:8:16
node1:4:8
node2:12:32
OR
n1:16:24
n2:16:32

4.5 Modifying SYSROOT file system

The SYSROOT file system is installed first time after cloning the file system from a compute node installation which is equipped with a hard disk. There are three methods by which an existing SYSROOT file system can be modified:

1. 'cd' into the SYSROOT dir and make changes (for simple tasks like modifying libs, adding new files, etc.)
2. 'chroot' into the SYSROOT dir and perform installation just like it is done in any standard root file system e.g. installing Intel Fortran Compiler.
   NOTE: this option will work only when both vertex node and computes nodes have the same architecture e.g. x86 or ppc. When compute nodes and vertex node have different architectures, follow option#3 below.
3. Modify the original image from which the SYSROOT was 'cloned' and then clone the latest file system onto the SYSROOT file system at vertex node. This is the last option when nothing else is feasible.

4.6 Adding or Deleting users

Use the standard Linux command to create/delete users. If using NIS (aka Yellow Pages) for user management, follow appropriate instructions on creating users in NIS. An example for creating user on Linux is given below.

Example:

Let’s say the new user id that needs to be created is “roy”. Follow steps given below on vertex node:

1. Create user id
```
 > useradd –u 502 –g 501 roy
```
   Here the new user is added to the existing group id ‘501’.
2. Generate RSA key-pair for this user under the default /home/roy/.ssh/
```
 > ssh-keygen –t rsa
```

NOTE: It is important that to be able to use Vertex system by the users, a RSA key pair should exist in default dir location /home/<user>/.ssh/ of all such users. Otherwise the applications executed by these users will not run on compute nodes.

5.0 KNOWN ISSUES

5.1 Expected output of “whoami”

Running the command “vrun whoami” should return the correct user who is running this command. Currently the output from this command is “root”. This needs to be fixed in later releases.

5.2 Expected output of “hostname”

Currently running the command “vrun hostname” returns the hostname of the compute node. Whereas this should return the “hostname” of the vertex node.