Notes on Portable Batch System (PBS)

Amit Jain
Department of Computer Science
Boise State University

Introduction

The Portable Batch System (PBS) is a system that can be used to manage the usage of a cluster. We are using the open source version of PBS (website: `http://www.openpbs.org`). PBS is a very general scheduling system that can be used for many setups other than clusters. Open PBS works well for 32 nodes or less. For more than 32 nodes, there is a commercial version of PBS.

Setup

The PBS system is very configurable. The best way is to install the system is to build it from the source. The system consists of three programs: `pbs_server`, `pbs_sched` and `pbs_mom`.

On a cluster, the master machine will run `pbs_server` and `pbs_sched`. Each node that can run applications will run the `pbs_mom` server. For example, in the lab, onyx runs all three servers, whereas the workstations (ws[01-32]) run the `pbs_mom` server.

On `onyx`, the start script is:
`/etc/rc.d/init.d/pbs`
Also`/etc/rc.d/rc5.d/S99pbs` is a link to since `onyx` is running at runlevel 5. Add another link from `/etc/rc.d/rc3.d` if you want to run PBS from run level 3 as well.

On the workstations, the start scripts are:
`/etc/rc.d/init.d/pbsmom`
Also `/etc/rc.d/rc5.d/S99pbsmom` is a link to since `onyx` is running at runlevel 5. Add another link from `/etc/rc.d/rc3.d` if you want to run PBS from run level 3 as well.

This allows the PBS system to start automatically when the machines boot.

PBS Configuration

The configuration of PBS is complex. Consult the PBS administrator's guide for more details about setting up queues, nodes to manage etc.

The PBS system keeps its information in `/var/spool/pbs` (unless you specified a different directory during the build process). The log files for the various servers are kept under that directory.

The list of machines to manage is kept in `/var/spool/pbs/server_priv/nodes`. Here is the `nodes` file on `onyx`.

```
ws00 master np=4
ws01 node
ws02 node
ws03 node
ws04 node
ws05 node
ws06 node
ws07 node
ws08 node
ws09 node
ws10 node
ws11 node
ws12 node
ws13 node
ws14 node
ws15 node
ws16 node
```

Here the first column is the hostname of the system. Note that listing `ws00` allows us to schedule execution on `onyx` as well. The second column is an arbitrary string, called a property. This allows a user to request certain number of nodes with a certain property. For example, a user can request `"nodes=1:master+16:node"` to request the entire cluster. The entry `np=4` with onyx specifies that there are 4 virtual processors on `onyx` and that allows PBS to start up to four jobs on `onyx`.

For each job that is scheduled, there is a file in `/var/spool/pbs/aux/` with the same name as the job name. For example, if the jobname is `115.onyx.boisestate.edu`, then the file `/var/spool/pbs/aux/115.onyx.boisestate.edu` lists the machines allocated to that job. Suppose we requested "nodes=1:master+4:node", then the allocated nodes might be.

```
[amit@onyx amit]:cat /var/spool/pbs/aux/115.onyx.boisestate.edu
ws00
ws04
ws03
ws02
ws01
[amit@onyx amit]:
```

PBS sets the environment variable `PBS_NODEFILE` to the pathname of the file containing the list of nodes allocated to the job. More about this later.

Running programs under PBS

Getting ready to run PBS jobs

To be able to run PBS jobs, you must check your `.bash_profile` and `.bashrc` files carefully. Any command that manipulates the terminal must be put in a conditional statement of the following form.

```
if test "$PBS_ENVIRONMENT" = "PBS_INTERACTIVE" -o -z "$PBS_ENVIRONMENT"
then
# set up the prompt to the hostname
#
PS1="[\u@\h \W]":
fi
```

The reason is that there is no terminal when you are running in batch mode. If you try to manipulate the terminal in batch mode, your login will fail and your batch job will not run. The environment variable `PBS_ENVIRONMENT` is set by PBS to be either `PBS_INTERACTIVE` or `PBS_BATCH`. When you are not running under PBS, then the variable is unset.

The `pvmrun` command

With PVM you need to create the PVM system with either the pvm console or the xpvm application. Once the PVM system is operational, then you run your program. This is undesirable for many reasons: If you forget to start the PVM system, your applications fails. If you left the PVM system running, then you will connect to what is already running. In order to avoid these problems, we have created a application `pvmrun`. Running `pvmrun` with the help option shows its usage.

```
[amit@onyx amit]:pvmrun -h
pvmrun: $Id: pvmrun.c,v 1.3 2003/11/17 23:48:17 amit Exp amit $
Usage: pvmrun -np <#copies> <executable> {<args>,...}
*****************************************************************
  For a pvm master/slave program: pvmrun -np 1 <executable> {<args>,...}
  For a pvm n-way spmd program: pvmrun -np n <executable> {<args>,...}

  The list of machines to use for PVM is determined as follows:
    if PBS_NODEFILE is set, then use the file specified by PBS
    else: first use .pvm_hosts, if found, in current directory
          then use .pvm_hosts, if found, in the user's home directory
          then use /usr/local/etc/machines.
*****************************************************************
[amit@onyx amit]:
```

The pvmrun program starts the pvm system for you, runs your application (either master/slave style or spmd style) and then cleanly shuts down the PVM system. If the PVM system is already running, it shuts it down and restarts it with the proper list of nodes to use.

You must use pvmrun in your PBS batch jobs.

Running batch jobs

Preparing a PBS batch job script

Any parallel program that takes more than a few minutes should normally be run as a PBS batch job. In order to run it as a PBS batch job, you will need to prepare a PBS batch script (which is just a shell script with some additional features). Here is a sample PBS batch job (∼amit/cs430/lab/pvm3/tools/psort.pbs):

```
#!/bin/sh
#PBS -l nodes=1:master+16:node
# This is a PBS job submission script. It asks for the master node (ws00)
# and 16 workstations in the PBS cluster to run the PVM application on.
#
# IMPORTANT NOTE:  Be sure to modify the "cd" command below to switch
# to the directory in which you are currently working!
#
#------------------------------------------------------------------------

cd /home/faculty/amit/cs430/lab/pvm3/tools
pvmrun -np 16 psort 20000000 16
```

The line starting with #PBS is a PBS directive. There are many PBS directives but the one we will use is mainly the one that lists the nodes that we need to run our program.

Here is another sample PBS batch job. Here the psum program is assumed to spawn processes to the 16 nodes.

```
#!/bin/sh
#PBS -l nodes=1:master+16:node
# This is a PBS job submission script. It runs a master/slave PVM program
# Note that even though we are specifying only one process to pvmrun, we
# need to reserve the appropriate number of nodes to match what psum requires.
#
# IMPORTANT NOTE:  Be sure to modify the "cd" command below to switch
# to the directory in which you are currently working!
#
#------------------------------------------------------------------------
```

```
cd /home/faculty/amit/cs430/lab/pvm3/parsum
pvmrun -np 1 psum 10000 16
```

Submitting a PBS batch job script

The command `qsub` can be used to submit a PBS job. Continuing with the example script `psort.pbs` from the previous subsection, we can submit it for execution as follows.

`cd ~amit/cs430/lab/pvm3/tools/` `qsub psort.pbs`

The status of a job can be checked with the `qstat` command. Using `qstat -n` also shows the nodes that were allocated to your job.

```
[amit@onyx tools]:qsub psort.pbs; qstat
116.onyx.boisestate.edu
Job id           Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - -----
116.onyx         psort.pbs        amit                    0 Q default
[amit@onyx tools]:qstat -n

onyx.boisestate.edu:
                                                            Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
116.onyx.boises amit     default  psort.pbs     --   17  --     -- 00:30 R   --
   ws00/0+ws16/0+ws15/0+ws14/0+ws13/0+ws12/0+ws11/0+ws10/0+ws09/0+ws08/0+ws07/0
   +ws06/0+ws05/0+ws04/0+ws03/0+ws02/0+ws01/0
[amit@onyx tools]:
```

You can delete jobs with the `qdel` command.

The standard output and standard error streams are redirected into the files `psort.pbs.oxxx` and `psort.pbs.exxx`, where `xxx` is the job number assigned by PBS.

In case of an error in running the job after it has been accepted in the queue, PBS sends an email to the user.

Running interactive jobs

To run an interactive PBS job, we use the -I option. In that case, the `qsub` command reads the PBS script to determine the resources requested but the ignores the rest of the PBS script and instead starts a new shell and connects standard in/out/err to that shell. The parameter `np=4` given in the PBS node file limits the number of interactive jobs to four (since interactive jobs run on the current machine). This is fine and not so dandy because now the user can start pvm or xpvm and allocate nodes without knowing what other users are doing.

To solve this dilemma, we have added scripts `pvm` and `xpvm` in `/usr/local/bin` so users run that script rather than the standard `pvm` and `xpvm` programs.

The new script `pvm` uses a command line parameter for the number of processors and creates a PBS script with the right directives on the fly. Then it starts an interactive PBS session using this script. At that point, the user is assured that they have the right number of nodes allocated to them. Then using the environment variable PBS_NODEFILE, the user can create a PVM virtual machine that is not shared with other users.

See the sample session below.

```
[amit@onyx amit]:pvm

####################################################
Invoke PVM using PBS.
Usage:  pvm  [-h] <#nodes>
Uses onyx + 4 nodes by default
####################################################

Using 4 hosts by default

*****************************************************
 Scheduling an interactive pvm session with PBS.
 Please end session by typing in exit.
 For running xpvm, use the following commands.
  export DISPLAY=localhost:16.0
  /usr/bin/xpvm $PBS_NODEFILE
 For running pvm, use the following command.
  /usr/bin/pvm $PBS_NODEFILE
 Please always halt the pvm system before exiting
   either the pvm console or xpvm!!!
 Or use the pvmrun command, which halts pvm properly
 Usage: pvmrun -np <#copies> <executable> {<args>,...}
*****************************************************

qsub: waiting for job 117.onyx.boisestate.edu to start
qsub: job 117.onyx.boisestate.edu ready

[amit@onyx amit]:pvm

.... pvm startup messages deleted ...

pvm> conf
5 hosts, 1 data format
                    HOST     DTID     ARCH   SPEED       DSIG
                    ws00    40000 LINUXI386    1000 0x00408841
                    ws04    80000 LINUXI386    1000 0x00408841
                    ws03    c0000 LINUXI386    1000 0x00408841
                    ws02   100000 LINUXI386    1000 0x00408841
                    ws01   140000 LINUXI386    1000 0x00408841
pvm> halt
Terminated
[amit@onyx amit]:exit

qsub: job 117.onyx.boisestate.edu completed
[amit@onyx amit]:
```

The user can run `xpvm` in a similar fashion.

1 GUI

The user can use `xpbs` for a graphical front-end to the PBS system. The user can also use `xpbsmon` to monitor the state of the cluster under the PBS system.

2 Documentation

The are man pages for pbs, qsub, qstat, qdel as well as various other parts of PBS.

Installation Issues for a Beowulf Cluster

These are some issues with a guarded Beowulf cluster with all the nodes on a private network and master with two network interfaces: one to a private network and one to the public network.

- If you have multiple network interfaces on the master machine (like on onyx: ws00 points to the private gigabit network and onyx points to the fast ethernet connected to the Internet), then you need to change the file `/var/spool/pbs/server_name` to contain `ws00` on all machines that will run PBS.
- PBS copies results back using the `pbs_rcp` program. In order for that to work, the nodes must have `rsh` access to the master on the private network. This can be done by adding the nodes to the `/etc/xinted.d/rsh` file and restarting the `xinted` service. For example, on onyx, the file looks like the following.

  ```
  # default: on
  # description: The rshd server is the server for the rcmd(3) routine and, \
  ```

```
#       consequently, for the rsh(1) program.  The server provides \
#       remote execution facilities with authentication based on \
#       privileged port numbers from trusted hosts.
service shell
{
        socket_type             = stream
        wait                    = no
        user                    = root
        log_on_success          += USERID
        log_on_failure          += USERID
        server                  = /usr/sbin/in.rshd
        disable                 = no
        only_from               = ws00 ws01 ws02 ws03 ws04 ws05 ws06 ws07 ws08
ws09 ws10 ws11 ws12 ws13 ws14 ws15 ws16 ws17 ws18 ws19 ws20 ws21 ws22
ws23 ws24 ws25 ws26 ws27 ws28 ws29 ws30 ws31 ws32 }
```

Appendix

The pbsget script for use with PBS

The latest version of the pvm/xpvm scripts for use with PBS are at
`http://cs.boisestate.edu/∼amit/research/beowulf/tools/pvm/pbspvm`.

```
#!/bin/sh
#vim: set ts=4:
#
# author: Amit Jain
# date: Tue Nov 18 17:05:23 MST 2003
# license: GNU General Public License
#          See http://www.gnu.org/licenses/gpl.txt for the
#          full text of the license.

echo
echo "#####################################################"
echo "Invoke PVM using PBS."
echo "Usage: " `basename $0` " [-h] -<#nodes>"
echo "Uses onyx + 4 nodes by default"
echo "#####################################################"
echo

arg=$1
arg=${arg:0:2}
if test "$arg" = "-h"
then
    echo "Usage: " `basename $0` " [-h] -<#nodes>"
    exit 1
fi

# determine number of nodes to allocate: default is 4
if test "$1" = ""
then
    p=4
    echo "Using $p hosts by default"
else
    arg=$1
    p=${arg:1}
fi
maxnodes=`wc -l /usr/local/etc/machines| awk '{print $1}'`
maxnodes=$[maxnodes-1]
if test $p -gt $maxnodes
```

```
then
    echo `basename $0` ": Maximum number of nodes allowed is $maxnodes!"
    exit 1
fi

# check if  PBS interactive job already running for this user
userid=`id -u`
if test -f /tmp/pvm.pbs.$userid
then
    echo "PVM interactive session already running!"
    echo "If that is not the case, remove /tmp/pvm.pbs.$userid"
    echo "and try again!"
    exit 1
fi

# create PBS batch file to reserve the right number of nodes
echo >> /tmp/pvm.pbs.$userid
echo "#!/bin/sh" >> /tmp/pvm.pbs.$userid
echo "#PBS -l nodes=1:master+$p:node" >> /tmp/pvm.pbs.$userid


disp=$DISPLAY

echo
echo "*****************************************************"
echo " Scheduling an interactive pvm session with PBS."
echo " Please end session by typing in exit."
echo " For running xpvm, use the following commands."
echo "  export DISPLAY=$disp "
echo "  /usr/bin/xpvm \$PBS_NODEFILE "
echo " For running pvm, use the following command."
echo "  /usr/bin/pvm \$PBS_NODEFILE "
echo " Please always halt the pvm system before exiting "
echo "   either the pvm console or xpvm!!!"
echo " Or use the pvmrun command, which halts pvm properly "
echo " Usage: pvmrun -np <#copies> <executable> {<args>,...}"
echo "*****************************************************"
echo


qsub -I /tmp/pvm.pbs.$userid

rm -f /tmp/pvm.pbs.$userid
```

Source code for pvmrun command

Source code for pvmrun command is available at
`http://cs.boisestate.edu/∼amit/research/beowulf/tools/pvm/pvmrun.c`.