Using the Platform LSF blaunch Framework

Platform LSF
Version 8.0
June 2011

Copyright

We’d like to hear from you

You can help us make this document better by telling us what you think of the content, organization, and usefulness of the information. If you find an error, or just want to make a suggestion for improving this document, please address your comments to doc@platform.com.

Your comments should pertain only to Platform documentation. For product support, contact support@platform.com.

Document redistribution and translation

Internal redistribution

Trademarks

Third-party license agreements

http://www.platform.com/Company/third.part.license.htm

Third-party copyright notices

http://www.platform.com/Company/Third.Party.Copyright.htm

Contents

Open MPI

LSF must be installed and running. You must build Open MPI according to the Open MPI documentation for implement Open MPI with LSF.

Open MPI 1.3.2 and up is tightly integrated with LSF’s blaunch functionality.

1. Run Open MPI jobs in LSF.

   ```
   bsub -n2 -o %J.out -e %J.err mpiexec mympi.out
   ```

Platform MPI

Platform MPI (formerly HP-MPI)is partially integrated with LSF.

1. Set the MPI_REMSH environment variable.

   ```
   MPI_REMSH=blaunch;export MPI_REMSH
   ```

2. Run your job. For example:

   ```
   bsub -n 16 -R "span[ptile=4]" $MPI_ROOT/bin/mpirun -lsb_mcpu_hosts a.out
   ```

   Using Platform MPI with Infiniband:

   ```
   bsub -n 16 -R "span[ptile=4]" $MPI_ROOT/bin/mpirun -lsb_mcpu_hosts -IBV a.out
   ```

MVAPICH

MVAPICH can be integrated with LSF.

1. Choose from two options:
   a) Change the MVAPICH source code (if you only want to run MVAPICH with LSF).

      Modify the MVAPICH source code: `RSH_CMD =`'`blaunch'` and build the package.

   b) Write a wrapper script.

      Wrap `/usr/bin/rsh` on the first execution host or all candidate execution hosts for `blaunch` as follows:

      Example wrapper script:

      cat /usr/bin/rsh
      ```
      #!/bin/sh
      #
      # wrapper /usr/bin/rsh
      # blaunch is used when applicable
      #
      if [ -z "$LSF_BINDIR" \
           -o -z "$LSB_JOBID" \
           -o -z "$LSB_JOBINDEX" \
           -o -z "$LSB_JOBRES_CALLBACK" \
           -o -z "$LSB_DJOB_HOSTFILE" ]; then
          RSH="/usr/bin/rsh.bin"
      else
          RSH=$LSF_BINDIR/blaunch
      fi
      $RSH $*
      ```

   c) If you wrote a wrapper script, specify host file with a script.

      Example:

      cat run.mvapich
      ```
      #! /bin/sh
      #BSUB -n 2
      #BSUB -o %J.out
      #BSUB -e %J.err
      #BSUB -R 'span[ptile=1]'
      mpirun_rsh -rsh -np $LSB_DJOB_NUMPROC -hostfile $LSB_DJOB_HOSTFILE mympi
      ```

2. Run **bsub**.

   For example, **bsub < run.mvapich**.

Intel MPI and mpich2

Intel MPI is a variation of MPICH2.This solution applies to either integration.

1. Create a wrapper script around mpdboot, without the daemonize option.

   It should:

   - loop all hosts and `blaunch` `mpd` without `-d` option in background
   - at the end, check whether the `mpd` ring is constructed correctly
   - exit 0 if correctly constructed, otherwise print out error

   Example:

```
#!/usr/bin/env python2.3
"""
mpdboot for LSF
    [-f | --hostfile hostfile]
    [-i | --ifhn=alternate_interface_hostname_of_ip_address
     -f | --hostfile hostfile]
    [-h]
"""
import re
import string
import time
import sys
import getopt
from time import ctime
from os       import environ, path
from sys      import argv, exit, stdout
from popen2   import Popen4
from socket   import gethostname, gethostbyname
def mpdboot():
    # change me
    MPI_ROOTDIR="/opt/mpich2"
    #
    mpdCmd="%s/bin/mpd" % MPI_ROOTDIR
    mpdtraceCmd="%s/bin/mpdtrace" % MPI_ROOTDIR
    mpdtraceCmd2="%s/bin/mpdtrace -l" % MPI_ROOTDIR
    nHosts = 1
    host=""
    ip=""
    localHost=""
    localIp=""
    found = False
    MAX_WAIT = 5
    t1 = 0
    hostList=""
    hostTab = {}
    cols = []
    hostArr = []
    hostfile = environ.get('LSB_DJOB_HOSTFILE')
    binDir = environ.get('LSF_BINDIR')
    if environ.get('LSB_MCPU_HOSTS') == None \
        or hostfile == None \
        or binDir == None:
        print "not running in LSF"
        exit (-1)
    rshCmd = binDir + "/blaunch"
    p = re.compile("\w+_\d+\s+\(\d+\.\d+\.\d+\.\d+")
#
    try:
        opts, args = getopt.getopt(sys.argv[1:], "hf:i:", ["help", "hostfile=", "ifhn="])
    except getopt.GetoptError, err:
        print str(err)
        usage()
        sys.exit(-1)
    fileName = None
    ifhn = None
    for o, a in opts:
        if o == "-v":
            version();
            sys.exit()
        elif o in ("-h", "--help"):
            usage()
            sys.exit()
        elif o in ("-f", "--hostfile"):
            fileName = a
        elif o in ("-i", "--ifhn"):
            ifhn = a
        else:
            print "option %s unrecognized" % o
            usage()
            sys.exit(-1)
    if fileName == None:
        if ifhn != None:
            print "--ifhn requires a host file containing 'hostname 
ifhn=alternate_interface_hostname_of_ip_address'\n"
            sys.exit(-1)
        # use LSB_DJOB_HOSTFILE
        fileName = hostfile
    localHost = gethostname()
    localIp = gethostbyname(localHost)
    pifhn = re.compile("\w+\s+\ifhn=\d+\.\d+\.\d+\.\d+")
    try:
        # check the hostfile
        machinefile = open(fileName, "r")
        for line in machinefile:
            if not line  or  line[0] == '#':
                continue
            line = re.split('#', line)[0]
            line = line.strip()
            if not line:
                continue
            if not pifhn.match (line):
                # should not have --ifhn option
                if ifhn != None:
                    print "host file %s not valid for --ifhn" % (fileName)
                    print "host file should contain 'hostname ifhn=ip_address'"
                    sys.exit(-1)
                host = re.split(r'\s+',line)[0]
                if cmp (localHost, host) == 0 \
                   or cmp(localIp, gethostbyname(host))== 0:
                    continue
                hostTab[host] = None
            else:
                # multiple blaunch-es
                cols = re.split(r'\s+\ifhn=',line)
                host = cols[0]
                ip = cols[1]
                if cmp (localHost, host) == 0 \
                   or cmp(localIp, gethostbyname(host))== 0:
                    continue
                hostTab[host] = ip
            nHosts += 1
            #print "line: %s" % (line)
        machinefile.close()
    except IOError, err:
        print str(err)
        exit (-1)
    # launch an mpd on localhost
    if ifhn != None:
        cmd = mpdCmd + " --ifhn=%s " % (ifhn)
    else:
        cmd = mpdCmd
    print "Starting an mpd on localhost:", cmd
    Popen4(cmd, 0)
    # wait til 5 seconds at max
    while t1 < MAX_WAIT:
        time.sleep (1)
        trace = Popen4(mpdtraceCmd2, 0)
        # hostname_portnumber (IP address)
        line = trace.fromchild.readline()
        if not p.match (line):
            t1 += 1
            continue
        strings = re.split('\s+', line)
        (basehost, baseport) = re.split('_', strings[0])
        #print "host:", basehost, "port:", baseport
        found = True
        host=""
        break
    if not found:
        print "Cannot start mpd on localhost"
        sys.exit(-1)
    else:
        print "Done starting an mpd on localhost"
    # launch mpd on the rest of hosts
    if nHosts < 2:
        sys.exit(0)
    print "Constructing an mpd ring ..."
    if ifhn != None:
        for host, ip in hostTab.items():
            #print "host : %s ifhn %s\n" % (host, ip)
            cmd="%s %s %s -h %s -p %s --ifhn=%s" % (rshCmd, host, mpdCmd, basehost, baseport, ip)
            #print "cmd:", cmd
            Popen4(cmd, 0)
    else:
        for host, ip in hostTab.items():
            #print "host : %s ifhn %s\n" % (host, ip)
            hostArr.append(host +  " ")
        hostList = string.join(hostArr)
        #print "hostList: %s" % (hostList)
        cmd="%s -z \'%s\' %s -h %s -p %s" % (rshCmd, hostList, mpdCmd, basehost, baseport)
        #print "cmd:", cmd
        Popen4(cmd, 0)
    # wait till all mpds are started
    MAX_TIMEOUT = 300 + 0.1 * (nHosts)
    t1 = 0
    started = False
    while t1 < MAX_TIMEOUT:
        time.sleep (1)
        trace = Popen4(mpdtraceCmd, 0)
        if len(trace.fromchild.readlines()) < nHosts:
            t1 += 1
            continue
        started = True
        break
    if not started:
        print "Failed to construct an mpd ring"
        exit (-1)
    print "Done constructing an mpd ring at ", ctime()
def usage():
    print __doc__
if __name__ == '__main__':
    mpdboot()
```

```
cat run.intelmpi
#! /bin/sh
#BSUB -n 2
#BSUB -o %J.out
#BSUB -e %J.err
mpdboot.lsf
mpiexec -np $NUMPROC mympi.out
mpdallexit
```

2. Run **bsub**.

   For example, **bsub < run.intelmpi**.

Index