# HowTo – HPL Over Intel MPI

#This is a step by step procedure of how to run the High Performance Linpack (HPL) #benchmark on a Linux cluster. This was done using Intel-MPI on a Linux cluster of 128 #nodes running Intel's Nehalem processor 2.93 MHz with 12GB of RAM on each node. #The operating system that was used is RedHat Enterprise Linux 5.3. The #interconnectivity between the nodes was via Infiniband 4x-DDR using the standard #RedHat EL 5.3 drivers.

#You can use my simple PHP web tool to enter you system specs and it will suggest for #you optimal input parameters for your HPL file before running the benchmark on the #cluster. The tool can be accessed via the URL below:

http://hpl-calculator.sourceforge.net

#First of all, use the HPL source code that came with the Intel MKL libraries to build #HPL, it's easier to setup than the one from the HPL website. Also make sure you have #python installed since Intel MPI depends on it heavily when launching the jobs. You #also need to setup SSH trust keys between the nodes as well as rsh trust just in case.

#This is the location of the HPL source code that came with MKL:
[root@master mp_linpack]# pwd
/usr/local/intel/11.1.038/mkl/10.2.0.013/benchmarks/mp_linpack

#You don't need to create your Make file from scratch, Intel has prepared one for you #with the MKL optimizations needed to save you the hassle of doing it manually. Also #notice that the "CC" variable points to "mpiicc" and not "mpicc" in the file:
[root@master mp_linpack]# ls Make.em64t
Make.em64t
[root@master mp_linpack]# grep mpiicc Make.em64t
CC = mpiicc

#You will need to point your Make file to the Intel MPI directory and 64-bit paths:
[root@master mp_linpack]# vi Make.em64t (Edit the below 3 lines to point to Intel MPI and 64-bit directories)

```
############################################
MPdir = /usr/local/intel/11.1.038/impi/3.2.1.009
MPinc = -I$(MPdir)/include64
MPlib = $(MPdir)/lib64/libmpi.a
############################################
```

#Make sure your environment variables are pointing to Intel MPI and Intel compiler
#before building HPL:
[root@node001 mp_linpack]$ env | grep PATH

PATH=/usr/local/intel/11.1.038/impi/3.2.1.009/bin64:/usr/local/bin:/bin:/usr/bin:/usr/local/pbs/bin:/usr/local/pbs/sbin:/usr/local/maui/sbin:/usr/local/maui/bin:/usr/local/uxcat/bin:/usr/local/uxcat/sbin:/usr/local/intel/11.1.038/Compiler/11.1/038/bin/intel64

LD_LIBRARY_PATH=/usr/local/intel/11.1.038/impi/3.2.1.009/lib64:/usr/local/intel/11.1.038/Compiler/11.1/038/lib/intel64:/usr/lib64:/usr/local/lib64:/usr/lib:/usr/local/intel/11.1.038/mkl/10.2.0.013/lib/em64t

#Build HPL over Intel MPI
[root@master mp_linpack]# make arch=em64t clean_arch_all
[root@master mp_linpack]# make arch=em64t

#Binary should be compiled and ready to be used now:
[root@master mp_linpack]# cd bin/em64t/
[root@master em64t]# ls
HPL.dat  xhpl
#After you have the HPL binary compiled and ready, you need to prepare Intel MPI
#daemons before running your job on the cluster:

#Make sure IPoIB is up
#(We are running standard Infiniband drivers and DAPL that came with RedHat EL 5.3)

#This is an example of for setting up 4 nodes
[root@master ~]# psh node001-node004 "reboot"
[root@master ~]# psh node001-node004 "ifup ib0"
[root@master ~]# psh node001-node004 "ifconfig ib0 | grep -w inet"
node001:  inet addr:10.2.136.1  Bcast:10.2.136.255  Mask:255.255.255.0
node004:  inet addr:10.2.136.4  Bcast:10.2.136.255  Mask:255.255.255.0
node002:  inet addr:10.2.136.2  Bcast:10.2.136.255  Mask:255.255.255.0
node003:  inet addr:10.2.136.3  Bcast:10.2.136.255  Mask:255.255.255.0

#Make sure your environment variables are set in your .cshrc in your home directory so it
#takes effect on all nodes when a job is launched remotely on nodes:
[chewbacca@node001 em64t]$ cat ~chewbacca/.cshrc | grep -v "#"
setenv PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/pbs/bin:/usr/local/pbs/sbin:/usr/local/maui/sbin:/usr/local/maui/bin:/usr/local/uxcat/bin:/usr/local/uxcat/sbin:/usr/local/intel/11.1.038/Compiler/11.1/038/bin/intel64

setenv LD_LIBRARY_PATH
/usr/local/intel/11.1.038/Compiler/11.1/038/lib/intel64:/usr/lib64:/usr/local/lib64:/usr/lib:/usr/local/intel/11.1.038/mkl/10.2.0.013/lib/em64t

```
setenv I_MPI_DEVICE "rdma:OpenIB-cma
source /usr/local/intel/11.1.038/impi/3.2.1.009/bin64/mpivars.csh

#Start Intel MPD daemons (They claim that mpirun launches these daemons for you
#automatically but it was buggy for me when testing it, so just launch it manually)
[chewbacca@node001 em64t]$ cat mpd.hosts
node001
node002
node003
node004
[chewbacca@node001 em64t]$ mpdboot -n 4 -f mpd.hosts

#Make sure the mpd daemons are up now:
[chewbacca@node001 em64t]$ mpdtrace
node001
node002
node003
node004
#Make sure your I_MPI_DEV variable is pointing to your Infiniband card so that Intel
#MPI knows what interface to use for its communication.

#In order to know what to set the I_MPI_DEV variable to, check the below file, in our
#case it is "OpenIB-cma"
[root@node001 ~]# cat /etc/ofed/dat.conf | grep ib0 | grep Open
OpenIB-cma u1.2 nonthreadsafe default libdaplcma.so.1 dapl.1.2 "ib0 0" ""
[root@node 001~]#

#Notice the syntax of using ":" when setting the I_MPI_DEVICE environment variable
#e.g. "rdma:OpenIB-cma", also make sure your I_MPI_ROOT is set as well.
[chewbacca@node001 em64t]$ env | grep I_MPI
I_MPI_ROOT=/usr/local/intel/11.1.038/impi/3.2.1.009
I_MPI_DEVICE=rdma:OpenIB-cma
[chewbacca@node001 em64t]$


#Make sure your PATH and LD_LIBRARY_PATH are pointing to the Intel MPI
#libraries, MKL, and Intel compiler libraries and binaries just in case.
[chewbacca@node001 em64t]$ env | grep PATH
PATH=/usr/local/intel/11.1.038/impi/3.2.1.009/bin64:/usr/local/bin:/bin:/usr/bin:/usr/loca
l/pbs/bin:/usr/local/pbs/sbin:/usr/local/maui/sbin:/usr/local/maui/bin:/usr/local/uxcat/bin:/
usr/local/uxcat/sbin:/usr/local/intel/11.1.038/Compiler/11.1/038/bin/intel64

LD_LIBRARY_PATH=/usr/local/intel/11.1.038/impi/3.2.1.009/lib64:/usr/local/intel/11.
1.038/Compiler/11.1/038/lib/intel64:/usr/lib64:/usr/local/lib64:/usr/lib:/usr/local/intel/11.
1.038/mkl/10.2.0.013/lib/em64t
```

#We have 8 CPUs per node, so list each node 8 times in your nodes file to launch 8 HPL
#instances on each node
[chewbacca@node001 em64t]$ cat nodes
node001
node001
node001
node001
node001
node001
node001
node001
node002
node002
node002
node002
node002
node002
node002
node002
node003
node003
node003
node003
node003
node003
node003
node003
node004
node004
node004
node004
node004
node004
node004
node004

#The mpirun that comes with Intel MPI seems to be a wrapper around mpiexec, so I just
#use mpiexec directly since mpirun was buggy. Also make sure you don't launch the job
#from the master node since it doesn't have Infiniband, launch it from a compute node
#that has Infiniband, e.g. node001:

[chewbacca@node001 em64t]$ which mpiexec
/usr/local/intel/11.1.038/impi/3.2.1.009/bin64/mpiexec
[chewbacca@node001 em64t]$ mpiexec -machinefile nodes  -n 32 ./xhpl | tee HPL.out

#This is the HPL input file that we ran on 128 nodes, each node has 12GB, the chosen N #was around 92% aligned with NB value of 224

[chewbacca@node001 em64t]$ cat HPL.dat

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
417536       Ns
1            # of NBs
224          NBs
0            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
16           Ps
64           Qs
16.0         threshold
1            # of panel fact
0            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
4            NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
0            RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
0            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
0            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

## #The HPL results when running over Intel MPI:

```
================================================================================
HPLinpack 2.0  --  High-Performance Linpack benchmark  --  September 10, 2008
Written by A. Petitet and R. Clint Whaley,  Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
Modified by Julien Langou, University of Colorado Denver
================================================================================
An explanation of the input/output parameters follows:
T/V    : Wall time / encoded variant.
N      : The order of the coefficient matrix A.
NB     : The partitioning blocking factor.
P      : The number of process rows.
Q      : The number of process columns.
Time   : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

N      :  417536
NB     :     224
PMAP   : Row-major process mapping
P      :      16
Q      :      64
PFACT  :    Left
NBMIN  :       4
NDIV   :       2
RFACT  :    Left
BCAST  :   1ring
DEPTH  :       0
SWAP   : Mix (threshold = 64)
L1     : transposed form
U      : transposed form
EQUIL  : yes
ALIGN  : 8 double precision words
--------------------------------------------------------------------------------
- The matrix A is randomly generated for each test.
- The following scaled residual check will be computed:
      ||Ax-b||_oo / ( eps * ( || x ||_oo * || A ||_oo + || b ||_oo ) * N )
- The relative machine precision (eps) is taken to be            1.110223e-16
- Computational tests pass if scaled residuals are less than                16.0
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00L2L4      417536   224    16    64            4701.73             1.032e+04
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0011152 ..... PASSED
================================================================================

Finished      1 tests with the following results:
              1 tests completed and passed residual checks,
              0 tests completed and failed residual checks,
              0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------
End of Tests.
================================================================================
```