HowTo - High Performance Linpack (HPL)

This is a step by step procedure of how to run HPL on a Linux cluster. This was done using the MVAPICH MPI implementation on a Linux cluster of 512 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, as root or normal user, you need to compile MVAPICH over Infiniband, Intel compiler, and Intel MKL to get the best results out of the Nehalem processor as Intel recommends. This will produce the optimal "mpirun" binary to be used to launch HPL on the cluster nodes.

Install the below development rpms for Infiniband since they will place a few header files that are needed during the build under /usr/include/infiniband:

[root@node lib64] rpm -ivh libibcommon-devel-1.1.1-1.el5.x86_64.rpm

[root@node lib64] rpm -ivh libibumad-devel-1.2.1-1.el5.x86_64.rpm

[root@node lib64] rpm -ivh libibverbs-devel-1.1.2-1.el5.x86_64.rpm

Compile and install MVAPICH (Options in red are required for TotalView debugger to work with MVAPICH)

```
cd /usr/src
wget http://mvapich.cse.ohio-state.edu/download/mvapich/mvapich-1.1.tgz
tar zxvf mvapich-1.1.tgz
cd mvapich-1.1/
cp make.mvapich.gen2 make.mvapich.intel
vi make.mvapich.intel    (Edit this file to point to our compilers and installation folder as below)

================================================================================
  Add intel compilers and remove XRC from the cflags
IBHOME=${IBHOME:-/usr/include/infiniband}
IBHOME_LIB=${IBHOME_LIB:-/usr/lib64}
PREFIX=${PREFIX:-/usr/local/mpi/mvapich/intel/1.1}
export CC=${CC:-/usr/local/intel/11.1.038/Compiler/11.1/038/bin/intel64/icc}
export CXX=${CXX:-/usr/local/intel/11.1.038/Compiler/11.1/038/bin/intel64/icpc}
export F77=${F77:-/usr/local/intel/11.1.038/Compiler/11.1/038/bin/intel64/ifort}
export F90=${F90:-/usr/local/intel/11.1.038/Compiler/11.1/038/bin/intel64/ifort}
```

export CFLAGS=${CFLAGS:--D${ARCH} ${PROCESSOR} ${PTMALLOC} -DEARLY_SEND_COMPLETION -DMEMORY_SCALE -DVIADEV_RPUT_SUPPORT -D_SMP_ -D_SMP_RNDV_ -DCH_GEN2 -D_GNU_SOURCE ${COMPILER_FLAG} -I${IBHOME}/include $OPT_FLAG}
==========
./configure --enable-debug --enable-sharedlib --with-device=ch_gen2 --with-arch=LINUX -prefix=${PREFIX} $ROMIO --without-mpe -lib="$LIBS" 2>&1 |tee config-mine.lo

export MPIRUN_CFLAGS="$MPIRUN_CFLAGS -g"
./make.mvapich.intel

Now do the HPL installation as a normal user, in this case I used my id chewbacca with the shared home directory being mounted from the master node of the cluster on all nodes. I did the compilation on the master node since it has full packages installed on it but ran the actual benchmark from a compute node since it has an Infiniband card and the master doesn't.

[root@node ~] su - chewbacca

[chewbacca@node ~]$ mkdir hpl

[chewbacca@node ~]$ cd hpl/

Download the source code for HPL

http://www.netlib.org/benchmark/hpl/hpl-2.0.tar.gz

Untar and prepare Make file for compilation

[chewbacca@node ~/hpl]$ tar -xzvf hpl-2.0.tar.gz

[chewbacca@node ~/hpl]$ cd hpl-2.0/setup

[chewbacca@node setup]$ sh make_generic **This generates a template for us**

[chewbacca@node setup]$ cp Make.UNKNOWN ../Make.Linux

[chewbacca@node setup]$ cd ..

[chewbacca@node hpl-2.0]$

You might need to install some compatibility libraries on all nodes depending on your system

yum install compat-gcc-34-g77-3.4.6-4.x86_64

Modify the Make file and point it to our compilers and libraries, also note that we tell it to use the MKL and BLAS libraries from Intel to enhance the performance on the Nehalem.

[chewbacca@node hpl-2.0]$ vi Make.Linux

ARCH = Linux

TOPdir = /home/ecc_11/chewbacca/hpl/hpl-2.0

MPdir = /usr/local/mpi/mvapich/intel/1.1

MPlib = -L$(MPdir)/lib

LAdir1 = /usr/local/intel/11.1.038/mkl/10.2.0.013/lib/em64t/

LAdir2 = /usr/local/intel/11.1.038/Compiler/11.1/038/

LAlib = -L$(LAdir1) -L$(LAdir2) -lclusterguide -lmkl_intel_lp64 -lmkl_sequential -lmkl_core

CC = $(MPdir)/bin/mpicc

LINKER = $(MPdir)/bin/mpicc

If they don't already exist, create symbolic links for Infiniband libraires so that the make won't fail, or install "libibverbs-devel" and "libibumad-devel" and the symbolic links will be placed and the libraries will be included automatically. This needs to be done as root if needed.

[root@node lib64] pwd

/usr/lib64

[root@node lib64] ln -s libibumad.so.1.0.3 libibumad.so

[root@node lib64] ln -s libibverbs.so.1.0.0 libibverbs.so

[root@node lib64] ls -l libibumad.so

lrwxrwxrwx 1 root root 18 Aug 4 07:19 libibumad.so -> libibumad.so.1.0.3

[root@node lib64] ls -l libibverbs.so

lrwxrwxrwx 1 root root 19 Aug 4 07:19 libibverbs.so -> libibverbs.so.1.0.0

Now back as a normal user, set you library path

[chewbacca@node ~]$ echo "setenv LD_LIBRARY_PATH /usr/local/intel/11.1.038/mkl/10.2.0.013/lib/em64t/:/usr/local/mpi/mvapich/intel/1.1/lib/shared:/usr/lib64/:/usr/local/lib64:/usr/lib:/usr/local/intel/11.1.038/Compiler/11.1/038/lib/intel64" >> ~chewbacca/.cshrc

[chewbacca@node ~]$ source .cshrc

Clean everything before building then build

[chewbacca@node ~]$ cd ~chewbacca/hpl/hpl-2.0

[chewbacca@node hpl-2.0]$ make arch=Linux clean_arch_all

[chewbacca@node hpl-2.0]$ make arch=Linux

Binary is now created, you need to run it from the same directory containing HPL.dat input file.

[chewbacca@node hpl-2.0]$ cd bin/Linux/

[chewbacca@node Linux]$ ls

HPL.dat xhpl

Generate a file containing the node names, for testing just create one with the localhost:

```
[chewbacca@node Linux]$ vi create-node-file
[chewbacca@node Linux]$ chmod 755 create-node-file
[chewbacca@node Linux]$ cat create-node-file
!/bin/bash

for i in `seq 4`

do echo localhost >> nodes

done
[chewbacca@node Linux]$ ./create-node-file
[chewbacca@node Linux]$ cat nodes
localhost
localhost
localhost
localhost
[chewbacca@node Linux]$
```

Generate the nodes file to be used for the actual run containing all 512 node names, each node name is repeated 8 times since we have 8 CPUs and want 8 HPL instances to run on each node:

```
[chewbacca@node Linux]$ cat create-allnodes-file
!/bin/bash

for node in `seq -w 512`
do
   for cpu in `seq 8`
    do
      echo node$node
    done

done > allnodes

[chewbacca@node Linux]$ chmod 755 create-allnodes-file
[chewbacca@node Linux]$ ./create-allnodes-file
node001
node001
node001
node001
node001
node001
node001
node001
node001
node002
……
```

Edit your HPL.dat file which is the input file you give to the HPL binary. The default HPL is set for 4 processes with very low N values which will run quick but with bad results. At this point you can use the PHP web tool to aid you in suggesting optimized input parameters. The tool should suggest to you the 4 most important parameters in the HPL input file which are N, NB, P, and Q. More details about these paramters can be found at the end of this document.

Run a test HPL benchmark to make sure binary is working, just run it on localhost with the default simple HPL.dat input file. Launch the binary from a node with and Infiniband (compute node) and run it as normal user not root.

```
[chewbacca@node Linux]$ ssh node001
 [chewbacca@node001 ~]$ cd hpl/hpl-2.0/bin/Linux/
[chewbacca@node001 Linux]$ env | grep LD                    (Make sure your library path is ok)
LD_LIBRARY_PATH=/usr/local/intel/11.1.038/mkl/10.2.0.013/lib/em64t/:/usr/local/mpi/mvapich/intel/1.1/lib/shared:/usr/lib64/:/usr/local/lib64:/usr/lib:/usr/local/intel/11.1.038/Compiler/11.1/038/lib/intel64

[chewbacca@node001 Linux]$ /usr/local/mpi/mvapich/intel/1.1/bin/mpirun -np 4 -hostfile nodes  ./xhpl | tee HPL.out
```

Finally run the HPL benchmark using the HPL.dat input file that has the optimized input parameters after you made sure that the binary works.

[chewbacca@node001 Linux]$ /usr/local/mpi/mvapich/intel/1.1/bin/mpirun -np 4096 -hostfile **allnodes** ./xhpl | tee HPL.out

Please find below some notes I wrote on how to pick the parameter values for the HPL.dat input file in order to optimize your results:

(**P * Q**) is the size of your grid which is equal to the number of processors your cluster has. In the 512 nodes cluster input file, the P and Q product is 4096 which is basically the number of processors we have (512*8). When picking these two numbers, try to make your grid as "square" as possible in shape. The HPL website mentions that best practice is to have it close to being a "square", thus P and Q should be approximately equal, with Q slightly larger than P.

Below are some of the possible combinations for P and Q for our 512 node cluster and the one that we ended up using is highlighted:

P	*	Q
1	*	4096
2	*	2048
4	*	1024
8	*	512
16	*	256
32	*	128
64	*	64

Another important parameter is "**N**", which is the size of your problem, and usually the goal is to find the largest problem size that would fit in your system's memory. Choose "N" to be close to your total memory size (double precision 8 bytes), but don't make it equal to 100% of the memory size since some of the memory is consumed by the system, so choose something like 90% of the size of the total memory size. If you choose a small value for "N", this will result in not enough work performed on each CPU and will give you bad results and low efficiency. If you choose a value of "N" exceeding your memory size, swapping will take place and the performance will go down.

Another parameter is "**NB**", which is the block size in the grid. Usually block sizes giving good results are within the [96,104,112,120,128, …, 256] range. For our cluster we used NB=224.

The calculation for the N value can be a bit confusing, so let's do an example for it. Let's try to find a decent value for N for our cluster 512 node cluster, something around 90% of the total memory of the system (double precision 8 bytes). Below is the equation to calculate that along with the steps needed to optimize the result:

sqrt((Memory Size in Gbytes * 1024 * 1024 * 1024 * Number of Nodes) /8) * 0.90

Let's do it step by step, so we have 512 nodes each with 12GB of memory:

(12 * 1024 * 1024 * 1024 * 512) /8 = 824633720832

Then the square root of that figure is ~ 908093 (Which is equivalent to 100% of total memory)

Then that figure times 0.9 is ~ 817283

Now we can optimize the value of N even more. Usually a better result can be obtained if N is NB aligned, in our case we chose NB to be 224 so we do the below step:

817283 / 224 ~ 3648

So finally we do 224 * 3648 = 817152 (This is the same result you will get when using our PHP web tool)

(This should be a decent pick for N and is around 90% of total memory of the system)

Note: Picking these values for the input file is the tricky part when running HPL and you will have to try several combinations until you are satisfied with the results, it is basically a skill that you master with practice.

Other terms that you might want to know for the Top500.org are:

- **Rpeak**
- **Rmax**
- **Efficiency**

Rpeak: It's basically the theoretical calculated performance of your system. The below example is for calculating Rpeak for our 512-nodes cluster.

Rpeak = CPU Speed Ghz * Number of Total Cores * Operations/Cycle

= 2.93 * (8*512) * 4

= 48005 TeraFlops

Rmax: It's the actual performance you get out of the system when you run HPL on it, in the case of our cluster it was 41770 TeraFlops.

Efficiency: The ratio of Rmax to Rpeak (Rmax/Rpeak). The efficiency can be dramatically affected by the type of interconnect you use in the cluster (e.g. Infiniband, Myrinet, Ethernet, etc…). In the case of our cluster we got around 87% efficiency according to the table below:

70%	72%	74%	76%	78%	80%	82%	84%	86%	88%	90%	92%	94%	96%	98%	100%
33603	34563	35523	36483	37443	38404	39364	40324	41284	42244	43204	44164	45124	46084	47044	48005

The actual input and output files that were used for our cluster can be found in the following page and you can use it as a template when running HPL on other clusters. You will need to tweak the value of "N" depending on the amount of memory that your new cluster has as well as the "P*Q" values depending on the number of processors that you have.

Please note that all of the above tedious calculations are done for you if you use our PHP web tool hosted on sourceforge in the link below:

http://hpl-calculator.sourceforge.net

Below is the HPL.dat file that was used for the 512-node Infiniband cluster using Nehalem 2.93 MHz processor with 12 GB of RAM on each node. The parameters in red are the ones that were suggested by our web tool and from several experimental runs they gave an optimized result.

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL-13.out      output file name (if any)
file            device out (6=stdout,7=stderr,file)
1             of problems sizes (N)
829248          Ns
1             of NBs
224             NBs
0             PMAP process mapping (0=Row-,1=Column-major)
1             of process grids (P x Q)
32              Ps
128             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)
128           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)
```

Best output for our cluster was the below (41.77 TeraFlops):

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00L2L4      829248   224    32   128          9103.52             4.177e+04
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=     0.0005126 ...... PASSED
================================================================================
```