Tutorial 3 (2019)

From ACE Lab

Follow the instructions below to complete this tutorial. The tutors are available to answer valid, well thought out questions. The tutors will not touch your keyboard or complete the tutorials for you. Answer questions provided in a text document and email to dmacleod@csir.co.za. Pay special attention to the instructions in red, you will need to demonstrate this to a tutor in order to complete that task.

Part 0 - Firewalld Fix

Firewalld is a firewall management solution available for many Linux distributions which acts as a frontend for the iptables packet filtering system provided by the Linux kernel. You've used this tool to facilitate the configuration of a NAT (enabling your internal network to get passthrough to the external network via you headnode).

If you have trouble resolving google.com but are able to ping 8.8.8.8. You most probably have a conflict on your firewall rules blocking you. Follow these instructions to fix it on you headnode. Notes: 1. Ensure that firewalld is running on you headnode

```
systemctl restart firewalld
systemctl status firewalld
```

should give you output without errors
2. Add your public interface to trusted zone to bypass the conflict.

```
firewall-cmd --zone=trusted  --change-interface=ens3
```

3. Check if the interface is on correct zone

```
firewall-cmd --get-active-zones
```

4. You should see

```
trusted
interfaces: ens3
```

Part 1 - HPL Benchmark

The High Performance LINPACK (HPL) benchmark is used to measure a system's floating point number processing power. The resulting score in, Floating Point Operations Per Second (FLOPS), is often used to roughly quantify the computational power of an HPC system. HPL requires math libraries to perform its floating point operations, and an MPI installation for communication in order to execute in parallel across multiple CPU cores (and hosts).

Important tip: if your compile fails, you should reset to a clean start point with 'make clean'.

1. Install OpenMPI and Basic Linear Algebra Subprograms (BLAS) library.

```
yum install openmpi-devel atlas-devel
```

Create a new bash session to load the module system you just installed.

2. Download and extract HPL from

```
http://www.netlib.org/benchmark/hpl/
```

3. Enter the directory and copy a template Makefile, the Makefiles are labeled Make.<arch>

```
cp setup/Make.Linux_PII_CBLAS_gm .
```

note that the Makefile suffix is used as the architecture identifier.

4. Edit the path of the build directory in your Makefile:

```
TOPdir =
```

5. Edit the Makefile to use the ATLAS math library you installed

```
LAdir       = /usr/lib64/atlas
LAlib        = $(LAdir)/libtatlas.so $(LAdir)/libsatlas.so
```

6. Edit the Makefile to use your OpenMPI compiler (to produce an MPI aware parallel code)

```
CC           = mpicc
LINKER       = mpicc
```

QUESTION 1:
Explain the difference between shared and distributed memory systems

_____

_____

_____

NB: by default the the OpenMPI you installed is not sourced in your environment ($PATH & $LD_LIBRARY_PATH) by default. Test this with 'which mpicc'. The compilation will fail if the mpicc command is not available, resulting in the error: "mpicc: Command not found". To add OpenMPI to your environment you need to load the appropriate module. Check which modules are available with

```
module avail
```

then load the OpenMPI module with

```
module load <module_name>
```

QUESTION 2:
Explain the difference between static and dynamic linked libraries

_____

_____

_____

7. Compile HPL

```
make arch=<arch>
```

to confirm that the compilation completed successfully, check that the **xhpl** executable was produced in bin/<arch>

8. The HPL.dat file defines how the HPL benchmark solves a large dense linear array of double precision floating point numbers. Therefore selecting the appropriate parameters in this file can have a massive effect on the FLOPS you obtain.

```
     The most important parameters are:
N:   defines the length of one of the sides of the 2D array to be solved.
     Therefore, problem size ∝ runtime ∝ memory usage ∝ <N>².
     For best performance N should be a multiple of NB.
NB:  defines the block (or chunk) size into which the array is divided.
     The optimal value is determined by the CPU architecture such that the block fits in cache (Google).
P&Q: define the domains (in two dimensions) for how the array is partitioned on a distributed memory system.
     Therefore P*Q = MPI ranks.
```

QUESTION 3:
What is a FLOP and how does it relate to the performance of supercomputers?

_____

_____

_____

9. Execute HPL on your headnode

```
mpirun -np <cores> ./xhpl
```

At this point you are ready to run HPL on your cluster, however you only have one compute node currently. Its time to deploy a second compute node in OpenNebula. Ask Israel or Matthew for assistance. Once your two compute nodes have been successful deployed, are accessible from the headnode, and added to SLURM you can continue with running HPL across multiple nodes.

10. Create a SLURM submission script to run your benchmark on your two compute nodes. The script should look as follows:

```
#!/bin/bash
#SBATCH --ntasks <MPI_RANKS>
#SBATCH -N <NODES>
#SBATCH -t 02:00:00
#SBATCH --job-name=hpl_benchmark

mpirun /path/to/xhpl
```

SLURM will populate the appropriate MPI parameters based on the resources you requested, so specifying ranks (-np) is not required.

11. Submit your job to SLURM:

```
sbatch <script>
```

12. Check the state of your job with

```
squeue
```

By default, SLURM will store the job output to a log file 'slurm-<job_id>.out'. Check the result of your benchmark here.

Part 2 - Optimising HPL

You now have a functioning HPL benchmark. However, using math libraries (BLAS, LAPACK, **ATLAS**) from a repository (yum) will not yield optimum performance, because these repos contain generic code compiled to work on all x86 hardware. Code compiled specifically for HPC hardware can use instruction sets like AVX, AVX2 and AVX512 (if available) to make better use of the CPU. A (much) higher HPL result is possible if you compile your BLAS library (such as ATLAS, GOTOBLAS, OpenBLAS or Intel MKL) from source on the hardware you intend to run the code on.

The VM's that make up your cluster are not the same architecture. In order to compile high performance codes for your compute nodes, you need to perform the next steps on your compute node.

1. Download OpenBLAS source code and compile

2. Recompile HPL using this new BLAS implementation (edit LAdir)

3. Rerun HPL on your cluster

It is useful to know what the theoretical FLOPS performance (RPeak) of your hardware is when trying to obtain the highest benchmark result (RMax). RPeak can be derived from the formula:

```
RPeak = CPU_frequency[GHz] * CPU_cores * OPS/cycle
```

Newer CPU architectures allow for 'wider' instrument sets which execute multiple instructions per CPU cycle. The table below shows the floating point operations per cycle of various instruction sets:

SSE4.2	4
AVX	8
AVX2	16
AVX512	32

You can determine your CPU model as well as the instructions supported on your **compute node** with the command

```
cat /proc/cpuinfo
```

For model name, you should see something like "... Intel Xeon E5-26.....". If instead you see "QEMU...", please call a tutor.

You can determine the maximum and base frequency of your CPU model on the Intel Ark website. Because HPL is a demanding workload, assume the CPU is operating at its base frequency. You should have everything you need to calculate the RPeak of your cluster. Typically an efficiency of at least 75% is considered adequate for Intel CPUs (RMax / RPeak > 0.75).

QUESTION 4:
What is the combined theoretical FLOPS (RPeak) of your two 8 core compute nodes?

_____

Part 3 - HPC Challenge

HPC Challenge (or HPCC) is benchmark suite which contains 7 micro-benchmarks used to test various performance aspects of your cluster. HPCC includes HPL which it uses to access FLOPs performance. Having successfully compiled and executed HPL, the process is fairly straight forward to setup HPCC (it uses the same Makefile structure).

1. Download HPCC from https://icl.utk.edu/hpcc/software/index.html

2. Extract then move into the hpl/ subdirectory.

3. Copy and modify the Makefile as your did for Part 1. Also, add '-std=c99' to the 'CCFLAGS' line.

4. Compile HPCC from the base directory using

```
make arch=<arch>
```

5. Edit the hpccinf.txt file (same as HPL.dat)

6. HPCC replies on the input parameter file 'hpccinf.txt' (same as HPL.dat). Run HPCC as you did HPL

7. Download this script with formats the output into a readable format https://tinyurl.com/y65p2vv5

8. Run the script with

```
./format.pl -w -f hpccoutf.txt
```

to see your benchmark result, your HPL score should be similar to your standalone HPL.