PAP laboratory assignments
Lab 3: Performance characteritzation of HPC clusters

Lluc Álvarez and Eduard Ayguadé

Spring 2019-20

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

1 - Objectives

The main objective of this laboratory assignment is to characterize the performance of HPC (High-Performance Computing) clusters. To do so, we will use three well-known benchmarks, Stream, Linpack and HPCG, to measure the performance of Boada for different types of workloads.

In particular, we will use Stream to measure the memory bandwidth of a cluster node and to understand the effects of Non-Uniform Memory Access (NUMA) configurations. Then, we will use Linpack to characterize the peak floating-point performance of the system, scaling the analysis from a single CPU to all the CPUs in two nodes. Finally, we will use HPCG to understand the floating-point performance of the cluster when executing irregular applications, focusing on how the performance scales from one CPU to all the CPUs in two nodes.

This document explains the steps that you should follow to compile, execute and analyze the performance of these benchmarks on Boada.

2 - Stream

Stream is a widely used benchmark to measure the memory bandwidth. This benchmark is written in OpenMP, and consists of four memory-intensive kernels: `Copy`, `Scale`, `Add`, and `Triad`.

1. Copy the file `/scratch/nas/1/pap0/sessions/stream.tar.gz` to your working directory.
2. Uncompress the file using the command `tar -zxvf stream.tar.gz`. This should generate a new directory called `stream`.
3. Enter the `stream` directory and compile the benchmark using the command `make`.
4. Copy the script to submit stream jobs from `/scratch/nas/1/pap0/sessions/submit-stream.sh` to your stream directory.
5. Launch the stream benchmark to the execution queue `execution2` with 1 thread using the command `qsub -l execution2 submit-stream.sh 1`. Note that the script receives one mandatory parameter to specify the number of threads that are used in the execution. Check the output of the execution to understand what the benchmark reports.
6. Do the following experiments and explain the memory bandwidth measured by the four kernels of the Stream benchmark:
   (a) Measure the memory bandwidth using 1, 2, 4, 8 and 16 threads and reason about the results that are obtained.
   (b) Boada uses DDR4-2400 and DDR4-2133 memories. According to the specifications, what is the peak memory bandwidth of these memories? Compare the results you have obtained with the peak memory bandwidth and reason about the differences.
   (c) Use `numactl` to experiment with the memory and thread allocations for the executions of the Stream benchmark. After reading the documentation (`man numactl`), modify the `submit-stream.sh` script to add the `numactl` command with the options `--cpubind` and `--membind`. Perform the following experiments and explain the results.
      - Allocate the memory and the threads in only one NUMA node.
      - Allocate the memory in one NUMA node and the threads in the other NUMA node.
      - Allocate the memory and the threads in the two NUMA nodes.

3 - Linpack

Linpack is the de facto standard benchmark to measure the peak floating-point performance of HPC systems. The benchmark is written in MPI and uses the mathematical libraries provided (and optimized) by the system. We will use Linpack to measure the performance of two nodes of Boada. To do so, you will have to follow a series of instructions to compile the benchmark, understand its parameters, and execute the suggested configurations.

1. Copy the file `/scratch/nas/1/pap0/sessions/hpl-2.3.tar.gz` to your working directory.
2. Uncompress the file using the command `tar -zxvf hpl-2.3.tar.gz`. This should generate a new directory called `hpl-2.3`.
3. Enter the `hpl-2.3` directory and generate the Makefile of the benchmark using the following commands:

   ```
   cd hpl-2.3/setup
   source make_generic
   cd ..
   cp setup/Make.UNKNOWN Make.Boada
   ```
4. Edit the generated Makefile `Make.Boada` to correctly set the following variables. Note that the `TOPdir` variable has to be set to the path of the hpl-2.3 directory, which will depend on your working directory.

   ```
   ARCH = Boada
   TOPdir = ${HOME}/path/to/hpl-2.3
   MPlib = /usr/lib/x86_64-linux-gnu/libmpich.so.12
   CC = mpicc.mpich
   ```
5. Compile the benchmark using the command `make arch=Boada`. This will generate a directory `bin/Boada` that contains a binary and a configuration file. Go to this directory with the command `cd bin/Boada`.
6. Copy the script to submit Linpack jobs from `/scratch/nas/1/pap0/sessions/submit-hpl.sh` to the directory that contains the binary and the configuration file.
7. Launch the Linpack benchmark to the execution queue `execution2` with the command `qsub -pe mpich 1 -l execution2 submit-hpl.sh 1 4`. This command does the following:

   - `qsub -pe mpich 1` launches the job to the execution queues preparing the MPI environment (`mpich`) in 1 node.
   - `-l execution2` specifies the execution queue where the job is sent. Please use `execution2` for all the experiments.
   - `submit-hpl.sh 1 4` specifies the script with the job (`submit-hpl.sh`), the number of nodes (1), and the number of processes (4).

   In general, Linpack uses P processes that can be distributed among N nodes. In order to specify P and N the aforementioned parameters have to be correctly set in the command line when submitting the job. The command line we have used before launches 4 processes in 1 node (P=4 and N=1). In general, the command to launch P processes in N nodes is `qsub -pe mpich N -l execution2 submit-hpl.sh N P`. So, for example, if you want to launch an execution with 8 processes distributed in 2 nodes (P=8 and N=2) you should use the command `qsub -pe mpich 2 -l execution2 submit-hpl.sh 2 8`.

8. The parameters of the execution are specified in the file `HPL.dat`. The format of this file and the meaning of each parameter is explained in another file called `TUNING` in the top directory. Open the file `HPL.dat` and look at the configuration options. The most relevant parameters are:

   - Problem size (`N`): The size of the problem. For our experiments we recommend to use just one problem size of N=8000.

Line of file	Parameter description	Default value	Recommended value
5	# of problems sizes (N)	4	1
6	Ns	29 30 34 35	8000

   - Number of blocks (`NBs`): The block size that each process uses. We recommend to use 4 block sizes, between 32 and 256.

Line of file	Parameter description	Default value	Recommended value
7	# of NBs	4	4
8	NBs	1 2 3 4	32 64 128 256

   - Process grid (`PxQ`): Parameter that specifies how the problem is partitioned among the processes. Given a number of processes N, the problem is partitioned vertically (P) and horizontally (Q). Note that it is mandatory that PxQ=N, otherwise the execution will fail. This is a parameter that needs to be explored, since every experiment will have a different best process grid. For instance, for 8 processes, there are 4 possible process grids: 1x8, 2x4, 4x2 and 8x1. Similarly, for 16 processes, there are 5 possible process grids: 1x16, 2x8, 4x4, 8x2 and 16x1. Every time you launch Linpack with a given number of processes you will have to change the process grid accordingly. To specify the parameters in the configuration file you should change the following lines. Note that the example values below are for 4, 8 and 16 processes, but you will have to come up with different process grids for the other number of processes you will use in your experiments.

Line of file	Parameter description	Default values for 4 processes	Values for 8 processes	Values for 16 processes
10	# of process grids (P x Q)	3	4	5
11	Ps	2 1 4	1 2 4 8	1 2 4 8 16
12	Qs	2 4 1	8 4 2 1	16 8 4 2 1

   - Other parameters: the file contains other parameters related to the algorithm itself. To reduce the execution time of the jobs we recommend you to change the following parameters:

Line of file	Parameter description	Default value	Recommended value
14	# of panel fact	3	1
16	# of recursive stopping criterium	2	1
20	# of recursive panel fact.	3	1

9. Do the following experiments and explain the results:

   - Measure the performance using 1, 2, 4, 6 and 8 processes running in one node. For each number of processes, what is the best configuration (NBs, process grid) that you have found?
   - Measure the performance using 2, 4, 8, 12 and 16 processes running in two nodes. For each number of processes, what is the best configuration (NBs, process grid) that you have found?
   - Plot the scalability, the execution time and the GFLOPS for all the experiments above with one and two nodes. Does the benchmark scale well?
   - Each node of Boada has 2 Intel Xeon E5-2609 v4 processors running at 1.70GHz. Compare the results you have obtained with the peak performance specifications of this processor.

3 - HPCG

HPCG is the de facto standard benchmark to measure the performance of irregular applications in HPC systems. The benchmark is written in MPI and OpenMP and uses the mathematical libraries provided (and optimized) by the system. We will use HPCG to measure the performance of two nodes of Boada. To do so, you will have to follow a series of instructions to compile the benchmark, understand its parameters, and execute the suggested configurations.

1. Copy the file `/scratch/nas/1/pap0/sessions/hpcg-3.1.tar.gz` to your working directory.
2. Uncompress the file using the command `tar -zxvf hpcg-3.1.tar.gz`. This should generate a new directory called `hpcg-3.1`.
3. Enter the `hpcg-3.1` directory and generate the Makefile of the benchmark using the following commands:

   ```
   cd hpcg-3.1
   cp setup/Make.MPI_GCC_OMP setup/Make.Boada
   ```

4. Edit the generated Makefile `setup/Make.Boada` to correctly set the following variables. Note that the `TOPdir` variable has to be set to the path of the hpcg-3.1 directory, which will depend on your working directory.

   ```
   TOPdir = ${HOME}/path/to/hpcg-3.1
   MPlib = /usr/lib/x86_64-linux-gnu/libmpich.so.12
   CXX = mpicxx.mpich
   ```

5. Compile the benchmark using the command `make arch=Boada`. This will generate a directory `bin` that contains a binary and a configuration file. Go to this directory with the command `cd bin`.
6. Copy the script to submit HPCG jobs from `/scratch/nas/1/pap0/sessions/submit-hpcg.sh` to the directory that contains the binary and the configuration file.
7. Launch the HPCG benchmark to the execution queue `execution2` with the command `qsub -pe mpich 1 -l execution2 submit-hpcg.sh 1 8`. This command does the following:

   - `qsub -pe mpich 1` launches the job to the execution queues preparing the MPI environment (`mpich`) in 1 node.
   - `-l execution2` specifies the execution queue where the job is sent. Please use `execution2` for all the experiments.
   - `submit-hpcg.sh 1 8` specifies the script with the job (`submit-hpcg.sh`), the number of processes (1), and the number of threads (8).

   In general, HPCG uses P processes that spawn T threads. Each process is allocated in a dedicated node, and the parallelism inside the node is exploited by the T threads spawned by each process. In order to specify P and T the aforementioned parameters have to be correctly set in the command line when submitting the job. The command line we have used before launches 1 process and 8 threads (P=1 and T=8). In general, the command to launch P processes with T threads is `qsub -pe mpich P -l execution2 submit-hpcg.sh P T`. So, for example, if you want to launch an execution with 2 processes and 4 threads (P=2 and T=4) you should use the command `qsub -pe mpich 2 -l execution2 submit-hpcg.sh 2 4`.

8. The parameters of the execution are specified in the file `hpcg.dat`. The third line of this file specifies the problem size (104x104x104 by default) and the fourth line specifies the time that the benchmark will be executed (60 seconds by default). We will use these default values to measure the GFLOPS Boada obtains running this benchmark. Note that, since the benchmark is stopped after 60 seconds, measuring the execution time of the benchmark is meaningless.

9. The execution of the benchmark generates an output with a summary of the results. The name of the output file has the format `HPCG-Benchmark_3.1_YEAR-MONTH-DAY_HOUR-MINUTE-SECOND.txt`. The output file contains the general results of the execution and, in addition, it also shows the execution time and the GFLOPS of the 4 main kernels that compose the HPCG benchmark: DDOT, WAXPBY, SpMV and MG. Open the output file generated by the execution you have done previously and understand its contents.

10. Do the following experiments and explain the results:

    - Measure the performance using 1 and 2 processes and 1, 2, 4 and 8 threads.
    - Plot the results and explain them. Does the benchmark scale when adding threads? And adding processes?
    - Inspect the output files and explain the performance of the 4 kernels that compose the benchmark. Do they scale? What is their contribution to the total execution time? Given the characteristics of the individual kernels, is it normal the scalability you observe in the whole benchmark? What would Dr. Gene Amdahl think about your results?
    - Compare the performance results of HPCG with the ones you have observed before with Linpack. Are they similar or different? Why?