Evaluation of performance and energy efficiency

December 22th 2022

Summary

CSUC benchmark suite 2022

With the present set of benchmarks, CSUC wants to evaluate the performance of the system that has been put out to tender. With this aim, CSUC has prepared a set of application benchmarks, which must be presented from the tender as described in the "Plec de clàusules administratives particulars (PCAP)". This set of scientific application benchmarks are a collection of jobs using the latest version of some of the most relevant scientific software at CSUC.

This set of benchmarks has been adapted to consider the typology of jobs which have spent most of the computational time during the last years. These sets are representative of the work performed by the scientific community nowadays are using the CSUC facilities. The frame within the work carried out includes several areas of knowledge, like Environmental Sciences, Biomedical and Life Sciences, Material Sciences and Astronomy, among others.

The selected set of programs and jobs is shown in the next table. It includes several kinds of methodologies and basis sets, while using four of the most utilized programs: *Orca*, *Quantum Espresso*, *Gromacs* and *OpenFOAM*.

Application	Version	Job
Orca	5.0.2	nematic
Quantum Espresso	7.0	ausurf
Gromacs CPU	2022	ion_channel
Gromacs GPU	2022	EAG1-channel
OpenFOAM	v2112	motorcycle

These benchmarks will be used by CSUC as a basis to compare and mark the different candidate systems offered by bidders.

The complete set of benchmarks can be obtained from the following link: https://confluence.csuc.cat/display/HPCKB/Benchmarks+2022

General rules

The aim of this series of tests is to compare the performance and power consumption of the different machines presented to the tender. Hence the benchmarks must be done following the instructions contained on this sheet as strictly as possible.

Software

All software should ideally be the same as that which is available in the system when delivered to CSUC.

The bidder should use the following version of the scientific software, which is the same used to. These versions are:

- Orca 5.0.2
- Quantum Espresso 7.0
- Gromacs 2022
- OpenFOAM v2112

Results obtained at CSUC using these versions will serve as a reference; bidders' results must agree numerically with these results.

With the only exception of Orca (which is distributed as a precompiled binary) all applications must be compiled using compilers and libraries that must be available in the system when delivered to CSUC.

Each library, compiler and other software must be identified, including the name, revision and its origin.

Results

The bidder must fill out the file *"Resum_característiques_tècniques_lotX.xlsx"* (where X is the lot number), containing a summary of the most relevant results, which accompanies this document.

CSUC also provides reference values for the results. Values obtained and presented by the candidates should agree within numerical uncertainties; otherwise wall time values won't be evaluated.

The calculations must have been run on a system identical or, at least, as similar as possible to that presented in the offer, and any deviation must be thoroughly documented and justified by the bidders.

The bidder must deliver a *tarball* containing the entire folder structure, including input files, scripts, output files, logs and *tarballs* generated by the scripts. Temporary files created in the temporary scratch directory of the job are not necessary. This *tarball* must be provided to CSUC with the rest of the requested technical documentation.

Evaluation of the benchmarks

In this section we give a detailed description of the methodology to evaluate of the performance of the different benchmarks from the CSUC benchmark suite.

There are two important parts in the benchmarks: the parallel and the wall time contributions. In the next subsections we explain how these two contributions are evaluated.

Evaluation of the wall time

One of the contributions to the score mark of the parallel benchmarks comes from the wall time of the point with the least number of cores, this contribution is evaluated in the following way:

- The bidder with the minimum wall time will obtain the maximum score
- The bidder with the maximum wall time will obtain the minimum score
- The rest of the bidders' scores will be assigned linearly between the two extreme values according to the following formula:

$$WT = \frac{T_{max} - T_{of}}{T_{max} - T_{min}}$$

Where T_{of} is the wall time of the lowest core count result of a given bidder.

For **Gromacs**, it is more convenient to use performance in nanoseconds of simulation per day instead of wall time. The same procedure applies, but with the maximum score being awarded to the highest performance in ns/day, minimum performance obtaining minimum score, etc.

Evaluation of the scalability

It is important for the CSUC to evaluate the scalability of the different performance tests. In order to obtain a single score for each benchmark we have defined a measure of the scalability in the following way:

$$S = 1 - \frac{A_{ideal} - A_{actual}}{A_{ideal}}$$

where

- A_{ideal} is the area under the ideal speed up vs number of cores curve
- A_{actual} is the area under the actual speed up vs number of cores curve

The actual speed up vs number of cores curve can be built from the timing data that must be given to the CSUC by the bidders where we can obtain the speed up of the calculation depending on the number of cores used in the execution of the job. This means that we have a discrete set of values that approximate the real speed up curve. From this set of values we can calculate approximately the area under the speed up curve using the trapezoidal rule, as

$$A_{actual} = \frac{1}{2}\sum_{i=1}^{N}(x_{i+1} - x_i)\big(f(x_{i+1}) + f(x_i)\big)$$

where

- $\{x_i\}$ are the different values of the number of cores used
- $\{f(x_i)\}$ are the value of the speed up of the job executed with x_i cores

The value of A_{ideal} can be evaluated easily considering that the ideal scalability curve for the speed up is a linear function (f(x)=x), so

$$A_{ideal} = \frac{1}{2}(x_N - x_1) * \big(f(x_N) + f(x_1)\big)$$

Considering these definitions one can see trivially that the scalability (S) ranges from 0 to 1 depending on how close to the ideal speed up is the actual one, for the extreme situations we have:

- An application where the speed up in terms of the number of cores follows the ideal curve (f(x) = x), i. e. with a perfect scaling behavior will have an S = 1 given that $A_{ideal} = A_{actual}$
- Alternatively, an application where there is no speed up in terms of the number of cores (f(x) = 1), will have an S = 0 given that $A_{actual} = 0$

Example of scalability evaluation

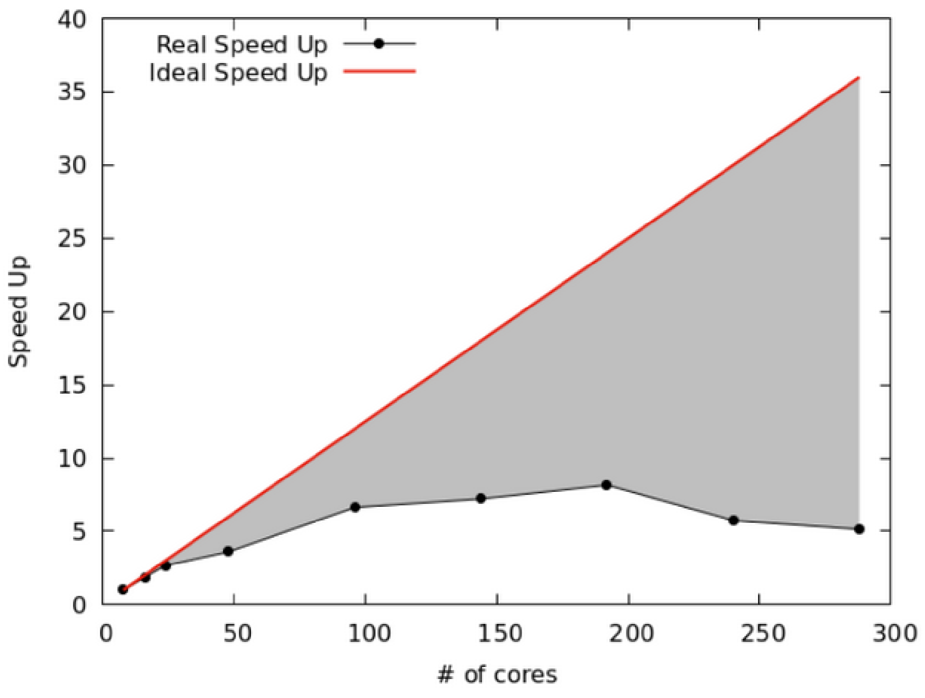
In this section we present a real example of how to evaluate the performance of an application using the methodology presented above. The application analysed is Quantum Espresso 7.0 executed in our current facility.

We have executed Quantum Espresso in a range of 8 to 288 cpu cores and we have obtained the following wall times and speed up in terms of the number of cores:

# of cores	**Wall Time (s)**	**Speed Up**

8	1435	1
16	775	1,85
24	538	2,67
48	399	3,59
96	216	6,64
144	199	7,21
192	176	8,15
240	250	5,74
288	279	5,14

Now, as we have explained before, we must evaluate the area between the ideal and actual speed up curves, which corresponds to the shaded region in the following plot.



Using the definitions given previously we need to evaluate the area under the ideal curve, the approximated area under the actual curve and finally evaluate the scalability according to the equation

If we do this in the present example we obtain an S = 0,32. This value reflects how far is the real scalability of a computation from the idealized situation.

Again, for **Gromacs**, performance in nanoseconds of simulation per day will be used instead of wall time. In this case, speed-up is directly proportional to performance; it is calculated as performance with *n* cores divided by performance with 1 core. Evaluation of scalability from speed-up values is otherwise identical to the other cases.

Parallel benchmark score

In order to determine the final score of each parallel benchmark the following formula will be used:

$$P = \frac{WT + 2 \cdot S}{3}$$

where WT and S are the wall time and scalability contributions respectively.

Evaluation of the energy efficiency

In order to evaluate the energy efficiency of the different machines offered to the CSUC the bidders must measure the energy consumption of the different jobs executed to evaluate the performance of the system, that is the same jobs specified in the previous section. The evaluation of the energy consumption must be done through the smart PDUs included in the offer in order to be able to reproduce the results by the CSUC HPC staff.

At the end the bidders should give to the CSUC a table for each benchmark summarizing the results that should include: the number of cores, the wall time to execute the job and the mean power used by the node(s) to execute the job (power used by all the working nodes). Here you will find an example of a real Quantum Espresso execution:

# of cores	**Wall time (s)**	**Average Power Usage (W)**
8	1435	318
16	775	432

24	538	512
48	399	540
96	216	1064
144	199	1482
192	176	2008
240	250	2521
288	279	3045

Using these data we can evaluate the energy efficiency in a similar way of the scalability. We can plot the curve of the Average power (in W) used by the nodes executing the job vs the number of cores used and calculate the area under the curve for each bidder's offer and assign a value for the energy efficiency according to the formula:

$$EE_{of} = 1 - \frac{A_{of}}{A_{max}}$$

Where

- A_{of} is the area of the offer considered
- A_{max} is the greatest area between all the offers

Benckmarks usage

Quantum Espresso

The first benchmark described here is a usual Quantum Espresso (QE) execution. QE is a density functional theory code using plane waves as function basis. This kind of jobs represent a large percentage of the cpu time employed by the users of CSUC's HPC service (mainly using VASP). The computational requirements and bottlenecks of VASP and QE are quite similar so we have decided to use the Open Source one in order to evaluate the performance of the System.

The chosen QE benchmark is part of the PRACE benchmark suite (https://repository.prace-ri.eu/git/UEABS/ueabs) and corresponds the the "small" test case which is a kind of job that reproduces nicely the typical size (in cores and memory usage) of the jobs executed by our users.

The input files can be downloaded from the following link.

The test should be executed with the number of cores specified in the file csuc_benchmarks_results.xlsx that must be presented by the bidders. It is included with the input files an example with the output of the job. The bidders must present also the output files of the different executions and must verify that the results are numerically correct.

In order to execute the job the bidders should execute a command like the following one:

```
srun -n $cores -N $nodes $path_to_QE_executable/pw.x -nk 2 -i
ausurf.in > ausurf-$cores.out
```

It is important to include the "-nk 2" flag given that the System has 2 k-points and neglecting this can produce poor performance of the test.

OpenFOAM

The OpenFOAM benchmark is a pre-meshed instance of MotorBike example and, basically, makes a 3 step process: Decompose(serial), simpleFOAM(500 step parallel) and Recompose(serial).

Usage:

- Download the benchmark from this link.
- Setup the OpenFOAM environment.
- Go to *concurs_short* folder.
- Set the MPIRUN environment variable with your MPI launch command: ex: " *mpirun –np* " or "*srun -n* ". The default command is "*mpirun -np <ncores>*"
- Launch test with <ncores>:
    a. Shared disk: Run ./*Run_FOAM <ncores>*
    b. Distributed disk:
        i. export *LOCALSCRATCH=/path/to/local_folder/* (same on all nodes)
        ii. Export DIST with the flag to launch 1 task per node (default is openmpi –pernode). Ex (SLURM srun): "*--ntasks-per-node=1*"
        iii. Run ./Run_dist_FOAM <ncores> [LOCALSCRATCH]
- The results will appear into bench_<ncores> folder.

For this benchmark, it is only required to report the timings of the simpleFoam execution in terms of the number of cores used at last step (500) and mean power consumption during the simpleFOAM execution.

ORCA

The ORCA benchmark consists in a MP2 single-point calculation for a liquid crystal molecule, that we'll refer to as *nematic*.

- Install the pre-compiled ORCA binaries in your test machine. To run ORCA in parallel, a MPI library is required. Reference results have been run with OpenMPI 4.1.2; other MPI libraries are acceptable as long as they are either 1) open source or 2) provided in the offer for use in the production cluster.
- Download the input files from this link. You'll find input files to run the benchmark on 1, 2, 4, 8, 12, 16, 24, 32, 40, 48, 64, 96 and 128 cores, plus a generic template called *nematic_XX.inp*. Packed with them you'll find an output file for reference and an example Slurm batch script.
- You'll have to run all the tests that fit in a single node of your machine, plus a whole-node test (prepared by editing the template file) if it's not already included in the provided list. For example, in a node with 72 cores, you'll have to run the benchmark on 1, 2, 4, 8, 12, 16, 24, 32, 40, 48, 60 and 72 cores (the last one done by editing the input template). Note substitution of results for different numbers of cores or interpolation of results are **not acceptable**.
- Set up the environment, making sure the **mpirun** executable is in path.
- Run orca the battery of tests with the command:

  ```
  /full/path/to/orca nematic_X.inp > nematic_X.out
  ```

  with X=1,2,4,8...

- The numerical result we will use to evaluate performance is **total run time** as provided by ORCA. This value can be found at the end of the output file.
- Pack the output files together with any relevant batch scripts and/or additional configuration information and deliver them together with your numerical results.

Gromacs CPU

The Gromacs CPU benchmark is a standard PRACE benchmark, *ion_channel*.

- Compile Gromacs version 2022 in your machine. Reference results have been trun with a compilation using GCC 11.2.0, OpenMPI 4.1.2 and internal FFTW 3.3.8; other compilers, MPI libraries and mathematical libraries are acceptable as long as they are either 1) open source or 2) provided in the offer for use in the production cluster.

  Note: the following Gromacs compilation options are suggested:

```
-DBUILD_SHARED_LIBS=off -DGMX_BUILD_OWN_FFTW=on -
DGMX_EXTERNAL_BLAS=off -DGMX_EXTERNAL_LAPACK=off -DGMX_MPI=on
–DGMX_OPENMP=on
```

This compilation can be used for both Gromacs benchmarks (CPU and GPU) if the following option is added:

```
–DGMX_GPU=CUDA
```

- Download the input file from this link. Packed with it you will find an output file for reference and two example Slurm batch scripts.
- You'll have to run a battery of tests in hybrid mode combining MPI and OpenMP. You'll run the test on the following number of cores: 4, 8, 16, 24, 32, 48, 64, 96, 128. All cases that can fit into a single node have to be run in that way, while larger cases can be run across the minimum number of nodes that provide the necessary number of cores. A fixed configuration of 4 OpenMP threads per MPI process will be used.
- Instead of running the same test and measuring wall time to completion, we will run all tests open-ended for half an hour and we will use Gromacs own performance evaluation (in nanoseconds per day) as the outcome.
- Set up the environment, making sure the **gmx_mpi** and **mpirun** executables are in path.
- Run Gromacs for the battery of tests with the command:

```
OMP_NUM_THREADS=4 mpirun -np X gmx_mpi mdrun -s
ion_channel.tpr -maxh 0.5 -g ion_channel_Y.log
```

  with X=1, 2, 4, 6, 8, 12, 16, 24, Y=4*X.

- The numerical result we will use to evaluate performance is **performance** in ns/day as calculated by Gromacs. This value can be found at the end of the output file.
- Pack the output files together with any relevant batch scripts and/or additional configuration information and deliver them together with your numerical results.

Gromacs GPU

The Gromacs GPU benchmark is strong scaling benchmark by PDC Centre, *eag1_channel*.

- Compile Gromacs version 2022 in your machine. Reference results have been run with a compilation using GCC 11.2.0, OpenMPI 4.1.2, CUDA Toolkit 11.60 and internal FFTW 3.3.8; other compilers, MPI libraries and mathematical libraries are acceptable as long as they are either 1) open source or 2) provided in the offer for use in the production cluster.

  Note: the following Gromacs compilation options are suggested:

```
-DBUILD_SHARED_LIBS=off -DGMX_BUILD_OWN_FFTW=on -
DGMX_EXTERNAL_BLAS=off -DGMX_EXTERNAL_LAPACK=off –DGMX_MPI=on
–DGMX_OPENMP=on –DGMX_GPU=CUDA
```

- Download the input file from this link. Packed with it you'll find an output file for reference and two example Slurm batch scripts.
- You'll have to run the test four times, using 1, 2, 3, 4, 5 and 6 nodes. In each case, you can use the two GPUs and two CPUs in these nodes as you see fit.
- Instead of running the same test and measuring wall time to completion, we will run all tests open-ended for half an hour, and we will use Gromacs own performance evaluation (in nanoseconds per day) as the outcome.
- Set up the environment, making sure the **gmx_mpi** and **mpirun** executables are in path.
- Run Gromacs for the battery of tests with the command:

  ```
  gmx_mpi mdrun -s eag1_channel.tpr -maxh 0.5 -g
  eag1_channel_X.log
  ```

  with X=1, 2, 3, 4, 5, 6. Note that the best parallel configuration will depend on the system, and as such we leave that open. Correct execution may require pinning and/or GPU use to be configured manually.

- The numerical result we will use to evaluate performance is **performance** in ns/day as calculated by Gromacs. This value can be found at the end of the output file.
- Pack the output files together with any relevant batch scripts and/or additional configuration information and deliver them together with your numerical results.