This is the title of the Proposal

Principal Investigator[1], First Collaborator[2], and Last Collaborator[1,2]

[1]Affiliation of the PI
[2]Other affiliation

Abstract

This is the abstract of the proposal: in this document we provide a template for CSCS Production Project Submission with guidelines, focusing on sections **Representative benchmarks and Scaling** and **Performance Analysis** in particular.

Background and Significance

The project proposal should be no longer than **10 A4 pages** including graphs and references, and must contain the following information:

- Abstract
- Background and significance
- Scientific goals and objectives
- Description of the research methods, algorithms, and code
- Parallelization approach, memory requirements
- Representative benchmarks and scaling
- Performance analysis
- Resource justification (annual node-hours and disk space)
  - ⋆ Visualization, pre- and post-processing needs
  - ⋆ Development and debugging requirements
- Project plan: tasks and milestones
- Previous results

Please follow the structure used in this template, which reflects the requirements reported on the page with the general instructions for Production Projects Submission.

Scientific Goals and Objectives

. . .

Research Methods, Algorithms and Code

Please insert in this section a description of the methods and algorithms of the code adopted for your computational study, with a brief list of the main scientific libraries that will be used.

Parallelization Approach and Memory Requirements

Please present in this section the parallel approach employed to address the proposed computational study with the selected code: in general for community codes this information is available on the home page of the code itself. For instance, on the CP2K home page one can read that *CP2K is written in Fortran 2003 and can be run efficiently in parallel using a combination of multi-threading, MPI, and CUDA*. If you don't use a community code, please report if the code employs MPI distributed parallelism or hybrid MPI/OpenMP, which type of MPI communication has been implemented and if the code is taking advantage of shared memory parallelism, GPU accelerators or OpenACC/CUDA.

Representative Benchmarks and Scaling

Please report in this section the results of the mandatory strong scaling tests performed with the selected code on a representative system to be investigated during your research activity. The goal is to choose the most efficient job size to run the performance analysis that will be reported in the next section. You should therefore select meaningful job sizes to simulate the representative system, compatible with reasonably short runtimes: the lowest number of nodes might be determined by memory and wall time constraints, while the highest number of nodes tested should allow you identify the job size at which you reach $\sim 50\%$ of the parallel efficiency with respect to ideal scaling. Weak scaling tests might be provided as well, in addition to the strong scaling data.

The wall time in seconds and the corresponding speed-up of the small example system shown in this template are reported in Table 1, while Figure 1 shows the scaling plot: we start the scaling test on 2 nodes, whose runtime has been used as a reference to compute the speed-up of larger job sizes. We then proceed doubling the number of nodes and checking the corresponding speed-up, until we are sure to have reached the $\sim 50\%$ limit in parallel efficiency (16 nodes in the small example below).

Nodes	Wall time (s)	Speed-up
2	1022	1.00
4	476	2.15
8	339	3.01
16	210	4.87
32	206	4.96

Table 1: Wall time and speed-up

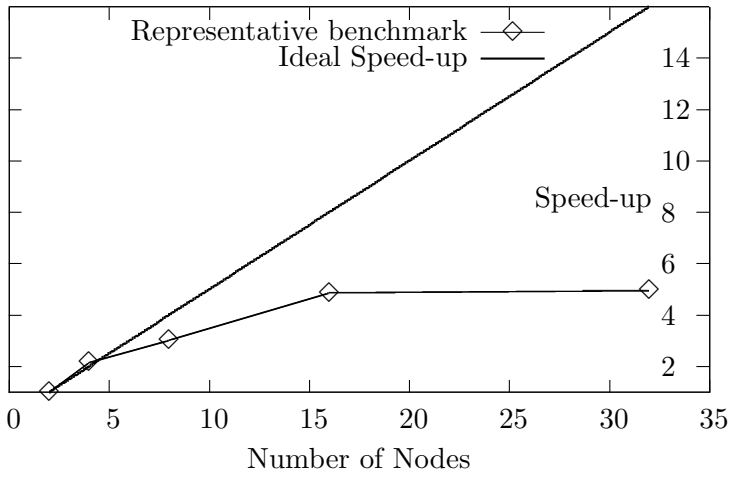

Figure 1: Strong scaling vs. ideal speed-up

Performance Analysis

Please insert within the present section a summary of the performance analysis conducted on the representative system at the optimal job size selected in the previous section, which reached $\sim 50\%$ parallel efficiency. You should have followed the instructions available on the section Performance Report to instrument the executable of the selected code with *Cray Performance and Analysis Tools*. The results of the simulation run with the instrumented executable are the report text file with extension `.rpt`, and the larger apprentice binary file with extension `.ap2`. Please make these two files available for inspection by the reviewers, either by enclosing them at submission time or indicating where they can be accessed for reading under your `$HOME` or `$PROJECT` (not `$SCRATCH`).

The summary data can be extracted using the following commands on the report text file, named `<report>.rpt` in the example:

```
grep -A 14 CrayPat/X <report>.rpt
grep \|USER <report>.rpt
grep \|MPI <report>.rpt
```

The summary should look like the example below:

```
CrayPat/X:  Version 6.4.5 Revision 87dd5b8  01/23/17 15:37:24
Experiment:                    lite  lite/gpu
Number of PEs (MPI ranks):       16
Numbers of PEs per Node:          1  PE on each of  16  Nodes
Numbers of Threads per PE:    1,114
Number of Cores per Socket:      12
Execution start time:  Tue Mar 28 15:15:55 2017
System name and speed:  nid02294  2601 MHz (approx)
Intel haswell CPU  Family:  6  Model: 63  Stepping:  2

Avg Process Time:      2,100 secs
High Memory:        13,977.3 MBytes     873.6 MBytes per PE
I/O Read Rate:     67.110363 MBytes/sec
I/O Write Rate:    19.512511 MBytes/sec

|  59.2% | 1,236.266484 | 110.728787 |  8.8% |          1.0 |USER

|  31.8% |   664.415775 |         -- |    -- |     35,648.0 |MPI_SYNC
|   2.8% |    58.511390 |         -- |    -- | 14,458,788.1 |MPI
```

Resource Justification

The resource request of the annual amount of node-hours should be clearly linked with the node-hours used by the representative benchmark: the number of node-hours consumed by a simulation is computed multiplying the number of nodes by the wall time expressed in hours. Please note that CrayPAT might add a non negligible overhead to the wall time: please report within this section if this happens in your case and then use the wall time of your scaling test to justify the request.

In the small example used throughout this template, the optimal job size of the representative benchmark is 16 nodes and the corresponding wall time reported in Table 1 is 210 s, which correspond to ~ 0.933 node-hours, as a result of the multiplication 16 nodes $\times \frac{210s}{3600\frac{s}{hour}}$. The benchmark is short and represents in general a small number of iterations (timesteps or an equivalent measure), while in a real production simulation we will need to extend it.

Therefore we will need to estimate how many iterations will be needed to complete a simulation in production. Furthermore, the project plan might contain multiples tasks, each of them requiring several sets of simulations to complete: therefore the annual resource request will sum up the corresponding numbers of node-hours obtained multiplying all these factors.

	First task	Second task
Simulations per task	2	4
Iterations per simulation	5000	10000
node-hours per iteration	0.933	0.933
Total node-hours	9333	37333

Table 2: Justification of the resource request

The values reported in Table 2 will sum up to a total of 46666 annual node-hours, resulting from the sum of the node-hours estimated to complete the first and the second task of the project.

You should insert in this section the request for long term storage as well, explaining your needs based on the I/O pattern of the representative benchmark that is reported in the performance analysis.

Visualization, pre- and post-processing

Please insert in this subsection the optional requirements for visualization, pre- and post-processing.

Development and debugging

Please insert in this subsection the optional requirements for development and debugging.

Project Plans: Tasks and Milestones

...

Results from Previous Allocations

Please list allocations requested, granted and used in your previous project (if applicable). You should also include a list of research publications that resulted from these allocations.

References

...