This is the title of the Proposal

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

[1]PI affiliation
[2]Other affiliation

Abstract

This is the abstract of the proposal: in this document we provide a template for sections **Representative benchmarks and Scaling** and **Performance Analysis** with guidelines ...

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 (link: see how to do it)
- Resource justification (annual node hours and disk space)
- Project plans: tasks and milestones
- Visualization, pre- and post-processing needs
- Previous results are mandatory! Please list allocations requested, granted and used in your previous project (if applicable)
- Research publications that resulted from these allocations
- Development and debugging requirements (not mandatory)

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 Method, Algorithms and Code

Please insert within this section a description of the methods and algorithms of the code that you plan to use in your computational study, including a brief list of the main libraries that will be used. E.g.: fftw, parallel-hdf5, petsc, ...

Parallelisation 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 plan to use a community code, please report if the code makes used of MPI distributed parallelism or hybrid MPI/OpenMP, which type of MPI communication has been employed 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 50% limit in parallel efficiency.

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

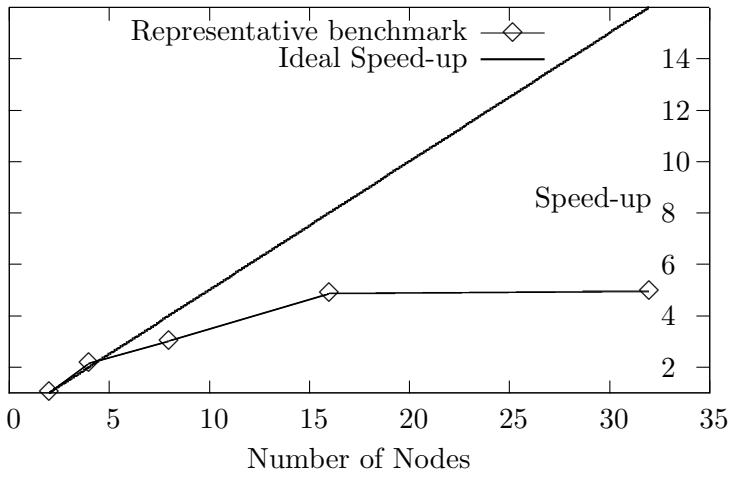

Figure 1: Strong scaling vs. ideal speed-up

Performance Analysis

Please insert within the present section a summary of the performance analysis conducted on the representative system at the optmila 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 of the CSCS User Portal, in order 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
```

Resources Justification

The resource request computation of the required time should start with the node hours needed for a single timestep. As an example, the values in Table 2 sum up to 107500 node hours.

	simulation type 1	simulation type 2
number of simulations	2	3
node hours per timestep	1	1.5
timesteps per simulation	20000	15000
total node hours	40000	67500

Table 2: Derivation of node hours

Project Plans: Tasks and Milestones

...

Visualization, pre- and post-processing

...

Results from Previous Allocations

...

Research publications from Previous Allocations

...

Development and debugging requirements

...

References