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 in particular on sections **Representative benchmarks and Scaling**, **Batch Job Summary Report** and **Resource Justification**.

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
- Research methods, algorithms and code parallelization (including memory requirements)
- Representative benchmarks and scaling
- Batch Job Summary Report
- Resource justification (annual node hours and disk space)
  - ⋆ Visualisation, 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 Production Projects Submission.

Scientific Goals and Objectives

...

Research Methods, Algorithms and Code Parallelization

Please insert in this section a description of the methods and algorithms of all the codes adopted for your computational study, justifying your choices and describing possible alternatives. Furthermore, please specify whether you are the main developer, a contributing developer or a user of the code. You should include a brief list of the main scientific libraries employed and a description of the parallelization approach, with specific memory and I/O requirements as well. In general community codes publish this information on their web sites. For instance, the CP2K home page reports that *CP2K is written in Fortran 2008 and can be run efficiently in parallel using a combination of multi-threading, MPI, and CUDA*. However, please clearly state if you use your own modified version of a community code and briefly describe why you are not using the public release of the software.

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 it makes use of shared memory parallelism, GPU accelerators or OpenACC/CUDA, with specific memory and I/O requirements as well.

Validation, Verification, Uncertainty Quantification, State of the art

Please explain how to validate your method against experiments or other established reference data and verify the numerical consistency of your model, citing the relevant references to peer reviewed papers. Provide parametric sensitivity analysis of your method, with estimates of the uncertainty of your predictions. Data driven uncertainty quantification is encouraged and for multiphysics/multiscale problems, please estimate the uncertainty of the full methods and software. Place the proposed research in the context of competing work, explaining the advantages and drawbacks of your approach.

Representative Benchmarks and Scaling

Please report in this section the results of the mandatory strong scaling tests performed with the selected code: you should report scaling data and plot for every representative system of your project. The goal is to choose the most efficient job sizes to run the simulations in production.

You should select meaningful job sizes to simulate the representative systems, compatible with reasonably short runtimes: the lowest number of nodes is determined in general by memory and wall time constraints, while the highest node counts should let you identify the optimal job size that you plan to use in production simulations and that maximizes the parallel efficiency (ratio of benchmark speed-up vs. linear speed-up), subject to problem-specific constraints (e.g. memory usage or time-to-solution requiring use of more nodes). However, the parallel efficiency must be at least $\sim 50\%$: please do not simply choose the largest job size that results in over 50% parallel efficiency without justification. If possible, please provide weak scaling tests as well, in addition to the required strong scaling data.

Table 1 reports the wall time τ in seconds, the corresponding speed-up σ and parallel efficiency η for a single representative system, with reference to the corresponding JobID. Figure 1 shows the scaling plot: we started the scaling test on a single node, taking this runtime 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 (8 nodes in the small example below).

Nodes	τ (s)	σ	η	JobID
1	763	1.00	1.00	37594624
2	469	1.63	0.82	37594295
4	282	2.71	0.68	37594123
8	182	4.19	0.52	37594007
16	133	5.78	0.36	37580699

Table 1: Time, speed-up, efficiency

Figure 1: Strong scaling of the representative benchmark

The ratio of the speed-up values (benchmark vs. ideal scaling) for each job size gives the corresponding parallel efficiency.

Please select the optimal job size that you plan to use in production simulations, which should also yield a parallel efficiency (ratio of achieved speed-up vs. linear speed-up) above the 50% limit for your representative system.

You are also requested to provide some key parameters of the batch jobs planned on the scratch filesystem during the development of the project, listed below:

- the number of jobs that you plan to run simultaneously
- the wall clock time of your typical submission in hours
- the expected job size of your typical job in nodes
- the memory per node requested by the typical job in GB
- the maximum number of input files read by a job
- the maximum number of output files written by a job
- the largest file size in GB
- the maximum number of files accessed simultaneously during a job
- the library used for I/O
- the maximum number of files on scratch at the same time
- the maximum size of all files on scratch at the same time

The data above will help defining the workflow of your project: if the projects will develop in multiple tasks with different workflows, then you should report the data for each task of the project separately. The I/O patterns are relevant for the scratch filesystem and should be specified explicitly, indicating the maximum number of files accessed simultaneously on the scratch filesystem (Table 2), toghether with the library used for I/O (e.g.: HDF5 / NetCDF / MPIIO / POSIX / ...). Please note that the last two lines of Table 2 refer to the overall sum of file number and size, not to individual jobs.

Table 2 provides a template of how the workflow parameters should be presented in the proposal. In our example, a template project will develop in two tasks that will be described in section **Project Plan**, therefore we provide two sets of workflow parameters in two columns, one for each task of the project. The overall usage of the scratch filesystem is reported for the two tasks together.

	First task	Second task
Number of simultaneous jobs	4	8
Typical wall clock time (hours)	6	12
Typical job size (nodes)	16	16
Memory per node (GB)	12	18
Maximum number of input files in a job	4	6
Maximum number of output files in a job	8	14
Largest file size (GB)	0.5	1.2
Maximum number of files accessed simultaneously in a job	16	16
Library used for I/O	HDF5	HDF5
Maximum number of files on scratch at the same time	200	
Maximum size of all files on scratch at the same time (GB)	680	

Table 2: Workflow parameters of the two tasks of a template project, providing the overall scratch filesystem usage

Batch Job Summary Report

Please report here the batch job summary report of each representative system at the optimal job size selected in the previous section, which should be above the 50% parallel efficiency. The batch job summary report can be found at the bottom of the Slurm output file of a successful job.

The batch job summary report should look like the example below:

```
Batch Job Summary Report (version 21.01.1) for Job "rfm_Cp2kGpuCheck_1_job" (37580699) on daint

Job information (1/3)
-------------------------------------------------------------------------------------------------------
             Submit             Eligible               Start                 End    Elapsed Time limit
------------------- ------------------- ------------------- ------------------- ---------- ----------
2022-04-08T00:58:26 2022-04-08T00:58:27 2022-04-08T06:32:50 2022-04-08T06:35:21   00:02:31   00:10:00
-------------------------------------------------------------------------------------------------------

Job information (2/3)
---------------------------------------------------------------
    Username      Account    Partition   NNodes         Energy
------------ ------------ ------------ -------- -------------
    jenscscs     jenscscs       normal       16    388.110 kJ

Job information (3/3) - GPU utilization data
-----------------------------------------------------
   Node name       Usage      Max mem Execution time
------------ ----------- ------------ --------------
    nid03948        46 %     2031 MiB       00:02:14
    nid03935        48 %     2061 MiB       00:02:14
    nid03938        45 %     2027 MiB       00:02:14
    nid03937        46 %     2027 MiB       00:02:14
    nid03943        45 %     2025 MiB       00:02:14
    nid03947        45 %     2025 MiB       00:02:14
    nid03940        46 %     2017 MiB       00:02:14
    nid03942        45 %     2021 MiB       00:02:14
    nid03945        45 %     2021 MiB       00:02:14
    nid03944        46 %     2027 MiB       00:02:14
    nid03949        45 %     2017 MiB       00:02:14
    nid03946        45 %     2017 MiB       00:02:14
    nid03936        48 %     2075 MiB       00:02:14
    nid03941        45 %     2025 MiB       00:02:14
    nid03939        45 %     2027 MiB       00:02:14
    nid03934        48 %     2067 MiB       00:02:14
```

Resource Justification

The request of the annual amount of node hours should be clearly linked with the node hours used by the representative benchmarks: the number of node hours consumed by a simulation is computed multiplying the number of nodes by the wall time expressed in hours.

In the small example used throughout this template, the optimal job size of the representative benchmark is 8 nodes and the corresponding wall time reported in Table 1 is 182 s, which is equivalent to ~ 0.4044 node hours, as a result of the following product:

$$0.4044 \text{ node hours } = 8 \text{ nodes } \times \frac{182s}{3600\frac{s}{hour}}$$

The benchmark is short and represents in general a small number of iterations (cycles, timesteps or an equivalent measure), while in a real production simulation we will need to extend it.

Therefore we will estimate how many iterations should be necessary to complete a simulation in production. Furthermore, the project plan might contain multiple tasks, each of them requiring several sets of simulations to complete: the annual resource request will sum the corresponding node hours obtained multiplying all the factors reported in Table 3.

	First task	Second task
Simulations per task	2	4
Iterations per simulation	5000	10000
node hours per iteration	0.4044	0.4044
Total node hours	4044	16176

Table 3: Justification of the resource request

The small example above will request a total of 20220 annual node hours, summing the node hours estimated to complete the first and the second task of the project (Table 3), in agreement with the **Project Plan**.

You should present in this section your request for long term storage as well, explaining your needs based on the I/O pattern of the representative benchmarks reported in the proposal.

Visualisation, pre- and post-processing

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

Development and debugging

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

Project Plan: Tasks and Milestones

Please report tasks and milestones of your project. When describing your project development, aside from laying out the tasks that take you from beginning to end, please mark key dates as well. An easy way to do this graphically is through the use of a Gantt chart: milestones charts are also useful to determine more accurately whether or not a project is on schedule.

Results from Previous Allocations

Please list here your past requests, granted projects and used allocations (if applicable). You should also include a list of research publications that resulted from past allocations.

References

...