**Guidelines:**

The project proposal should be 10 A4 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 (if applicable)
* Development and debugging requirements (if applicable)
* Project plan: tasks and milestones
* Previous results

Please follow the structure used in this template, which reflects the requirements reported on Produc-

tion Projects Submission.

**Template**

**This is the title of the Proposal**

Principal Investigator1, First Collaborator2, and Last Collaborator1,2

1Affiliation of the PI

2Other affiliation

**Abstract (Max 200 Words)**

This is the abstract of the proposal: in this document we provide a template for CSCS Pro-

duction Project Submission with guidelines, focusing in particular on sections Representative

benchmarks and Scaling, Batch Job Summary Report and Resource Justification.

***5 pages for “Background and Significance” and “Scientific Goals and Objectives”:***

**Background and Significance**

Describe the proposed research and the main scientific/technical advances you will achieve with the requested allocation. The justification of the requested resources must be clearly linked to the software performance evaluation.

**Scientific Goals and Objectives**

Please extensively describe your scientific case. The scientific case remains the most relevant part of the proposal.

**Research Methods, Algorithms and Code Parallelization (*max 1 page*)**

Please insert in this section a description of the methods and algorithms of all the codes adopted foryour 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](https://www.cp2k.org/) 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 theselected 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 ∼ 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 ∼ 50% limit in parallel efficiency (8 nodes in the small example below).



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.



**Batch Job Summary Report (if available)**

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:



**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:



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.



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 (if applicable)**

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

**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.

**References**

**Must not be references of your own work but a list of relevant references from the literature!**

**Results from Previous Allocations can be added as Annexed**

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.