Final author response for:

"Implementation and scaling of the fully coupled Terrestrial Systems Modeling Platform (TerrSysMP) in a massively parallel supercomputing environment – a case study on JUQUEEN (IBM BlueGene/Q)"

F. Gasper, K. Goergen, S. Kollet, P. Shrestha, M. Sulis, J. Rihani, M. Geimer, and S. Kollet

Dear Editors and Referees,

We would like to thank the Editor, Associate Editor, and the anonymous Referee #1 and #2 for their careful and constructive feedback. In the following *final author response,* we are addressing all comments of the interactive public discussion. All of the suggestions and comments have helped considerably to improve the manuscript and were considered in the revised version.

The Referees' original comments are formatted in blue, while our response is formatted in black font.

Anonymous Referee #1

1 additional information desired for

I)

section 2.4, page 3555, scaling study setup (2): processor-distribution is based on powers of 2 which is said to imply a non optimal usage of allocated compute resources. Further explanation for this fact would be good.

In the manuscript the explanation was revised as follows:

"In this setup, the compute resource allocation is not possible in an optimal sense, since the load distribution between the component models does not follow powers of two."

This also follows from "scaling study setup (1)" (which is based on the distribution-method outlined in 3.2) the load of the models. Thus also the resource allocation is distributed roughly like: 75% / 12.5% / 12.5% which is not representable by power of twos.

II)

section 3.1, page 3557: compiler options guiding the compiler to make use of architecture-specific benefits are mentioned that give a speedup of roughly two. Since this is an impressive number and comes for free (no src code modifications needed), the reader should be informed about the options to reproduce or try for his own code.

We compiled our code with IBMs XL-Compilers and the options:

-O3 -qhot -qarch=qp -qtune=qp

The flags are now included in the manuscript.

Be aware, that only Blue Gene/Q machines might benefit from this compiler options.

III)

sections 3.2 / 3.4: the weak scaling tests were performed for an idealized 24h simulation (cf. figure 5) but the optimal load balancing seems to be determined for a much shorter simulation (cf. timings within figure 3). Why is a short simulation done a-priori representative for a whole day simulation? What if the setup changes over time (e.g. different parametrizations and/or model internal schemes)? What if different physical aspects appear over the time of 24 simulated hours - does this change the optimal load balancing?

The screenshot in Figure 3 is actually only a showcase for the method described in Section 3.2 and profiles only 6 h simulation time (please also note in this context that timings in Figure 3 are LateSender times and do not directly lead to overall timings).

For the actual load balancing we used a profile over 24h but only on the first scaling step.

In the manuscript the explanation was change as follows:

"[…], this method is only precise if the actual setup is traced/profiled. In order to determine the distribution for our test setup 1, 24 hours were traced on scaling step 1. Since we are simulating an idealized test case (flat geometry with homogeneous vegetation) we assumed negligible influence on the load distribution with increasing domain sizes."

IV)

section 3.2, page 3558, last paragraph: the improved load balance was found for a "characteristic test case", how is this case related to the fully coupled weak scaling tests done later on? What does the improved load balancing look like? Is it one of the design described in table 1?

This "characteristic test case" was a real data test case (not idealized but also fully coupled) we used for implementing and porting. This is now made clear in the manuscript.

Improved load balancing means, that the LateSender times were reduced as described in sect. 3.2 and the "scaling study setup (1)" was also determined in this way.

V)

figure 3: the impact of the improved load balancing is not equal for the whole MPMD setup. CLM (program_off) and ParFlow (main) times decrease by a factor of 4, whereas COSMO (lmorg->organize_dynamics and lmorg->organize_physics) times increase. Maybe this should be explained further by comparing the used resources for the experiments ran for figure 3 (which might also explain the topology plot in the third column of the cube view).

The timings in Figure 3 are LateSender (waiting) times and not overall runtimes. The timings (seconds) in Figure3 are also accumulated over the ranks, which is one of the reasons why they increase in COSMO (has more ranks in 3b) and decrease in ParFlow and CLM (has less ranks in 3b).

The impact of the load balancing is equal for the whole MPMD setup and accounts for about 30% less runtime.

As we realize (also from reviewer comments 1.III and 2.I) that Figure 3 causes some confusion, we revised the screenshots and caption.

VI)

in general: file I/O was disabled as far as possible for the scaling tests. It would be good to have at least a brief comment on the changed scaling behaviour if file I/O is turned on. Does I/O for example hinder scaling by introducing additional synchronisation points for all MPI-tasks? Would it be possible to use an adjusted load balance / task mapping to overcome some issues related to file I/O?

It is difficult to say how the scaling behavior would change if file I/O was turned on, since obviously, it depends on the number of output intervals and the amount of variables to be written. In general, I/O, especially parallel I/O, is fast on JUQUEEN (using a General Parallel File System) and also additional synchronization points would not introduce much overhead.

In the manuscript the explanation why file I/O is disabled was revised as follows:

"Note that in the presented scaling study, file I/O is disabled as far as possible. The reason for this is the missing parallel file I/O in some component models and memory limitations in case of large domain sizes."

Parallel file I/O and associated performance monitoring and tuning is beyond the scope of this study and subject of future work.

2 clarifications needed

I)

figure 3: the binaries in the second column of the cube viewer should be matched to the models (lmorg = COSMO, main = ParFlow, program_off = CLM) to see the differences when task distribution is changed

Agreed

3 typos

I)

section 2.4, page 3555, line 18: "... powers of two are also used for the ..."

Agreed

II)

table 1(a), page 3567: #processors for scaling step 1 should be "24x16/8x8/8x8" according to the text

Agreed

III)

table 1(b), page 3567: #processors for scaling step 1 should be "16x16/16x8/16x8" according to the text

Agreed

Anonymous Referee #2

1 Major comments

I)

Abstract: The abstract is, I think, too general. The specific findings of this study should be summarized in the abstract.

Agreed. The abstract has been revised according to the Referee's comments.

II)

p.3555, l.10-13: I suppose that the “minimal wait states” described here corresponds to the “optimized balancing” discussed in section 3.2 and Fig. 3B? If so, the link should be made; if not, then I do not understand how the “minimal wait states” was reached.

This is correct. A link was made.

III)

p.3557, l.16: The mapfile, assigning MPI ranks to actual CPU cores to optimize the communication pattern on the 5-D torus, is mentioned there but its impact, that may be I think extremely important, is not described in the rest of the paper. More detail on how this mapfile was defined, if so, for the weak scaling experiment should be provided.

We did not use a strategy to find a perfect match between the 5D torus and the software-wise domain decomposition or communication pattern, thus we cannot provide details on the impact. Using such a strategy would also result in very long queuing times, since a certain “shape” (characteristic hardware arrangement) has to be requested from the scheduler. Our strategy is to generate the mapfile on-the-fly with an arbitrary shape.

The manuscript was extended by the following information:

“In order to allocate the resources in a performant way, an algorithm is used that first queries the assigned shape and then arranges the resources in a way that an executable is distributed to adjacent nodes. This usually ensures low latencies within the 5D torus interconnect.”

IV)

p.3558, l.5-12: Even if one can understand that in Fig. 3a the load is not ideally balanced while in Fig. 3b, the load is improved, it is not clear to me what is the method or principle to follow to improve the load balance. More details should be provided on this important issue.

The strategy is basically a combination of a profiling/tracing (ideally with LateSender metric) and the knowledge of the communication schematic of the coupled model system and its component models. The manuscript was extended by the following information:

"For example, if Parflow has 30% LateSender waiting time in the corresponding receive call from CLM and CLM is also waiting, it is clear, that COSMO needs about 30% more resources from, for example, Parflow. This might have to be iterated a few times, especially if the speedup saturates."

V)

p.3559, l.2: I suppose that the fact that only 2304 (CLM, ParFlow) and 1152 (COSMO) number of grid cells were reached is linked to the drastic increase of initialization time of CLM described on p.3561. If so, the link between the two sections of the paper should be made; it not, it should be explained why more grid cells were not reached.

While bigger domain sizes are becoming more and more impractical because of the drastically increasing initialization time, this is not the reason for the maximum cell numbers.

In the manuscript the explanation was added as follows:

"A further scaling was not possible at this point because also in the component model CLM 3.5, arrays with global domain size are used. It appears that in newer CLM versions this bottle neck has been removed. Further scaling steps might be possible after a newer CLM version has been implemented into TerrSysMP."

VII)

p.3559, l.8-11: This sentence is not clear at all. I suppose that it means that if the workaround is applied and if a reduced number of processes per node (nnpn) is defined, than this applies to all component models in the coupled system, even the ones not requiring it? This is hard to understand at this point in the text as the workaround is not described yet.

Agreed; as also pointed out in your comment 2.II, the effect of nppn and the workaround needs to be described earlier in the text and was modified accordingly in the manuscript.

VIII)

p.3573, Fig.5: the legend on the right of the Fig. 5 is very hard to read.

We increased the font size and made sure that everything is readable after the final typesetting.

2 Minor comments

I)

p.3551, l.19: I do not understand the sentence "remain independent supporting interchangeable executables as a major advantage"; even if the OASIS approach allows to change the different executables of a coupled system, it of course does not guarantee that two executables are interchangeable.

In the manuscript this sentence was changed to:

“Therefore component models remain independent which allows for interchangeable executables as a major advantage”

II)

p.3552, l.22: Writing that “each MPI-process can only access 1GB” is a bit mis-leading as there exists a workaround based on the number of processes per node (nppn) is detailed later on p.3559.

Agreed. This has been clarified.

III)

p.3555, l.26: “T” and “nbn” appearing in formula (1) are not explained.

This has been fixed in the manuscript.