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

F. Gasper[1,2], K. Goergen[2,3,4], P. Shrestha[3], M. Sulis[3], J. Rihani[3], M. Geimer[4], and S. Kollet[1,2]

[1]Agrosphere (IBG-3), Forschungszentrum Jülich GmbH, Jülich, Germany

[2]Centre for High-Performance Scientific Computing in Terrestrial Systems (HPSC TerrSys), ABC/J Geoverbund, Germany

[3]Meteorological Institute, University of Bonn, Bonn, Germany

[4]Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Jülich, Germany

Keywords: coupled terrestrial modeling, high-performance computing, parallel scaling and performance, profiling and tracing, code optimization

Abstract

Continental-scale hyper-resolution simulations constitute a grand challenge in characterizing non-linear feedbacks of states and fluxes of the coupled water, energy, and biogeochemical cycles of terrestrial systems. Tackling this challenge requires advanced coupling and supercomputing technologies for earth system models that are discussed in this study, utilizing the example of the implementation of the newly developed Terrestrial Systems Modeling Platform (TerrSysMP) on JUQUEEN (IBM Blue Gene/Q) of the Jülich Supercomputing Centre, Germany. The applied coupling strategies rely on the Multiple Program Multiple Data (MPMD) paradigm using the OASIS suite of external couplers, and require memory and load balancing considerations in the exchange of the coupling fields between different component models and the allocation of computational resources, respectively. Using the advanced profiling and tracing tool Scalasca to determine an optimum load balancing leads to a 19% speedup. In massively parallel supercomputer environments, the coupler OASIS-MCT is recommended, which resolves memory limitations that may be significant in case of very large computational domains and exchange fields. However, model I/O and initialization in the peta-scale range require still major attention, as they constitute true big data challenges in the light of future exa-scale compute resources. Based on a factor-two speedup due to compiler optimizations, a refactored coupling interface using OASIS-MCT and an optimum load balancing, the problem size in a weak scaling study can be increased by a factor of 64 from 512 to 32768 processes while maintaining parallel efficiencies above 80% for the component models.

1 Introduction

In studies of the terrestrial hydrologic, energy and biogeochemical cycles, integrated multi-physics simulation platforms take a central role in characterizing non-linear interactions, variances and uncertainties of system states and fluxes in reciprocity with observations. Recently developed integrated simulation platforms attempt to honor the complexity of the terrestrial system across multiple time and space scales from the deeper subsurface including groundwater dynamics into the atmosphere (Anyah et al., 2008 ; Fersch et al., 2013 ; Keyes et al., 2013 ; Maxwell et al., 2007 ; Maxwell et al., 2011 ; Shrestha et al., 2014). Technically, the application of these new generations of terrestrial modeling systems over regional climate-scale or micro-scale (e.g. large eddy simulation) requires porting of the system to supercomputing environments, while ensuring ideally a high degree of efficiency in the utilization of, for example, standard Linux clusters and massively parallel resources alike. With such complex applications, a systematic scaling study and performance analysis including profiling and tracing is crucial for understanding the runtime behavior, to identify optimal model settings, and an efficient identification of bottlenecks in the program's parallelism. On sophisticated leadership-class supercomputers, such as the 28-rack 5.0 Petaflops (Linpack performance) IBM Blue Gene/Q JUQUEEN of the Jülich Supercomputing Centre (JSC) (Germany) used in this study, this is a challenging task, in particular, when a coupled model system consisting of an external coupler integrated with different component models is to be analyzed.

There exist a number of studies dealing with the detailed strong and weak scaling behavior of various simulation platforms in hydrology and reactive solute transport, such as Hammond et al. (2014) ; Kollet et al. (2010) ; Mills et al. (2007). In these studies the focus has been placed on the parallel efficiency of solution algorithms including preconditioners for various classes and systems of partial differential equations in global implicit and explicit solution approaches. In the presented study, the focus is shifted from the analysis of parallel solver and preconditioner performance toward the challenges and parallel efficiency of coupling different component models externally as part of the development of (regional) earth system models.

58 The challenges and intricacies of coupling technologies of earth system models were reviewed by
59 Valcke et al. (2012), who focused on the central features of different established systems consisting of
60 data transfers, re-gridding, time step management, and parallel efficiency. Prominent examples of
61 coupled modeling systems are the Community Climate System Model, CCSM (Gent, 2006), and the
62 Earth System Modeling Framework, ESMF (Hill et al., 2006), which have also been shown to scale to
63 processor numbers on the order of 10^4. As a matter of fact Dennis et al. (2007) explicitly discuss the
64 application of ultra high-resolution CCSM on the Blue Gene platform and the required preparations
65 with regard to ,for example, memory allocations and parallel I/O due to this unique supercomputer
66 architecture.

67 The need for high- or hyper-resolution coupled simulations of the terrestrial system originates from the
68 multi-scale, non-linear processes and feedbacks of the water, energy, and biogeochemical cycles in
69 and between the subsurface, land surface, and atmosphere (Wood et al., 2011). As a matter of fact,
70 *ab initio* simulations would require spatial resolutions in the sub-millimeter and sub-second ranges, in
71 order to resolve ,for example, non-local reactive transport process in porous media (Yang et al., 2013)
72 and turbulent exchange between the land surface and the atmosphere (Shao et al., 2013). Additionally,
73 heterogeneity of the terrestrial system exists at all spatial scales resulting in variances and residence
74 time distributions of system's states and fluxes spanning orders of magnitude (Kirchner et al., 2000).
75 Thus, resolving all pertinent processes at their respective support scales and adequately honoring
76 cross-scale heterogeneity of the terrestrial system constitutes a grand challenge that may be tackled
77 by efficiently utilizing massively parallel supercomputing environments (Kollet et al., 2010).

78 The issue that subsurface hydrologic models usually run on a relatively small scale with high
79 resolution, while atmospheric models operate on a very big/continental scale, leads to unsolved
80 questions regarding the coupling of those models. A solution by upscaling the hydrology model to a
81 continental scale lacks adequate scaling laws for the continuity equations of variably saturated
82 subsurface flow (e.g., Richards' equation). Also, the downscaling of the atmospheric model to a
83 regional scale remains challenging due to the representation of turbulence and the lower boundary
84 condition in atmospheric models, that is, the land surface. A straightforward way to combine both

models in a soil-vegetation-atmosphere system is to increase the size of the hydrology model to a continental scale, but leaving the resolution high. This requires computational resources only massively parallel supercomputers like JSC's JUQUEEN can provide.

In this study, we present our experiences from porting, tuning, and scaling the parallel Terrestrial Systems Modeling Platform (TerrSysMP) (Shrestha et al., 2014) from commodity Linux clusters to the massively parallel supercomputing environment JUQUEEN, the IBM Blue Gene/Q system of JSC. We aim at addressing and highlighting general technical aspects that have to be considered in designing, porting, or refactoring fully coupled geoscience models to highly scalable High Performance Computing (HPC) architectures. The study also demonstrates how an optimal resource allocation may be achieved for such a complex modeling system with heterogeneous computing loads between the different component models, and gives an example for a weak scaling study of the highly scalable model system TerrSysMP.

2 TerrSysMP, compute environment, and experiment design

In this section, the modeling platform consisting of the different component models and coupling technologies is introduced, followed by a description of the hardware characteristics of the JUQUEEN (IBM Blue Gene/Q) supercomputer environment used in this study. The modeling platform was instrumented with performance analysis tools, which are also outlined here. The design of the numerical experiments for the ensuing scaling, profiling, and tracing analyses is detailed, including remarks on an ad-hoc *a priori* load balancing of the different component models.

2.1 The Terrestrial Systems Modeling Platform, TerrSysMP

The parallel Terrestrial Systems Modeling Platform (v1.0) consists of the numerical weather prediction system (COSMO, v4.11) of the German Weather Service (Baldauf et al., 2011), the Community Land Model (CLM, v3.5) (Oleson et al., 2008), and the variably saturated surface-subsurface flow code ParFlow (v3.1) (Jones and Woodward, 2001 ; Kollet and Maxwell, 2006). For details with regard to the different component models, the reader is referred to the aforementioned publications. In TerrSysMP,

110 these component models were integrated in a scale consistent way conserving moisture and energy
111 from the subsurface across the land surface into the atmosphere (Fig. 1). The interested reader is
112 referred to Shrestha et al. (2014) for a detailed description of the modeling system. Each component
113 model is itself parallel and has been demonstrated to scale efficiently to a large number of parallel
114 tasks (e.g.,Kollet et al., 2010).

115 *Figure 1*

116 In order to couple differently structured component models to simulate complex systems, it is
117 necessary to match a specified interface to exchange fluxes and states. Tailoring this interface
118 exclusively for a certain model environment does not provide the flexibility and compatibility that is
119 needed for various scientific modeling platforms. The obvious solution is a coupling strategy that
120 abstracts that interface via synchronous data-exchange, time step management, grid-transformation
121 and interpolation methods, and I/O with a low cost and strong stability on different computing
122 environments.

123 In TerrSysMP, the interface abstraction relies on the Multiple Program Multiple Data (MPMD)
124 execution model, which forms the basis of the external Ocean-Atmosphere-Sea-Ice-Soil coupler,
125 OASIS (Valcke, 2013). With the MPMD functionality, which is offered by most MPI-implementations, it
126 is possible to run several executables within the same global MPI_COMM_WORLD communicator.
127 This functionality affords a coupler that has an external "view" of all component models reflecting the
128 key requirement of high modularity and is especially useful in coupling of component models with fast
129 development cycles and heterogeneous computation loads (Chang et al., 1997). The implementation
130 of the coupler is almost non-invasive. Therefore component models remain independent which allows
131 for interchangeable executables as a major advantage. Thus, OASIS links the aforementioned
132 component models as independent executables, and can be implemented in two different versions,
133 OASIS3 and OASIS3-MCT (OASIS3 including the Model Coupling Toolkit libraries). In case of
134 OASIS3, the coupler is implemented as an additional independent executable, while in case of
135 OASIS3-MCT, the coupler is attached to each individual component model as a library. The impact of

coupling with OASIS-3 or OASIS3-MCT in massively parallel computer environments is discussed in detail in the sections below.

It is important to note that coupling independent executables based on the MPMD paradigm may confront the developer and user with basic technical drawbacks that need to be considered in the initial design of the modeling platform. For example, the MPMD-functionality might not be available or well supported on every machine, especially in case of customized MPI implementations. Additionally, the assigned computational resources, that is, the number of parallel tasks per component executable, are fixed at run time and, thus, load balancing between them has to be performed a priori. Moreover, component models with relatively small computational load, even after load balancing, are constantly blocking resources and use up allocated core-hours that cannot be made available to other users.

2.2 Characteristics of JUQUEEN Blue Gene/Q

JUQUEEN is an IBM BlueGene/Q system with 458,752 cores and 448 TB main memory with a Linpack performance of 5.0 Petaflops. This makes JUQEEN currently (Nov. 2013) the 8th fastest supercomputer in the world (Top500.org, 2013).

Supercomputers like JUQUEEN have very special characteristics. Most remarkable is the trade-off in clock rate (1.6 GHz) for lower power/cooling requirements and improved system's reliability. This trade-off is compensated by the large number of cores and also the 4-way simultaneous multithreading (SMT) of the 64Bit PowerPC A2 processors. The IBM BlueGene/Q architecture is based on *nodes* which contain one CPU with 16 cores and 16 GB main memory. 32 of those nodes are assembled in one (water cooled) *nodeboard*, which is also the smallest allocation unit for jobs. One *rack* consists of 8 I/O nodes and 2 *midplanes* containing 16 nodeboards each. Compared to standard Linux clusters, the IBM BlueGene/Q series is an architecture with very low memory per core. The 16 GB RAM per node are distributed to 16 (64 with SMT4) cores and have a static mapping. Thus, each MPI-process can only access 1 GB (256 MB with SMT4). While there is a workaround to enable more memory per core, described later in the text, this is the most challenging constraint and discussed in following sections.

An important feature of BlueGene/Q is the very fast interconnect, which links all nodes via a 5D torus (electrical signaling within a midplane, optical signaling beyond midplanes). The 512 nodes of a midplane are connected in a 4x4x4x4x2 configuration and allow for a very high peak bandwidth (40 GB/s per node). The mapping of requested hardware allocations is left to the LoadLeveler job scheduling system, which generally prioritizes large jobs (with maximum wall clock time), but smaller jobs can be placed in the gaps. The mapping to the 5D torus can be a critical task for communication intensive programs, however, requesting a certain configuration (*shape*) can result in increased queuing times.

2.3 Performance analysis

As a profiling and tracing tool for analyzing the runtime behavior of TerrSysMP, to identify performance bottlenecks and determine the optimum (static) load balancing, that is, resources allocation, for each experiment setup, Scalasca 1.4.3 was used. Scalasca (Geimer et al., 2012) is a portable open-source toolset which can be used to analyze the performance behavior of parallel applications written in C, C++ and Fortran which are based on the parallel programming interfaces MPI and/or OpenMP. It has been specifically designed for use on large-scale HPC systems such as the IBM Blue Gene series, but is also well-suited for small- and medium-scale systems. Scalasca supports an incremental performance-analysis procedure, combining runtime summaries (profiles) suitable to obtain a performance overview with in-depth studies of concurrent behavior via event tracing. A distinctive feature of Scalasca is its scalable automatic trace analysis (Geimer et al., 2010), which scans event traces of parallel applications for wait states that occur, for example, as the result of unevenly distributed workloads. Such wait states can present major obstacles to achieving good performance.

The typical Scalasca workflow is as follows: Before any performance data can be collected, the target application is instrumented, that is, probes are inserted into the application to intercept important events. Scalasca supports various ways to accomplish this task, for example, using automatic compiler-based instrumentation, library interposition, or via source-to-source transformation. At

runtime, these probes trigger the collection of performance events to – by default – generate a profile measurement providing a performance overview. Based on the initial profile results, the measurement configuration can be optimized to reduce measurement perturbation, for example, by filtering small but frequently executed functions. In-depth analyses of the performance behavior can then be performed by collecting and automatically analyzing event traces, which allow to distinguish between wait states and actual communication or synchronization time as well as to determine their root causes and activities on the critical path (Böhme et al., 2010 ; Böhme et al., 2012).

To obtain information about the allocated memory, only an interface provided by IBM can be used (#include <spi/include/kernel/memory.h>). This is due to the fact that the compute-nodes of JUQUEEN use a specific compute node kernel with reduced functionality that does not offer generic memory interfaces making the use of conventional memory tools impossible.

2.4 Scaling study experimental design

To identify scalability and performance limitations of TerrSysMP when going to very large model domains either by increasing the spatial resolution or expanding the model domain to for example, continental scales, a weak scaling study with an idealized test case was developed. In the scaling study, the two-dimensional horizontal extent of the model domain (*nx*, *ny*) was increased by a factor of 4 for each scaling step (doubling every dimension). The number of cells in vertical dimension, *nz*, remained constant for every scaling step with ParFlow *nz*=30, CLM *nz*=10, and COSMO *nz*=40. All models use a two-dimensional processor topology and in the first scaling step, one Blue Gene/Q nodeboard with 32 nodes and 512 physical CPU cores was used. The allocated resources are doubled in each dimension as well and, thus, the patch-size (grid-cells per task) for every MPI rank remains constant throughout the scaling experiment.

Time stepping remains constant across all scaling steps and is based on the physical processes simulated and applied solution algorithms of the different component models. In the atmospheric model COSMO, the time step size, Δt, is strongly determined by the spatial discretization and was fixed at Δt=10 s. Time integration of the relevant exchange fluxes with the land surface and subsurface

model CLM and ParFlow is performed by OASIS over a 900 s interval, which simultaneously constitutes the constant time step size of CLM and ParFlow. 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.

The scaling study is performed with two different setups in terms of grid size and processor allocation (Table 1).

Table 1

1) In the first setup, a grid size, n, is used that is closely related to real-data test cases used by Shrestha et al. (2014) for development and testing of TerrSysMP. The initial scaling step consists of nx=ny=288 grid-cells for CLM and ParFlow with a lateral spatial discretization of $\Delta x=\Delta y=0.5\ km$ and nx=ny=144 for COSMO with a lateral spatial discretization of $\Delta x=\Delta y$=1 km. An optimal hardware distribution was used, which was predicted with profiles from the analysis tool Scalasca and the method described in Section 3.2. The profiling showed minimal wait states (critical path) with a processor allocation (starting with one nodeboard/512 MPI-ranks) of 8x8=64 for CLM and ParFlow and 24x16=384 for COSMO. This results in patch sizes of (288x288x30)/64=38880 grid-cells for ParFlow, (288x288x10)/64=12960 for CLM and (144x144x40)/384=2160 for COSMO.

2) In the second scaling setup, the grid sizes, n, and number of processors, np, are expressed as a power of two to provide a more standardized experiment for better comparability. 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. The first step has grid sizes of 256x256 for ParFlow and CLM and 128x128 for COSMO. The 512 MPI ranks (one nodeboard) are distributed as: 16x8=128 for ParFlow and CLM and 16x16=256 for COSMO. This results in patch sizes of (256x256x30)/128=15360 grid-cells for ParFlow, (256x256x10)/128=5120 for CLM and (128x128x40)/256=2560 for COSMO.

In both setups, the parallel efficiency E [%] in our study is defined as:

$$E(n, nb) = \frac{T(n,1)}{T(nb \cdot n, nb)} * 100 \quad (1)$$

Where *T* is time, *n* is the problem size and *nb* is the number of nodeboards. Thus, in case of perfect parallel scaling and efficiency ,that is, zero communication overhead, the simulation platform would exhibit an efficiency value of *E*=100%.

3 Results

In this section, the implementation and building process of TerrSysMP is described, followed by an introduction of an *ad hoc* load balancing approach for MPMD programs with the usage of performance analysis tools. The execution of the designed scaling study and the reason why first attempts failed due to memory restrictions are also presented in this section. This is followed by the advancements with the new OASIS version with results and discussion.

3.1 TerrSysMP implementation

For coupled systems with independently developed model codes, it is unlikely that all components are initially ready and efficient for various computing sites, compilers and libraries. In order to reach an optimum single-node and component-model performance, TerrSysMP was initially ported to use IBM XL compilers that may produce executables with the most efficient hardware utilization. To improve the usability of the complete model system, which is developed in a standard Linux cluster environment, fully automatized script-based install procedures allow for a very efficient and fast application deployment. The most current release version of the TerrSysMP system is retrieved from a master GIT repository and adjusted for the build environment for the machine, in our case JUQUEEN, that is, little/big-endianness, library-paths and data-structures, similar to the GNU *autoconf* software configuration package. Optional/experimental features (e.g., OASIS3-MCT, etc.) are also available for integration during this procedure. In a second step, the complete model system is built and the run-time environment (model settings, forcing data and job-scripts, etc.) is set up. In order to preserve portability and legacy code, TerrSysMP does not make use of hardware intrinsics or interfaces to IBM-APIs (i.e., L1P prefetcher, atomic operations, etc.). However, there are compiler options, which guide

the compiler to make use of architecture-specific benefits and help with constraints, in our case: *-O3 -qhot -qarch=qp -qtune=qp*. The usage of these options enables a speedup of roughly a factor of two for TerrSysMP. To allow for easy regression testing during model development and for first-time users familiarizing themselves with the system, forcing data and model settings for well-defined real-data and idealized test cases as well as reference results are provided.

3.2 Optimum resource allocations for MPMD

As already briefly mentioned in the explanation of TerrSysMP's coupling scheme in Section 2.1, in most MPMD implementations, the resource allocation or association of hardware nodes to a certain application is fixed during runtime. Usually, in many MPI implementations a different number of executables is started through the invocation of the MPI parallel job launcher; processes are then mapped onto the computational resources allocated by the job scheduler. On IBM Blue Gene/Q, a mapfile has to be used in conjunction with MPMD to explicitly assign MPI ranks to the actual CPU cores. This mapfile may either be set up before job submission to optimize the communication pattern on the 5D torus network topology of the BG/Q, or the resources are assigned automatically by the scheduler. The latter was used to define the mapfiles. 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.

This setup combined with CPU affinity means that a load balancing between the component models during runtime is not possible and assigned resources are fixed. Thus, no dynamic load-balancing algorithms are applicable. Since simulations may run for several hours, unbalanced resource assignments have a strong impact on the parallel efficiency. Therefore, determining an approximate load for every component model and applying a static load balancing in advance is a necessary condition for an efficient utilization of resources. For TerrSysMP, using a profiling tool (on JUQUEEN for example Scalasca) in conjunction with a graphical tool to visualize the profile (here CUBE-QT (Song and Wolf, 2004)), provides a complete picture of the time spent within the individual models and

routines. With detailed knowledge of the synchronization and communication-structure (Fig. 2) of the coupled system (or a critical-path analysis available in the newest Scalasca implementation), one can identify which models are waiting for completion of others and, thus, are under- or overloaded. 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.

Figure 2

Figure 3 is a showcase for this workflow and shows two CUBE-QT screenshots of the fully coupled TerrSysMP. In Fig. 3a, the load is not ideally balanced and the topology view (right) shows more cores with higher load in the relevant functions than in the optimized balancing of Fig. 3b. In both screenshots, the metric *Late Sender* was chosen and, thus, the displayed (accumulated) timings are equivalent to this particular wait state (receiver waits for sender).

Figure 3

With this complete picture of TerrSysMP, it was possible to determine an improved load balance for the test setup 1 in Section 2.4 and also characteristic real data test cases reacting positively to this approach. For example, compared to established balancing methodologies based on component intrinsic timing routines a 19% speedup was reached in this example. However, 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 in 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.

3.3 Advanced coupling interface

Nowadays, parallel scientific software applications are targeted mostly at architectures such as commodity Linux clusters with fast interconnects, which are used regularly without major problems. However, utilizing massively parallel supercomputers requires different approaches, not only because

315 of the architecture, but also because of complicated communication patterns, data-structures and
316 distinct optimization that may be possible or necessary. The individual component models, which are
317 used in TerrSysMP, are well tested at many different supercomputing sites, but coupling them
318 especially with a highly resolved hydrologic model based on an external coupler adds an additional
319 level of complexity.

320 TerrSysMP was first developed for a standard Linux cluster and then ported to JSC's IBM Blue
321 Gene/Q supercomputer JUQUEEN. A comparably small reference test case scaled reasonably well.
322 However, in order to use TerrSysMP as a model for large-scale, hyper-resolution simulations, the
323 applicability for much bigger domain sizes had to be explored. Scaling studies as described in Section
324 2.4 with resolutions from *nx*=*ny*=288 (CLM, ParFlow) / *nx*=*ny*=144 (COSMO) ideally up to *nx*=*ny*=9216
325 (CLM, ParFlow) / *nx*=*ny*=4608 (COSMO) were planned, while *nx*=*ny*=2304 (CLM, ParFlow) /
326 *nx*=*ny*=1152 (COSMO) were actually reached.

327 During initial scaling tests, an increase in problem size by a factor of four in the second scaling step
328 led to stalled simulations due to insufficient main memory. In contrast to most standard Linux clusters,
329 the IBM Blue Gene/Q uses a static memory map, which means that the nodes' memory is equally
330 distributed across the processes running on that node in MPI parallel setup (see also Section 2.2).This
331 configuration is fixed and cannot change during a simulation. Since the standalone external coupler
332 OASIS3 is only running with a single process, it can only use 1/16th of the RAM of an individual node
333 if all 16 CPU cores per node are to be used, which results in 1 GB using OASIS3 as the coupler,
334 although the rest of the node is unused (only one and the same executable may run on an individual
335 node). A workaround for enabling more memory to one CPU is to reduce the number of processes per
336 node (*nppn*), with the side effect, that this configuration obviously decreases the parallel efficiency of
337 the modeling system, especially because this process count also applies to all CPUs and thus, also to
338 all other component models.. For OASIS3, reducing *nppn* to 4 and using only 1/4th of the nodes CPUs
339 results in 4GB of RAM which are available per process. Thus, for applications with large memory
340 requirements, such as TerrSysMP, the resource usage when coupling with OASIS3 may be inefficient
341 in non-standard supercomputer environments.

342 Investigating the memory problems further with JUQUEEN's memory-tracking interface, which
343 provides information on the actually allocated amount of memory, showed that in each coupling time
344 step, OASIS3 receives several arrays from each sending process of a certain component model. It
345 then repartitions all these local parts from the domain decomposition of each individual component
346 model into the full domain. In subsequent steps, re-gridding and also weighting algorithms are
347 performed. Then, the global domain is partitioned again into local parts and sent forward to the
348 receiving component model processes. The aforementioned memory transgression occurred due to
349 the use of arrays with the size of the complete model domain. This usually does not pose problems for
350 smaller domain sizes, especially on general-purpose Linux clusters, which usually provide more than
351 2 GB RAM per core including dynamic memory allocations. However, on JUQUEEN the allocation of
352 global domain sizes prohibits an extensive weak scaling. For example, if one would need to use just
353 one of JUQUEEN's racks, each process is allowed to store only 8192 double values as a local
354 partition in order to enable one node to gather a global domain. This limitation of the single-threaded
355 concept of OASIS3 indicates that it is (at least with regard to massively parallel supercomputers) only
356 applicable to medium grid sizes and processor counts.

357 In September 2012 CNRS/CERFACS released a new version of OASIS, namely OASIS3-MCT (since
358 May 2013 OASIS3-MCT_2.0), which now relies on the Model Coupling Toolkit, MCT (Larson et al.,
359 2005). In the new version, OASIS is not a standalone coupler, but a library that is included in the
360 different component models. The actual interface basically remains the same, which makes porting to
361 this new version straightforward. Implementing the coupling within a library leads to a parallel OASIS,
362 since the library is part of each process, which overcomes computational as well as bandwidth
363 bottlenecks. But most importantly, each process can send its data to the targeted processes without
364 the need for repartitioning a global array. This renders the coupling thinner and consumes only few
365 extra resources. Figure 4 shows an illustration of the coupling with a) OASIS3 and b) OASIS3-MCT.
366 With this newly designed coupling interface, scaling to very large model domains is possible.

367 *Figure 4*

3.4 Weak scaling study

By using OASIS3-MCT, the model system allows for domain sizes up to a resolution of *nx*=*ny*=2304 (CLM, ParFlow) and *nx*=*ny*=1152 (COSMO) grid points, which constitutes an increase in the problem size by a factor of 64 as compared to the unit reference test cases applying the original OASIS3 coupling. A further scaling was not possible at this point because also in the component model CLM3.5, 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.

The scaling plot (Fig. 5a) of setup design 1 (Table 1a) shows that the dynamic model kernels, here called *driver* routines, scale well, which is essential for extended hyper-resolution runs in the context of large-scale integrated terrestrial simulations. CLM has a parallel efficiency of almost 100% (98% in the largest run) due to its 1D isolated column physics with no communication overhead. The driver takes only a couple of seconds even in the larger runs. The COSMO driver has a parallel efficiency of slightly above 92% (largest run; see dotted lines in Fig. 5a for driver efficiencies), but is the component with the heaviest compute load, therefore dictating the total calculation time. The ParFlow driver scales less well with about 82% parallel efficiency (largest run).

Figure 5 shows which bottlenecks eventually arise in the larger scaling steps preventing the coupled system from efficient scaling. The initialization time of CLM increases drastically with each step. An analysis of the code revealed that during initialization, the load-balancing algorithm is redundantly done by every rank and dependent on the global grid size *n* and the number of processors *np*. Since both grow with a factor of 4 between each scaling step, the initialization time in theory increases with a factor of 16. The actual increase of the initialization time is a factor of 14.41 between the last two steps. The scaling plot (Fig. 5b) of setup design 2 (Table 1b) shows a similar behavior. Only ParFlow shows a decrease in parallel efficiency (68% in the largest run), which indicates a higher sensitivity to communication with a larger number of MPI ranks (Kollet et al., 2010). Additionally, the initialization time determined by CLM is higher because of the larger number of CLM ranks. The overall calculation

394 time is slightly higher than in setup approach 1, since the patch-size of the limiting component model
395 COSMO is larger.

396 *Figure 5*

397 **4 Summary and conclusions**

398 TerrSysMP was successfully ported to the massive parallel IBM BG/Q system JUQUEEN of the Jülich
399 Supercomputing Centre. In comparison to the domain sizes that could be run using the initial coupling
400 with OASIS3, the problem size could be increased by a factor of 64 while still maintaining very good
401 scaling factors and hence a high parallel efficiency using OASIS3-MCT. The study demonstrated that
402 an in-depth consideration of the hardware features and software environment is necessary to
403 efficiently operate fully coupled model systems based on the MPMD paradigm on massively parallel
404 architectures such as JUQUEEN. This is irrespective of the individual component model's
405 performance, as the coupling process adds significant additional complexity. Applying OASIS3 in
406 standard Linux cluster environments for external coupling is appropriate for medium domain sizes on
407 the order of 256 MPI ranks. Beyond medium domain sizes, OASIS3-MCT affords efficient coupling in
408 standard and massively parallel computer environments by overcoming mainly RAM-dependent
409 limitations. MPMD load balancing can be performed efficiently with profiling tools, such as Scalasca, to
410 optimize MPMD resource allocation and solve configuration restrictions, such as static resource
411 mapping. However, despite TerrSysMP's encouraging weak-scaling performance of the dynamic
412 kernels of the different components models, initialization and I/O need to be reconciled for processor
413 counts beyond one BG/Q midplane (8192 cores), which are required for large-scale hyper-resolution
414 simulations. Currently, the applicability of TerrSysMP is explored for fully coupled terrestrial
415 simulations over the pan European continent and simulations of a regional scale *virtual reality*.

416 **Acknowledgements**

417 We would like to thank the John von Neumann Institute for Computing and the Forschungszentrum
418 Jülich for providing the required compute time for the projects JICG43 ("Fractal Scaling of

419 Hydrodynamics at the Catchment Scale") and HBN24 (“Development, testing, and application of an
420 integrated soil-vegetation-atmosphere model”) that were utilized in this study. The financial support by
421 the SFB/TR 32 "Pattern in Soil-Vegetation-Atmosphere Systems: Monitoring, Modeling, and Data
422 Assimilation” funded by the Deutsche Forschungsgemeinschaft (DFG) and the Geoverbund ABC/J
423 (http://www.geoverbund-abcj.de) is gratefully acknowledged.

References

Anyah, R. O., Weaver, C. P., Miguez-Macho, G., Fan, Y., and Robock, A.: Incorporating water table dynamics in climate modeling: 3. Simulated groundwater influence on coupled land-atmosphere variability, Journal of Geophysical Research-Atmospheres, 113, Artn D07103 Doi 10.1029/2007jd009087, 2008.

Baldauf, M., Seifert, A., Forstner, J., Majewski, D., Raschendorfer, M., and Reinhardt, T.: Operational Convective-Scale Numerical Weather Prediction with the COSMO Model: Description and Sensitivities, Monthly Weather Review, 139, 3887-3905, Doi 10.1175/Mwr-D-10-05013.1, 2011.

Böhme, D., Geimer, M., Wolf, F., and Arnold, L.: Identifying the Root Causes of Wait States in Large-Scale Parallel Applications, Parallel Processing (ICPP), 2010 39th International Conference on, 2010, 90-100,

Böhme, D., Wolf, F., De Supinski, B. R., Schulz, M., and Geimer, M.: Scalable Critical-Path Based Performance Analysis, Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International, 2012, 1330-1340,

Chang, C.-C., Czajkowski, G., Eicken, T. v., and Kesselman, C.: Evaluating the performance limitations of MPMD communication, Proceedings of the 1997 ACM/IEEE conference on Supercomputing, San Jose, CA, 1997.

Dennis, J. M., Jacob, R., Vertenstein, M., Craig, T., and Loy, R.: Toward an ultra-high resolution community climate system model for the BlueGene platform, SciDac 2007: Scientific Discovery Through Advanced Computing, 78, U247-U251, Doi 10.1088/1742-6596/78/1/012030, 2007.

Fersch, B., Wagner, S., Rummler, T., Gochis, D., and Kunstmann, H.: Impact of groundwater dynamics and soil-type on modelling coupled water exchange processes between land and atmosphere, Climate and Land Surface Changes in Hydrology, 359, 140-145, 2013.

Geimer, M., Wolf, F., Wylie, B. J. N., Abraham, E., Becker, D., and Mohr, B.: The Scalasca performance toolset architecture, Concurrency and Computation-Practice & Experience, 22, 702-719, Doi 10.1002/Cpe.1556, 2010.

Geimer, M., Saviankou, P., Strube, A., Szebenyi, Z., Wolf, F., and Wylie, B. J. N.: Further Improving the Scalability of the Scalasca Toolset, Applied Parallel and Scientific Computing, Part II, 7134, 463-473, 2012.

Gent, P. R.: Preface to special issue on Community Climate System Model (CCSM), Journal of Climate, 19, 2121-2121, Doi 10.1175/Jcli9020.1, 2006.

Hammond, G. E., Lichtner, P. C., and Mills, R. T.: Evaluating the performance of parallel subsurface simulators: An illustrative example with PFLOTRAN, Water Resources Research, 50, 208-228, DOI: 10.1002/2012WR013483, 2014.

Hill, C., DeLuca, C., Balaji, V., Suarez, M., da Silva, A., Sawyer, W., Cruz, C., Trayanov, A., Zaslavsky, L., Hallberg, R., Boville, B., Craig, A., Collins, N., Kluzek, E., Michalakes, J., Neckels, D., Schwab, E., Smithline, S., Wolfe, J., Iredell, M., Yang, W. Y., Jacob, R., and Larson, J.: Implementing applications with the Earth System Modeling Framework, Applied Parallel Computing: State of the Art in Scientific Computing, 3732, 563-572, 2006.

Jones, J. E., and Woodward, C. S.: Newton-Krylov-multigrid solvers for large-scale, highly heterogeneous, variably saturated flow problems, Advances in Water Resources, 24, 763-774, Doi 10.1016/S0309-1708(00)00075-0, 2001.

Keyes, D. E., McInnes, L. C., Woodward, C., Gropp, W., Myra, E., Pernice, M., Bell, J., Brown, J., Clo, A., Connors, J., Constantinescu, E., Estep, D., Evans, K., Farhat, C., Hakim, A., Hammond, G., Hansen, G., Hill, J., Isaac, T., Jiao, X. M., Jordan, K., Kaushik, D., Kaxiras, E., Koniges, A., Lee, K., Lott, A., Lu, Q. M., Magerlein, J., Maxwell, R., McCourt, M., Mehl, M., Pawlowski, R., Randles, A. P., Reynolds, D., Riviere, B., Rude, U., Scheibe, T., Shadid, J., Sheehan, B.,

Shephard, M., Siegel, A., Smith, B., Tang, X. Z., Wilson, C., and Wohlmuth, B.: Multiphysics simulations: Challenges and opportunities, International Journal of High Performance Computing Applications, 27, 4-83, Doi 10.1177/1094342012468181, 2013.

Kirchner, J. W., Feng, X. H., and Neal, C.: Fractal stream chemistry and its implications for contaminant transport in catchments, Nature, 403, 524-527, Doi 10.1038/35000537, 2000.

Kollet, S. J., and Maxwell, R. M.: Integrated surface-groundwater flow modeling: A free-surface overland flow boundary condition in a parallel groundwater flow model, Advances in Water Resources, 29, 945-958, Doi 10.1016/J.Advwatres.2005.08.006, 2006.

Kollet, S. J., Maxwell, R. M., Woodward, C. S., Smith, S., Vanderborght, J., Vereecken, H., and Simmer, C.: Proof of concept of regional scale hydrologic simulations at hydrologic resolution utilizing massively parallel computer resources, Water Resources Research, 46, Artn W04201 Doi 10.1029/2009wr008730, 2010.

Larson, J., Jacob, R., and Ong, E.: The Model Coupling Toolkit: A new fortran90 toolkit for building multiphysics parallel coupled models, International Journal of High Performance Computing Applications, 19, 277-292, Doi 10.1177/1094342005056115, 2005.

Maxwell, R. M., Chow, F. K., and Kollet, S. J.: The groundwater-land-surface-atmosphere connection: Soil moisture effects on the atmospheric boundary layer in fully-coupled simulations, Advances in Water Resources, 30, 2447-2466, Doi 10.1016/J.Advwatres.2007.05.018, 2007.

Maxwell, R. M., Lundquist, J. K., Mirocha, J. D., Smith, S. G., Woodward, C. S., and Tompson, A. F. B.: Development of a Coupled Groundwater-Atmosphere Model, Monthly Weather Review, 139, 96-116, Doi 10.1175/2010mwr3392.1, 2011.

Mills, R. T., Lu, C., Lichtner, P. C., and Hammond, G. E.: Simulating subsurface flow and transport on ultrascale computers using PFLOTRAN, SciDac 2007: Scientific Discovery Through Advanced Computing, 78, U387-U393, Doi 10.1088/1742-6596/78/1/012051, 2007.

Oleson, K. W., Niu, G. Y., Yang, Z. L., Lawrence, D. M., Thornton, P. E., Lawrence, P. J., Stockli, R., Dickinson, R. E., Bonan, G. B., Levis, S., Dai, A., and Qian, T.: Improvements to the Community Land Model and their impact on the hydrological cycle, Journal of Geophysical Research-Biogeosciences, 113, Artn G01021 Doi 10.1029/2007jg000563, 2008.

Shao, Y. P., Liu, S. F., Schween, J. H., and Crewell, S.: Large-Eddy Atmosphere-Land-Surface Modelling over Heterogeneous Surfaces: Model Development and Comparison with Measurements, Boundary-Layer Meteorology, 148, 333-356, Doi 10.1007/S10546-013-9823-0, 2013.

Shrestha, P., Sulis, M., Masbou, M., Kollet, S., and Simmer, C.: A scale-consistent Terrestrial Systems Modeling Platform based on COSMO, CLM and ParFlow, Mon. Weather Rev., 0027-0644, doi:10.1175/MWR-D-14-00029.1, 2014

Song, F., and Wolf, F.: CUBE User Manual, University of Tennessee, Innovative Computing Laboratory, 2004.

Top500.org, Top 500 Supercomputer Sites: http://www.top500.org/, access: November 2013, 2013.

Valcke, S., Balaji, V., Craig, A., DeLuca, C., Dunlap, R., Ford, R. W., Jacob, R., Larson, J., O'Kuinghttons, R., Riley, G. D., and Vertenstein, M.: Coupling technologies for Earth System Modelling, Geoscientific Model Development, 5, 1589-1596, Doi 10.5194/Gmd-5-1589-2012, 2012.

Valcke, S.: The OASIS3 coupler: a European climate modelling community software, Geoscientific Model Development, 6, 373-388, Doi 10.5194/Gmd-6-373-2013, 2013.

Wood, E. F., Roundy, J. K., Troy, T. J., van Beek, L. P. H., Bierkens, M. F. P., Blyth, E., de Roo, A., Doll, P., Ek, M., Famiglietti, J., Gochis, D., van de Giesen, N., Houser, P., Jaffe, P. R., Kollet, S., Lehner, B., Lettenmaier, D. P., Peters-Lidard, C., Sivapalan, M., Sheffield, J., Wade, A., and Whitehead, P.: Hyperresolution global land surface modeling: Meeting a grand challenge

519 for monitoring Earth's terrestrial water, Water Resources Research, 47, Artn W05301 Doi
520 10.1029/2010wr010090, 2011.

521 Yang, X. F., Scheibe, T. D., Richmond, M. C., Perkins, W. A., Vogt, S. J., Codd, S. L., Seymour, J. D.,
522 and McKinley, M. I.: Direct numerical simulation of pore-scale flow in a bead pack: Comparison
523 with magnetic resonance imaging observations, Advances in Water Resources, 54, 228-241,
524 Doi 10.1016/J.Advwatres.2013.01.009, 2013.

525

Table Captions

Table 1: Summary of experimental design setup for scaling studies.

Tables

a) **Design 1** **Scaling step**	**1**	**2**	**3**	**4**
#gridcells per dimension (COSMO/ CLM/ ParFlow)	144 / 288 / 288	288 / 576 / 576	576 / 1152 / 1152	1152 / 2304 / 2304
#processors (COSMO/ CLM/ ParFlow)	24x16 / 8x8 / 8x8	48x32 / 16x16 / 16x16	96x64 / 32x32 / 32x32	192x128 / 64x64 / 64x64
cores	512	2048	8192	32768
nodeboards	1	4	16	64
midplanes	1/16	1/4	1	4 (2 racks)
b) **Design 2** **Scaling step**	**1**	**2**	**3**	**4**
#gridcells per dimension (COSMO/ CLM/ ParFlow)	128 / 256 / 256	256 / 512 / 512	512 / 1024 / 1024	1024 / 2048 / 2048
#processors (COSMO/ CLM/ ParFlow)	16x16 / 8x16 / 8x16	32x32 / 32x16 / 32x16	64x64 / 64x32 / 64x32	128x128 / 128x64 / 128x64
cores	512	2048	8192	32768
nodeboards	1	4	16	64
midplanes	1/16	1/4	1	4 (2 racks)

Table 1.

Figure Captions

Figure 1. Schematic of interaction processes between TerrSysMP component models.

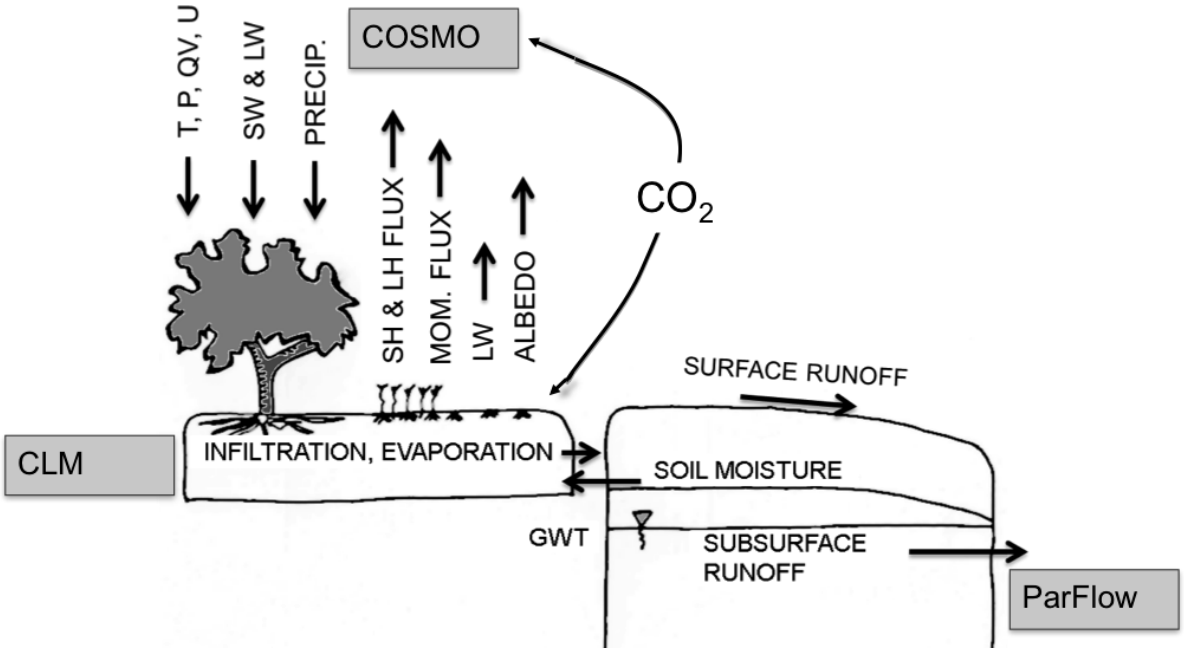
Figure 2. Schematic of the synchronization- and communication-structure. CLM receives first from COSMO before receiving from ParFlow, thus, wait states in ParFlow are indicating an overloaded COSMO. CLM calculation is very fast, but COSMO and ParFlow are idle during this time. CLM sends first to COSMO before sending to ParFlow. CLM is idle during COSMO and ParFlow computation.

Figure 3. CUBE screenshots of the fully coupled TerrSysMP after 6 hour simulation time. In a) each component model is naively distributed to one third of the resources (processor distribution: 192 COSMO, 160 ParFlow, 160 CLM). In b) the resources are distributed according to load, thus, the *Late Sender* wait state is significantly reduced (processor distribution: 384 COSMO, 80 ParFlow, 48 CLM). The topology view in b) shows fewer cores with *Late Sender* wait states where receivers are waiting for senders in the relevant functions. The unit of the middle view is *Late Sender* waiting time (accumulated over all CPUs). The units in the left and right view are percent.

Figure 4. Schematic of the coupling in TerrSysMP with OASIS3 (left) and OASIS3-MCT (right). OASIS3 is a separate executable and coupling arrays are repartitioned to the full domain by OASIS. OASIS3-MCT is part of each component model and coupling arrays only consist of the local fraction of the full domain and are routed by OASIS to the destination processor.

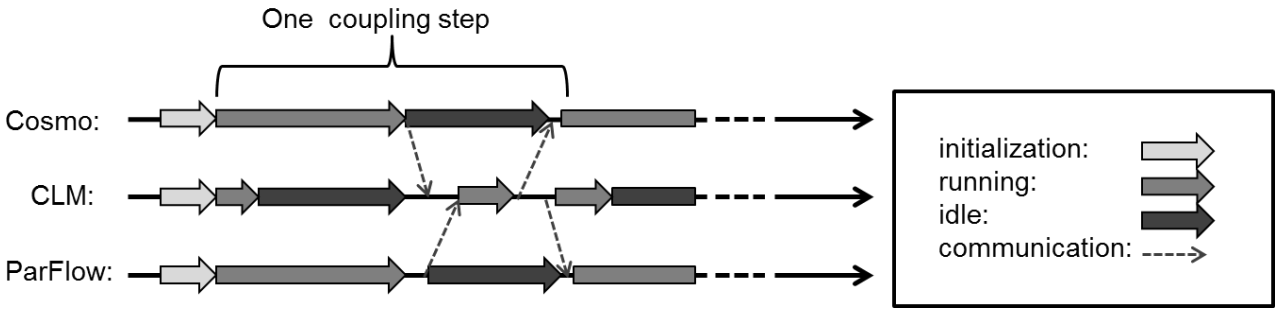
Figure 5. Idealized TerrSysMP weak-scaling study results with a) setup-design 1 (nx=ny=288, 288, and 144 for ParFlow, CLM and COSMO, respectively) and b) setup-design 2 (nx=ny=256, 256, and 128 for ParFlow, CLM and COSMO, respectively). The dotted lines show the absolute timings of the individual component models (green/COSMO is bounding the calculation time). The colored areas show the stacked absolute timings of the calculation, initialization and finalization time. The solid lines show the parallel efficiency of the relevant components on the secondary axis. The computational problem size, n, as well as the assigned CPU cores, np, is increasing with a factor of 4 between each step.

556 **Figures**



557

558 Figure 1.



559

560 Figure 2.

a)

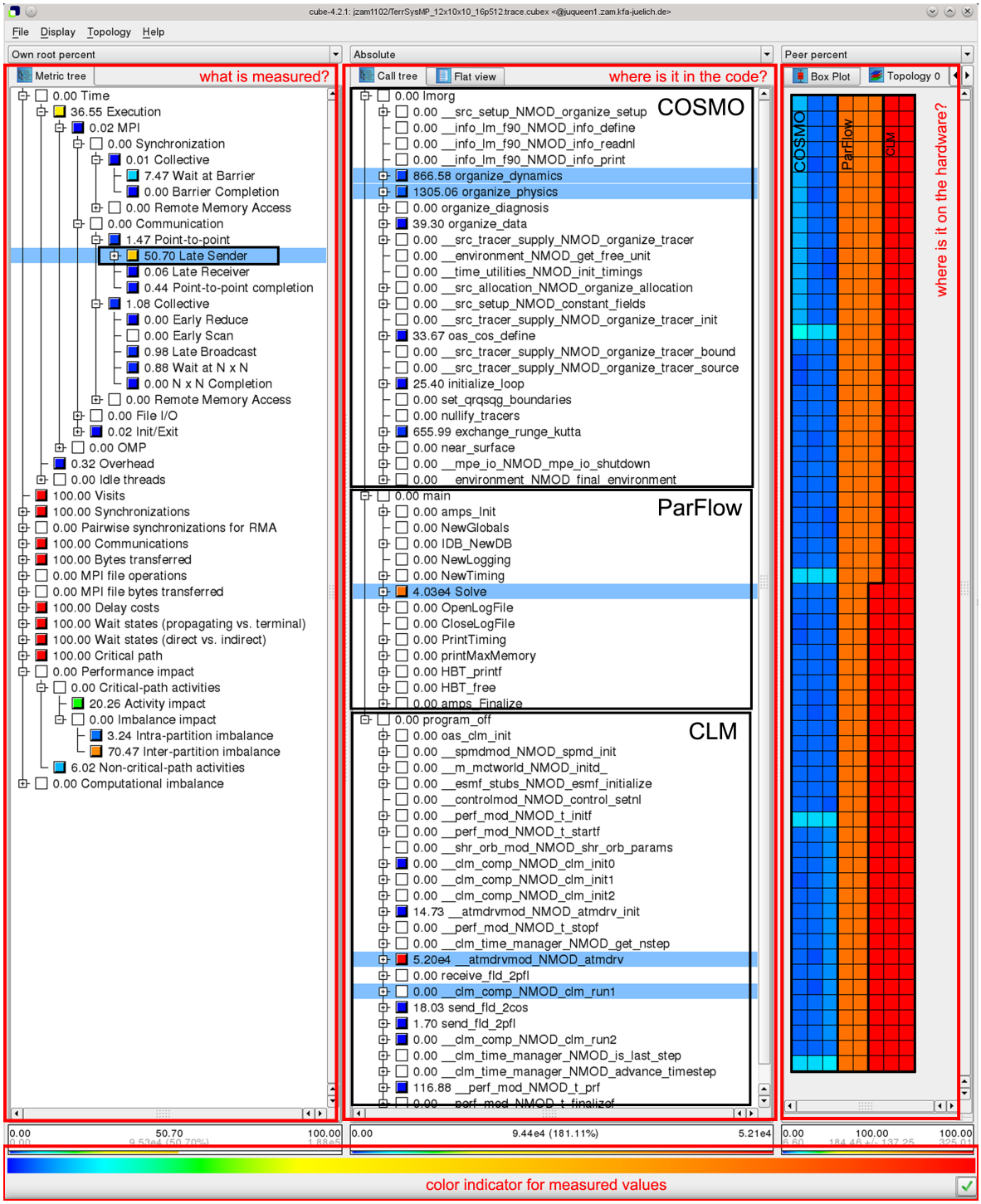

cube-4.2.1: jzam1102/TerrSysMP_12x10x10_16p512.trace.cubex <@juqueen1.zam.kfa-juelich.de>
File Display Topology Help
Own root percent
Absolute
Peer percent
Metric tree
what is measured?
Call tree
Flat view
where is it in the code?
Box Plot
Topology 0
where is it on the hardware?
COSMO
ParFlow
CLM
0.00 Time
36.55 Execution
0.02 MPI
0.00 Synchronization
0.01 Collective
7.47 Wait at Barrier
0.00 Barrier Completion
0.00 Remote Memory Access
0.00 Communication
1.47 Point-to-point
50.70 Late Sender
0.06 Late Receiver
0.44 Point-to-point completion
1.08 Collective
0.00 Early Reduce
0.00 Early Scan
0.98 Late Broadcast
0.88 Wait at N x N
0.00 N x N Completion
0.00 Remote Memory Access
0.00 File I/O
0.02 Init/Exit
0.00 OMP
0.32 Overhead
0.00 Idle threads
100.00 Visits
100.00 Synchronizations
0.00 Pairwise synchronizations for RMA
100.00 Communications
100.00 Bytes transferred
0.00 MPI file operations
0.00 MPI file bytes transferred
100.00 Delay costs
100.00 Wait states (propagating vs. terminal)
100.00 Wait states (direct vs. indirect)
100.00 Critical path
0.00 Performance impact
0.00 Critical-path activities
20.26 Activity impact
0.00 Imbalance impact
3.24 Intra-partition imbalance
70.47 Inter-partition imbalance
6.02 Non-critical-path activities
0.00 Computational imbalance
0.00 lmorg
0.00 __src_setup_NMOD_organize_setup
0.00 __info_lm_f90_NMOD_info_define
0.00 __info_lm_f90_NMOD_info_readnl
0.00 __info_lm_f90_NMOD_info_print
866.58 organize_dynamics
1305.06 organize_physics
0.00 organize_diagnosis
39.30 organize_data
0.00 __src_tracer_supply_NMOD_organize_tracer
0.00 __environment_NMOD_get_free_unit
0.00 __time_utilities_NMOD_init_timings
0.00 __src_allocation_NMOD_organize_allocation
0.00 __src_setup_NMOD_constant_fields
0.00 __src_tracer_supply_NMOD_organize_tracer_init
33.67 oas_cos_define
0.00 __src_tracer_supply_NMOD_organize_tracer_bound
0.00 __src_tracer_supply_NMOD_organize_tracer_source
25.40 initialize_loop
0.00 set_qrqsqg_boundaries
0.00 nullify_tracers
655.99 exchange_runge_kutta
0.00 near_surface
0.00 __mpe_io_NMOD_mpe_io_shutdown
0.00 __environment_NMOD_final_environment
0.00 main
0.00 amps_Init
0.00 NewGlobals
0.00 IDB_NewDB
0.00 NewLogging
0.00 NewTiming
4.03e4 Solve
0.00 OpenLogFile
0.00 CloseLogFile
0.00 PrintTiming
0.00 printMaxMemory
0.00 HBT_printf
0.00 HBT_free
0.00 amps_Finalize
0.00 program_off
0.00 oas_clm_init
0.00 __spmdmod_NMOD_spmd_init
0.00 __m_mctworld_NMOD_initd_
0.00 __esmf_stubs_NMOD_esmf_initialize
0.00 __controlmod_NMOD_control_setnl
0.00 __perf_mod_NMOD_t_initf
0.00 __perf_mod_NMOD_t_startf
0.00 __shr_orb_mod_NMOD_shr_orb_params
0.00 __clm_comp_NMOD_clm_init0
0.00 __clm_comp_NMOD_clm_init1
0.00 __clm_comp_NMOD_clm_init2
14.73 __atmdrvmod_NMOD_atmdrv_init
0.00 __perf_mod_NMOD_t_stopf
0.00 __clm_time_manager_NMOD_get_nstep
5.20e4 __atmdrvmod_NMOD_atmdrv
0.00 receive_fld_2pfl
0.00 __clm_comp_NMOD_clm_run1
18.03 send_fld_2cos
1.70 send_fld_2pfl
0.00 __clm_comp_NMOD_clm_run2
0.00 __clm_time_manager_NMOD_is_last_step
0.00 __clm_time_manager_NMOD_advance_timestep
116.88 __perf_mod_NMOD_t_prf
0.00 __perf_mod_NMOD_t_finalizef
0.00
50.70
100.00
9.53e4 (50.70%)
0.00
9.44e4 (181.11%)
5.21e4
0.00
100.00
100.00
184.46 +/- 137.25
color indicator for measured values

Figure 3a.

b)

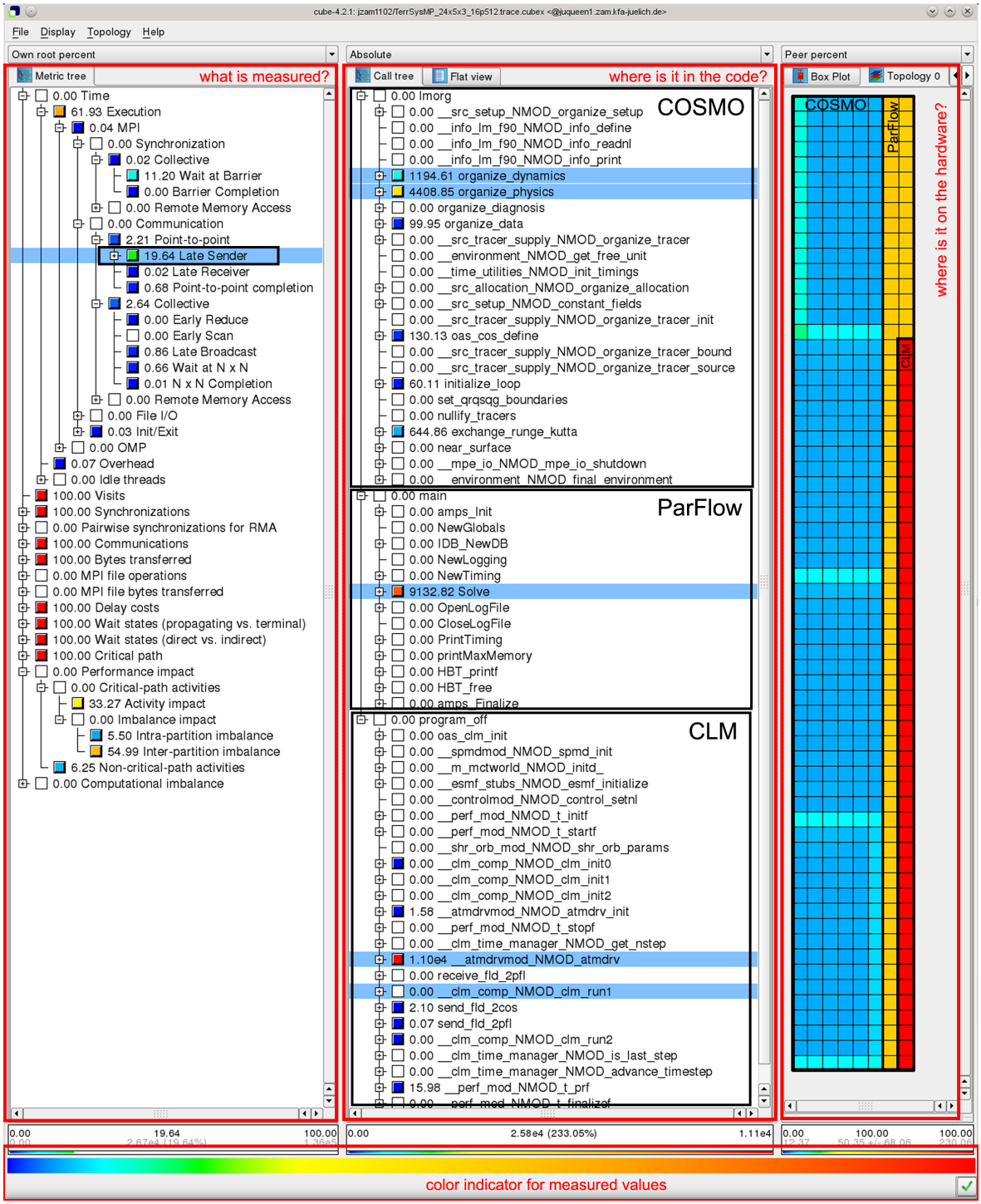
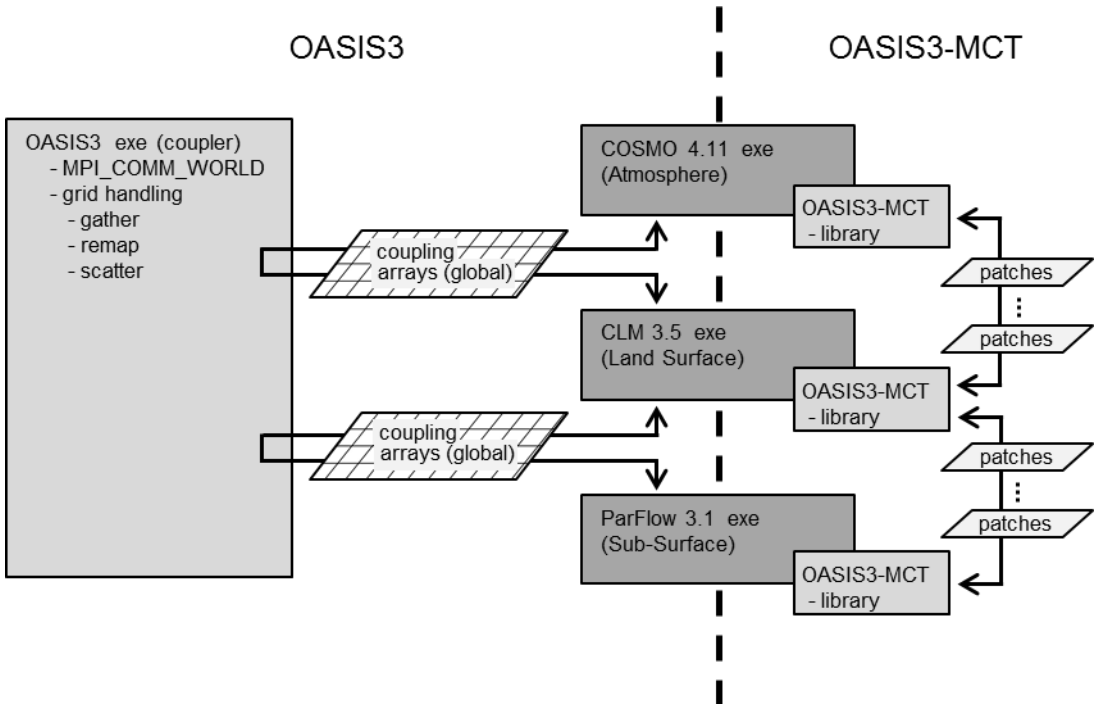

cube-4.2.1: jzam1102/TerrSysMP_24x5x3_16p512.trace.cubex <@juqueen1.zam.kfa-juelich.de>
File Display Topology Help
Own root percent
Absolute
Peer percent
Metric tree
what is measured?
Call tree
Flat view
where is it in the code?
Box Plot
Topology 0
where is it on the hardware?
COSMO
ParFlow
CLM
0.00 Time
61.93 Execution
0.04 MPI
0.00 Synchronization
0.02 Collective
11.20 Wait at Barrier
0.00 Barrier Completion
0.00 Remote Memory Access
0.00 Communication
2.21 Point-to-point
19.64 Late Sender
0.02 Late Receiver
0.68 Point-to-point completion
2.64 Collective
0.00 Early Reduce
0.00 Early Scan
0.86 Late Broadcast
0.66 Wait at N x N
0.01 N x N Completion
0.00 Remote Memory Access
0.00 File I/O
0.03 Init/Exit
0.00 OMP
0.07 Overhead
0.00 Idle threads
100.00 Visits
100.00 Synchronizations
0.00 Pairwise synchronizations for RMA
100.00 Communications
100.00 Bytes transferred
0.00 MPI file operations
0.00 MPI file bytes transferred
100.00 Delay costs
100.00 Wait states (propagating vs. terminal)
100.00 Wait states (direct vs. indirect)
100.00 Critical path
0.00 Performance impact
0.00 Critical-path activities
33.27 Activity impact
0.00 Imbalance impact
5.50 Intra-partition imbalance
54.99 Inter-partition imbalance
6.25 Non-critical-path activities
0.00 Computational imbalance
0.00 lmorg
0.00 __src_setup_NMOD_organize_setup
0.00 __info_lm_f90_NMOD_info_define
0.00 __info_lm_f90_NMOD_info_readnl
0.00 __info_lm_f90_NMOD_info_print
1194.61 organize_dynamics
4408.85 organize_physics
0.00 organize_diagnosis
99.95 organize_data
0.00 __src_tracer_supply_NMOD_organize_tracer
0.00 __environment_NMOD_get_free_unit
0.00 __time_utilities_NMOD_init_timings
0.00 __src_allocation_NMOD_organize_allocation
0.00 __src_setup_NMOD_constant_fields
0.00 __src_tracer_supply_NMOD_organize_tracer_init
130.13 oas_cos_define
0.00 __src_tracer_supply_NMOD_organize_tracer_bound
0.00 __src_tracer_supply_NMOD_organize_tracer_source
60.11 initialize_loop
0.00 set_qrqsqg_boundaries
0.00 nullify_tracers
644.86 exchange_runge_kutta
0.00 near_surface
0.00 __mpe_io_NMOD_mpe_io_shutdown
0.00 __environment_NMOD_final_environment
0.00 main
0.00 amps_Init
0.00 NewGlobals
0.00 IDB_NewDB
0.00 NewLogging
0.00 NewTiming
9132.82 Solve
0.00 OpenLogFile
0.00 CloseLogFile
0.00 PrintTiming
0.00 printMaxMemory
0.00 HBT_printf
0.00 HBT_free
0.00 amps_Finalize
0.00 program_off
0.00 oas_clm_init
0.00 __spmdmod_NMOD_spmd_init
0.00 __m_mctworld_NMOD_initd_
0.00 __esmf_stubs_NMOD_esmf_initialize
0.00 __controlmod_NMOD_control_setnl
0.00 __perf_mod_NMOD_t_initf
0.00 __perf_mod_NMOD_t_startf
0.00 __shr_orb_mod_NMOD_shr_orb_params
0.00 __clm_comp_NMOD_clm_init0
0.00 __clm_comp_NMOD_clm_init1
0.00 __clm_comp_NMOD_clm_init2
1.58 __atmdrvmod_NMOD_atmdrv_init
0.00 __perf_mod_NMOD_t_stopf
0.00 __clm_time_manager_NMOD_get_nstep
1.10e4 __atmdrvmod_NMOD_atmdrv
0.00 receive_fld_2pfl
0.00 __clm_comp_NMOD_clm_run1
2.10 send_fld_2cos
0.07 send_fld_2pfl
0.00 __clm_comp_NMOD_clm_run2
0.00 __clm_time_manager_NMOD_is_last_step
0.00 __clm_time_manager_NMOD_advance_timestep
15.98 __perf_mod_NMOD_t_prf
0.00 __perf_mod_NMOD_t_finalizef
0.00
19.64
100.00
2.67e4 (19.64%)
1.36e5
0.00
2.58e4 (233.05%)
1.11e4
0.00
100.00
100.00
12.37
50.35 +/- 68.06
230.06
color indicator for measured values

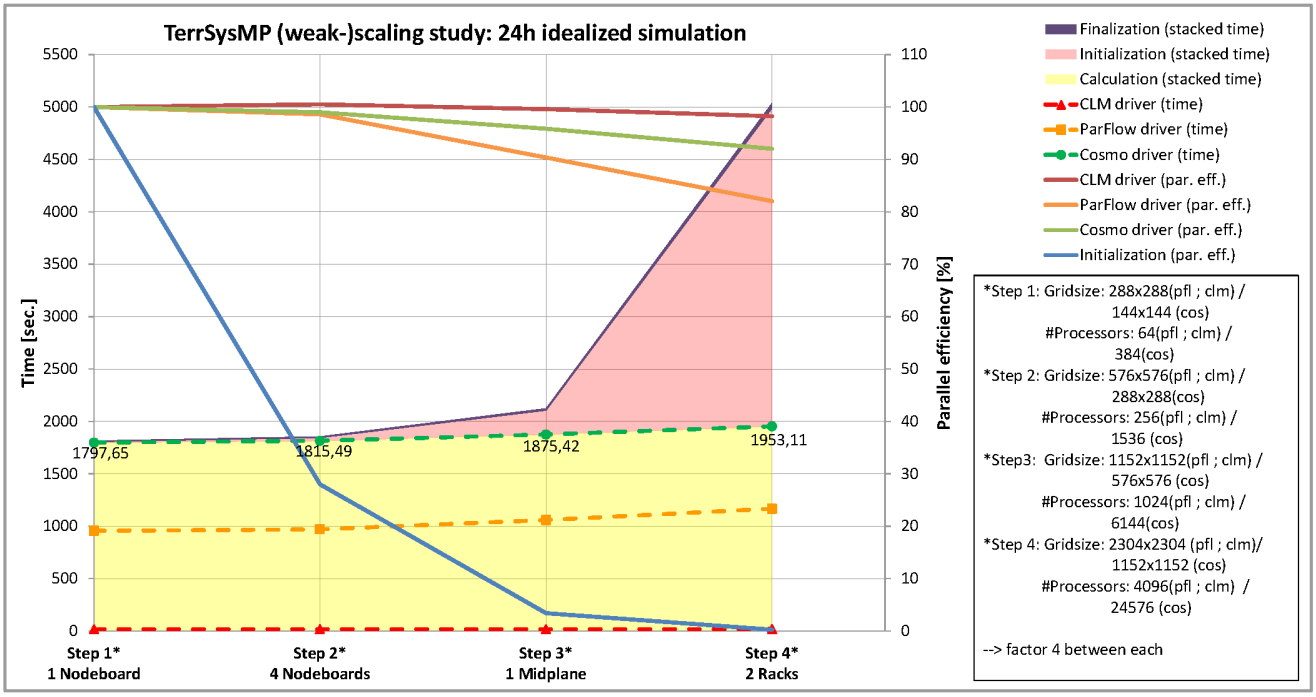
Figure 3b.



567

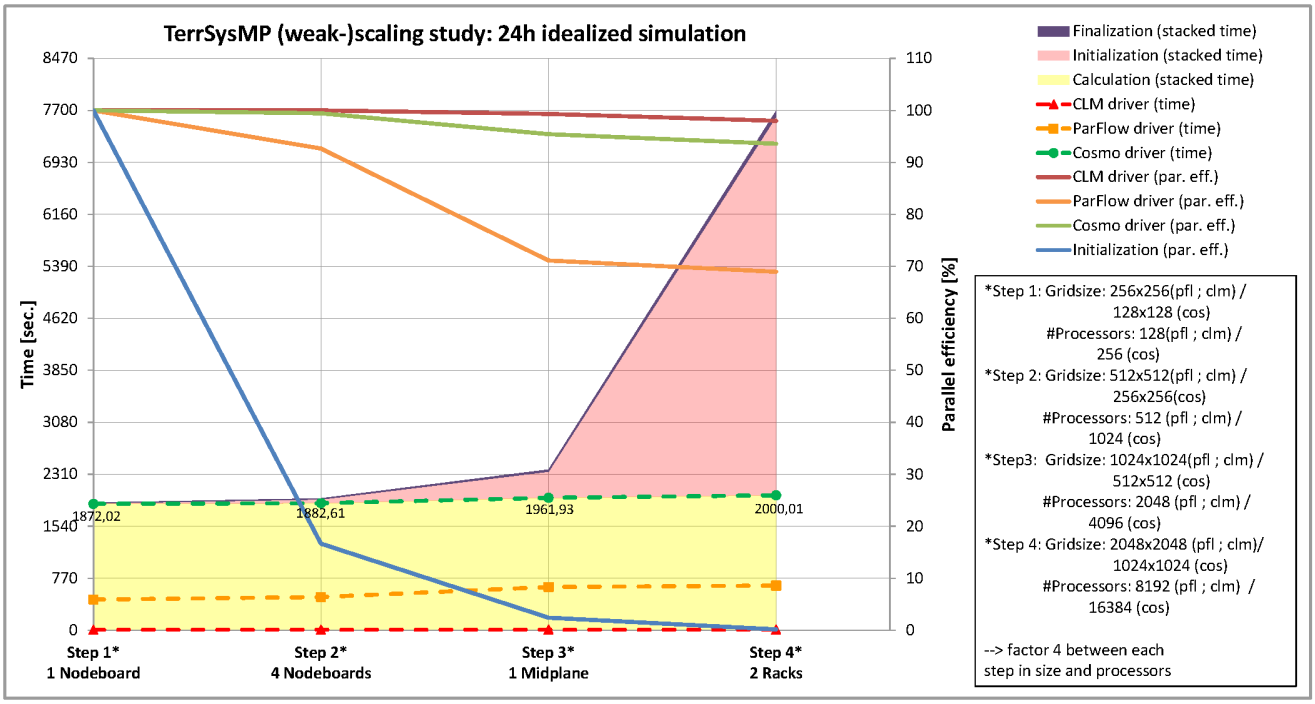
568 Figure 4.

569 a)



570

571 b)



572

573 Figure 5.