Scalability Tuning of the Load Balancing and Coupling Framework FD4

Matthias Lieber, Wolfgang E. Nagel, and Hartmut Mix

Center for Information Services and High Performance Computing (ZIH),
Technische Universität Dresden, 01062 Dresden, Germany
E-mail: {matthias.lieber, wolfgang.nagel, hartmut.mix}@tu-dresden.de

In this paper, we discuss the scalability tuning of the HPC software framework FD4. The framework provides dynamic load balancing and model coupling for multiphase and multiphysics simulations. We first investigate scalability bottlenecks using the Vampir performance analysis tool-set on the BlueGene/Q system at JSC. Then, we describe and evaluate our optimized algorithms: A new hierarchical 1D partitioning algorithm for SFC-based dynamic load balancing and a new method to organize the coupling metadata. The final scalability benchmark shows that the overhead of FD4 has been reduced by a factor of 3.3 at 262 144 ranks, which leads to a considerably increased scalability of the whole application.

1 Introduction

Driven by the growing capacity of supercomputers and the increasing knowledge about the underlying processes, simulation models become more and more complex. For example in the atmospheric sciences, models are coupled to incorporate more elements of the geosphere in the simulations, e.g. climate models or earth systems models[1]. Additionally, more complex descriptions of individual phenomena are included in the models. Running such model systems efficiently on large scale supercomputers is a challenging task and requires highly tuned methods for data decomposition, communication, load balancing, and model coupling, which should be implemented preferably in an independent and reusable software framework. In this paper, we analyze scalability bottlenecks of the framework FD4 and describe our improvements. The work has been carried out on the BlueGene/P and BlueGene/Q systems at Jülich Supercomputing Centre. The architecture of these systems offers an excellent platform to prepare applications and algorithms for the Exascale era, where the number of nodes is expected to grow even further[2].

FD4 (Four-Dimensional Distributed Dynamic Data structures[3,4]) has been developed for dynamic load balancing in the model system COSMO-SPECS, which describes the interaction between aerosols, clouds, and precipitation with a high level of detail[5]. It consists of the COSMO model[6], a non-hydrostatic limited-area atmospheric model, and the spectral cloud microphysics model SPECS[7]. SPECS introduces 11 new variables to describe three types of hydrometeors (water droplets, frozen particles, and insoluble particles). These variables are each discretized into 66 size classes, leading to a high amount of data per grid cell. Since SPECS' computational costs are very high and strongly depend on the spatially and temporally varying presence of hydrometeors, a new version called COSMO-SPECS+FD4[3] with dynamic load balancing has been developed: SPECS has been separated from the static partitioning of COSMO and decomposed in three dimensions into small, rectangular blocks transparently managed by FD4, see Fig. 1. FD4 provides dynamic load balancing of the blocks, communication between neighbor blocks located on

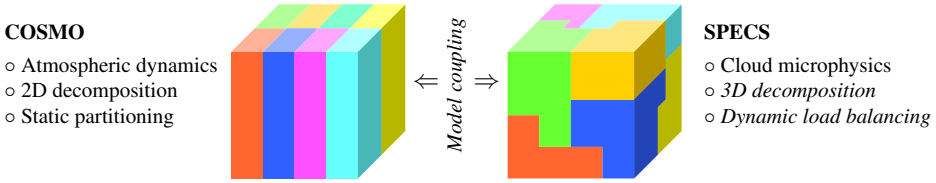


Figure 1. Coupling concept of COSMO-SPECS+FD4. The boxes illustrate exemplary 8-way partitionings of the 3D computational domain. Services provided by FD4 are written in *italics*.

different MPI processes, and coupling to the static partitioning of the atmospheric model. After first scalability experiences with COSMO-SPECS+FD4, we observed that much attention needs to be paid to the load balancing and coupling methods in order to prevent an increasing overhead that impedes any benefit of the load balancing.

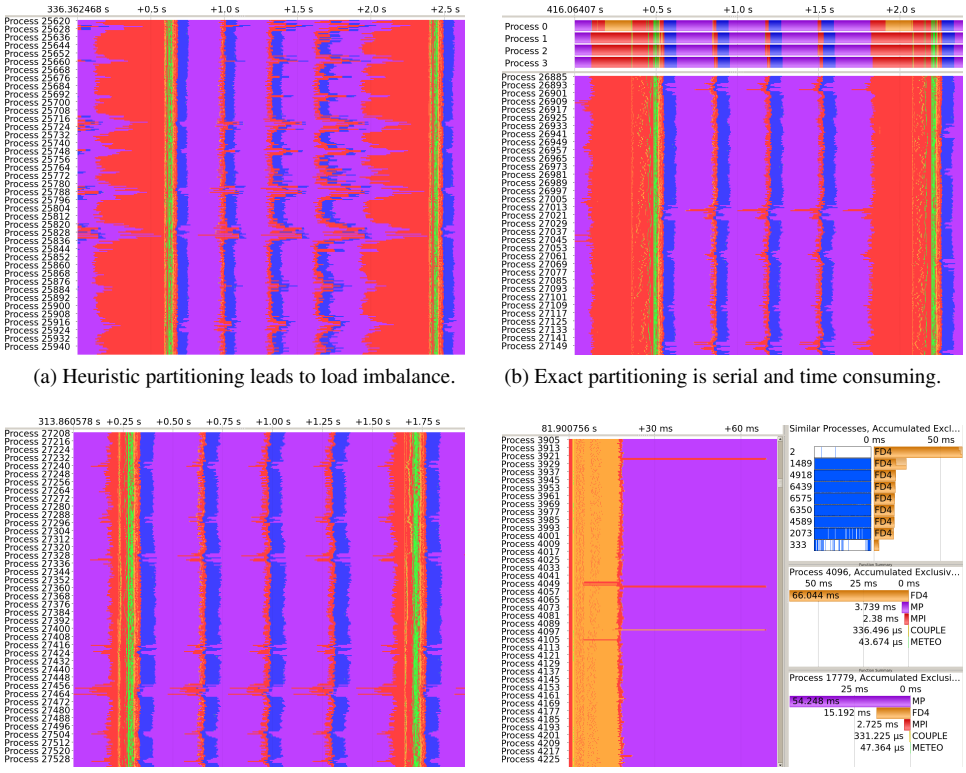
2 Scalability Bottleneck Analysis of FD4

In this section, we discuss the scalability bottlenecks we detected in FD4. For a detailed analysis, we used the Vampir performance analysis tool-set[8,9]. The VampirTrace[10] performance monitor records application events from all processes, like function calls and MPI messages, along with their time stamp to an OTF[11] trace file. This data is then displayed with the Vampir[12] trace visualizer by means of several charts like timelines or statistics. For our analysis, we performed runs with 32 768 processes on BlueGene/Q. We recorded MPI events and certain functions of COSMO-SPECS+FD4. Additionally, we added user-defined counters, e.g. to observe the number of FD4 blocks per process over time. To reduce the trace file size, only a few time steps (e.g. the last 3 out of 180) have been recorded. This selective instrumentation caused a runtime overhead of less than 1% with additional trace writing time of approx. 90 s. In spite of the large number of application processes, the compressed trace files have a total size of approx. 4 GiB only.

We ran COSMO-SPECS+FD4 with an artificial benchmark scenario: An initial heat bubble leads to the formation of a precipitating cumulus cloud after 30 min simulated time. This basic problem can be replicated arbitrarily along the horizontal axes to scale the grid size. Note, that the benchmark does not perform any I/O of model data. For the bottleneck analysis, we used a horizontal grid of 512×256 cells with 48 vertical layers containing 128 replicated clouds, resulting in extremely small COSMO partitions of $2 \times 2 \times 48$ cells.

2.1 Analysis of Dynamic Load Balancing

Since the workload of the blocks varies over time, COSMO-SPECS+FD4 regularly performs dynamic load balancing. The workload (i.e. weight) of a block is determined by measuring the time of the microphysics computations. FD4 uses a Hilbert space-filling curve[13] (SFC) for dynamic load balancing. The SFC linearizes the 3D block decomposition of the grid and, thus, reduces the partitioning problem from three to one dimension. Then, the remaining 1D partitioning problem is to find consecutive partitions in the block vector with a minimal maximum load among the partitions (i.e. bottleneck). Current published algorithms are either fast, parallel heuristics[14] or serial exact methods[15].



(a) Heuristic partitioning leads to load imbalance.

(b) Exact partitioning is serial and time consuming.

(c) New scalable high-quality partitioning.

(d) High duration of coupling in a few processes.

Figure 2. Vampir performance analysis of COSMO-SPECS+FD4 running on BlueGene/Q with 32 768 ranks.

Figure 2(a) shows a Vampir timeline chart for a 1% subset of the 32 768 processes. The time interval is approx. one time step, which starts at ca. 0.6 s and ends at ca. 2.4 s. Within a time step, first COSMO is computed (green), then SPECS performs four micro time steps (blue and purple), and finally dynamic load balancing is carried out. All MPI communication is shown in red. In Fig. 2(a), a parallel heuristic is used for partitioning. However, the achieved load balance is rather poor leading to large waiting time after SPECS.

Figure 2(b) shows the same time step when using an exact partitioning method (adapted from Pınar and Aykanat's exact bisection algorithm[15]). While the load balance is clearly better, the partitioning algorithm requires substantial serial computation time on rank 0, as shown in the upper part of the chart (orange). This leads to large waiting time in all other ranks (red). Additionally, all block weights need to be transferred to rank 0 first, which causes high network traffic and non-scalable memory consumption. Overall, the benefit of the exact method is reduced strongly due to the great overhead at large scale.

As Fig. 2(c) shows, we solved this problem with a new algorithm that is scalable and provides high load balance at the same time. The algorithm combines the approaches of the heuristic and the exact method. We briefly describe the algorithm in Sec. 3.1.

2.2 Analysis of Model Coupling

FD4 separates the decomposition of SPECS from the COSMO model. Consequently, each process owns one dynamic partition of SPECS that does not necessarily overlap with the process' static COSMO partition. Since data needs to be transferred between both models each time step, model coupling techniques are required[16], i.e. firstly map the partitionings to find communication partners (handshaking) and secondly transfer the data. In general, handshaking needs to be done only once at initialization for static partitions. But in applications with dynamic load balancing, handshaking needs to be repeated every time the partitioning is changed and, thus, becomes critical to performance. A naive implementation of handshaking would, for example, search the list of COSMO partitions for each local FD4 block to find overlaps. Of course, this algorithm is not scalable at all. We optimized the handshaking by performing this matching with the smallest bounding box containing all local blocks. This approach strongly reduces the handshaking time in most cases. However, as shown in Fig. 2(d), we observed that at certain time steps a very small number of (obviously arbitrary) processes require a larger handshaking time. In this example, two ranks consumed around 66 ms for handshaking (i.e. FD4 without MPI), while most of the processes required 15 ms only. Of course, this imbalance leads to waiting time in the coupling data transfer and at subsequent synchronization points. We found out, that the slow processes have disconnected FD4 partitions with parts far apart in the block decomposition. This leads to a large bounding box and, thus, increased handshaking time.

3 Improving Scalability of FD4

In this section, we briefly describe the algorithms we developed to improve the scalability of dynamic load balancing and model coupling in FD4 and evaluate the improvement.

3.1 Improvement of Dynamic Load Balancing

We developed a novel hierarchical 1D partitioning algorithm that combines the scalability of the parallel heuristic and the high quality of the exact method. Our algorithm starts with a parallel heuristic, but creates only $G < P$ partitions, where P is the total number of processes. For sufficiently small G, this will create almost optimal partitionings[14]. Each of the G coarse partitions is assigned a group of processes. In the seconds phase, each group independently partitions their coarse partition using an exact method. This way, we reduce the communication costs for gathering the block weights and the computation costs for the exact method, since it is run for a subset of blocks and processes only. Because no global vector of block weights is assembled, memory consumption is reduced in comparison to running the exact method for all processes. For a full description of the method, we refer to Ref. 4.

The group count G is like a slide control and tunes the influence of the heuristic versus the exact method. Therefore, we performed a series of runs on 65 536 processes with a horizontal grid of 512×512 cells and varied G from 16 to 8192. The grid was decomposed into 786 432 FD4 blocks for balancing SPECS. Figure 3 shows the total runtime of COSMO-SPECS+FD4 divided into main components (without initialization). Additionally, three important dynamic load balancing metrics are presented: runtime of the load

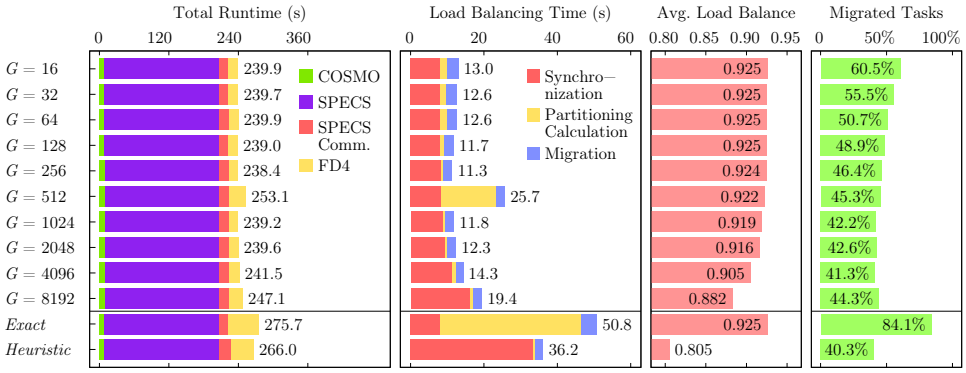


Figure 3. Influence of the group count G on the hierarchical 1D partitioning algorithm in COSMO-SPECS+FD4 with 65 536 processes on BlueGene/Q. The exact method and the heuristic are included as reference.

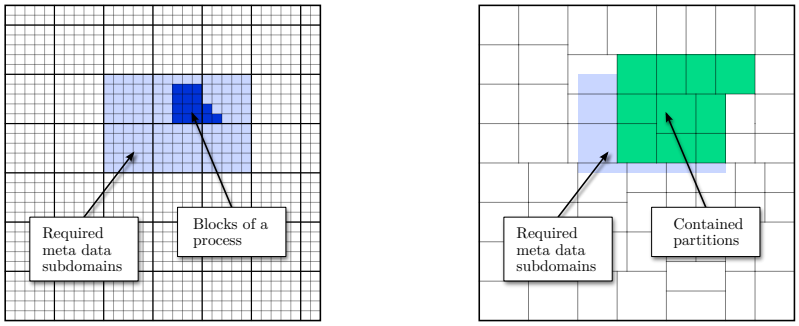
balancing, average load balance (defined as average process load divided by maximum load), and the average amount of migration per time step. Note, that dynamic load balancing is carried out every time step in this benchmark, i.e. 180 times. The results show, that our new method is almost as fast as the heuristic in terms of partitioning calculation, but achieves the optimal average load balance. At the optimal group count $G = 256$, the runtime of the application is reduced by more than 10% compared to the existing partitioning methods. The increased partitioning calculation time at $G = 512$ is a reproducible artifact resulting from some collectives on subcommunicators when distributing the result to all ranks. The results for average load balance and migrated tasks illustrate the slide control feature of G.

In practice, performing load balancing at every time step generates noticeable overhead. To reduce the number of load balancing invocations, FD4 is able to decide automatically if load balancing is beneficial. In this *auto-mode*, FD4 weighs the time lost due to imbalance against the time required for load balancing. Both times are measured at runtime and a history from the last 4 load balancing invocations is kept. Using the *auto-mode*, the execution time of the benchmark in Fig. 3 is reduced further to 234.9 s.

3.2 Improvement of Model Coupling

Handshaking with disconnected FD4 partitions that cause large bounding boxes has been tuned by replacing the simple bounding box with a cluster of a limited number of boxes. The clustering leads to small additional costs, but effectively solves the load imbalance issue when performing handshaking.

However, the total costs for handshaking are still very high at large scale, because every partition of the model coupled to FD4 (i.e. COSMO) has to be checked for overlap with the local blocks. We developed a general method[4] to dramatically reduce the search space by spatially decomposing the meta data that describes the coupled partitions (i.e. position, size, variables, owner). The so-called meta data subdomains (MDSDs) are rectangular subsets of the domain and cover multiple FD4 blocks. They contain the descriptions of



(a) Required meta data subdomains.

(b) Contained coupled partitions.

Figure 4. Exemplary illustration of meta data subdomains.

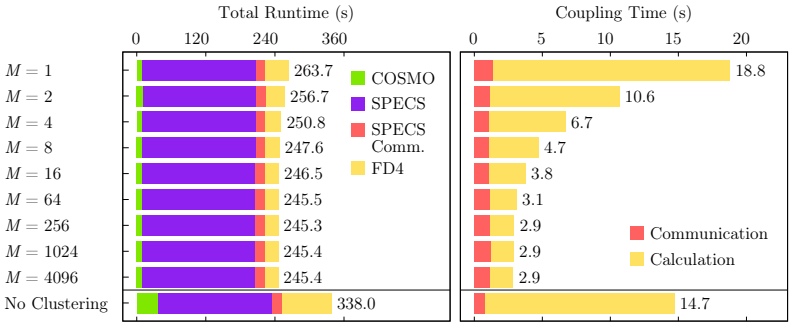


Figure 5. Influence of the meta data subdomain count M on the runtime of COSMO-SPECS+FD4 and the coupling time with 131 072 processes on BlueGene/Q. The original method without clustering is included as reference.

all partitions that begin within the region of the MDSD, i.e. each partition is assigned to a unique MDSD. For handshaking, only the MDSDs that overlap with the FD4 blocks and additionally the MDSDs at the spatially next lower coordinates need to be evaluated, given that no coupled partition is larger than the MDSDs (see Fig. 4). Since the MDSD decomposition is straightforward, the necessary MDSDs can be identified in constant time. Thus, the effort for handshaking does not depend on the total number of coupled partitions. The advantage of the MDSD concept is that arbitrary partition structures are supported, i.e. more complex geometries than rectangular partitions as in COSMO.

We evaluated the improvement achieved by the clustering and the meta data subdomains by running the COSMO-SPECS+FD4 benchmark on 131 072 ranks with a horizontal grid of 1024×512 cells. Fig. 5 shows that the calculation part of coupling (i.e. handshaking) is strongly accelerated with increasing number of MDSDs. Without the optimizations, the imbalanced handshaking (see Sec. 2.2) leads to waiting time in COSMO and FD4 load balancing, which increases the respective runtimes.

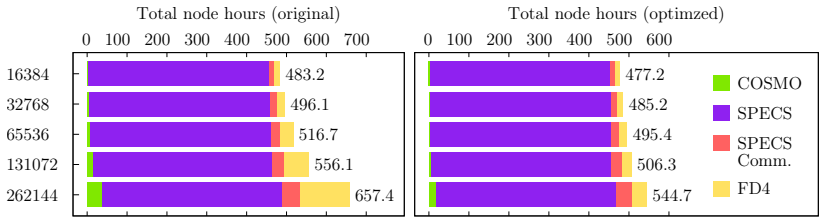


Figure 6. Strong scalability comparison of original COSMO-SPECS+FD4 and the tuned version on BlueGene/Q. Runtimes times number of nodes (32 processes each) are shown (without initialization).

3.3 Scalability Benchmark

We evaluated the scalability of COSMO-SPECS+FD4 with a strong scaling benchmark using a horizontal grid size of 1024×1024 cells and 3 145 728 FD4 blocks. Figure 6 compares the results for the original version and the version incorporating the optimizations described above. In both versions, the *auto-mode* to trigger dynamic load balancing was used. The speed-up of the original version from 16 384 to 262 144 ranks is 11.8, while a speed-up of 14.0 is achieved with the optimized version. At 262 144 processes, the overhead of FD4 has been reduced by a factor of 3.3, not considering the side-effects of handshaking imbalance.

4 Conclusion and Outlook

We discussed scalability optimizations for the HPC software framework FD4 that combines dynamic load balancing and model coupling. Scalability bottlenecks were analyzed using the Vampir performance analysis tools-set and optimizations were developed, implemented, and tested in FD4. The results show exemplary, that complex coupled simulation models with dynamic workload behavior can be run efficiently on large-scale HPC systems. FD4 is available as open source[17].

Future scalability improvements include the elimination of global meta data to reduce the memory consumption and global communication overheads. This would benefit the dynamic load balancing, since no global partitioning information needs to be distributed to all ranks, as well as the model coupling, since no global coupling meta data is stored on a single rank. Therefore, the meta data subdomains can be used to store local meta data only and dynamically redistribute the meta data after load balancing.

Acknowledgments

We thank the Jülich Supercomputing Centre, Germany, for access to JUGENE and JUQUEEN and the German Weather Service (Deutscher Wetterdienst) for providing the COSMO model. Furthermore, we want to thank Verena Grützun, Ralf Wolke, and Oswald Knoth for their support regarding the COSMO-SPECS model. This work was funded by the German Research Foundation (DFG), grant No. NA 711/2-1.

References

1. Warren M. Washington, Lawrence Buja, and Anthony Craig, *The computational future for climate and Earth system models: on the path to petaflop and beyond*, Philosophical Transactions A, **367**, no. 1890, 833–846, 2009.
2. Jack Dongarra et al., *The International Exascale Software Project Roadmap*, Int. J. High Perform. C., **25**, no. 1, 3–60, 2011.
3. Matthias Lieber, Verena Grützun, Ralf Wolke, Matthias S. Müller, and Wolfgang E. Nagel, *Highly Scalable Dynamic Load Balancing in the Atmospheric Modeling System COSMO-SPECS+FD4*, in: Applied Parallel and Scientific Computing, vol. 7133 of *LNCS*, pp. 131–141, 2012.
4. Matthias Lieber, *Dynamische Lastbalancierung und Modellkopplung zur hochskalierbaren Simulation von Wolkenprozessen*, Dissertation, Technische Universität Dresden, 2012.
5. Verena Grützun, Oswald Knoth, and Martin Simmel, *Simulation of the influence of aerosol particle characteristics on clouds and precipitation with LM–SPECS: Model description and first results*, Atmos. Res., **90**, no. 2-4, 233–242, 2008.
6. "Consortium for small-scale modeling", `http://www.cosmo-model.org`.
7. Martin Simmel and Sabine Wurzler, *Condensation and activation in sectional cloud microphysical models*, Atmos. Res., **80**, no. 2-3, 218–236, 2006.
8. Wolfgang E. Nagel, Alfred Arnold, Michael Weber, Hans-Christian Hoppe, and Karl Solchenbach, *VAMPIR: Visualization and Analysis of MPI Resources*, Supercomputer 63, **XII**, no. 1, 69–80, 1996.
9. Andreas Knüpfer, Holger Brunst, Jens Doleschal, Matthias Jurenz, Matthias Lieber, Holger Mickler, Matthias S. Müller, and Wolfgang E. Nagel, *The Vampir Performance Analysis Tool-Set*, in: Tools for High Performance Computing, pp. 139–155, Springer. 2008.
10. "VampirTrace website", `http://www.tu-dresden.de/zih/vampirtrace`.
11. "OTF website", `http://www.tu-dresden.de/zih/otf`.
12. "Vampir website", `http://vampir.eu`.
13. James D. Teresco, Karen D. Devine, and Joseph E. Flaherty, "Partitioning and dynamic load balancing for the numerical solution of partial differential equations", in: Numerical Solution of Partial Differential Equations on Parallel Computers, vol. 51 of *LNCSE*, pp. 55–88. Springer, 2006.
14. Serge Miguet and Jean-Marc Pierson, "Heuristics for 1D rectilinear partitioning as a low cost and high quality answer to dynamic load balancing", in: High-Performance Computing and Networking, vol. 1225 of *LNCS*, pp. 550–564. Springer, 1997.
15. Ali Pınar and Cevdet Aykanat, *Fast optimal load balancing algorithms for 1D partitioning*, J. Parallel Distrib. Comput., **64**, no. 8, 974–996, 2004.
16. Jay Larson, Robert Jacob, and Everest Ong, *The Model Coupling Toolkit: A New Fortran90 Toolkit for Building Multiphysics Parallel Coupled Models*, Int. J. High Perf. Comput. Appl., **19**, no. 3, 277–292, 2005.
17. "FD4 website", `http://wwwpub.zih.tu-dresden.de/~mlieber/fd4/`.