ENGINEERING MATHEMATICS
AND COMPUTING LAB

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

A parallel solution scheme for multiphysics evolution problems using OpenPALM

Martin Wlotzka, Vincent Heuveline

Preprint No. 2014-01

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)

www.emcl.iwr.uni-heidelberg.de

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)
ISSN 2191–0693
Preprint No. 2014-01

EMCL publications are split into two categories: Papers and Preprints.
Under the category Papers all publications that were published in journals, books etc. are listed.
The category Preprint Series contains publications that were accepted for the Preprint Series of the EMCL.
The EMCL Preprint Series was published under the roof of the Karlsruhe Institute of Technology (KIT) until April 30, 2013. As from May 01, 2013 it is published under the roof of Heidelberg University.

Affiliation of the Authors

Martin Wlotzka[a,1], Vincent Heuveline[a]

[a] *Engineering Mathematics and Computing Lab (EMCL),*
Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Germany
[1] *Corresponding Author: Martin Wlotzka, martin.wlotzka@uni-heidelberg.de*

Impressum

Heidelberg University
Interdisciplinary Center for Scientific Computing (IWR)
Engineering Mathematics and Computing Lab (EMCL)

Speyerer Str. 6,
69115 Heidelberg
Germany

A parallel solution scheme for multiphysics evolution problems using OpenPALM

Martin Wlotzka and Vincent Heuveline

Engineering Mathematics and Computing Lab (EMCL),
Interdisciplinary Center for Scientific Computing (IWR),
Heidelberg University, Germany

ABSTRACT

Multiphysics models seek to accurately represent complex phenomena by incorporating all relevant physical aspects. Therefore, multiphysics simulations often require the use of high performance computing resources. As an example application, we consider a natural convection problem, where we couple a fluid dynamics model and a temperature evolution model. In order to exploit the parallelism of operator splitting schemes, we propose to use the OpenPALM coupler tool for coupling individual model implementations. We carried out performance tests of the coupled models with a varying number of MPI processes and compared them to a standard implementation. Our results show that the coupling approach is superior over the standard implementation with respect to parallel efficiency, especially when using a higher number of MPI processes.

1 INTRODUCTION

Multiphysics systems consist of more than one component governed by its own principle for evolution or equilibrium. As an example, we consider a natural convection problem, where the wind, pressure and temperature are the variables of interest. For the individual components there are often well-established solvers and simulation codes available. This emphasizes the reuse of software in operator splitting coupling methods. Current practices for multiphysics coupling seek to find a balance between performance, software reuse and numerical accuracy. The OpenPALM coupling tool allows to perform parallel multiphysics simulations employing specialized model implementations as well as legacy codes.

2 PROBLEM FORMULATION, DISCRETIZATION AND SOLUTION SCHEME

2.1 Natural convection problem

According to the Boussinesq approximation, we assume density variations being sufficiently small to be neglected, except in the buoyancy term. We consider a 2D scenario of wind dynamics in the form of the natural convection problem

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \Delta \mathbf{u} - \frac{1}{\rho} \nabla \hat{p} = \frac{\hat{\theta}}{\theta_0} g \mathbf{e}_z \,, \tag{1a}$$

$$\nabla \cdot \mathbf{u} = 0 \,, \tag{1b}$$

$$\partial_t \theta + (\mathbf{u} \cdot \nabla)\theta - \alpha \Delta \theta = 0 \,. \tag{2}$$

The unknowns $\mathbf{u}$, $p = p_0 + \hat{p}$ and $\theta = \theta_0 + \hat{\theta}$ denote the velocity, pressure and potential temperature, respectively, where the given ground states are indicated by the zero-subscript and the deviations from the ground states are denoted with the hat symbol. The constants denote the

density ρ, the kinematic viscosity ν, the gravitation acceleration g and the thermal diffusivity α. We use a rectangular domain Ω for our computations. The no-slip condition $\mathbf{u} = 0$ is imposed on the whole boundary. For the temperature, we impose the Dirichlet boundary conditions $\theta = \theta_{\mathrm{hot}}$ and $\theta = \theta_{\mathrm{cold}}$ on the left and right side of the domain, respectively, and the homogeneous Neumann boundary condition $\nabla \mathbf{u} \cdot \mathbf{n} = 0$ on the thermally insulated top and bottom side. As initial conditions, we take $\mathbf{u}_0 \equiv 0$ and a linear temperature profile between the hot and the cold side of the domain.

2.2 Discretization

The implementations of the fluid dynamics model and the temperature evolution model are based on spatial Galerkin finite element discretizations of (1a,1b) and (2). We use inf-sup-stable Q2 / Q1 Taylor-Hood elements (Taylor and Hood, 1973, Ern and Guermond, 2010) for velocity and pressure, respectively, and Q2 elements for the temperature. As time stepping scheme, we use the Crank-Nicolson method. The discretized fluid dynamics model reads

$$\begin{aligned}
&(\mathbf{u}_h^{n+1}, \mathbf{v}_h) + \frac{\Delta t}{2}\Big[((\mathbf{u}_h^{n+1} \cdot \nabla)\mathbf{u}_h^{n+1}, \mathbf{v}_h) \\
&\qquad + \nu(\nabla \mathbf{u}_h^{n+1}, \nabla \mathbf{v}_h) - \frac{1}{\rho}(p_h^{n+1}, \nabla \cdot \mathbf{v}_h)\Big] \\
&\qquad = (\mathbf{u}_h^{n}, \mathbf{v}_h) - \frac{\Delta t}{2}\Big[(\mathbf{u}_h^{n} \cdot \nabla)\mathbf{u}_h^{n}, \mathbf{v}_h) + \nu(\nabla \mathbf{u}_h^{n}, \nabla \mathbf{v}_h) \\
&\qquad\qquad - \frac{1}{\rho}(p_h^{n}, \nabla \cdot \mathbf{v}_h)\Big] + \Delta t\Big(\frac{\hat{\theta}}{\theta_0} g \mathbf{e}_z, \mathbf{v}_h\Big) \quad \forall\, \mathbf{v}_h\,,
\end{aligned} \tag{3a}$$

$$(\nabla \cdot \mathbf{u}_h^{n+1}, q_h) = 0 \quad \forall\, q_h\,, \tag{3b}$$

where Δt is the timestep size, $\mathbf{u}_h^{n+1}$ and p_h^{n+1} denote the velocity and pressure of the new timestep, and $\mathbf{u}_h^{n}$ and p_h^{n} denote the solution of the old timestep. The finite element spaces are defined on a rectangular mesh Ω_h with cell size $h > 0$. The discretized temperature evolution model reads

$$\begin{aligned}
&(\theta_h^{n+1}, \phi_h) + \frac{\Delta t}{2}\Big[((\mathbf{u} \cdot \nabla)\theta_h^{n+1}, \phi_h) + \alpha(\nabla \theta_h^{n+1}, \nabla \phi_h)\Big] \\
&\qquad = (\theta_h^{n}, \phi_h) - \frac{\Delta t}{2}\Big[((\mathbf{u} \cdot \nabla)\theta_h^{n}, \phi_h) + \alpha(\nabla \theta_h^{n}, \nabla \phi_h)\Big] \quad \forall\, \phi_h\,,
\end{aligned} \tag{4}$$

where θ_h^{n+1} denotes the temperature of the new timestep, and θ_h^{n} is the solution from the old timestep. Note that the system (3a,3b) is nonlinear in the velocity variable, whereas (4) is a purely linear equation for the temperature. Therefore, we use Newton's method to solve the fluid dynamics model. In each Newton step, a linear system with the Jacobian matrix of (3a,3b) has to be solved. As this Jacobian matrix is non-symmetric in general, we use the GMRES Krylov subspace method with incomplete LU factorization (ILU) preconditioner (Saad, 2000) to compute the Newton updates. We also use an ILU-preconditioned GMRES solver to solve the temperature model (4), which is non-symmetric, too. Both models are implemented with the HiFlow^3 finite element package (Anzt et al., 2012).

2.3 Solution scheme

We consider a solution scheme which makes up the full model by coupling individual model components for the fluid dynamics part (3a,3b) and for the temperature evolution part (4). In contrast to implementing the solvers for the coupled models in one single monolithic application, we split the models and their implementations into separate applications. This provides a layer of parallelism, since the model components can be solved concurrently in each timestep by means of

an operator splitting approach. The solution scheme is sketched in Algorithm 1. It is a first order in time scheme for the coupled evolution problem (Keyes et al., 2011).

Algorithm 1 Concurrent operator splitting scheme

1: Given initial states $\mathbf{u}_h^0$, p_h^0 and θ_h^0.
2: **for** n=0,1,...,N **do**
3: Solve (3a,3b) using $\theta = \theta_h^n$ for one timestep to obtain $\mathbf{u}_h^{n+1}$ and p_h^{n+1}.
4: Solve (4) using $\mathbf{u} = \mathbf{u}_h^n$ for one timestep to obtain θ_h^{n+1}.
5: **end for**

In order to maintain these advantages in the simulations, we propose the use of the OpenPALM coupler tool.

3 SIMULATION WITH OPENPALM

3.1 Concept of OpenPALM

The fundamental concept of OpenPALM (Buis et al., 2006) is to consider multiphysics simulations as a coupled application. The models are assembled in a coupling algorithm, and OpenPALM controls their execution and interaction. Each model can be implemented individually. This offers the possibility to develop specialized solvers for the coupled models or to reuse existing codes with only minimal modifications. OpenPALM features two levels of parallelism. On the one hand, models can run concurrently on separate compute nodes. On the other hand, OpenPALM is able to couple models which are internally parallelized, using MPI, OpenMP as well as accelerators like GPU or MIC.

The OpenPALM coupler and the two model implementations are each compiled into their own executable. The simulations are then carried out as a multiple program multiple data (MPMD) application, which is outlined in Figure 1.

We use two independent model implementations with HiFlow[3] for the fluid dynamics and for the temperature evolution. Both models are parallelized using MPI based on a domain decomposition. Hereby, the models may use individual decompositions with a different number of subdomains. Generally, the nonlinear fluid dynamics model requires a much higher computational effort for solving one timestep than the linear temperature evolution model. As the model components need to exchange their solutions after each timestep, they should consume the same time-to-solution per timestep. Therefore, we run the fluid dynamics model on a bigger number of MPI processes than the temperature evolution model, so that their time-to-solution is balanced.

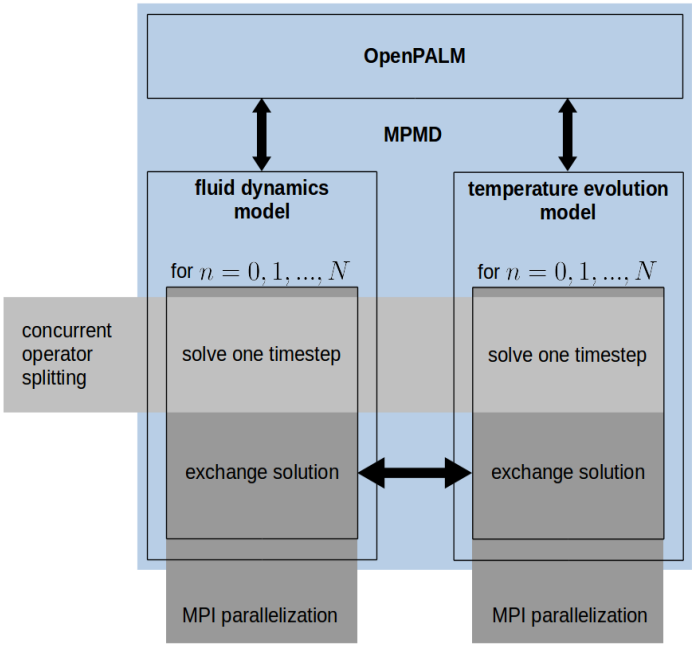

Figure 1: MPMD coupling scheme using OpenPALM.

3.2 *Model implementation details*

In OpenPALM terminology, a model component which can be executed in a coupling algorithm is called a unit. We designed two units for the fluid dynamics model and for the temperature evolution model. OpenPALM offers a graphical user interface named PrePALM which can be used to define the execution scheme of the units. Figure 2 shows the scheme we used for our simulations. The blue and red vertical bars depict the two independent execution branches of the application. The boxes sitting on the branches represent the units. Each unit has a variety of plugs in pink, black and green color, which are connected to each other. Plugs on the top side of a unit represent input data, and plugs on the bottom side represent output data. The color of the plugs indicates the data type, e.g. pink plugs for double precision floating point variables and green plugs for integer variables. By connecting the plugs, units can communicate with each other. OpenPALM provides two basic communication primitives for sending and receiving data, namely *PALM_Put* and *PALM_Get*. These communication routines can be used in the unit's source code to perform the actual exchange of the velocity and temperature data. As we run the models in parallel according to an individual domain decomposition for each model, the data to be exchanged is distributed among the MPI processes. OpenPALM is able to derive communication paths between the individual processes of parallel units. This avoids to collect the distributed data on one process before the communication, and to broadcast it afterwards. Instead, *PALM_Put* and *PALM_Get* transfer the data in portions according the intersection of the individual data distributions in the units. In Figure 2, the thick pink connection indicates the transfer of the distributed velocity and temperature data between the parallel units, and the thin connections indicate single process transfer of auxiliary data which the units need to set up their model.

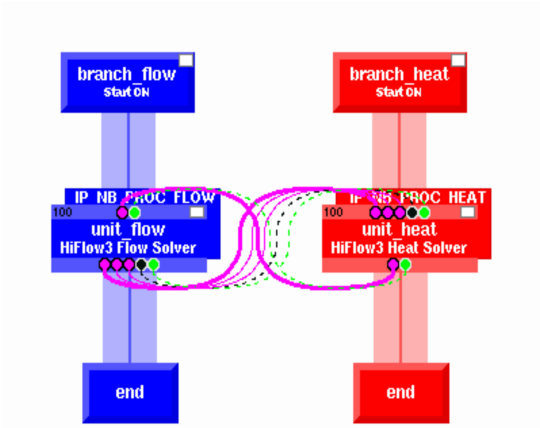

Figure 2: Execution scheme defined in PrePALM.

4 NUMERICAL EXPERIMENTS AND RESULTS

kinematic viscosity	$\nu = 1.57 \times 10^{-5}\,\mathrm{m}^2/\mathrm{s}$
thermal diffusivity	$\alpha = 1.9 \times 10^{-5}\,\mathrm{m}^2/\mathrm{s}$
heated wall	$\theta_{\mathrm{hot}} = 283.15\,\mathrm{K}$
cooled wall	$\theta_{\mathrm{cold}} = \theta_0 = 273.15\,\mathrm{K}$
timestep size	$\Delta t = 0.05\,\mathrm{s}$
degrees of freedom	$N = 3.41 \times 10^6$

Table 1: Scenario parameters.

We ran a series of simulations for the natural convection scenario with the parameters listed in Table 1. The Rayleigh number of the fluid is $\mathrm{Ra} \approx 10^9$, and the flow yields a Reynolds number of $\mathrm{Re} \approx 10^5$. Figure 3 shows three snapshots from the simulations. In the initial phase, buoyancy drives the main flow and recirculation areas evolve in the corners of the domain. The recirculation vortices propagate in horizontal direction and revolve the main flow direction temporarily. Finally, hot and cold vortices create turbulent behaviour.

We carried out performance tests using a discretization with 2.36 million degrees of freedom (DoF) for the fluid model and 1.05 million DoF for the temperature model, which amounts to 3.41 million DoF in total. We ran four test series with a total number of MPI processes ranging from 64 to 512. We varied the number of processes for the fluid dynamics model and for the temperature evolution model to achieve a balanced time-to-solution. Note that the OpenPALM coupler tool itself always runs on one process. For comparison, we also implemented the full model in a single

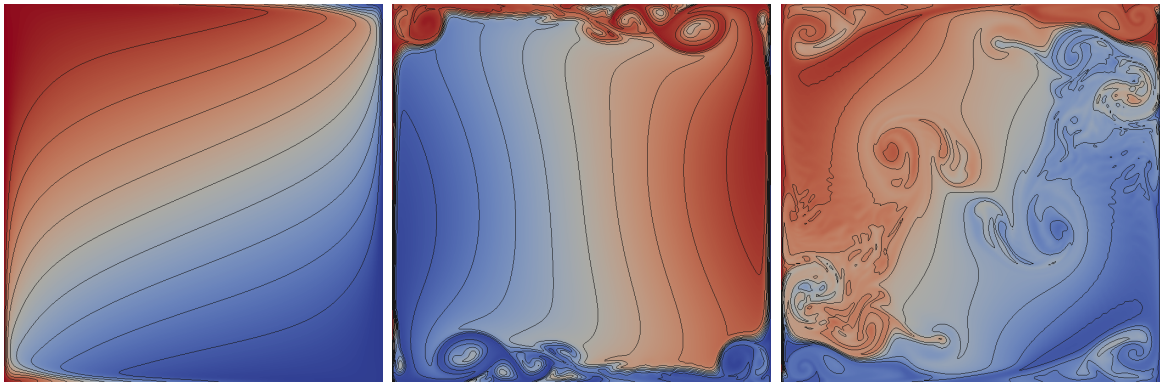

Figure 3: Natural convection simulation snapshots.

executable without OpenPALM. All tests were run on the JUROPA supercomputer at Jülich Supercomputing Center.

In Figures 4 and 5 we show the runtime performance of our tests. Each of the four diagrams visualizes the results of one of the test series, comprising the configurations with the same total number of MPI processes. The data points are distinguished by the distribution of the processes on the fluid model and on the temperature model for the specific configurations. For example, 55+8+1 means that the fluid model ran on 55 processes, the temperature model ran on 8 processes, and the OpenPALM coupler tool ran on one process, adding up to 64 processes in total. The diagrams show the individual runtime of the two models in blue and orange, respectively. As the models exchange their solutions and hence require a synchronization after each timestep, the overall runtime of the coupled models equals the maximum of the individual model runtimes. The runtime consumed by the data exchange itself is typically three to four orders of magnitude smaller than the computation time and therefore negligible. In addition, the diagrams show as black horizontal line the runtime of the single executable implementation for comparison.

For the configurations with 64 processes in total, the runtime of the fluid model decreases and the runtime of the temperature model increases monotonely when shifting processes from the temperature model to the fluid model, i.e. from left to right in the upper diagram. We could achieve slightly smaller overall runtimes for the coupled models compared to the single executable implementation.

Using 128 processes in total, the runtime of the fluid model also decreases when adding processes, with an exception for the 119+8+1 configuration. At the same

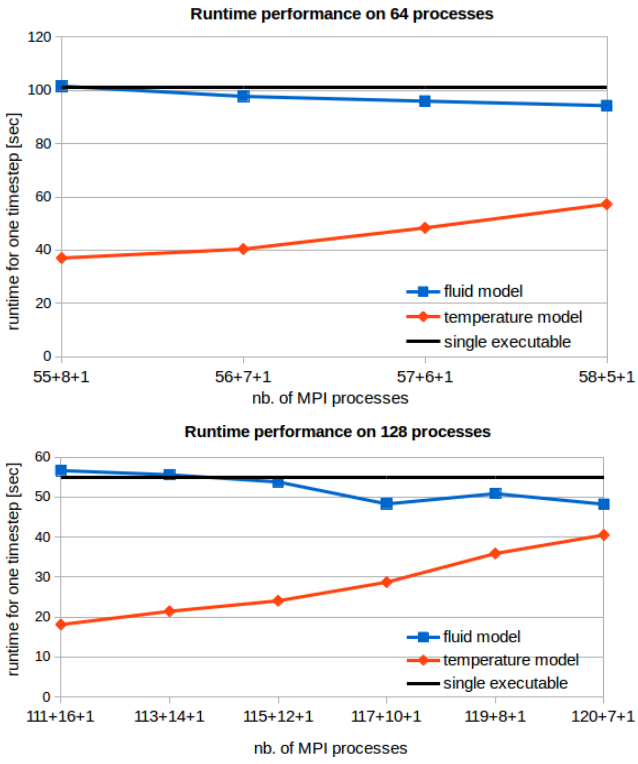

Figure 4: Runtime performance of the test configurations with 64 and 128 MPI processes.

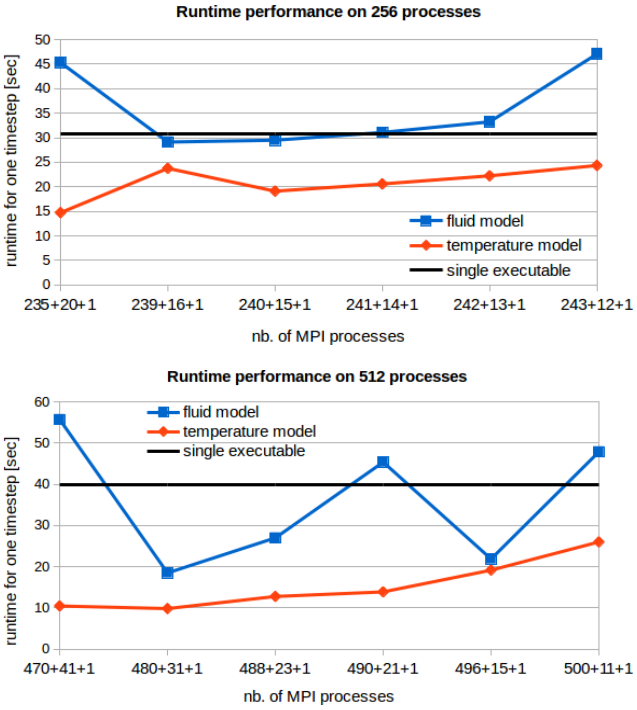

Figure 5: Runtime performance of the test configurations with 256 and 512 MPI processes.

time, the runtime for the temperature model increases monotonely.

The 256-process-tests show a different behaviour. The runtime of the fluid model takes its minimum at the 239+16+1 configuration, where the temperature model has a local runtime maximum. When further adding processes to the fluid model, its runtime as well as the temperature model's runtime increase.

For the configurations with 512 processes in total, the runtime of the fluid model exhibits strong variations. However, the temperature model runtime increases monotonely when shifting processes to the fluid model. The 480+31+1 configuration showed the lowest overall runtime of the coupled models, which was less than half of the single executable runtime.

To assess the performance of our implementations, we evaluated the parallel speedup and efficiency with respect to the 64-process-configuration. The speedup is defined as $S_n = T_{64}/T_n$ and the efficiency is $E_n = S_n/n$, where T_n is the runtime using $n = 64, 128, 256, 512$ processes in total. For the coupled models, we took the best configuration from each of the four test series to compute speedup and efficiency. Figure 6 shows the efficiency of the coupled models and of the single executable implementation in blue and black color, respectively. The data shows a decrease of efficiency for both the coupled models and the single executable with increasing number of processes. However, the single executable implementation suffers from a severe efficiency drop when going from 256 to 512 processes, while the coupled models' efficiency exhibits only a small decrease. This is reflected by the runtime ratio $T_n^{\text{s.e.}}/T_n^{\text{c.m.}}$ of the the single executable and the coupled models, which is depicted in orange color on Figure 5. The ratio jumps to a value greater than two for the best configuration with 512 processes in total, meaning that the single executable implementation consumes more than two times the runtime of the coupled models.

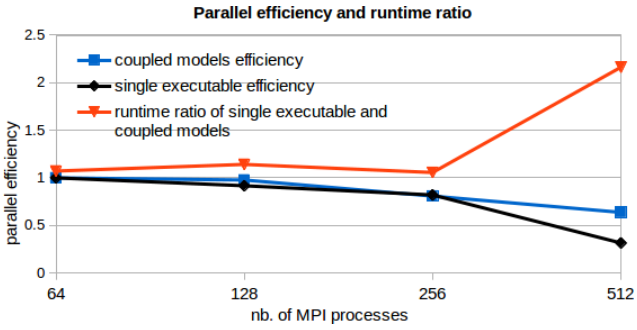

Figure 6: Efficiency and runtime ratio of single executable and coupled models implementation.

5 CONCLUSION

As an example of a multiphysics evolution problem, we gave the problem formulation of a natural convection wind dynamics scenario. The fluid dynamics are modeled by the instationary Navier-Stokes equations using the Boussinesq approximation for the momentum equation, and the temperature evolution is described by an instationary convection diffusion equation. We proposed

a first order in time operator splitting scheme which offers great flexibility for implementing the individual model components. We described the finite element discretization and the solvers we used for the implementation. We explained the concept of the OpenPALM software coupler tool, which we employed to couple our individual fluid and temperature models. For comparison, we also developed a single executable implementation including both models.

We carried out four series of performance tests, where we measured the runtime for computing one timestep using the coupled models as well as the single executable implementation. We varied the total number of MPI processes, ranging from 64 to 512, and we tested different parallel configurations for the OpenPALM-coupled model. In each of the test series, we could determine a configuration for the coupled models which showes a better runtime performance than the corresponding single executable run. However, except for the 64-process-tests, finding the best configuration required to investigate several configurations for each of the test series, as their performance varies in a non-monotonic way. In particular the scalability of the fluid dynamics model depends strongly on the distribution of the MPI processes. A reason for this unpredictable behaviour lies in the mapping of the MPI processes onto the compute nodes. The compute nodes of the JUROPA supercomputer are each equipped with two Intel Xeon X5570 quad-core processors, so there are eight cores per node. In the test runs, we allocated all eight cores on the nodes we used, putting one MPI process per core. Therefore, depending on the specific configuration at hand, the JUROPA scheduler maps MPI processes from both the fluid model and the temperature model onto the same compute node, or from one of the models and from the OpenPALM coupler. It even happens that some MPI processes of the two models, or of one of the models and OpenPALM, are mapped onto the cores of the same CPU on one of the nodes. As the models each run in their own executable, it means that the CPU is shared between the MPI processes of two distinct executables. This may result in bottlenecks with respect to computational effort, inter-node communication or memory usage.

Nevertheless, we could prove the potential of the coupling approach to yield a better runtime performance than the single executable implementation. In particular for the 512 processes tests, the coupled models show their superiority with respect to the parallel efficiency. This is due to the fact that the coupled models run concurrently in independent executables, requiring synchronization only after each timestep. Also, the size of the equation system and therefore the computational effort for each of the coupled models is smaller than for the single executable implementation.

As future work to improve the efficiency and to reduce the disruptive impact of the mapping of MPI processes to CPU cores, we aim for a hybrid MPI and OpenMP parallelization of the models. This allows to put one MPI process per CPU, hereby preventing from sharing the same CPU between the executables of the coupled models, while still exploiting all cores with the proper number of OpenMP threads.

Acknowledgements

Work on the OpenPALM coupler tool is funded by the German Research Foundation (DFG) as part of the DFG-project “Land use change and management effect on soil N_2O emissions in the Hai He river basin”.

The authors gratefully acknowledge the computing time granted by the John von Neumann Institute for Computing (NIC) and provided on the supercomputer JUROPA at Jülich Supercomputing Centre (JSC).

References

Anzt, H., Augustin, W., Baumann, M., Gengenbach, T., Hahn, T., Helfrich-Schkarbanenko, A., Heuveline, V., Ketelaer, E., Lukarski, D., Nestler, A., Ritterbusch, S., Ronnas, S., Schick, M., Schmidtobreick, M., Subramanian, C., Weiss, J.-P., Wilhelm, F., Wlotzka, M. (2012). HiFlow^3: A Hardware-Aware Parallel Finite Element Package. Springer, Tools for High Performance Computing 2011, p139-151

Buis, S., Piacentini, A., Déclat, D., The PALM Group (2006). PALM: A computational framework for assembling high-performance computing applications. Concurrency and Computation: Practice and Experience, Vol. 18-2, p231-245, doi:10.1002/cpe.914

Ern, A., and Guermond, J.-L. (2010). Theory and Practice of Finite Elements. Springer, Applied Mathematical Sciences, Vol. 159

Keyes, D.E., McInnes, L.C., Woodward, C., Gropp, W.D., 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., Jordan, K., Kaushik, D., Kaxiras, E., Koniges, A., Lee, K., Lott, A., Lu, Q., Magerlein, J., Maxwell, R., McCourt, M., Mehl, M., Pawlowski, R., Peters, A., Reynolds, D., Riviere, B., Rüde, U., Scheibe, T., Shadid, J., Sheehan, B., Shephard, M., Siegel, A., Smith, B., Tang, X., Wilson, C., Wohlmuth, B. (2011). Multiphysics Simulations: Challenges and Opportunities. Tech. Rep. ANL/MCS-TM-321, Argonne National Laboratory

Saad, Y. (2000). Iterative Methods for Sparse Linear Systems (2nd edition).

Taylor, C., and Hood, P. (1973). A numerical solution of the Navier-Stokes equations using the finite element technique. Elsevier, Computers & Fluids, Vol. 1

Preprint Series of the Engineering Mathematics and Computing Lab

recent issues

No. 2013-04 Nicolai Schoch, Stefan Suwelack, Rüdiger Dillmann, Vincent Heuveline: Simulation of Surgical Cutting in Soft Tissue using the Extended Finite Element Methods (X-FEM)

No. 2013-03 Martin Wlotzka, Vincent Heuveline, Edwin Haas, Steffen Klatt, David Kraus, Klaus Butterbach-Bahl, Philipp Kraft, Lutz Breuer: Dynamic Simulation of Land Management Effects on Soil N2O Emissions using a coupled Hydrology-Ecosystem Model on the Landscape Scale

No. 2013-02 Martin Baumann, Jochen Förstner, Jonas Kratzke, Sebastian Ritterbusch, Bernhard Vogel, Heike Vogel: Model-based Visualization of Instationary Geo-Data with Application to Volcano Ash Data

No. 2013-01 Martin Schindewolf, Björn Rocker, Wolfgang Karl, Vincent Heuveline: Evaluation of two Formulations of the Conjugate Gradients Method with Transactional Memory

No. 2012-07 Andreas Helfrich-Schkarbanenko, Vincent Heuveline, Roman Reiner, Sebastian Ritterbusch: Bandwidth-Efficient Parallel Visualization for Mobile Devices

No. 2012-06 Thomas Henn, Vincent Heuveline, Mathias J. Krause, Sebastian Ritterbusch: Aortic Coarctation simulation based on the Lattice Boltzmann method: benchmark results

No. 2012-05 Vincent Heuveline, Eva Ketelaer, Staffan Ronnas, Mareike Schmidtobreick, Martin Wlotzka: Scalability Study of HiFlow^3 based on a Fluid Flow Channel Benchmark

No. 2012-04 Hartwig Anzt, Armen Beglarian, Suren Chilingaryan, Andrew Ferrone, Vincent Heuveline, Andreas Kopmann: A unified Energy Footprint for Simulation Software

No. 2012-03 Vincent Heuveline, Chandramowli Subramanian: The Coffee-table Book of Pseudospectra

No. 2012-02 Dominik P.J. Barz, Hendryk Bockelmann, Vincent Heuveline: Electrokinetic optimization of a micromixer for lab-on-chip applications

No. 2012-01 Sven Janko, Björn Rocker, Martin Schindewolf, Vincent Heuveline, Wolfgang Karl: Software Transactional Memory, OpenMP and Pthread implementations of the Conjugate Gradients Method - a Preliminary Evaluation

No. 2011-17 Hartwig Anzt, Jack Dongarra, Vincent Heuveline, Piotr Luszczek: GPU-Accelerated Asynchronous Error Correction for Mixed Precision Iterative Refinement

No. 2011-16 Vincent Heuveline, Sebastian Ritterbusch, Staffan Ronnås: Augmented Reality for Urban Simulation Visualization

No. 2011-15 Hartwig Anzt, Jack Dongarra, Mark Gates, Stanimire Tomov: Block-asynchronous multigrid smoothers for GPU-accelerated systems

No. 2011-14 Hartwig Anzt, Jack Dongarra, Vincent Heuveline, Stanimire Tomov: A Block-Asynchronous Relaxation Method for Graphics Processing Units

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)

www.emcl.iwr.uni-heidelberg.de