Proceedings of ALGORITMY 2016
pp. 23–32

A HYBRID PARALLEL NUMERICAL ALGORITHM FOR THREE-DIMENSIONAL PHASE FIELD MODELING OF CRYSTAL GROWTH

PAVEL STRACHOTA AND MICHAL BENEŠ*

Abstract. We introduce a hybrid parallel numerical algorithm for solving the phase field formulation of the anisotropic crystal growth during solidification. The implementation is based on the MPI and OpenMP standards. The algorithm has undergone a number of efficiency measurements and parallel profiling scenarios. We compare the results for several variants of the algorithm and decide on the most efficient solution.

Key words. crystal growth, finite volume method, hybrid parallelization, MPI, OpenMP, phase field methods

AMS subject classifications. 65Y05, 65M08, 80A22, 74N05

1. Introduction. Numerical solution of evolutionary problems for PDE in three dimensions requires a large amount of computational resources. Parallel implementation of the numerical solvers therefore became an ubiquitous task. In this paper, the phase field model of crystal growth and its numerical solution described in [23, 22], [21] is revisited. In the first part, we review the problem formulation and the used numerical schemes. Results of the selected simulations are presented and the features of the presented approach are summarized. In the second part, we focus on the design of the hybrid MPI/OpenMP parallel implementation of the numerical algorithm and its performance on modern HPC clusters of ccNUMA nodes.

2. Problem Formulation. The phase field model [3] of the anisotropic [2, 21] Stefan problem [13] with surface tension is posed in a domain $\Omega \subset \mathbb{R}^3$ and time interval $\mathcal{J} = (0, T)$

$$\frac{\partial u}{\partial t} = \Delta u + L\frac{\partial p}{\partial t} \qquad \text{in } \mathcal{J}\times\Omega, \tag{2.1}$$

$$\alpha\xi^2\frac{\partial p}{\partial t} = \xi^2\nabla\cdot T^0\left(\nabla p\right) + f\left(u, p, \nabla p; \xi\right) \qquad \text{in } \mathcal{J}\times\Omega, \tag{2.2}$$

$$b_c\left(u\right) = 0 \qquad \text{on } \mathcal{J}\times\partial\Omega, \tag{2.3}$$

$$T^0\left(\nabla p\right)\cdot\boldsymbol{n} = 0 \qquad \text{on } \mathcal{J}\times\partial\Omega, \tag{2.4}$$

$$u|_{t=0} = u_{ini},\ \ p|_{t=0} = p_{ini} \qquad \text{in } \Omega \tag{2.5}$$

with the boundary condition operator b_c in (2.3) given by either $b_c\left(u\right) = u - u_{\partial\Omega}$ or $b_c\left(u\right) = \nabla u\cdot\boldsymbol{n}$. u represents the temperature field and p the phase field implicitly determining the diffuse phase interface Γ by the relation $\Gamma\left(t\right) = \left\{\boldsymbol{x}\in\mathbb{R}^3\middle|\, p\left(t,\boldsymbol{x}\right) = \frac{1}{2}\right\}$. The model parameters involve the melting point of the material u^*, the latent heat L, the attachment kinetics coefficient α, positive constants a, b, β [3] and the parameter ξ controlling the recovery of the sharp interface model [4]. The anisotropic operator T^0 (see [1, 2, 5]) is derived from the *dual* Finsler metric $\phi^0\left(\boldsymbol{\eta}^*\right)$, $\boldsymbol{\eta}^*\in\mathbb{R}^3$ as

*Dept. of Mathematics, Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague, (`pavel.strachota@fjfi.cvut.cz`).

$T^0(\boldsymbol{\eta}^*) = \phi^0(\boldsymbol{\eta}^*)\,\phi^0_\eta(\boldsymbol{\eta}^*)$ where $\phi^0_\eta = \left(\partial_{\eta_1^*}\phi^0, \partial_{\eta_2^*}\phi^0, \partial_{\eta_3^*}\phi^0\right)^{\mathrm{T}}$. The anisotropic surface energy ψ is incorporated into the model by the choice $\phi^0(\boldsymbol{\eta}^*) = |\boldsymbol{\eta}^*|\,\psi\left(-\frac{\boldsymbol{\eta}^*}{|\boldsymbol{\eta}^*|}\right)$ [17, 10]. For the possible forms of ψ, see e.g. [16].

The problem (2.1)–(2.5) describes the evolution of a two-phase single system where the solid region (a crystal) grows from an initial solid nucleus under supercooling conditions.

3. Numerical Solution. The spatial discretization is based on the cell centered finite volume method on a structured rectangular mesh $\mathcal{T}$ [9], using fourth order multipoint (MPFA) or second order central difference (CDFA) flux approximation schemes (for details see [24, 23]). In general, the resulting semidiscrete scheme is in the form

$$m(K)\,\frac{\mathrm{d}u^h_K}{\mathrm{d}t}(t) = \sum_{\sigma\in\mathcal{E}_K} F_{K,\sigma}(t,u_K) + Lm(K)\,\frac{\mathrm{d}p^h_K}{\mathrm{d}t}(t), \qquad \forall K\in\mathcal{T} \tag{3.1}$$

$$\alpha\xi^2 m(K)\,\frac{\mathrm{d}p^h_K}{\mathrm{d}t}(t) = \xi^2 \sum_{\sigma\in\mathcal{E}_K} T^0_{K,\sigma}(t,p_K) + m(K)\,f\left(u^h_K, p^h_K, \nabla^h p^h_K;\xi\right) \tag{3.2}$$

where $\mathcal{E}_K$ is the set of all faces of the finite volume $K\in\mathcal{T}$, m and $\tilde{m}$ are 3D and 2D measures in $\mathbb{R}^3$, respectively,

$$F_{K,\sigma}(t,u_K) = \tilde{m}(\sigma)\,\nabla^h u^h_\sigma(t)\cdot\boldsymbol{n}_{K,\sigma}, \qquad T^0_{K,\sigma}(t,p_K) = \tilde{m}(\sigma)\,T^0\left(\nabla^h p^h_\sigma(t)\right)\cdot\boldsymbol{n}_{K,\sigma}$$

are the numerical fluxes across the face $\sigma\in\mathcal{E}_K$,

$$\nabla^h p^h_K(t) = \left(\delta^h_1 p^h_K(t), \delta^h_2 p^h_K(t), \delta^h_3 p^h_K(t)\right)^{\mathrm{T}}$$

approximates $\nabla p(t,\boldsymbol{x}_K)$ at the point $\boldsymbol{x}_K\in K$, and

$$\nabla^h u^h_\sigma(t) = \left(\delta^h_1 u^h_\sigma(t), \delta^h_2 u^h_\sigma(t), \delta^h_3 u^h_\sigma(t)\right)^{\mathrm{T}}, \qquad \nabla^h p^h_\sigma(t) = \left(\delta^h_1 p^h_\sigma(t), \delta^h_2 p^h_\sigma(t), \delta^h_3 p^h_\sigma(t)\right)^{\mathrm{T}}$$

approximate $\nabla u(t,\boldsymbol{y}_\sigma)$, $\nabla p(t,\boldsymbol{y}_\sigma)$ at the point $\boldsymbol{y}_\sigma\in\sigma$ and $\boldsymbol{n}_{K,\sigma}$ is the unit normal to σ pointing out of K.

The integration in time is performed by the explicit 4th order Runge-Kutta-Merson (RKM) solver [6] with automatic adjustment of time step τ in agreement with the stability condition $\tau < ch^2$, $c>0$, $h=\max_{K\in\mathcal{T}}\operatorname{diam} K$.

The numerical schemes have been verified in a number of ways. For the isotropic case $\phi^0(\boldsymbol{\eta}^*) = |\boldsymbol{\eta}^*|$, the experimental order of convergence has been measured [23] in a specific situation [19] when the analytical solution is known. In the anisotropic cases, a number of studies have been performed, providing visual evidence of convergence with both flux approximation variants and testifying to the advantages of the MPFA scheme [21, 23]. In addition, many simulations have been carried out to study the capabilities of the phase field model. Some sample results are presented in Figure 3.1. Another important simulation in Figure 3.2 confirms the long term transformation of the crystal into the equilibrium Wulff shape [10] whose visualization is also provided. Such a simulation requires a very large amount of time to complete even with a parallel algorithm.

4. Hybrid Parallel Implementation. The numerical algorithm implemented in C/C++ as the InterTrack software package uses the MPI standard [14] for work sharing among multiple processes running in a distributed memory environment. Each

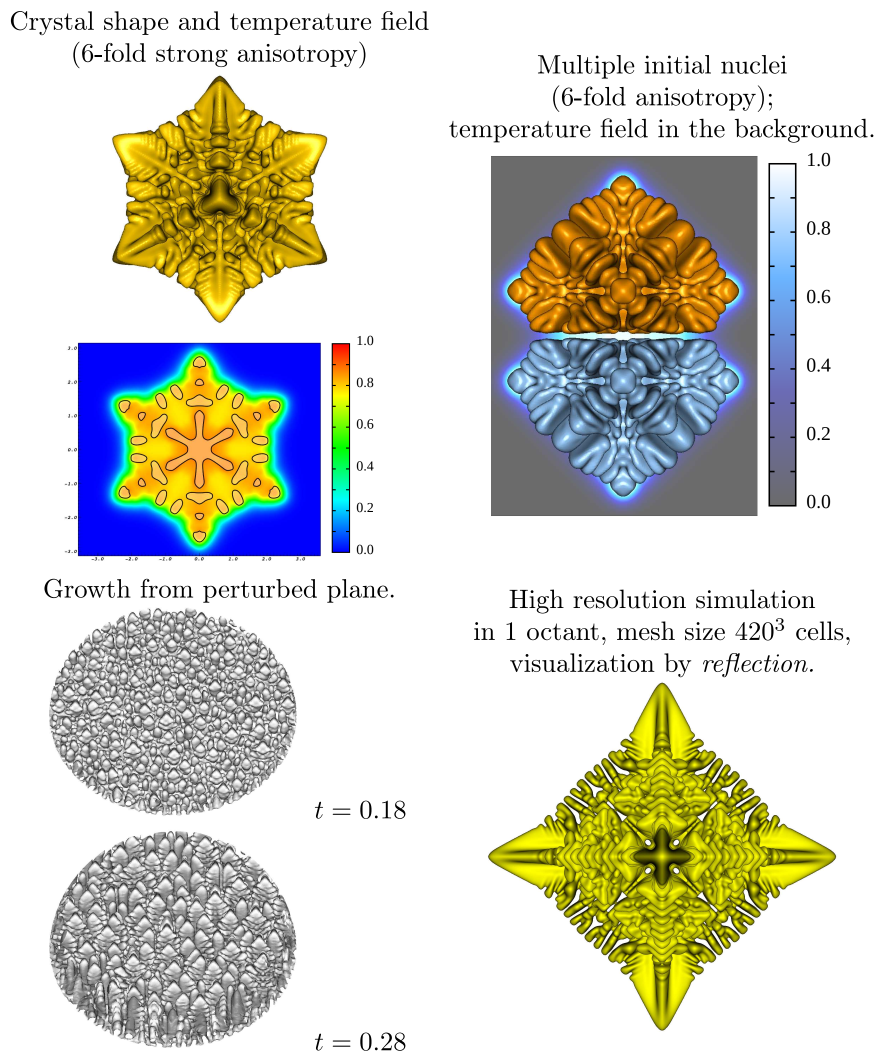

Crystal shape and temperature field
(6-fold strong anisotropy)
1.0
0.8
0.6
0.4
0.2
0.0
Multiple initial nuclei
(6-fold anisotropy);
temperature field in the background.
1.0
0.8
0.6
0.4
0.2
0.0
Growth from perturbed plane.
$t = 0.18$
$t = 0.28$
High resolution simulation
in 1 octant, mesh size 420^3 cells,
visualization by *reflection*.

Figure 3.1. *Sample results of crystal growth simulations. For detailed explanation of simulation settings, see [21].*

process can optionally further employ the OpenMP [8] technology for multithreaded parallel execution on SMP (Symmetric MultiProcessing) and ccNUMA (Cache Coherent Non-Uniform Memory Architecture, [20]) systems. As MPI is also routinely used to run more processes on a single multicore compute node, it is possible to use both technologies in a number of combinations on the same system.

Implementation of the Method of Lines. Separate treatment of the spatial and temporal discretization described in Section 3 is known as the method of lines [18]. The semidiscrete scheme (3.1)–(3.2) is effectively a system of ordinary differential equations which can be solved by any suitable method. Following this idea, the RKM solver is implemented as a separate and completely general solver of an ODE system

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{f}\left(t, \boldsymbol{x}\right). \tag{4.1}$$

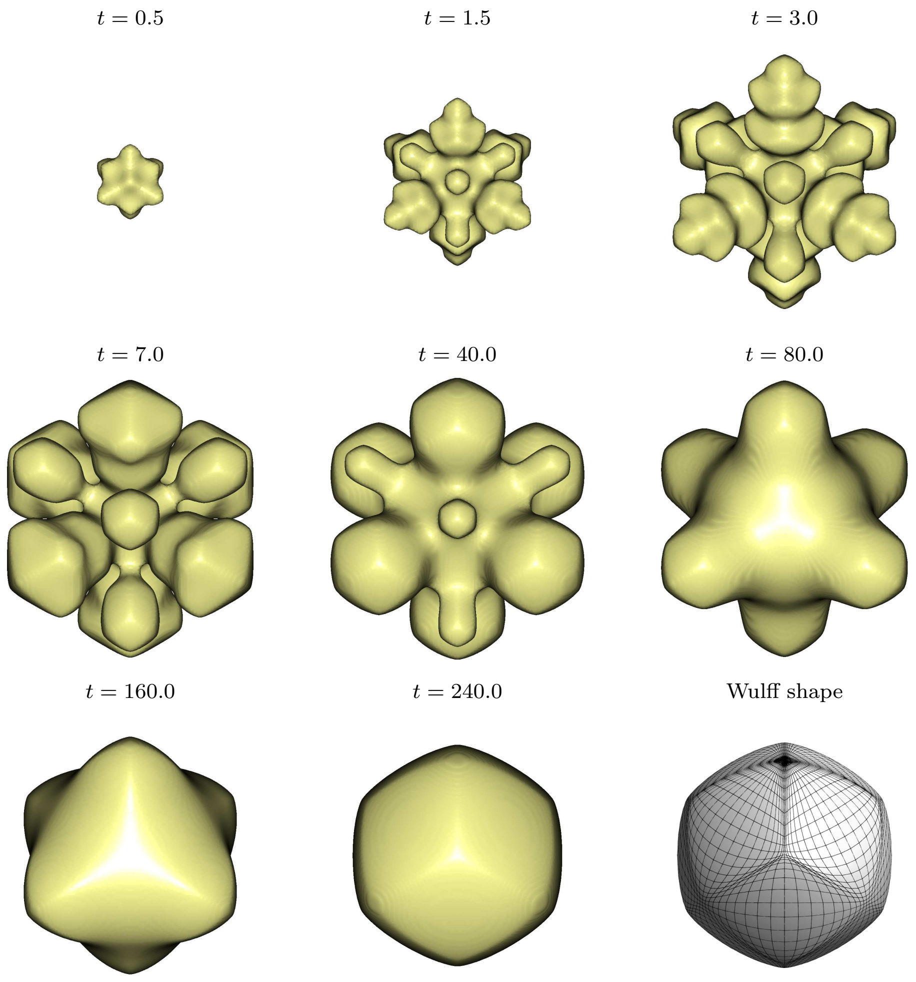


FIGURE 3.2. *Asymptotic convergence to the Wulff shape under thermal insulation conditions, 6-fold anisotropy. Parallel computation on four HP xw9400 compute nodes, each equipped by two dual core AMD Opteron 2216 @ 2.4 GHz processors. Mesh size* 144^3 *cells, total computation time 316 hours.*

However, the components of the vector $\boldsymbol{x}$ are allowed to be scattered into many *chunks* located in one contiguous memory area. Each chunk itself is a 1D array. This design is suitable for the method of lines where the spatial discretization uses ghost (auxiliary) cells to implement boundary conditions. These cells represent "holes" in the contiguous array that are not part of the solution vector $\boldsymbol{x}$. The whole finite volume scheme has to be contained in the right hand side $\boldsymbol{f}$ which is called by the RKM solver.

MPI Parallelization. Work sharing among individual processes (MPI ranks) is achieved by 1D domain decomposition along the z axis. Each rank solves the problem on its own sub-domain. However, on the boundaries, solution data from the neighboring subdomains are needed in each evaluation of $\boldsymbol{f}$ to calculate the numerical fluxes $F_{K,\sigma}$ and $T^0_{K,\sigma}$. The ghost cells that surround the subdomain from all sides are used with advantage *both* for halo exchange with the neighbors *and* for implementing the boundary conditions. As a result, the numerical scheme has the same form for all cells and can be written in the code at a single location. The structure of the right hand side $\boldsymbol{f}$ can be summarized in the following steps:

1. Prepare the boundary conditions by setting the appropriate ghost cells to the correct values.
2. Use nonblocking point to point (P2P) MPI communication to receive the halo data from the neighbors into the appropriate ghost cells.
3. Evaluate the finite volume scheme for all cells in the subdomain in three nested loops over the directions z, y, x (in this order).

MPI collective communication is also used in the RKM solver to adjust the time step in all ranks correctly (see Figure 4.1). All I/O operations are performed by a single (master) rank.

OpenMP Parallelization. Multithreaded execution is performed in OpenMP within a parallel region created by the directive `#pragma omp parallel`. To minimize the overhead of frequent creation and destruction of threads, the parallel region should be as large as possible. In InterTrack, the parallel region encompasses the main loop of the RKM solver (again, see Figure 4.1) . Inside this loop, work sharing is achieved by parallelizing the suitable `for` loops by `#pragma omp for`. This is used directly in the solver for preparing the arrays which are passed as the $\boldsymbol{x}$ argument to the right hand side $\boldsymbol{f}$. The right hand side is called by all threads and is required to implement work sharing on its own by using (orphaned) OpenMP constructs inside its body. After the evaluation of the $\boldsymbol{K}_i$ coefficients in the Runge-Kutta method, an error estimate is calculated by using the maximum norm of a given expression (Figure 4.1; for the design of adaptive time step RK solvers, see e.g. [7]). The maximum has to be collected from all threads and all MPI ranks by a reduction operation. This is implemented natively in MPI and OpenMP Version 3.0 or higher. For OpenMP Versions <3.0, a critical section is used instead.

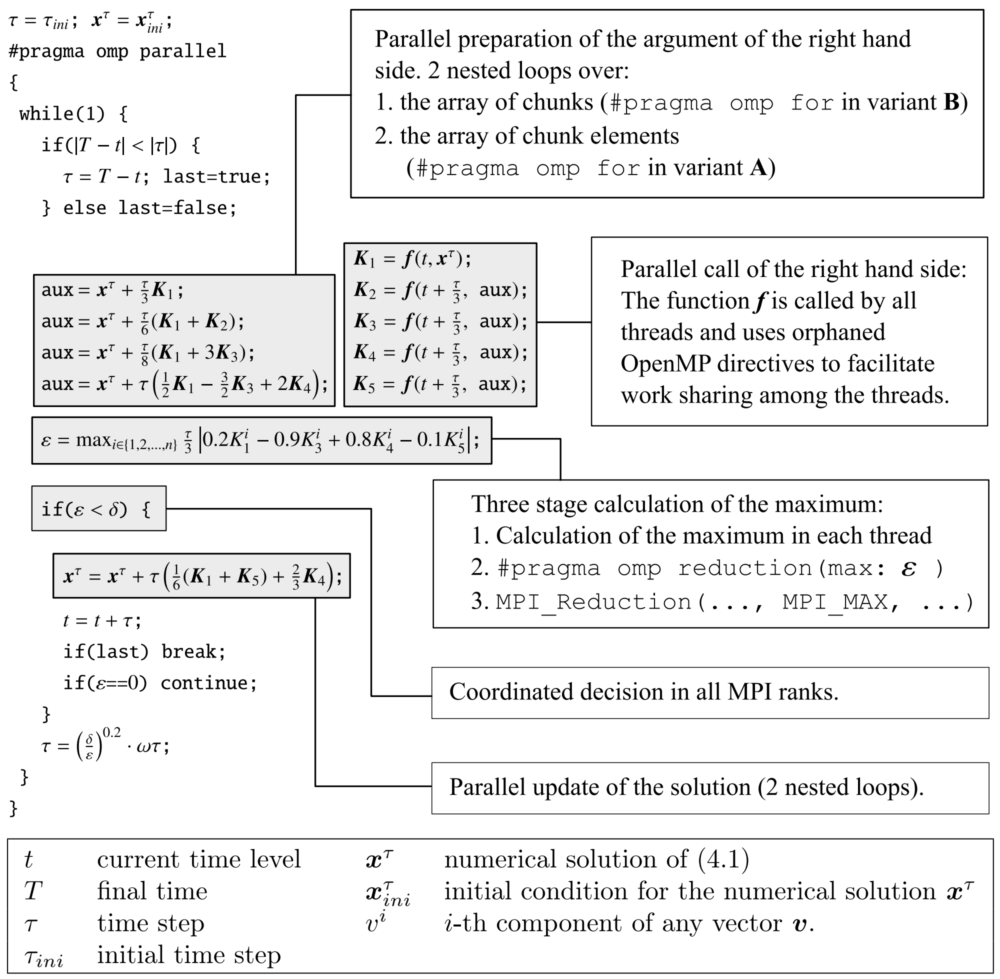
5. Parallel Computation Performance.

In this section, we discuss a number of parallel runs in different configurations and assess the *relative parallelization efficiency* of the run 2 with respect to a reference run 1

$$E_{\mathrm{rel}} = \frac{t_1}{t_2} \cdot \frac{p_1}{p_2} \cdot 100\%$$

where t_1, t_2 are computation times in two different parallelization settings and p_1, p_2 are the respective numbers of cores used.

MPI Parallelization Efficiency. The first presented test in Figure 5.1 shows parallel efficiency of a pure MPI-parallelized computation on a single compute node. The performance is very good and even superlinear speedup [11] can be observed. Fluctuations in efficiency with increasing number of cores p may partly be accounted to load imbalance when the mesh resolution is not divisible by p. However, the results of profiling Intertrack on a larger number of cores indicate that there are possibilities for improvement (Figure 5.2). On 64 cores and 4 nodes, the point to point communication overhead is still small compared to the time spent in collective MPI calls where no significant amounts of data are transferred. As collective operations represent an implicit barrier, this observation again implies load imbalance. Apart from the chosen mesh size, it can also be caused by several other factors such as extra boundary conditions evaluation in the first and last subdomain and dependence of computation duration on the input values.



```
$\tau = \tau_{ini}$; $\boldsymbol{x}^\tau = \boldsymbol{x}^\tau_{ini}$;
#pragma omp parallel
{
 while(1) {
   if($|T-t| < |\tau|$) {
      $\tau = T - t$; last=true;
   } else last=false;
      $\boldsymbol{K}_1 = \boldsymbol{f}(t, \boldsymbol{x}^\tau)$;
      aux = $\boldsymbol{x}^\tau + \frac{\tau}{3}\boldsymbol{K}_1$;
      $\boldsymbol{K}_2 = \boldsymbol{f}(t + \frac{\tau}{3}$, aux);
      aux = $\boldsymbol{x}^\tau + \frac{\tau}{6}(\boldsymbol{K}_1 + \boldsymbol{K}_2)$;
      $\boldsymbol{K}_3 = \boldsymbol{f}(t + \frac{\tau}{3}$, aux);
      aux = $\boldsymbol{x}^\tau + \frac{\tau}{8}(\boldsymbol{K}_1 + 3\boldsymbol{K}_3)$;
      $\boldsymbol{K}_4 = \boldsymbol{f}(t + \frac{\tau}{3}$, aux);
      aux = $\boldsymbol{x}^\tau + \tau\left(\frac{1}{2}\boldsymbol{K}_1 - \frac{3}{2}\boldsymbol{K}_3 + 2\boldsymbol{K}_4\right)$;
      $\boldsymbol{K}_5 = \boldsymbol{f}(t + \frac{\tau}{3}$, aux);
   $\varepsilon = \max_{i\in\{1,2,\dots,n\}} \frac{\tau}{3}\left|0.2K^i_1 - 0.9K^i_3 + 0.8K^i_4 - 0.1K^i_5\right|$;
   if($\varepsilon < \delta$) {
      $\boldsymbol{x}^\tau = \boldsymbol{x}^\tau + \tau\left(\frac{1}{6}(\boldsymbol{K}_1 + \boldsymbol{K}_5) + \frac{2}{3}\boldsymbol{K}_4\right)$;
      $t = t + \tau$;
      if(last) break;
      if($\varepsilon$==0) continue;
   }
   $\tau = \left(\frac{\delta}{\varepsilon}\right)^{0.2} \cdot \omega\tau$;
 }
}
```

Parallel preparation of the argument of the right hand side. 2 nested loops over:
1. the array of chunks (`#pragma omp for` in variant **B**)
2. the array of chunk elements
   (`#pragma omp for` in variant **A**)

Parallel call of the right hand side: The function $\boldsymbol{f}$ is called by all threads and uses orphaned OpenMP directives to facilitate work sharing among the threads.

Three stage calculation of the maximum:
1. Calculation of the maximum in each thread
2. `#pragma omp reduction(max:` ε `)`
3. `MPI_Reduction(..., MPI_MAX, ...)`

Coordinated decision in all MPI ranks.

Parallel update of the solution (2 nested loops).

t	current time level	$\boldsymbol{x}^\tau$	numerical solution of (4.1)
T	final time	$\boldsymbol{x}^\tau_{ini}$	initial condition for the numerical solution $\boldsymbol{x}^\tau$
τ	time step	v^i	i-th component of any vector $\boldsymbol{v}$.
τ_{ini}	initial time step		

FIGURE 4.1. *Hybrid parallel OpenMP/MPI Runge-Kutta-Merson solver.*

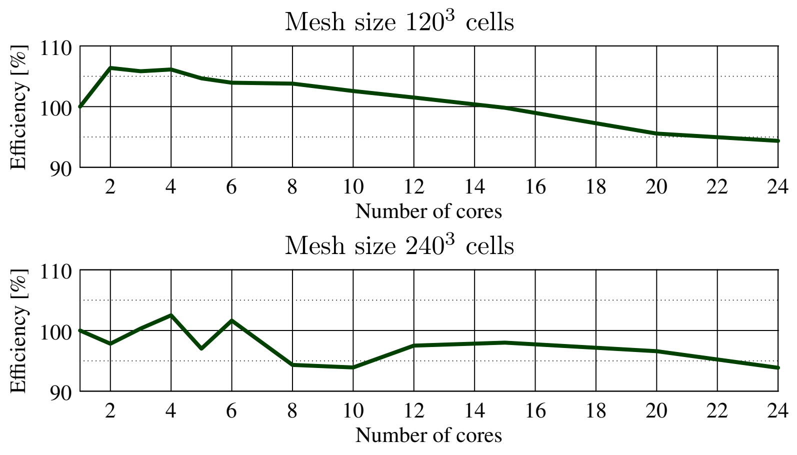


FIGURE 5.1. *Parallel efficiency of an MPI computation on different numbers of cores on a single ccNUMA node with two 12-core AMD Opteron 6172 @ 2.1 GHz, 12×512 KB L3 cache, CentOS 5.5 Linux, Intel C/C++ compiler 10.0, LAM MPI 7.1.4.*

Total cores	32	64	64
Distribution	2 nodes × 2 CPU × 8 cores	4 nodes × 2 CPU × 8 cores	4 nodes × 2 CPU × 8 cores
Mesh size	240 × 240 × 240	420 × 420 × 420	480 × 480 × 480
Computation	91.00 %	79.60 %	85.30 %
MPI collective	6.98 %	16.87 %	11.50 %
MPI P2P	2.02 %	3.43 %	3.20 %

FIGURE 5.2. *Allinea performance reports for* InterTrack *on the* ANSELM *cluster. Each node equipped by two 8-core Intel XEON Sandy Bridge E5-2665 @ 2.4GHz, 20 MB L3 cache.*

Hybrid Parallelization Efficiency. Hybrid parallelization theoretically improves load balancing. Using larger subdomains may diminish the relative differences in computation time. In addition, using an appropriate scheduling strategy for loop parallelization (e.g. dynamic instead of static [15]) can help. Surprisingly, the results in Figures 5.4 and 5.5 show that the performance of the hybrid algorithm actually goes down with increasing number of threads and decreasing number of MPI ranks. It seems to be generally worse than the performance of pure MPI parallelization on a single node. However, a deeper analysis reveals the following problems:

- In the numerical scheme, there is a natural need to share some variables between several functions. The original code used static variables, but the OpenMP–parallelized version requires these variables to be private for each thread. A straightforward solution is to declare them as `threadprivate`, but the overhead associated with their use is excessive (smaller bars in Figure 5.4). A much better solution is to use a structure of automatic (local) private variables and pass a pointer to this structure to any function that works with these variables (larger bars in Figure 5.4).
- When using MPI only, the communication overhead may be compensated by the use of separate address spaces which implicitly rules out cacheline conflicts and frequent inefficient memory accesses into a different NUMA domain (provided that the processes are bound to cores or sockets). This is why MPI parallelization is usually very efficient even on a single node. With OpenMP, on the other hand, one needs to be careful. The first implementation of the RKM solver (variant **A**) parallelizes the `for` loop over the chunk elements (which could possibly help if the number of chunks were very small). In contrast to that, the code in the right hand side $\boldsymbol{f}$ parallelizes the outermost (or optionally the second outermost) loop and accesses the chunk elements sequentially. The result is a cross pattern of memory use (Figure 5.3, left) leading to inevitable accesses across NUMA domains and poor cache utilization.

With the above in mind, we also tested variant **B** of the RKM solver parallelizing the loop over the array of chunks (Figure 5.3, right). The resulting efficiency measurement is in Figure 5.6 which proves that now pure MPI, pure OpenMP, and any possible combination thereof perform equally well.

All computations were executed with the `OMP_PROC_BIND` environment variable set to `TRUE`, binding the OpenMP threads to CPU cores. MPI process binding capabilities were not used.

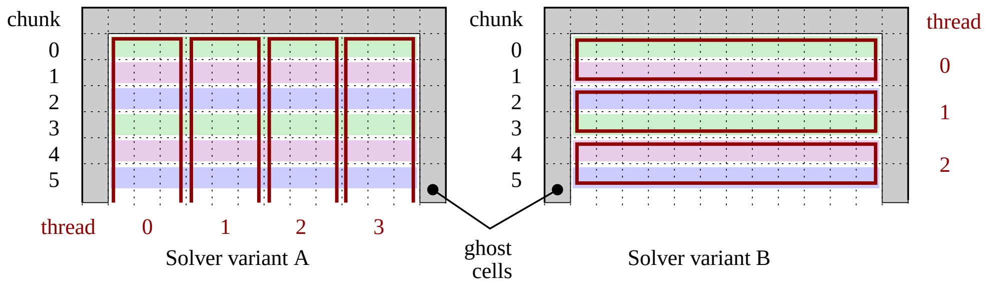


FIGURE 5.3. *Schematic of mesh partitioning and thread correspondence in variants **A** and **B** of the hybrid parallel OpenMP/MPI RKM solver.*

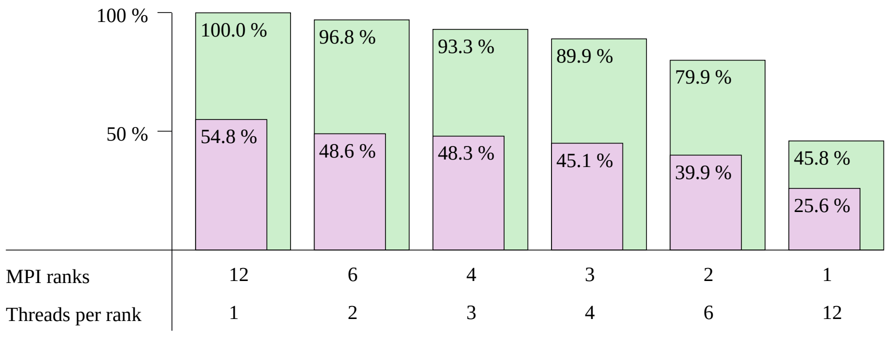


FIGURE 5.4. *Relative parallel efficiency of hybrid OpenMP/MPI computations on a single ccNUMA node with two 6-core AMD Opteron 2427 @ 2.2GHz, 6×512 KB L2 cache, CentOS 5.5 Linux, Intel C/C++ compiler 10.0, LAM MPI 7.1.4. Reference value 100% is for pure MPI run. Mesh size 120^3 cells, RKM solver variant **A**. Two series with a different technique of variable sharing between functions: 1) `private` automatic variables and pointer passing (larger bars), 2) static `threadprivate` variables (smaller bars).*

Transfer Layer Comparison. In Figure 5.7, the relative efficiency of multi-node computation is evaluated for different communication interconnects. In particular, gigabit Ethernet and 40Gbit Infiniband are compared. For more than two nodes, Infiniband allows slightly better performance, but the difference is not very significant. That is in accordance to the profiling results in Figure 5.2 where the communication only accounts for a small fraction of time.

6. Conclusion.

We have developed and tested a hybrid parallel algorithm for numerical simulation of crystal growth. The algorithm exhibits satisfactory scaling up to approximately 100 cores with pure MPI, which is sufficient for our current purposes. Based on the efficiency measurements, the additional OpenMP parallelization has been tuned to provide very good performance on many-core systems. If needed, hybrid parallelization can possibly significantly extend the scaling on clusters of ccNUMA nodes as it reduces the problem of inefficient 1D domain decomposition into very thin slices. In addition, the generic parallel RKM solver can easily be carried over to other problems.

Acknowledgments. This work has been supported by the project of the Czech Science Foundation no. GB14-36566G Multidisciplinary research centre for advanced materials.

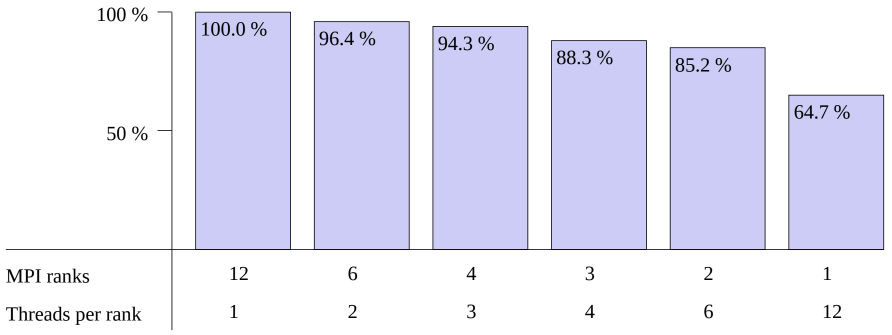


FIGURE 5.5. *Relative parallel efficiency of hybrid OpenMP/MPI computations on a single ccNUMA node with two 6-core AMD Opteron 2427 @ 2.2GHz, 6×512 KB L2 cache. Reference value 100% is for pure MPI run. Mesh size* 240^3 *cells, RKM solver variant* $\boldsymbol{A}$.

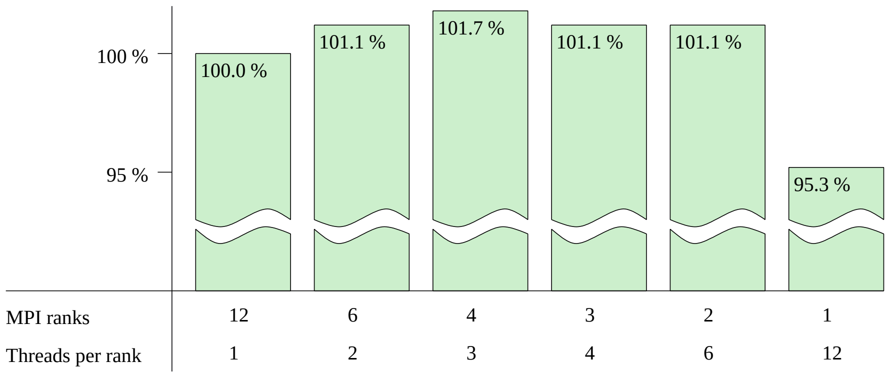


FIGURE 5.6. *Relative parallel efficiency of hybrid OpenMP/MPI computations on a single ccNUMA node with two 6-core AMD Opteron 2427 @ 2.2 GHz, 6×512 KB cache, CentOS 5.5 Linux, Intel C/C++ compiler 10.0, LAM MPI 7.1.4. Reference value 100% is for pure MPI run. Mesh size* 120^3 *cells, RKM solver variant* $\boldsymbol{B}$.

REFERENCES

[1] G. Bellettini and M. Paolini. Anisotropic motion by mean curvature in the context of Finsler geometry. *Hokkaido Math. J.*, 25(3):537–566, 1996.

[2] M. Beneš. Anisotropic phase-field model with focused latent-heat release. In *FREE BOUNDARY PROBLEMS: Theory and Applications II*, volume 14 of *GAKUTO International Series in Mathematical Sciences and Applications*, pages 18–30, 2000.

[3] M. Beneš. Mathematical and computational aspects of solidification of pure substances. *Acta Math. Univ. Comenianae*, 70(1):123–151, 2001.

[4] M. Beneš. Diffuse-interface treatment of the anisotropic mean-curvature flow. *Appl. Math-Czech.*, 48(6):437–453, 2003.

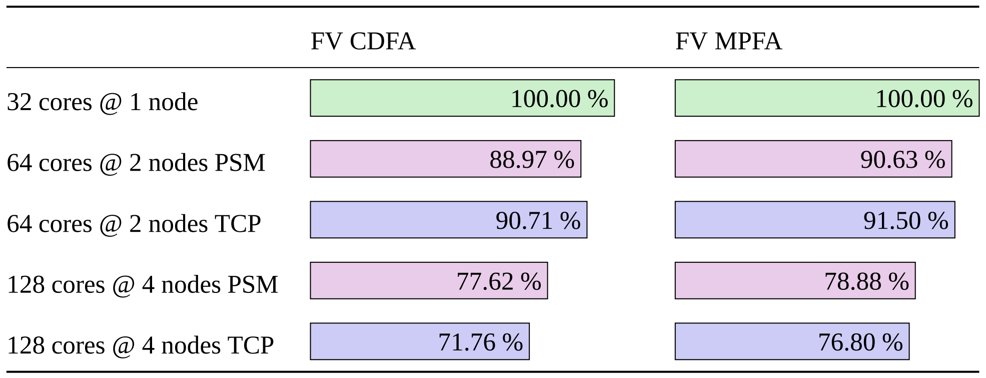
[5] M. Beneš. Computational studies of anisotropic diffuse interface model of microstructure formation in solidification. *Acta Math. Univ. Comenianae*, 76:39–59, 2007.

[6] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations.* Wiley, Chichester, 2003.

[7] J. Christiansen. Numerical solution of ordinary simultaneous differential equations of the 1st order using a method for automatic step change. *Numer. Math.*, 14:317–324, 1970.

[8] L. Dagum and R. Menon. Openmp: An industry standard API for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998.

[9] R. Eymard, T. Gallouët, and R. Herbin. Finite volume methods. In P. G. Ciarlet and J. L. Lions, editors, *Handbook of Numerical Analysis*, volume 7, pages 715–1022. Elsevier, 2000.



	FV CDFA	FV MPFA
32 cores @ 1 node	100.00 %	100.00 %
64 cores @ 2 nodes PSM	88.97 %	90.63 %
64 cores @ 2 nodes TCP	90.71 %	91.50 %
128 cores @ 4 nodes PSM	77.62 %	78.88 %
128 cores @ 4 nodes TCP	71.76 %	76.80 %

TCP TCP/IP protocol over gigabit Ethernet
PSM Performance Scaled Messaging (interface to Intel Infiniband HCA [12]).

FIGURE 5.7. *Comparison of relative parallel efficiency of MPI computations on multiple cc-NUMA nodes of the* Hyperion *cluster, using different OpenMPI transfer layers. Each node equipped by two 16-core AMD Opteron 6272 @ 2.1 GHz, 16 MB L3 cache, Intel QLE7340 True Scale Fabric HCA. Infiniband switch QLogic 12200, CentOS 6.5 Linux, gcc 4.4.3, OpenMPI 1.6.5. Reference value 100% is for the run on 32 cores on a single node.*

[10] M. E. Gurtin. *Thermomechanics of Evolving Phase Boundaries in the Plane.* Oxford Mathematical Monographs. Oxford University Press, 1993.

[11] J. L. Gustafson. Fixed time, tiered memory, and superlinear speedup. In *Proc. 5th Distributed Memory Computing Conference*, pages 1255–1260, 1990.

[12] Intel Corporation. Intel true scale fabric architecture: Enhanced HPC architecture and performance, 2013. White paper.

[13] A. M. Meirmanov. *The Stefan Problem.* De Gruyter Expositions in Mathematics. Walter de Gruyter, 1992.

[14] Message Passing Interface Forum. MPI: A message-passing interface standard version 3.1, 2015.

[15] OpenMP Architecture Review Board. OpenMP application program interface version 4.0, July 2013.

[16] M. PunKay. Modeling of anisotropic surface energies for quantum dot formation and morphological evolution. In *NNIN REU Research Accomplishments*, pages 116–117. University of Michigan, 2005.

[17] S. L. R. E. Napolitano. Three-dimensional crystal-melt Wulff-shape and interfacial stiffness in the Al-Sn binary system. *Phys. Rev. B*, 70(21):214103, 2004.

[18] W. E. Schiesser. *The Numerical Method of Lines: Integration of Partial Differential Equations.* Academic Press, San Diego, 1991.

[19] A. Schmidt. Computation of three dimensional dendrites with finite elements. *J. Comput. Phys.*, 125:293–3112, 1996.

[20] P. Stenström, T. Joe, and A. Gupta. Comparative performance evaluation of cache-coherent NUMA and COMA architectures. In *ISCA '92 Proceedings of the 19th annual international symposium on Computer architecture*, volume 20, pages 80–91. ACM New York, NY, USA, May 1992.

[21] P. Strachota. *Analysis and Application of Numerical Methods for Solving Nonlinear Reaction-Diffusion Equations.* PhD thesis, Czech Technical University in Prague, 2012.

[22] P. Strachota and M. Beneš. A multipoint flux approximation finite volume scheme for solving anisotropic reaction-diffusion systems in 3D. In J. Fořt, J. Fürst, J. Halama, R. Herbin, and F. Hubert, editors, *Finite Volumes for Complex Applications VI - Problems & Perspectives*, pages 741–749. Springer, 2011.

[23] P. Strachota and M. Beneš. Design and verification of the MPFA scheme for three-dimensional phase field model of dendritic crystal growth. In A. Cangiani, R. L. Davidchack, E. Georgoulis, A. N. Gorban, J. Levesley, and M. V. Tretyakov, editors, *Numerical Mathematics and Advanced Applications 2011: Proceedings of ENUMATH 2011, Leicester, September 2011*, pages 459–467. Springer Berlin Heidelberg, 2013.

[24] P. Strachota, M. Beneš, and J. Tintěra. Towards clinical applicability of the diffusion-based DT-MRI visualization algorithm. *J. Vis. Commun. Image R.*, 23(2):387–396, 2012.