UPPSALA
UNIVERSITET

IT Licentiate theses
2012-003

Towards an Adaptive Solver for High-Dimensional PDE Problems on Clusters of Multicore Processors

MAGNUS GUSTAFSSON

UPPSALA UNIVERSITY
Department of Information Technology

Towards an Adaptive Solver for High-Dimensional PDE Problems on Clusters of Multicore Processors

Magnus Gustafsson
magnus.gustafsson@it.uu.se

March 2012

Division of Scientific Computing
Department of Information Technology
Uppsala University
Box 337
SE-751 05 Uppsala
Sweden

http://www.it.uu.se/

Dissertation for the degree of Licentiate of Licentiate of Philosophy in Scientific Computing

ISSN 1404-5117
Printed by the Department of Information Technology, Uppsala University, Sweden

Abstract

Accurate numerical simulation of time-dependent phenomena in many spatial dimensions is a challenging computational task apparent in a vast range of application areas, for instance quantum dynamics, financial mathematics, systems biology and plasma physics. Particularly problematic is that the number of unknowns in the governing equations (the number of grid points) grows exponentially with the number of spatial dimensions introduced, often referred to as the curse of dimensionality. This limits the range of problems that we can solve, since the computational effort and requirements on memory storage directly depend on the number of unknowns for which to solve the equations.

In order to push the limit of tractable problems, we are developing an implementation framework, HAParaNDA, for high-dimensional PDE-problems. By using high-order accurate schemes and adaptive mesh refinement (AMR) in space, we aim at reducing the number of grid points used in the discretization, thereby enabling the solution of larger and higher-dimensional problems. Within the framework, we use structured grids for spatial discretization and a block-decomposition of the spatial domain for parallelization and load balancing. For integration in time, we use exponential integration, although the framework allows the flexibility of other integrators to be implemented as well. Exponential integrators using the Lanzcos or the Arnoldi algorithm has proven a succesful and efficient approach for large problems. Using a truncation of the Magnus expansion, we can attain high levels of accuracy in the solution.

As an example application, we have implemented a solver for the time-dependent Schrödinger equation using this framework. We provide scaling results for small and medium sized clusters of multicore nodes, and show that the solver fulfills the expected rate of convergence.

To my family

List of Papers

This thesis is based on the following papers, referred to in the text by their Roman numerals.

I M. Gustafsson and S. Holmgren. An implementation framework for solving high-dimensional PDEs on massively parallel computers. In *Numerical Mathematics and Advanced Applications 2009. Proceedings of the 8th European Conference on Numerical Mathematics and Advanced Applications, ENUMATH 2009*, pp. 417–424, Springer-Verlag, Berlin, 2010. [1]
Contributions: The author of this thesis did the implementation work and wrote the manuscript in discussion with the second author of the paper. The ideas were developed in collaboration between the authors.

II M. Gustafsson, K. Kormann and S. Holmgren. Communication-efficient algorithms for numerical quantum dynamics. In *Proceedings of PARA 2010: State of the Art in Scientific Computing, 10th International Workshop*, Lecture Notes in Computer Science, Vol. 7134, pp. 368–378, Springer Berlin / Heidelberg, 2012.
Contributions: The ideas were developed in close collaboration between the authors of the paper. The author of this thesis did most of the implementation work and prepared the manuscript in close collaboration with the second author, in discussion with the third author.

III M. Gustafsson, A. Nissen and K. Kormann. Stable difference methods for block-structured adaptive grids. Technical report 2011–022, Department of Information Technology, Uppsala University, 2011.
Contributions: The ideas were developed in close collaboration between the authors of the paper. The author of this thesis developed the code infrastructure and did the bulk part of the implementation. The manuscript was written in close collaboration between all three authors of the paper.

IV M. Gustafsson, J. Demmel and S. Holmgren. Numerical evaluation of the Communication-Avoiding Lanczos algorithm. Technical report 2012–001, Department of Information Technology, Uppsala University, 2012.
Contributions: The ideas were developed in collaboration with the second author. All implementation was done by the author of this thesis. The manuscript was prepared by the author of this thesis in discussion with the other authors of the paper.

Reprints were made with permission from the publishers.

[1] With kind permission from Springer Science and Business Media.

Contents

Chapter 1

Introduction

Accurate and efficient numerical simulation of time-dependent partial differential equations (PDEs) is a demanding task in scientific computing, in particular when the number of spatial degrees of freedom is large. Massive-scale computing resources, high-order accurate numerical techniques and adaptive schemes in space and time are required in order to solve such problems efficiently. As high-performance computers increase in number of processors, cores and memory capacity, the attainable problem size is constantly growing. However, recent development trends in high-performance computers towards clusters of multi-core nodes have introduced new challenges due to multi-level parallelism and complex memory hierarchies. Aiming at scalability on massive-scale computers, it is important that the numerical methods are implemented with the underlying parallel architectures in mind and that communication between processors and between cores is minimized.

High-dimensional problems (i.e. problems that need to be discretized in more than 3 spatial dimensions) appear in a vast range of research applications, such as quantum dynamics [34], systems biology [14], plasma physics [19] and financial mathematics [22]. The general class of problems that are considered in this thesis are d-dimensional, time-dependent, linear PDEs

$$\frac{\partial u}{\partial t} = \hat{P}(\mathbf{x}, t)\, u, \tag{1.1}$$

where $\hat{P}$ is the continuous linear operator describing the dynamics in the model. Discretizing (1.1) in space, leaving the time variable continuous, transforms the PDE into a large semi-discrete system of ordinary differential equations (ODEs), an approach generally referred to as the method of lines. We have

$$\frac{du}{dt} = P(\mathbf{x}, t)\, u. \tag{1.2}$$

where P is the discretized linear operator matrix, possibly dependent on space as well as time.

As an illustrating example of a high-dimensional PDE-problem, consider the time-dependent

Schrödinger equation (TDSE), governing the quantum dynamics of molecular processes,

$$\mathrm{i}\hbar\frac{\partial\psi(\mathbf{x},t)}{\partial t} = H\psi(\mathbf{x},t), \tag{1.3}$$

$$\psi(\mathbf{x},0) = \psi_0, \tag{1.4}$$

for $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and time $t \geq 0$. Here, $\hbar$ is Planck's reduced constant and H is the Hamiltonian operator, describing the kinetic and potential energy of the system,

$$H = -\sum_{i=1}^{d} \frac{\hbar^2}{2m_i}\frac{\partial^2}{\partial x_i^2} + V(\mathbf{x},t), \tag{1.5}$$

where m_i is the mass corresponding to dimension i, and V the potential modeling interactions within the system. The solution to equation (1.3), ψ, is referred to as the wave function and its L_2 norm is finite and preserved.

The remainder of this thesis is organized as follows. First, in Chapter 2 we give an overview of spatial discretization methods for high-dimensional PDEs, including a discussion on structured adaptive mesh refinement. In Chapter 3 we describe how exponential integration can be used for efficient time-propagation of large, linear PDE problems. Implementation aspects of HAParaNDA, our large-scale framework for adaptive and parallel discretization of high-dimensional problems are summarized in Chapter 4. We conclude by giving a summary of the attached papers in Chapter 5 followed by an outline of future work in Chapter 6.

Chapter 2

Spatial Discretization

In the framework that we present in this thesis, we have restricted the spatial discretization to structured grids. Compared to unstructured grids, there is a performance advantage with structured grids since the relations between all grid points are implicit in the grid description. With this restriction, we are free to use any discretization technique we find suitable, e.g. finite difference methods, finite element methods, finite volume methods, pseudospectral methods or radial basis functions. In this thesis we focus on high-order accurate finite difference methods.

Spatial discretization of PDEs using grid-based techniques is often straightforward to implement and computationally efficient. However, due to the exponential growth in grid size as the number of spatial dimensions grows, the total grid size quickly grows prohibitively large. On todays medium sized clusters (on the order of 1000 cores), detailed simulation of up to 6–7 degrees of freedom may be within reach for direct solution. In order to approach problems with even larger number of spatial dimensions, application-specific approximative schemes are necessary, aspects that are not discussed any further in this thesis.

2.1 Finite Differences

Finite difference methods are used to discretize the spatial derivatives that occur in a PDE like (1.1). In each grid point, a weighted sum of the surrounding grid points is computed, approximating the value of the derivative in that point. An operator that arises from a finite difference discretization can be viewed either as a band-diagonal matrix or as a stencil operator applied in each individual grid point. The numerical efficiency can be increased by using high-order terms and, in principle, any order of accuracy can be attained [15].

Since the finite difference approximation in each grid point only depends on the nearest-neighbor points in the grid, it is a simple matter to parallelize the sparse matrix-vector multiplication (SpMV) that corresponds to applying the stencil over the entire grid. However, the data access pattern of a stencil-like operator becomes problematic with

higher dimensional grids. To understand why, it is important to understand how data are stored in a computers memory and how the cache subsystems respond to data accesses.

When a d-dimensional array is stored in the memory of a computer it is mapped to a one-dimensional array, usually according to a row-major or column-major order. We assume a row-major storage format in the remainder of this section. For instance, accessing the elements in a two-dimensional array, there will be a stride of unit length between the elements in each row and a stride equal to the row length between each element in a column. Extending this to higher dimensions, the stride between consecutive elements in a particular dimension would be the product of the grid length in all the lower dimensions.

Modern cache-based computers exploit temporal locality of data accesses based on the idea that if some data is used at some point, it is expected that it will be used again in a near future. In order to avoid having to fetch the data repeatedly from memory, wasting many clock cycles of latency for each access, the data is stored temporarily in a much faster cache memory. However, due to the small size of the cache memory, it is not possible to store a large number of data values. If the span of memory locations that need to be accessed repeatedly is larger than the size of the cache, the data that has been stored in the cache will constantly be overwritten by new accesses. In the end, this will lead to poor overall performance since a lot of data has to be constantly reloaded from the slow memory. In order to improve cache-reuse for stencil computations, optimizations such as cache-blocking, cache-tiling and time-skewing are often used [9, 10, 29, 30]. For general sparse matrices, transformations such as the Cuthill-McKee algorithm [7] can be used to reduce the matrix to a more condensed band matrix.

2.2 Summation-by-parts Operators

Summation-by-parts (SBP) operators are finite difference schemes with central stencils in the interior and one-sided stencils on the boundaries of a (sub)domain. Combined with simultaneous approximating term (SAT) weak boundary conditions this discretization leads to provably time-stable numerical approximations. The combined scheme, referred to as SBP-SAT, can be used in single- as well as in multi-block configurations. In Paper IV we describe how this can be implemented in conjuction with a space-adaptive refinement scheme.

The SBP-SAT technique allows for high-order accurate numerical approximations. However, there will be a loss of accuracy along the block boundaries in a multi-block configuration; with an interior accuracy of order $2m$ the global accuracy will be of order $m + 2$.

2.3 Structured Adaptive Mesh Refinement

Adaptive mesh refinement (AMR) is typically used when the solution to a problem is confined to small regions of the spatial domain (localized solutions) or if there are artefacts

in the solution, such as shocks or oscillations, that require higher resolution only when and where they occur. By adapting the mesh to the solution, the computational resources can be utilized more efficiently since fewer grid points are needed to represent the solution. This effectively leads to a reduction in computational effort and memory requirements. The goal is to use as few grid points as possible to achieve a specified tolerance, without wasting too much effort on finding a suitable grid decomposition. For time-dependent problems, the grid also needs to be dynamically re-adapted as the solution propagates in space.

Structured adaptive mesh refinement (SAMR) is a particular form of AMR, where the logical decomposition of the spatial domain is restricted to rectangular patches. Each patch represents a structured grid discretization of a rectangular subdomain. The SAMR approach will greatly simplify the implementation with lower complexity and management overhead compared to a general AMR implementation. The remainder of this thesis will focus on SAMR.

Approaches to SAMR The approach to SAMR that has become most commonly used was originally developed by Berger and Oliger [4] and later on modified by Berger and Colella [2]. In this approach, the mesh is adapted by superimposing refined, logically rectangular patches on top of a coarse grid. This can be done iteratively by adding more patches on top of the refined grid patches until the grid represents the solution well enough. During computations, the solution is maintained on all levels of refinement, interpolating/projecting the solution between the levels. In the original algorithm by Berger and Oliger [4], the patches were allowed to be arbitrarily oriented with respect to the underlying grid patches. Berger and Colella [2] restricted the patches to be aligned with each other, a modification that greatly simplified the algorithm and reduced the overhead of managing the grid patches [28].

Block-wise SAMR is based on the Berger-Colella approach but introduces further restrictions on the patches in order to simplify the implementation and reduce the overhead. In this approach, the patches are required to conform to a block-decomposition of the grid and grid refinement is performed block-wise rather than point-wise. As such, if just a single point in a block requires refinement, the whole block is refined, either by increasing the number of grid points within the block or by generating more blocks. The subdomain that is represented by (the conjunction of) the superimposed patch(es) corresponds exactly to the subdomain represented by the coarser patch.

Dynamic Load Balancing As the mesh is adapted to the solution, data dependencies are introduced within and between the patches of the mesh. Since these data-dependencies are not known before-hand, they must be accounted for during the adaptation step. Furthermore, the data dependencies change dynamically as the mesh is propagated in time with the solution. Therefore, an important aspect of any AMR-implementation is to exploit the locality of data and to balance the workload evenly among the processors. The load balancing aspect is particularly important on large scale parallel computers but as parallelism increases at all levels (even in desktop computers and laptops), workload distribution needs to be considered also at small scale. Space-filling

curves [32] are recursive locality-preserving mappings from a multidimensional order to a linear order. Once the multidimensional structure has been linearised, the workload is balanced by distributing equal portions of the linear order to the processors. There are many flavors of space-filling curves with varying complexity and characteristics. For load-balancing purposes, Hilbert and Morton ordering are most commonly used [32]. The Morton (z-code) ordering is simpler to derive than the Hilbert ordering, but the Hilbert ordering preserves locality better.

Chapter 3

Exponential Integrators for Time Propagation

If P is independent of time, the general solution to (1.2) is given by

$$u(t) = e^{Pt}u(0). \tag{3.1}$$

On the other hand, if P is time-dependent the system of ODEs cannot be solved exactly using (3.1). However, the exponential formulation can be used successively on small time intervals if the time-dependent operator is expanded using the Magnus expansion [24]. At time step k we have

$$u_{k+1} = e^{\Omega_k}u_k, \tag{3.2}$$

where Ω_k is a sum of nested commutators of P, evaluated at different temporal locations. Truncating after the first term yields

$$\Omega_k = h_t P\left(t_n + \frac{h_t}{2}\right), \tag{3.3}$$

corresponding to the second-order accurate exponential midpoint rule. Here, h_t is the time step size and t_n is the time at time interval n. For the Schrödinger equation, Hochbruck and Lubich [17] developed efficient time-integration techniques using the Magnus expansion and exponential integrators. Furthermore, Kormann *et al.* [21] demonstrated how a truncated Magnus expansion was used to develop an efficient and accurate h,p-adaptive time-integrator with global error control.

The complexity of typical methods for explicitly computing the exponential of a general $n \times n$ matrix A is $O(n^3)$ [25]. Exploiting sparsity and structure of the matrix helps reducing the computational cost, but in general the exponential e^A is dense even if the matrix A is sparse and it is not feasible to compute and store the full exponential for large problems [25]. Instead, for large sparse problems, a common approach is to compute an approximate matrix exponential using Krylov subspace methods for eigendecomposition [25, 31]. The Lanczos algorithm (Hermitian matrices) or the Arnoldi algorithm

(general matrices) are commonly used for this purpose, computing an approximate eigenvalue decomposition by projecting the original problem onto a Krylov subspace of much lower dimension.

Let A be an $n \times n$ matrix and v_1 a normalized vector of length n. Then, after m iterations of Lanczos/Arnoldi with v_1 as initial vector, an orthonormal basis V_m^A spanning the Krylov subspace $\mathcal{K}_m(A, v_1)$ has been generated and along with it a corresponding projection matrix H_m^A, satisfying the relation

$$AV_m^A = V_m^A H_m^A + h_{m+1,m} v_{m+1} e_m^T. \tag{3.4}$$

The eigenvalues of H_m^A are approximate eigenvalues of A (Ritz values) and the corresponding eigenvectors of A (Ritz vectors) can be generated from the basis V_m^A and each respective eigenvector of H_m^A (c.f. Paper IV for more details on Ritz values and Ritz vectors).

Rewriting (3.2) in terms of the resulting Krylov basis, we have (c.f. [16, 25, 31])

$$u_{k+1} = V_m^{\Omega_k} \exp\left(H_m^{\Omega_k}\right) e_1 ||u_k||_2, \tag{3.5}$$

where $V_m^{\Omega_k}$ is the orthonormal basis spanning the Krylov subspace $\mathcal{K}_m(\Omega_k, u_k)$, $H_m^{\Omega_k}$ the corresponding projection matrix of Ω_k onto $\mathcal{K}_m$ and e_1 is the first column of the unit matrix I_m.

Chapter 4

HAParaNDA

The development of efficient numerical scientific software for computation on large scale parallel clusters is a complex and time consuming process. Deep knowledge of the underlying computer architecture and communication protocols is required as well as insight into the numerical methods and the application. Furthermore, numerical scientific implementations are often developed towards a given application and lack flexibility as to reuse the code for solving other problems. Our goal is to provide application experts with tools that enable flexible, simple and intuitive implementation of complex problems, relaxing them from low level implementation details and performance optimization. This functionality should be supplied without sacrificing the time-to-solution performance.

We are currently in the course of developing an implementation framework with these requirements in mind. HAParaNDA[1] is implemented in C++ and provides flexible constructs for setting up and solving high-dimensional time-dependent linear PDEs, using block-decomposed structured grids for spatial discretization. Adaptive mesh refinement is not yet implemented in HAParaNDA, but Section 4.3 briefly describes the algorithms and data structures needed to extend the existing code to support SAMR.

For the Schrödinger equation, we have implemented exponential integrators (cf. Chapter 3) based on the Lanczos algorithm for time-propagation, an approach that we believe will be feasible also for other applications. However, the flexibility of the HAParaNDA framework allows easy extension of other time-stepping methods if needed.

4.1 Framework

The concept of an object oriented software framework can be described as a "reusable, semi-complete application that can be specialized to produce custom applications" [13]. From this standpoint, our framework should provide the user with a hierarchy of abstract classes that define an intuitive interface for expressing high-dimensional time-dependent

[1] HAParaNDA is an acronym for High-Dimensional Parallel Numerical Dynamics. The original Haparanda is a town in the far northern parts of Sweden and has nothing to do with our implementation framework.

PDEs, with some key kernels implemented for use of-the-shelf, yet allow flexible extension by adding new classes with specialized properties.

Within our framework, there are three basic abstractions to consider; `Mesh`, `TimePropagator` and `LinearOperator`. `Mesh` is the base class for block-decomposed grids. Depending on the problem, the user can choose to use an `EquidistantMesh`, an `AdaptiveMesh` or implement a specialized mesh. The base class `TimePropagator` is further specialized by adding another level of abstract classes. Currently, we have included the `ExponentialIntegrator`, which in turn has been specialized with a Lanczos and an Arnoldi implementation. The `LinearOperator` base class defines the interface to applying the the linear operator matrix to a `Mesh`. Currently, we provide an implementation for distributed equidistant block-decomposed grids using stencil operators.

4.2 Implementation and parallelization

High-order finite difference stencils have been implemented for discretization of spatial derivatives but, as mentioned in the previous section, the framework allows for other approximation schemes to be used if wanted. In (1.2) P corresponds to an $N \times N$ sparse operator matrix, N being the total number of grid points in the discretized domain ($N = \prod_{i=1}^{d} n_i$ in the simplest case of an equidistant grid). When the grid is equidistant, we exploit the structure of the linear operator and implement it as a stencil operator, storing just the finite difference weights and the space/time-dependent diagonal terms. This "matrix free" approach releives us from having to store the entire $N \times N$ matrix and efficiency is improved compared to a general sparse matrix-vector multiplication. For problems using adaptive meshes, we can still implement a stencil-like operator, but the stencils will be larger and more complex (cf. Paper III for details regarding the stencils in the SBP-SAT implementation of SAMR).

We parallelize HAParaNDA at two levels; between nodes and within nodes. Between compute nodes we use MPI (Message Passing Interface) for distribution of data and synchronization of computations among the processors. Within each multicore node we use OpenMP for multi-threaded work sharing of compute-intensive loops. This type of hybrid parallelization has advantages compared to an MPI-only implementation, such as better memory efficiency, better mapping to the multicore architecture in the nodes and better scalability [27].

We use a block-structured decomposition of the grid, subdividing the computational domain into equally-sized blocks and distributing them to the compute nodes. The stencil operation described in Section 4.1 is parallelized locally in each node by slicing the corresponding block in the least unit-stride dimension. Along block boundaries, the data dependencies of the finite difference stencil will require neighboring blocks to exchange the data values that are closest to the corresponding boundary. The number of data values that need to be sent depends on the order of the finite difference approximations; in general p layers of data are affected, where the order of accuracy is $2p$. In Paper I we describe an algorithm for exchanging data one dimension at a time in order to reduce the memory usage overhead due to data duplication. This turned out *not* to be a succesful strategy

when going to larger scale clusters, and in particular not in combination with the s-step matrix powers kernel that we used in Paper II. Therefore, we abandoned this idea and instead start communication in all dimensions at once and overlap the communication latency with the computations that are not affected by block boundaries.

4.3 Adaptive Mesh Refinement

Currently, we have not implemented adaptive mesh refinement in parallel at large scale. We have developed a serial prototype version of SAMR with SBP-SAT operators and in this section we describe the data structures and algorithms that are necessary in order to make the transition towards a parallel implementation of those techniques in HAParaNDA.

4.3.1 Grid Representation and Refinement

We build upon the block-structured decomposition of the grid described above and let all blocks represent the same number of grid points. Refinement is carried out by binary splitting of blocks one dimension at a time in an anisotropic manner (i.e. blocks are not refined uniformly in all dimensions). As a block is split, two new blocks are generated in place of the original block, thereby increasing the spatial resolution. In higher dimensions, we expect this conservative scheme to generate fewer grid blocks than if isotropic refinement is used.

Due to the anisotropic refinement, each grid block corresponds to a logical d-dimensional hyperrectangle (d-orthotope) in space. We do not put any restrictions on the aspect ratio (the quotient of the length of the edges in each respective dimension) of the blocks but we enforce a 2:1 refinement ratio between blocks that share a block edge.

In order to keep track of the relationships between the blocks, we use binary kd-trees [1]. Each anisotropic refinement results in a block being split into two, thus generating two children nodes to the node corresponding to the block that was split. In the kd-tree, we distinguish between internal nodes and leaf nodes. The internal nodes are merely logical entities for representation of the dependencies between the blocks in the grid and do not contain any data. All data are stored in the leaf nodes of the kd-tree.

4.3.2 Load Balancing

The organization of the grid blocks using kd-trees naturally translates to a Morton ordering [32]. For every block that is split (regardless of which dimension), each of its children nodes is assigned a bit value 0 or 1 based on their position in space. Aggregating the bit values from the top to the bottom of the kd-tree, each leaf node gets a unique bit index corresponding to its Morton ordering in the tree. Thus, by maintaining this small piece of extra information in the kd-tree, we immediately get the linear order corresponding to

a Morton ordering of the grid blocks, which we then can use to distribute the workload among the processors.

4.4 Related Work

Adaptive mesh refinement has been an active area of research for several decades. As a result, many software packages providing AMR-functionality have arised, most of which are available to the public as open source libraries. In this section we will give an overview of some of the available libraries and put them in contrast to the features of our AMR-strategies.

The most widely used strategy for SAMR is the patch-based hierarchical grid refinement introduced by Berger and Colella [2]. Among this class of SAMR-packages are Chombo [6], SAMRAI [33], AMROC [11] and AMRClaw [3, 5].

In contrast, Racoon [12] and PARAMESH [23] use a block-wise refinement of a flat grid where the refinement of a block results in the block being logically replaced with a number of new blocks with higher resolution. Moreover, they both apply a strictly isotropic refinement, which means that 4 (in 2D) or 8 (in 3D) new blocks will be created on each refinement. Extending this to d dimensions, 2^d new blocks would be generated on each refinement, resulting in a massive fan-out in the number of created blocks in higher dimensions. This is the motivation behind the anistropic refinement strategy that we use.

Chapter 5

Summary of Attached Papers

Paper I

In this paper we present an early version of the HAParaNDA code (even though it is not referred to as HAParaNDA in the paper). We describe the basic implementation choices that we made in order to achieve scalability on small scale clusters for numerical quantum dynamics problems on equidistant cartesian grids and discuss the parallelization issues related to clusters of multicore nodes. We show strong scaling up to 256 cores as long as each processor has enough work to do and close to ideal weak scaling up to 32 processors (128 cores), which was the largest number of processors that we had access to at that time.

Paper II

This paper extends the discussion on exponential integrators and brings the performance analysis to larger scale clusters. We evaluate different communication-optimized variants of the Lanczos algorithm (among them the s-step Lanczos algorithm by Kim and Chronopoulos [20]) in order to see how the performance is affected by reducing the amount of communication. An important observation in this paper was that the s-step algorithm manages to avoid communication in the scalar products to some extent but since it requires an extra SpMV in each outer iteration the overall runtime is close to the same as for the standard algorithm.

Paper III

This report describes a prototype MATLAB implementation of a block-adaptive solver using the SBP-SAT technique for coupling of non-conforming grid blocks (i.e. grid blocks that differ in refinement in one or more dimensions). In this paper, we describe how to

construct a suitable grid to a given function and how to impose necessary restrictions on the resulting grid. We use binary *kd*-trees for organizing the grid and a 2:1 refinement ratio between neighboring blocks. Between blocks that are conforming, we use central finite differences. This paper is an extension of the work by Nissen *et al.* [26] where stability was proven for multi-block grids. We prove stability also for more elaborate grid structures and show that our method works for longer time integrations.

Paper IV

The Lanczos algorithm is not expected to scale very well when going towards massive scale parallel computers, due to frequent vector updates and inner products. A remedy to this, first proposed by Kim and Chronopoulos [20] (also discussed in paper II) and more recently by Hoemmen [18], is to unroll the iterations and compute several vectors of the Krylov subspace in one sweep. This means that s SpMVs $[Av_k, A^2v_k, \ldots, A^sv_k]$ are computed directly in a matrix powers kernel and then all s vectors are orthogonalized as a block with respect to all the previously computed blocks of basis vectors. This avoidance of computation is expected to pay off at all levels, between nodes in a distributed parallel environment as well as locally within each compute node due to the smaller amount of data movement between levels in the memory hierarchy. In this paper, we have implemented the Communication-Avoiding Lanczos algorithm (CA-Lanczos) and evaluated its numerical performance in conjunction with some commonly used orthogonalization and restarting techniques. Our findings showed that CA-Lanczos can successfully replace the original Lanczos algorithm in many cases, with a potential gain in performance and no significant loss of accuracy.

Chapter 6

Future Work

In this thesis we have described the current status of the development of the HAParaNDA framework for numerical solution of high-dimensional time-dependent PDEs. However, there are many important aspects that we have not yet been able to address and in this section we give an overview of the research directions that we plan for the future.

Large-scale implementation of spatial adaptivity So far, the numerical scheme for spatial adaptivity that we presented in Paper III has only been implemented as a prototype code in MATLAB. The long-term goal is to have this functionality implemented at large scale within the HAParaNDA framework. In principle, the current implementation of the framework allows this type of block-adaptive scheme to be deployed, but there are still some obstacles to overcome before this can be done. Most importantly, we need to find an efficient and flexible way to represent the SBP-SAT operators and generalize them to higher dimensions.

Parallel implementation of CA-Lanczos for exponential integration In Paper IV, we studied the numerical performance of the Communication-Avoiding Lanczos algorithm [18]. Our results were promising and we plan to implement this variant of Lanczos in our exponential integrator for a potential increase in performance and scalability. This also inlcudes developing a parallel implementation of the Tall-Skinny QR algorithm for efficient orthogonalization of the basis vectors in parallel. We expect the CA-Lanczos algorithm to be more stable and more efficient than the s-step algorithm that we implemented in Paper II.

Auto-tuning In the current implementation of HAParaNDA, the user must provide the framework with some basic information about the architecture on which it is compiled, such as cache-sizes and the number of sockets and cores in each shared-memory node. Based on this information, the framework strives to optimize the performance by blocking and tiling techniques (c.f. Section 2.1). However, due to the complexity of modern computer architectures it might be difficult to make an optimal deterministic choice of optimization parameters. By means of *auto-tuning* (see e.g. [8]) we can instruct the code

(compile time or runtime) to automatically find the parameter values that gives the best performance on a given architecture. Ideally, all performance-critical parameters would be determined automatically at compile time, but this is a far-fetched goal that has not yet been adressed.

Evaluation on massive-scale computers Even though we have shown good scalability on small and medium-sized clusters, there might be design choices that need to be reconsidered as we approach massive scale clusters. In particular, communication patterns between nodes and in the memory hierarchy must be monitored and analyzed in detail in order to identify performance bottlenecks that hamper scalability.

Acknowledgements

Many thanks to my advisor Sverker Holmgren for being supportive and help me develop my ideas. The freedom you have given me to find my own way towards this thesis has been invaluable. Further thanks to my co-advisor Michael Thuné for your support and for bringing structure into my work.

Martina, your company, your love and your understanding have helped me all along the way. Vanja, thank you for all the joy you bring and for giving me a smile when I need it the most.

Mom and dad, thank you for your support in life and for always being there. Mia, Sofia and Niclas, growing up with you was a pleasure. Thank you all for giving me perspective and teaching me the true values of life.

I would like to thank Katharina Kormann for great collaboration in many parts of my work, for reading this manuscript and for many fruitful discussions. I am very grateful to Anna Nissen, Elias Rudberg, Emanuel Rubensson and Martin Tillenius for invaluable discussions and ideas for how to develop my work further.

Part of the work in Paper IV was done during a 10 week visit at University of California, Berkeley. Many thanks to James Demmel and the other members of the Berkeley Benchmarking and Optimization Group for hosting me during these weeks and for helping me develop my ideas.

Thank you all at TDB for creating a good working environment. Many ideas have sprung out of the coffee breaks and box lunches.

Financial support from Ångpanneföreningens Forskningsstiftelse, Anna-Maria Lundin's stipendiefond and stiftelsen Lars Hiertas minne have made my travels in work possible.

Bibliography

[1] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18:509–517, 1975.

[2] M. J. Berger and P. Collela. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82:64–84, 1989.

[3] M. J. Berger and R. J. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM Journal on Numerical Analysis*, 35(6):2298–2316, 1998.

[4] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equation. *Journal of Computational Physics*, 53:484–512, 1984.

[5] R. Blikberg and T. Sørevik. Load balancing and OpenMP implementation of nested parallelism. *Parallel Computing*, 31(10-12):984–998, 2005.

[6] P. Colella, D. T. Graves, N. D. Keen, T. J. Ligocki, D. F. Martin, P. W. McCorquodale, D. Modiano, P. O. Schwartz, T. D. Sternberg, and B. V. Straalen. Chombo software package for AMR applications - design document, `https://seesar.lbl.gov/anag/chombo/chombodesign-3.0.pdf`, April 2009.

[7] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, ACM '69, pages 157–172. ACM, 1969.

[8] K. Datta. *Auto-tuning Stencil Codes for Cache-Based Multicore Platforms*. PhD thesis, Computer Science Division, University of California, Berkeley, December 2009.

[9] K. Datta, S. Kamil, S. Williams, L. Oliker, J. Shalf, and K. Yelick. Optimization and performance modeling of stencil computations on modern microprocessors. *SIAM Review*, 51(1):129–159, 2009.

[10] K. Datta, M. Murphy, V. Volkov, S. Williams, J. Carter, L. Oliker, D. Patterson, J. Shalf, and K. Yelick. Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pages 4:1–4:12, 2008.

[11] R. Deiterding. Detonation structure simulation with AMROC. In L. Yang, O. Rana, B. Di Martino, and J. Dongarra, editors, *High Performance Computing and Communications*, volume 3726 of *Lecture Notes in Computer Science*, pages 916–927. Springer Berlin / Heidelberg, 2005.

[12] J. Dreher and R. Grauer. Racoon: A parallel mesh-adaptive framework for hyperbolic conservation laws. *Parallel Computing*, 31(8-9):913–932, 2005.

[13] M. Fayad and D. C. Schmidt. Object-oriented application frameworks. *Communications of the ACM*, 40(10):32–38, 1997.

[14] L. Ferm, A. Hellander, and P. Lötstedt. An adaptive algorithm for simulation of stochastic reaction-diffusion processes. *Journal of Computational Physics*, 229(2):343 – 360, 2010.

[15] B. Fornberg. Calculation of weights in finite difference formulas. *SIAM Review*, 40(3):685–691, 1998.

[16] M. Gustafsson. A pde solver framework optimized for clusters of multi-core processors. Master's thesis, School of Enginering, Uppsala University, Januari 2009.

[17] M. Hochbruck and C. Lubich. On magnus integrators for time-dependent schrödinger equations. *SIAM Journal on Numerical Analysis*, 41(3):945–963, 2004.

[18] M. F. Hoemmen. *Communication-Avoiding Krylov Subspace Methods*. PhD thesis, EECS Department, University of California, Berkeley, April 2010.

[19] M. Holmström. Hybrid modeling of plasmas. In G. Kreiss, P. Lötstedt, A. Målqvist, and M. Neytcheva, editors, *Numerical Mathematics and Advanced Applications 2009*, pages 451–458. Springer, Berlin, 2010.

[20] S. K. Kim and A. T. Chronopoulos. A class of Lanczos-like algorithms implemented on parallel computers. *Parallel Computing*, 17(6-7):763–778, 1991.

[21] K. Kormann, S. Holmgren, and H. O. Karlsson. Global error control of the time-propagation for the Schrödinger equation with a time-dependent Hamiltonian. *Journal of Computational Science*, 2:178 – 187, 2011.

[22] G. Linde, J. Persson, and L. von Sydow. A highly accurate adaptive finite difference solver for the Black–Scholes equation. *International Journal of Computer Mathematics*, 86:2104–2121, 2009.

[23] P. MacNeice, K. M. Olson, C. Mobarry, R. deFainchtein, and C. Packer. Paramesh : A parallel adaptive mesh refinement community toolkit. *Computer Physics Communications*, 126:330–354, 2000.

[24] W. Magnus. On the exponential solution of differential equations for a linear operator. *Communications on Pure and Applied Mathematics*, 7(4):649–673, 1954.

[25] C. Moler and C. van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

[26] A. Nissen, G. Kreiss, and M. Gerritsen. Stability at nonconforming grid interfaces for a high order discretization of the Schrödinger equation. Technical Report 2011-017, Department of Information Technology, Uppsala University, 2011.

[27] R. Rabenseifner, G. Hager, and G. Jost. Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes. In *Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on*, pages 427–436, 2009.

[28] J. Rantakokko and M. Thuné. Parallel structured adaptive mesh refinement. In R. Trobec, M. Vajteršic, and P. Zinterhof, editors, *Parallel computing*, pages 147–173. Springer London, 2009.

[29] G. Rivera and C.-W. Tseng. A comparison of compiler tiling algorithms. In S. Jähnichen, editor, *Compiler Construction*, volume 1575 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 1999.

[30] G. Rivera and C.-W. Tseng. Tiling optimizations for 3D scientific computations. In *Supercomputing, ACM/IEEE 2000 Conference*, SC '00, pages 32:1–32:23, Nov. 2000.

[31] R. B. Sidje. Expokit: A software package for computing matrix exponentials. *ACM Transactions on Mathematical Software*, 24(1):130–156, 1998.

[32] J. D. Teresco, K. D. Devine, and J. E. Flaherty. Partitioning and dynamic load balancing for the numerical solution of partial differential equations. In A. M. Bruaset and A. Tveito, editors, *Numerical Solution of Partial Differential Equations on Parallel Computers*, volume 51, pages 55–88. Springer Berlin, 2006.

[33] A. M. Wissink, R. D. Hornung, S. R. Kohn, S. S. Smith, and N. Elliott. Large scale parallel structured AMR calculations using the SAMRAI framework. In *Proceedings of the 2001 ACM/IEEE conference on Supercomputing*, SC '01, pages 22:1–22:11, 2001.

[34] A. Zewail. Laser femtochemistry. *Science*, 242:1645–1653, 1988.

Recent licentiate theses from the Department of Information Technology

2012-002 Fredrik Bjurefors: *Measurements in Opportunistic Networks*

2012-001 Gunnika Isaksson-Lutteman: *Future Train Traffic Control – Development and deployment of new principles and systems in train traffic control*

2011-006 Anette Löfström: *Intranet Use as a Leadership Strategy*

2011-005 Elena Sundkvist: *A High-Order Accurate, Collocated Boundary Element Method for Wave Propagation in Layered Media*

2011-004 Niclas Finne: *Towards Adaptive Sensor Networks*

2011-003 Rebecka Janols: *Tailor the System or Tailor the User? How to Make Better Use of Electronic Patient Record Systems*

2011-002 Xin He: *Robust Preconditioning Methods for Algebraic Problems, Arising in Multi-Phase Flow Models*

2011-001 David Eklöv: *Efficient Methods for Application Performance Analysis*

2010-005 Mikael Laaksoharju: *Let Us Be Philosophers! Computerized Support for Ethical Decision Making*

2010-004 Kenneth Duru: *Perfectly Matched Layers for Second Order Wave Equations*

2010-003 Salman Zubair Toor: *Managing Applications and Data in Distributed Computing Infrastructures*

2010-002 Carl Nettelblad: *Using Markov Models and a Stochastic Lipschitz Condition for Genetic Analyses*

UPPSALA
UNIVERSITET

Department of Information Technology, Uppsala University, Sweden