FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG
INSTITUT FÜR INFORMATIK (MATHEMATISCHE MASCHINEN UND DATENVERARBEITUNG)

Lehrstuhl für Informatik 10 (Systemsimulation)

Massively Parallel Multilevel Finite Element Solvers on the Altix 4700

Tobias Gradl, Ulrich Rüde

Technical Report 07-12

Massively Parallel Multilevel Finite Element Solvers on the Altix 4700

Tobias Gradl, Ulrich Rüde

Chair for System Simulation, University Erlangen-Nuremberg, Cauerstr. 6, 91058 Erlangen, Germany

Introduction

In the most recent TOP-500 list, published in June 2007, the HLRB II at the Leibniz Computing Center of the Bavarian Academy of Sciences is ranked at position 10 for solving a linear system with 1.58 million unknowns at a rate of 56.5 Teraflops in the Linpack benchmark. However, this impressive result is of little direct value for scientific applications. There are few real life problems that could profit from the solution of a general dense system of equations of such a size.

Typical supercomputer applications today fall primarily in two classes. They are either variants of molecular dynamics simulations or they require the solution of sparse linear systems as they e.g. arise in finite element problems for the solution of partial differential equations (PDEs). These two application classes become apparent when one reviews the history of the Gordon Bell prize, the most prestigious award in high end computing. All Gordon Bell prizes fall in either of these two categories. It is also interesting to see the correlation between architecture and application. For example, when the Earth simulator, a classical vector architecture, was leading the TOP-500 list, the Bell prize was awarded to applications with significant PDE content. More recently, the prize has been awarded for molecular dynamics based applications, since this is the realm of the IBM/ Blue Gene systems that have been leading the list in the past few years. This, however, is not an indication that the interest in fast PDE solvers has declined, and therefore we will report here on our results for a massively parallel finite element solver for elliptic PDEs.

The HLRB II system is an SGI-Altix that went into operation in September 2006 with 4096 processors and an aggregate main memory of 17.5 Terabytes ("phase 1"). In April 2007 this system was upgraded to 9728 cores and 39 Terabytes of main memory ("phase 2"). In particular in terms of available main memory, this is currently one of the largest computers in the world. Though the HLRB II is a general purpose supercomputer, it is especially well suited for finite element problems, since it has a large main memory and a high bandwidth. With this article we would like to demonstrate the extraordinary power of this system for solving finite element problems, but also which algorithmic choices and implementation techniques are necessary to exploit this architecture to its full potential.

The test problem reported in this article is a finite element discretization on tetrahedral 3D finite elements for a linear, scalar, elliptic PDE in 3D, as it could be used as a building block in numerous more advanced applications. We have selected this problem since it has a wide range of applications, and also, because it is an excellent test example for any high performance computer architecture.

Algorithms for very large scale systems

In this article we focus on multigrid algorithms (1,2), since these provide mathematically the most efficient solvers for systems originating from elliptic PDEs. Since multigrid algorithms rely on using a hierarchy of coarser grids, clever data structures must be used and the parallel implementation must be designed carefully so that the communication overhead remains minimal. This is not easy, but our results below will demonstrate excellent performance on solving linear systems with up to $3 \cdot 10^{11}$ unknowns and for up to almost 10,000 processors.

Before we turn to the techniques that make this possible, we would like to comment on why we do not use domain decomposition methods that might be suggested as an alternative to using multigrid. In particular, it may be argued that it is easier to implement parallel domain decomposition efficiently than parallel multigrid, since it avoids the need of a grid hierarchy. However, the price for the ease of implementation is a deterioration of the convergence rate that makes plain domain decomposition methods unsuitable on massively parallel systems. Of course, they can be improved by using an auxiliary coarse space within each iteration of the algorithm. In our view, however, with a coarse space, domain decomposition methods lose much of their advantage, since they will suffer from essentially the same bottleneck as multigrid.

We wish to point out that the need for global communication is a fundamental feature of elliptic PDEs and so there is no hope to get around it by any algorithm. Multigrid methods seem to use the minimal amount of computation and also the minimal amount of communication that is necessary to solve the problem. Using a hierarchical mesh structure is essential not only to keep the operation count optimal, but also to keep for the amount of communication minimal. The difficulty is to organize and implement this efficiently. Using a mesh hierarchy can only be avoided if one is willing to pay by using more iterations. In total this may therefore lead to even more communication. We believe that the computational results below demonstrate that multigrid is the method of choice for solving extremely large PDE problems in parallel.

Hierarchical Hybrid Grids

HHG ("Hierarchical Hybrid Grids") (1,2,3) is a framework for the multigrid solution for finite element (FE) problems. FE methods are often preferred for solving elliptic PDEs, since they permit flexible, unstructured meshes. Among the multigrid methods, *algebraic* multigrid also supports unstructured grids automatically. *Geometric* multigrid, in contrast, relies on a given hierarchy of nested grids. On the other hand, geometric multigrid achieves a significantly higher performance in terms of unknowns computed per second.

HHG is designed to close this gap between FE flexibility and geometric multigrid performance by using a compromise between structured and unstructured grids: a coarse input FE mesh is organized into the grid primitives: vertices, edges, faces, and volumes that are then refined in a structured way as indicated in Fig 1. This approach preserves the flexibility of unstructured meshes, while the regular internal structure allows for an efficient implementation on current computer architectures, especially on parallel computers.

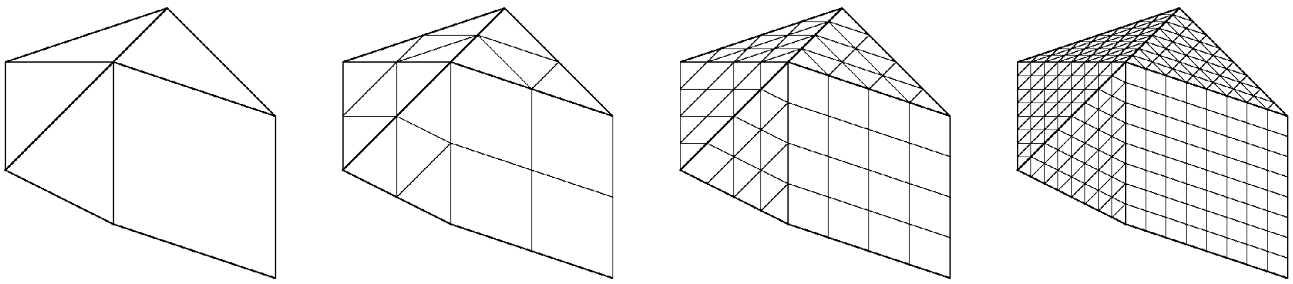

Figure 1: Regular refinement example for a two-dimensional input grid. Beginning with the input grid on the left, each successive level of refinement creates a new grid that has a larger number of interior points with structured couplings.

Parallelization

To exploit high-end computers, the programs must be parallelized using message passing. The HHG framework is an ideal starting point for this, since the mesh partitioning can be essentially accomplished on the level of the coarse input grid, that is, with a grid size that can still be handled efficiently by standard mesh partitioning software like Metis. Figure 2a shows a simple 2D example of such a grid distribution. Two triangular elements are assigned to the two processes P_0 and P_1. The unknowns on the edge between the elements are coupled to both elements and are thus needed by both processes. This introduces communication (Fig 2b) and is equivalent to using ghost nodes, as is typical in parallel mesh algorithms. The edge data structure itself can be assigned to any one of the two processors.

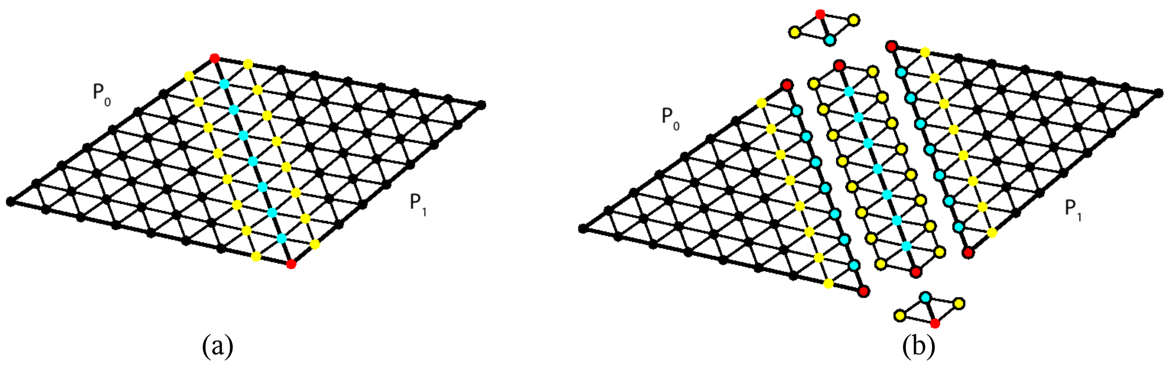

Figure 2: Grid distribution among processes. The encircled nodes are ghost values.

In order to avoid excessive latency, the algorithmic details and communication must be designed carefully. The multigrid solver uses a Gauß-Seidel smoother that traverses the grid points in the order of the primitives of the coarse input mesh: vertices - edges - faces - volumes. During the update of any such group, no parallel communication is necessary, because a vertex, for example, can only be connected to another vertex indirectly via an edge. This means that, rather than sending many small messages, each type of primitive can have its parallel dependencies updated as a single large message, which greatly reduces communication latency. However, a true Gauß-Seidel sweep traversing over the grid points still requires too many communication steps during each iteration, since neighboring grid points might belong to different processes. The current HHG implementation ignores a few such dependencies, thus giving the smoother the characteristics of a Jacobi iteration at the affected points. Numerically, this leads to a slight deterioration of the convergence rate, but the reduction in execution speed more than outweighs this effect.

World record in linear system solving

In our lagest computation to date, we have used 9170 cores of HLRB II and HHG to solve a finite element problem with 307 billion unknowns in 93 seconds run time. The problem itself is artificially designed by meshing a cube. This is necessary to ease the construction of problems with varying mesh size for our scalability study. However, the HHG data structures would allow for an arbitrary tetrahedral input mesh.

We believe that this is the largest finite element system that has been solved to date. Additionally, we point out that the absolute times to solution are still fast enough to leave room for using this solver as a building block in e.g. a time stepping scheme.

The results in Table 1 additionally show the results of a scaling experiment from 4 to 9170 compute cores. The amount of memory per core is kept constant and the problem size is chosen to fill as much of the available memory as possible. If the program were perfectly scalable, the average time per V-cycle would stay constant throughout the table, because the ratio of problem size (i.e. workload) versus number of cores (i.e. compute power) stays constant. Near perfect scaling is seen as measure of the quality of an algorithm and its implementation. For the HLRB II in installation phase 1 the computation time increases only by a factor of 2.2 when scaling from 4 to 3825 cores. This is still not perfect but in our view acceptable, especially when compared to other algorithms and especially in terms of the absolute compute time. Note that perfect scalability is the more difficult to achieve the faster a code is.

While every core of HLRB II phase 1 still had its own memory and network interface, the new dual-core configuration provides less bandwidth per core since two cores must share an interface. Additionally, a part of the installation is now configured as so-called "high density partitions" where two dual-core processors and thus four cores share one interface. Benchmark results including these high density partitions are marked with an asterisk in table 1. The timings for 64, 504 and 2040 cores show that the dual-core processors of phase 2 account for approximately 39% deterioration in runtime compared to phase 1. Scaling on the regular partitions shows a runtime increase from 4.93 s on 64 cores to 6.33 s on 6120 cores. On the high density partitions, the runtime deteriorates to 7.06 s on just 128 cores but then increases only slightly further to 7.75 sec for our largest runs.

Number of cores	Number of unknowns ($\cdot 10^6$)	Average time per V-cycle (sec)		Time to solution ($\|r\| < 10^{-6} \|r_0\|$)	
		Phase 1	Phase 2	Phase 1	Phase 2
4	134.2	3.16	6.38 *	37.9	76.6 *
8	268.4	3.27	6.67 *	39.3	80.0 *
16	536.9	3.35	6.75 *	40.3	81.0 *
32	1,073.7	3.38	6.80 *	40.6	81.6 *
64	2,147.5	3.53	4.93	42.3	59.2
128	4,295.0	3.60	7.06 *	43.2	84.7 *
252	8,455.7	3.87	7.39 *	46.4	88.7 *
504	16,911.4	3.96	5.44	47.6	65.3
2040	68,451.0	4.92	5.60	59.0	67.2
3825	128,345.7	6.90		82.8	
4080	136,902.1		5.68		68.2
6120	205,353.1		6.33		76.0
8152	273,535.7		7.43 *		89.2 *
9170	307,694.1		7.75 *		93.0 *

Table 1: Scaleup results for HHG. With a convergence rate of 0.3, 12 V-cycles are necessary to reduce the starting residual by a factor of 10^{-6}. The entries marked with * correspond to runs on (or including) high density partitions with reduced memory bandwidth per core.

Conclusions

The HHG framework and the HLRB II have been used to solve a finite element problem of world-record size, exceeding previous results by more than an order of magnitude, see (2,5). HHG draws its power from using a multigrid solver that is especially designed and carefully optimized for current, massively parallel high performance architectures. The SGI Altix-architecture is found to be well-suited for large scale iterative FE solvers.

References

(1) B. Bergen, T. Gradl, F. Hülsemann, U. Rüde. *A Massively Parallel Multigrid Method for Finite Elements,* Computing in Science and Engineering 8:56-62, 2006.

(2) B. Bergen, F. Hülsemann, U. Rüde: *Is* 1.7 x 10^{10} *Unknowns the Largest Finite Element System that Can Be Solved Today?* Proceedings of the ACM/IEEE 2005 Supercomputing Conference , Seattle, 12.-18. Nov. 2005, ISBN 1-59593-061-2

(3) G. Hager, B. Bergen, P. Lammers, G. Wellein. *Taming the Bandwidth Behemoth. First Experiences on a Large SGI Altix System.* In: inSiDE, 3(2):24, 2005.

(4) B. Bergen, G. Wellein, F. Hülsemann, U. Rüde: *Hierarchical Hybrid Grids — Achieving TERAFLOP Performance on Large Scale Finite Element Simulation*, International Journal of Parallel, Emergent and Distributed Systems, 22(4):311-329, 2007.

(5) M.F. Adams, H.H Bayraktar, T.M. Keaveny, P. Papadopoulos: *Ultrascalable implicit finite element analyses in solid mechanics with over a half a billion degrees of freedom*, ACM/IEEE Proceedings of SC2004: High Performance Networking and Computing, 2004.