UCHPC – UnConventional High Performance Computing for Finite Element Simulations

Stefan Turek, Dominik Göddeke, Christian Becker, Sven H.M. Buijssen, Hilmar Wobker
Applied Mathematics, Dortmund University of Technology, Germany
ture@featflow.de, dominik.goeddeke@math.tu-dortmund.de

Processor technology is still dramatically advancing and promises enormous improvements in processing data for the next decade. These improvements are driven by parallelisation and specialisation of resources, and 'unconventional hardware' like GPUs or the Cell processor can be seen as forerunners of this development. At the same time, much smaller advances are expected in moving data; this means that the efficiency of many simulation tools – particularly based on Finite Elements which often lead to huge, but very sparse linear systems – is restricted by the cost of memory access. We explain our approach to combine efficient data structures and multigrid solver concepts, and discuss the influence of processor technology on numerical and algorithmic developments. Concepts of 'hardware-oriented numerics' are described and their numerical and computational characteristics is examined based on implementations in FEAST, a high performance solver toolbox for Finite Elements which is able to exploit unconventional hardware components as 'FEM co-processors', on sequential as well as on massively parallel computers. Finally, we demonstrate prototypically how these algorithmic and computational concepts can be applied to solid mechanics problems, and we present simulations on heterogeneous parallel computers with more than one billion unknowns.

1 Introduction and Motivation

Software development for FEM problems has traditionally focused on improving the numerical methodology. Hardware aspects played only a minor role, since codes automatically ran faster with each new generation of processors. This trend has come to an end, as physical limitations (heat, leaking voltage, pin limits) have led to a paradigm change: Performance improvements are no longer driven by frequency scaling, but by parallelism and specialisation. Soon, CPUs will have tens of parallel cores, with hundreds to follow. Future massively parallel chip designs will likely be heterogeneous and contain general and specialised cores with non-uniform memory access (NUMA) to local storage/caches on the same chip. Both Intel and AMD have announced such chip projects, and current designs exhibit NUMA features already.

Specialised co-processors such as the accelerator boards offered by ClearSpeed, or Hypertransport couplings of CPUs with FPGAs (e. g. Cray's XD1 blades) are forerunners of this development and a good test environment for future systems. Similarly, multimedia processors such as the Cell BE or graphics processor units (GPUs) are of particular interest as they are optimised for high-bandwidth access to reasonably large local memories.

The parallelisation and specialisation of compute resources results in a major challenge for the programming model, in particular for clusters and supercomputers, where the coarse-grained parallelism (handled by message passing among distributed memories via MPI) must interact with the fine-grained on-chip parallelism and heterogeneity on the nodes. This task is particularly difficult for the sparse (linear) equation systems resulting from Finite Element discretisations of partial differential equations (PDEs). In contrast to the well-established BLAS and LAPACK libraries, different data formats and computation schemes with little standardisation co-exist, and performance is limited by the memory bandwidth and not by the raw compute power. This well-known 'memory wall problem' is further aggravated, as memory performance improves at a much slower pace than (peak) CPU performance.

Analogously to the coarse grained parallelism on the cluster level, compilers for standard languages are not expected to be able to locally parallelise existing data intensive codes efficiently in a fully automated way: This task requires explicit knowledge of the data flow and data reuse patterns and is particularly hard in the presence of heterogeneous memory hierarchies. New languages and frameworks with data and task parallel intrinsics (e. g. HPF, co-array Fortran, or, in particular for memory intensive tasks, Sequoia [9]) can perform better in this situation. New languages, however, imply the reimplementation of significant portions of an application, which is prohibitive for established and actively used codes.

Our hypothesis is that in the area of high performance Finite Element simulations, further performance improvement can only be achieved by 'hardware-oriented numerics'. This means that numerical and algorithmic foundation research must involve long-term technology trends. Data locality techniques with both spatial and temporal blocking are widely considered a very important design principle in this context. Similar in spirit is the work of Rüde et al. on patchwise multigrid smoothers and hierarchical hybrid grids [3]. Douglas and Thorne, and Douglas, Rüde et al. present cache-oriented multigrid solver components [6–8]. Butarri, Dongarra et al. discuss the impact of multicore architectures on mathematical software, in particular for (dense) BLAS and LAPACK-based applications [4]. Finally, Keyes and Colella et al. survey trends towards petascale computing for FEM [5, 13].

The above observations have led to the development of FEAST (Finite Element Analysis & Solution Tools), a software toolbox providing Finite Element discretisations and corresponding optimised parallel solvers for PDE problems [1, 16]. FEAST combines modern numerical techniques with a hardware-efficient implementation for a wide range of HPC architectures. In particular, FEAST contains mechanisms to include unconventional hardware – like GPUs and forthcomingly Cell BE processors – as 'FEM co-processors' in such a way that complex simulations can directly benefit from hardware acceleration without having to change application code.

2 FEAST - Finite Element Analysis & Solution Tools

The core features of FEAST in the context of this paper are as follows:

Separation of Structured and Unstructured Data FEAST covers the computational domain with a collection of quadrilateral subdomains. The subdomains form an unstructured coarse mesh (cf. figure 1), and each subdomain is independently refined in a generalised tensor product fashion, possibly combined with $r/h/rh$-adaptivity and anisotropic refinement. The resulting mesh is used to discretise the domain with Finite Elements. This approach caters to the contradictory needs of flexibility in the discretisation and efficient implementation: Instead of keeping all data in one general, homogeneous data structure, FEAST stores only local FE matrices and vectors (corresponding to subdomains) and thus maintains a clear separation of structured and unstructured parts of the domain. The nonzero pattern of the local matrices is known a priori, which is exploited to optimise data structures and linear algebra routines (*Sparse Banded BLAS*). In view of the memory wall problem, it is worth mentioning that optimised linear algebra operations acting locally on the subdomains can be implemented with direct block memory transfers and without any pointer chasing, thus maintaining both spatial and temporal locality. On the other hand, the flexibility of the unstructured coarse mesh allows to resolve even complex geometries.

Parallel Multigrid Solvers In the parallelisation of Finite Element codes, numerical robustness, numerical efficiency and (weak) scalability are often contradictory properties. For the problems we are concerned with in the (wider) context of this paper, multigrid methods are obligatory from a numerical point of view. In parallel, the strong recursive character of 'optimal' serial multigrid smoothers (e. g. ILU) is usually relaxed to a block-Jacobi approach to decouple the subdomains, as strong recursion between the subdomains leads to poor weak scalability due to high communication requirements.

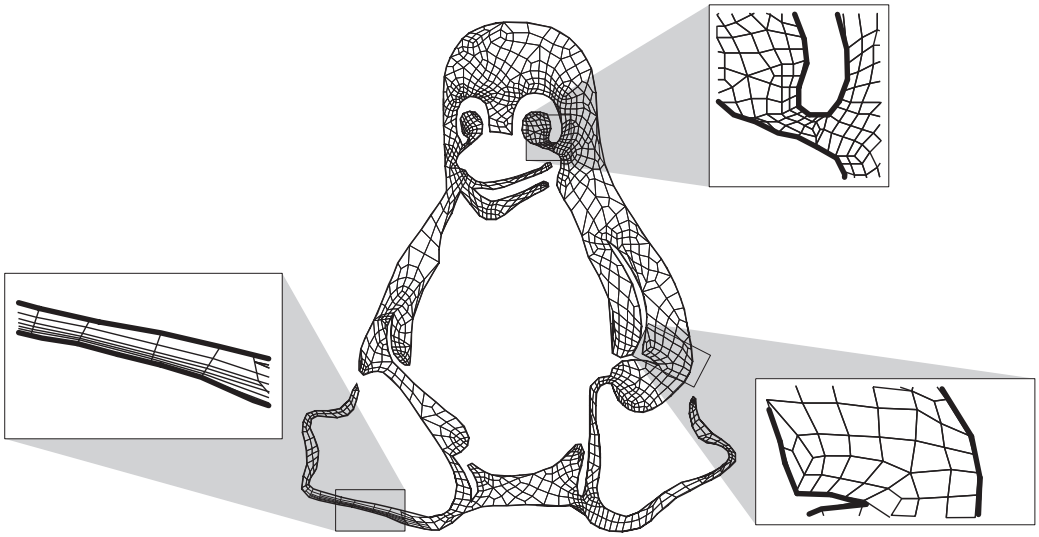

Figure 1: Exemplary Tux geometry. Important features of FEAST's discretisation approach are highlighted: Globally unstructured, locally structured grids; anisotropic refinement and regular refinement to generalised tensorproduct grids.

Block-Jacobi smoothers, on the other hand, lead to poor convergence in the presence of (even local) anisotropies, and hence dramatically reduce the numerical efficiency of the parallel solver [15, 16].

To address these contradictory needs, FEAST employs a generalised multigrid / domain decomposition concept called ScaRC [14]. The basic idea is to apply a *global* multigrid algorithm which is smoothed in an additive manner by *local* multigrid iterations acting on each subdomain independently. This cascaded multigrid scheme is very robust as local irregularities are 'hidden' from the outer solver. The global multigrid provides strong global coupling (as it acts on all levels of refinement), and it exhibits good parallel efficiency by design, both in the traditional and in the numerical sense.

From an implementational point of view, global matrix-vector operations are performed by a series of local operations on matrices representing the restriction of the 'virtual' global matrix on each subdomain. Local information is exchanged only over boundaries of neighbouring subdomains. There is only an implicit overlap, the domain decomposition is implemented via special boundary conditions in the local matrices. Several subdomains can be grouped together and treated within one MPI process.

Scalar and Vector-Valued Problems The guiding idea to treating vector-valued problems with FEAST is to rely on the modular, highly optimised and fully tested core routines for the scalar case in order to formulate robust schemes for a wide range of applications, rather than using the best suited numerical scheme for each application and go through the optimisation process over and over again. Vector-valued PDEs as they arise in Computational Solid Mechanics and Computational Fluid Dynamics can be rearranged and discretised in such a way that the resulting discrete equation systems consist of blocks that correspond to scalar subequations. Due to this special block-structure, all operations required to solve the systems can be implemented as a series of operations for scalar systems (e. g. matrix-vector operations, dot products and grid transfer operations in multigrid), taking advantage of the highly tuned linear algebra components in FEAST. Moreover, entire (scalar) subsystems can be treated with core FEAST scalar solvers.

Co-processor acceleration In FEAST, co-processors are not integrated on the kernel level, but on the level of local solvers for local subproblems in the global, parallel solver scheme: Instead of accelerating individual linear algebra operations, we accelerate entire multigrid solvers for local subproblems [11]. This concentrates sufficient fine-grained parallelism in a separate task and minimises the overhead of repeated co-processor configuration and data transfer in case of co-processors that are integrated in the system via a bandwidth bottleneck, for instance the PCIe bus for GPUs. The abstraction layer of the suggested 'minimally invasive' integration encapsulates heterogeneities of the system on the node level, so that MPI sees a globally homogeneous system, while the local solver components cleverly encapsulate the heterogeneity within the node. To benefit from co-processor acceleration, application code does not need to be changed at all. The reduced precision of several co-processor architectures is addressed by a mixed precision iterative refinement scheme. The current implementation supports GPUs [11], and a Cell backend is being developed.

Applications Two important classes of applications have recently been built on top of FEAST: The fluid dynamics code FEASTFLOW solves the transient incompressible Stokes and Navier-Stokes equations. It computes velocity fields and pressure distributions for Newtonian fluids like gases, water and many other liquids in a fully stabilised, implicit way. The solid mechanics code FEASTSOLID solves static and transient elasticity problems for small and finite deformations. Beside linear constitutive relations (Hooke, St. Venant-Kirchhoff), the nonlinear Neo-Hooke law is supported. Incompressible and nearly incompressible materials are handled by a mixed displacement/pressure formulation. Other applications, e. g. Fluid-Solid-Interaction (FSI) and different methods, e. g. Lattice Boltzmann methods (LBM), are actively being developed.

3 Results

In this section, we present some (prototypical) FEAST results which demonstrate the promising potential of our approach for future challenging problems.

Table 1 presents some performance data for solving the Poisson problem on the Tux geometry (see figure 1). The coarse grid is refined 8, 9 and 10 times yielding a maximum problem size of 1.4 billion unknowns, and either 63 or 127 compute processes are used (we do not count the master process which synchronises the parallel computations). The data illustrates several important observations: FEAST runs on three important classes of HPC architectures [2]: Supercomputers (IBM p690, JUMP, Jülich, Germany), vector machines (NEC SX8, HLRS, Stuttgart, Germany) and commodity based clusters (Opteron-based, Infiniband interconnects, LiDO, ITMC TU Dortmund, Germany). On the vector and

#DOF	Arch.	63p		127p	
		sec	MFlop/s	sec	MFlop/s
90,317,056	Power4	170.4	5,252	116.4	7,698
	NEC	156.7	5,376	91.9	9,167
	Opteron	65.2	13,412	40.5	21,586
361,120,256	Power4	-	-	269.6	11,912
	NEC	222.8	13,511	128.0	23,519
	Opteron	192.6	16,299	104.7	29,974
1,444,185,088	Power4	-	-	-	-
	NEC	595.0	22,328	364.9	36,415
	Opteron	-	-	615.6	25,637

Table 1: Performance results for solving the Poisson problem on the Tux geometry, due to insufficient local node memory some configurations could not be computed on all systems.

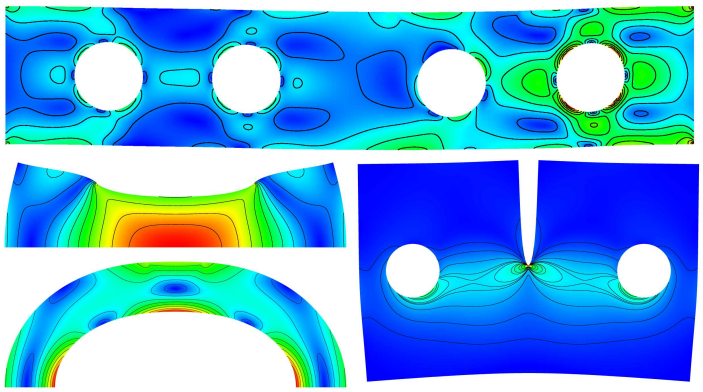

Figure 2: Displacements and von Mises stresses for four prototype configurations (STEELFRAME, BLOCK, PIPE and CRACK), computed with the GPU-accelerated solid mechanics application FEASTSOLID.

commodity architectures, FEAST exhibits good strong scalability and the local CPU efficiency for the commodity based architecture is close to the peak memory bandwidth. Performance on the JUMP system is disappointing due to the lack of time to profile the code for bottlenecks. The vector machine achieves the best local performance, even better results can be expected with the pursued further adaption to this special architecture.

Figure 2 shows examples of linearised elasticity computations, obtained by the application code FEASTSOLID. The four bodies consist of compressible material and are exposed to static external loads. The figure depicts the resulting displacements and the von Mises stresses. All four configurations were computed on a 16 node Opteron cluster, with a dualcore Santa Rosa CPU and a NVIDIA Quadro 5600 GPU per node. Speed-up is illustrated in the left plot in figure 3. Overall, we achieve a speed-up of 2.6 comparing the GPU-accelerated solver with a single core, and 1.6 against two cores. Detailed analysis [12] reveals that the two cores are much faster than expected based on pure bandwidth considerations, because the concurrency in the node helps overlapping data transfer and computation. We are currently investigating hybrid communication models for this three-way parallelism; i. e. many nodes via MPI, several heterogeneous resources within the node, and many multiprocessors in the GPU. The second important observation is that the achieved speed-up by a factor of 2.6 is driven by a local speed-up of 9 for the parts of the application (the local solvers per subdomain) that can benefit from hardware acceleration. In case of the linearised elasticity solver, 2/3 of all operations can be accelerated, so this factor limits the achievable performance gain of the entire solver to a factor of 3 (Amdahl's Law). The factor of 2.6 is thus already a very good number, especially in our 'minimally invasive' approach for which no application code needs to be changed. Consequently, 'hardware-oriented numerics' means that improved numerical techniques that increase the accelerable portion of the solution process need to be investigated.

Figure 3 (right) demonstrates good weak scalability for the linearised elasticity solver on up to 64 GPU-accelerated nodes. We execute the solver either on two CPUs per node, or on one GPU per node [12].[1] Previous work on other clusters showed that FEAST scales very well in the weak sense on up to 256 nodes [1] (see also section 2), both without and with GPU acceleration [10].

[1] As we do not have access to enough nodes with up-to-date hardware, the experiments were performed on NVIDIA Quadro 1400 GPUs, which are four hardware generations old. The CPUs in the cluster however are only one generation behind, which explains the comparatively weak speed-up.

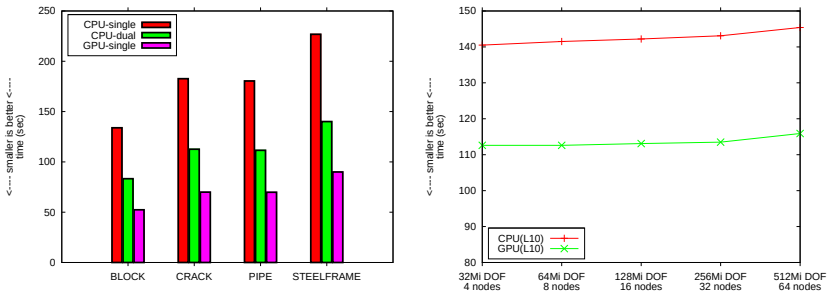

Figure 3: Timings for a singlecore, a dualcore and a GPU solver (left), weak scalability results (right). All computations were performed with the GPU-accelerated solid mechanics application FEASTSOLID.

4 Conclusions

Looking at current trends in numerical simulation methods and high performance computing techniques, dedicated hardware-oriented numerics seems to be a key tool to significantly improve FEM software packages for highly challenging problems from engineering disciplines and life sciences. Otherwise, the general expectation that with better hardware the simulation of more complex problems comes automatically into reach cannot be met any more due to the increasing gap between sustained and peak performance. In our contribution, we have outlined the usage of GPUs as representative 'FEM coprocessors': We presented prototypical simulations of elliptic model problems with more than one billion unknowns and examples from solid mechanics. These results demonstrate how highly sophisticated hardware-oriented numerics can be realised on heterogeneous parallel systems.

Acknowledgements

Parts of this work are based on a joint collaboration with Robert Strzodka (Max Planck Center, Max Planck Institut Informatik) and Patrick McCormick and Jamaludin Mohd-Yusof (Los Alamos National Laboratory).

We would like to thank NVIDIA for donating development hardware.

References

[1] C. Becker. *Strategien und Methoden zur Ausnutzung der High-Performance-Computing-Ressourcen moderner Rechnerarchitekturen für Finite Element Simulationen und ihre Realisierung in FEAST (Finite Element Analysis & Solution Tools)*. PhD thesis, Universität Dortmund, Fachbereich Mathematik, Logos Verlag, Berlin, 2007.

[2] C. Becker, S. H. Buijssen, and S. Turek. FEAST: development of HPC technologies for FEM applications. In W. Nagel, D. Kröner, and M. Resch, editors, *High Performance Computing in Science and Engineering '07*, pages 503–516. Springer, Berlin, 2007.

[3] B. Bergen, F. Hülsemann, and U. Ulrich Rüde. Is 1.7×10^{10} unknowns the largest finite element system that can be solved today? In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, 2005.

[4] A. Buttari, J. Dongarra, J. Kurzak, J. Langou, P. Luszczek, and S. Tomov. The impact of multicore on math software. In *Proceedings of PARA 2006, Umea Sweden*, 2006.

[5] P. Colella, T. H. Dunning, W. D. Gropp, and D. E. Keyes. A science–based case for large–scale simulation. Technical report, DOE Office of Science, 2003. `http://www.pnl.gov/scales`.

[6] C. C. Douglas, J. Hu, W. Karl, M. Kowarschik, U. Rüde, and C. Weiß. Fixed and adaptive cache aware algorithms for multigrid methods. In E. Dick, K. Riemslagh, and J. Vierendeels, editors, *Multigrid Methods VI*, volume 14, pages 87–93. Springer, 2000.

[7] C. C. Douglas, J. Hu, M. Kowarschik, U. Rüde, and C. Weiß. Cache optimization for structured and unstructured grid multigrid. *Electronic Transactions on Numerical Analysis*, 10:21–40, 2000.

[8] C. C. Douglas and D. T. Thorne. A note on cache memory methods for multigrid in three dimensions. *Contemporary Mathematics*, 306:167–177, 2002.

[9] K. Fatahalian, T. Knight, M. Houston, M. Erez, D. R. Horn, L. Leem, J. Y. Park, M. Ren, A. Aiken, W. J. Dally, and P. Hanrahan. Sequoia: Programming the memory hierarchy. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, 2006.

[10] D. Göddeke, R. Strzodka, J. Mohd-Yusof, P. McCormick, S. H. Buijssen, M. Grajewski, and S. Turek. Exploring weak scalability for FEM calculations on a GPU-enhanced cluster. *Parallel Computing*, 33(10–11):685–699, 2007.

[11] D. Göddeke, R. Strzodka, J. Mohd-Yusof, P. McCormick, H. Wobker, C. Becker, and S. Turek. Using GPUs to improve multigrid solver performance on a cluster. *accepted for publication in the International Journal of Computational Science and Engineering*, Special Issue on Implementational Aspects in Scientific Computing, 2008.

[12] D. Göddeke, H. Wobker, R. Strzodka, J. Mohd-Yusof, P. McCormick, and S. Turek. Co-processor acceleration of an unmodified parallel solid mechanics code with FEASTGPU. *accepted for publication in the International Journal of Computational Science and Engineering*, 2008.

[13] D. E. Keyes. Terascale implicit methods for partial differential equations. *Contemporary Mathematics*, 306:29–84, 2002.

[14] S. Kilian. *ScaRC: Ein verallgemeinertes Gebietszerlegungs-/Mehrgitterkonzept auf Parallelrechnern*. PhD thesis, Universität Dortmund, Fachbereich Mathematik, 2001.

[15] B. F. Smith, P. E. Bjørstad, and W. D. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.

[16] S. Turek, C. Becker, and S. Kilian. Hardware–oriented numerics and concepts for PDE software. *Future Generation Computer Systems*, 22(1-2):217–238, 2003.