Computers & Fluids 80 (2013) 327–332

Contents lists available at SciVerse ScienceDirect

Computers & Fluids

journal homepage: www.elsevier.com/locate/compfluid

Towards a complete FEM-based simulation toolkit on GPUs: Unstructured grid finite element geometric multigrid solvers with strong smoothers based on sparse approximate inverses

M. Geveler *, D. Ribbrock, D. Göddeke, P. Zajac, S. Turek

Institute of Applied Mathematics, TU Dortmund University of Technology, Germany

ARTICLE INFO

Article history:
Received 15 September 2011
Received in revised form 28 January 2012
Accepted 30 January 2012
Available online 9 February 2012

Keywords:
Unstructured grids
Multigrid solvers
Sparse matrices
Finite elements
Strong smoothers
GPU computing
SPAI
SAINV

ABSTRACT

We describe our FE-gMG solver, a finite element geometric multigrid approach for problems relying on unstructured grids. We augment our GPU- and multicore-oriented implementation technique based on cascades of sparse matrix–vector multiplication by applying strong smoothers. In particular, we employ Sparse Approximate Inverse (SPAI) and Stabilised Approximate Inverse (SAINV) techniques. We focus on presenting the numerical efficiency of our smoothers in combination with low- and high-order finite element spaces as well as the hardware efficiency of the FE-gMG. For a representative problem and computational grids in 2D and 3D, we achieve a speedup of an average of 5 on a single GPU over a multithreaded CPU code in our benchmarks. In addition, our strong smoothers can deliver a speedup of 3.5 depending on the element space, compared to simple Jacobi smoothing. This can even be enhanced to a factor of 7 when combining the usage of approximate inverse-based smoothers with clever sorting of the degrees of freedom. In total the FE-gMG solver can outperform a simple (multicore-) CPU-based multigrid by a total factor of over 40.

1. Introduction

Finite element methods (FEM) are a highly accurate flexible and theoretically rigorous instrument for solving partial differential equations (PDEs) that arise in many fields. Examples include high-order and non-conforming elements over arbitrarily unstructured geometries, adaptivity, a priori/a posteriori error estimation and special pressure-Schur-complement preconditioning in the solution of the Navier–Stokes equations [1].

A time consuming step within the solution pipeline of FEM is the linear system solver. Both numerical efficiency and hardware efficiency have to be addressed simultaneously to achieve a good total efficiency: Geometric Multigrid (gMG) solvers can treat the arising sparse linear systems in a number of iterations that is independent of the grid width. In combination with high-order finite elements, even superlinear convergence effects can be obtained [2]. However, the numerical efficiency and robustness of multigrid methods strongly depends on the smoothing operator, see Section 3.2.

Over the past several years, graphics processors (GPUs) have made the transition to a valuable and increasingly accepted general purpose computing resource, both on standalone workstations and in large-scale HPC installations. The main reason why GPUs excel at many HPC workloads that provide ample parallelism is that their design is fundamentally different from commodity CPU architectures: Instead of minimising the latency of a single task, they maximise the overall throughput of a large set of identical tasks, and the chips' ratio of functional units to control logic is much more favourable. For memory-bound problems, the GPU boards' more hard-wired memory lanes allow for a higher signal quality, and thus more aggregated memory bandwidth. We refer to a recent article by Garland and Kirk [3] for technical details and a concise description of the hardware-software model of *throughput-oriented computing*.

2. Solution approach

We present and evaluate an augmented version of a previously proposed implementation technique for FE-gMG (Finite Element Geometric Multigrid) solvers for PDE problems discretised on *unstructured* grids. Our target architectures are fine-grained (manycore) GPUs – at this point, we focus entirely on the solver performance, evaluating it for different finite element spaces. Since the smoother is the most critical part concerning performance and robustness of the gMG, we evaluate numerically efficient

* Corresponding author.
E-mail address: markus.geveler@math.tu-dortmund.de (M. Geveler).

0045-7930/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compfluid.2012.01.025

smoothers based on approximate inverses (SPAI and SAINV), see Section 3.2.

In our approach, performance-critical components of the solver pipeline, the smoother, grid transfer, coarse-grid solver and defect operators, are entirely based on cascades of sparse matrix–vector multiplications (SpMV). This consequent reduction to one performance-critical kernel within the multigrid solver has surprisingly many beneficial properties: The multigrid solver needs to be implemented only once, and is completely oblivious of the underlying finite element space, and even oblivious of the dimension of the computational domain (2D, 3D). Furthermore, our implementation replaces many specialised kernels with one central, well-understood and well-optimised parallel kernel (see Section 3.1), which is favourable in terms of software maintainability, performance-tuning and the adoption of GPUs in multigrid and finite element codes.

2.1. Related work

On GPUs, multigrid methods have received only moderate attention during the past several years, at least compared to the total number of papers concerned with sparse linear system solvers or finite element/volume/difference discretisations. The first to implement multigrid solvers for flow simulation in computer graphics entirely on GPUs were Bolz et al. [4] and Goodnight et al. [5]. They used geometric and algebraic multigrid (aMG) for finite-difference type discretisations. More recent publications presenting applications that require multigrid solvers are supersonic flows (aMG, unstructured grids [6]), (interactive) flow simulations for feature film (aMG/gMG, structured [7,8]), out-of core multigrid for gigapixel image stitching (gMG/aMG, structured [9]), image denoising and optical flow (gMG/aMG, structured [10]), power grid analysis (aMG, structured/unstructured [11]) and electric potential in the human heart (aMG, unstructured [12]). This last paper is similar in spirit to our work, since the authors also reduce (almost) the entire multigrid algorithm to sequences of sparse matrix–vector multiplications. The important difference (besides aMG vs. gMG) is that they use a specifically designed, problem-specific data layout in their SpMV implementation whereas we go further and use a layout that has been shown to deliver superior performance for a wide range of non-zero patterns. Also very closely related is the work by Heuveline et al., who pursue many different fine-grained parallel preconditioning techniques based on multicolouring: Simultaneously to our work, they have started to evaluate their preconditioners as smoothers in the multigrid context [13]. In summary, we can say that previous publications describing multigrid on GPUs either target algebraic multigrid, or are limited to structured grid geometric multigrid and low-order discretisations. To the best of our knowledge, together with Heuveline and his co-workers we are the first to present *geometric MG with strong smoothers for high-order unstructured grid FEM on GPUs.*

2.2. Contribution and paper outline

The paper at hand is the second in a series concerning FE-gMG on GPUs, significantly expanding a previous conference proceedings [14] and the initial publication on the topic which did not address strong smoothers [15]. The FE-gMG in this paper is based on a better matrix storage format (ELLPACK-T, see Section 3.1), and we focus on the evaluation of different preconditioners (Jacobi, SPAI and SAINV) in combination with different conforming finite element spaces (Q_1 and Q_2). Finally, all results are calculated on a newer generation of hardware (especially Fermi-type GPUs), see Section 4.

Throughout our computations, we solely concentrate on the solution of the linear system, which means, that all needed matrices (e.g. stiffness- and transfer-matrices as well as the preconditioners given by sparse approximate inverses within the smoother) are preassembled; this topic is subject to another publication under preparation. This is justified since in many practical scenarios, the linear solver dominates the total execution time and the combined effects of advanced smoothers, suitable numbering of the degrees of freedom and feasible utilisation of hardware acceleration have to be analysed. Here, the former increases numerical efficiency while simultaneously increasing the complexity of the SpMV operation by increasing the number of non-zeros and altering the sparsity pattern; the latter two are strongly related to each other since the sorting strategy directly influences matrix-bandwidth which is a major criterion for the performance of the GPU SpMV. In our publications mentioned above, we have already analysed the impact of the numbering of the degrees of freedom especially when using the GPU. Hence, in this paper, we concentrate on the other aspects mentioned above. We continue by describing the components of the FE-gMG solver in detail in Section 3, where we focus on the smoother and grid transfer operators. Especially, Section 3.1 is dedicated to the SpMV kernel and the implementational aspects of FE-gMG. In Section 4 we present results for our approach applied to a common model problem. Finally, we give a concise conclusion in Section 5.

3. FE-gMG – Finite Element Geometric Multigrid

3.1. SpMV kernel

We do not utilise the 'standard' CSR format, but rather the ELLPACK-T format proposed and exemplarily implemented by Vazques et al. [16]. In our experience, ELLPACK(-T) leads to significantly higher computational throughput, independent of the architecture and even for sequential code.

With the ELLPACK-T format, the sparse matrix–vector multiplication $\mathbf{y} = A\mathbf{x}$ is performed by computing each entry y_i of the result vector $\mathbf{y}$ independently. In general, this results in a regular access pattern on the data of $\mathbf{y}$ and A. In contrast, the access pattern on $\mathbf{x}$ depends highly on the non-zero structure of A, and due to the indirect addressing, memory access can be arbitrarily scattered.

The ELLPACK-T based SpMV kernel is mapped to the GPU architecture by launching one or more device threads for the calculation of an entry y_i, resulting in fully coalesced memory access to the matrix and the vector $\mathbf{y}$ due to the column-major ordering used. The access to the array $\mathbf{x}$ can be cached via the texture cache on the GPU to improve efficiency. On the FERMI generation of GPUs, the device-wide L2-cache is well utilised. No synchronisation between threads is necessary. The threads in one CUDA warp do not diverge because flow instructions are not necessary which would cause serialisation. Every warp finishes execution directly when all non-zero entries in the rows of its threads are completely processed. Because of this, only warps with a high relative non-zero count within their rows execute longer compared to average warps.

Finally, the ELLPACK-T format augments the former ELLPACK-R by storing t elements of one row contiguously in memory and thus allows for multiple threads in one warp to process one row.

3.2. Smoothing operator

As a key component for multigrid, our *smoother* is realised as a damped preconditioned Richardson iteration, using SpMV to plug in the preconditioner and for defect calculation:

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \omega M(\mathbf{b} - A\mathbf{x}^k)$$

Here, $A\mathbf{x} = \mathbf{b}$ is the linear system to solve and $M \approx A^{-1}$ a preassembled (sparse) preconditioner that approximates the inverse of the