Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.2015.w10si

Vol. **8**, No. 1, pp. 22-46
February 2015

Towards Textbook Efficiency for Parallel Multigrid

Björn Gmeiner[1,*], Ulrich Rüde[2], Holger Stengel[3], Christian Waluga[1] and Barbara Wohlmuth[1]

[1] *Institute for Numerical Mathematics (M2), Technische Universität München, Boltzmannstrasse 3, D-85748 Garching b. München, Germany.*
[2] *Department of Computer Science 10, FAU Erlangen-Nürnberg, Cauerstraße 6, D–91058 Erlangen, Germany.*
[3] *Erlangen Regional Computing Center (RRZE), FAU Erlangen-Nürnberg, Martensstraße 1, D–91058 Erlangen, Germany.*

Received 5 December 2013; Accepted 29 July 2014

Abstract. In this work, we extend Achi Brandt's notion of textbook multigrid efficiency (TME) to massively parallel algorithms. Using a finite element based geometric multigrid implementation, we recall the classical view on TME with experiments for scalar linear equations with constant and varying coefficients as well as linear systems with saddle-point structure. To extend the idea of TME to the parallel setting, we give a new characterization of a work unit (WU) in an architecture-aware fashion by taking into account performance modeling techniques. We illustrate our newly introduced parallel TME measure by large-scale computations, solving problems with up to 200 billion unknowns on a TOP-10 supercomputer.

AMS subject classifications: 65N55, 68W10

Key words: Multigrid, parallel computing, textbook efficiency, finite element method.

1. Introduction

Asymptotic complexity bounds for iterative solvers that contain unspecified constants provide only heuristic criteria to design efficient solvers [10, Chapter 14]. This simple observation motivated Brandt to coin the term *textbook multigrid efficiency* (TME) to characterize truly fast multigrid algorithms. In [9] he writes

> *Textbook multigrid efficiency means solving a discrete PDE problem with a computational effort that is only a small (less than 10) multiple of the operation count associated with the discretized equations itself.*

*Corresponding author. *Email addresses:* gmeiner@ma.tum.de (B. Gmeiner), ulrich.ruede@fau.de (U. Rüde), waluga@ma.tum.de (C. Waluga), wohlmuth@ma.tum.de (B. Wohlmuth)

In the context of Brandt's TME, computational *efficiency* is evaluated in terms of the number of floating point operations (FLOPS) that are needed to obtain a solution with a certain accuracy. To achieve TME, the number of FLOPS to solve the discretized system must be a small multiple of the FLOPS needed for an application of the discretized operator. Consequently, the elementary work unit (WU) is defined as the cost of one such operator application.

Unfortunately on modern architectures, the TME notion falls short in predicting the performance of an iterative solver since it does not take into account the parallel efficiency and an architecture-aware algorithm design. Nowadays even multigrid algorithms that are TME efficient are not necessarily fast. Thus there is a need to extend the TME metric so that it becomes possible to predict the time to solution also on parallel computer architectures.

The main contribution of our work lies in adapting the TME paradigm to the issues originating from the ongoing transition to massively parallel multicore systems with complex communication networks and with a hierarchical memory architecture. On such modern machines, the simple count of floating point operations of an algorithm correlates poorly with actual computational cost. In particular, energy consumption emerges increasingly as the fundamental bottleneck and as the limiting resource of large scale computing. Since the use of energy is predominantly caused by data movement, the computational cost is strongly affected by moving data. Here data movement includes parallel communication in a large supercomputer cluster, the communication between chips and memory modules in a computer node, and also the access of each processing core to the register banks or cache memory within the chip itself.

In this paper, we focus on linear finite elements (FE) on semi-structured hierarchical simplicial meshes. These restrictions yield the advantage that the complex and computationally expensive logistics of manipulating unstructured grids can be avoided on the finer levels of the multigrid hierarchy. Our implementation is based on the Hierarchical Hybrid Grid (HHG) framework [3,6], a geometric multigrid library implemented in C++ and using the MPI message passing system and hybrid programming. Using HHG, we recently demonstrated that it is possible to solve linear elliptic problems with more than 10^{12} unknowns on current TOP 10 supercomputers, using several 100 000 processor cores [18,19]. The efficiency of the framework relies on the systematic performance- and architecture- aware co-design of the multigrid algorithm and its implementation. In particular, the matrix-free design avoids the explicit storage of the stiffness matrix and thus circumvents the severe efficiency penalty in terms of data storage and redundant data transport. In the course of our work, we also demonstrate an enhanced method to deal with variable coefficients in our stencil-based framework. For illustration, we complement our considerations with large-scale computations.

Before we proceed, let us discuss some related work. Due to the extensive amount of literature dealing with multigrid methods, we must restrict ourselves to recent contributions on parallel realizations for high-end distributed memory architectures. For a further discussion of parallelization techniques for multigrid methods, we refer to [13,24,35]. To the authors' best knowledge, the first parallel multigrid solvers capa-