Parallel Computing 50 (2015) 1–24

Contents lists available at ScienceDirect

Parallel Computing

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

On the scalability of inexact balancing domain decomposition by constraints with overlapped coarse/fine corrections

Santiago Badia [a,b,*], Alberto F. Martín [a,b], Javier Principe [a,b]

[a] Centre Internacional de Mètodes Numèrics en Enginyeria (CIMNE), Parc Mediterrani de la Tecnologia, UPC, Esteve Terradas 5, 08860 Castelldefels, Spain
[b] Universitat Politècnica de Catalunya, Jordi Girona 1-3, Edifici C1, Barcelona, 08034 Spain

ARTICLE INFO

Article history:
Received 24 October 2014
Revised 24 September 2015
Accepted 29 September 2015
Available online 22 October 2015

Keywords:
Domain decomposition
Inexact solvers
BDDC
Parallelization
Overlapping
Scalability

ABSTRACT

In this work, we analyze the scalability of inexact two-level balancing domain decomposition by constraints (BDDC) preconditioners for Krylov subspace iterative solvers, when using a highly scalable asynchronous parallel implementation where fine and coarse correction computations are overlapped in time. This way, the coarse-grid problem can be fully overlapped by fine-grid computations (which are embarrassingly parallel) in a wide range of cases. Further, we consider inexact solvers to reduce the computational cost/complexity and memory consumption of coarse and local problems and boost the scalability of the solver. Out of our numerical experimentation, we conclude that the BDDC preconditioner is quite insensitive to inexact solvers. In particular, one cycle of algebraic multigrid (AMG) is enough to attain algorithmic scalability. Further, the clear reduction of computing time and memory requirements of inexact solvers compared to sparse direct ones makes possible to scale far beyond state-of-the-art BDDC implementations. Excellent weak scalability results have been obtained with the proposed inexact/overlapped implementation of the two-level BDDC preconditioner, up to 93,312 cores and 20 billion unknowns on JUQUEEN. Further, we have also applied the proposed setting to unstructured meshes and partitions for the pressure Poisson solver in the backward-facing step benchmark domain.

1. Introduction

In order to deal with increasing levels of complexity in the simulation of phenomena governed by partial differential equations (PDEs), computational engineering and science must advance in the development of numerical algorithms and implementations that will efficiently exploit the ever-increasing amount of computational resources. The growth in computational power that resulted from Moore's law passes now through increasing the number of cores in a chip, instead of making cores faster. As a result, the next generation of supercomputers, able to reach 1 exaflop/s, is expected to reach billions of cores. The efficient exploitation of billion-fold levels of concurrency is a big challenge. The advance of large scale scientific computing will be strongly related to the ability to efficiently exploit these extreme core counts [1].

The time spent in an implicit simulation at the linear solver relative to the overall execution time grows with the size of the problem and the number of cores [2]. For extreme scale implicit simulations, a massively parallel linear solver is a key component.

* Corresponding author at: Centre Internacional de Mètodes Numèrics en Enginyeria (CIMNE), Parc Mediterrani de la Tecnologia, UPC, Esteve Terradas 5, Castelldefels, 08860 Spain. Tel.: +34 934134108.
E-mail addresses: sbadia@cimne.upc.edu (S. Badia), amartin@cimne.upc.edu (A.F. Martín), principe@cimne.upc.edu (J. Principe).

http://dx.doi.org/10.1016/j.parco.2015.09.004
0167-8191/

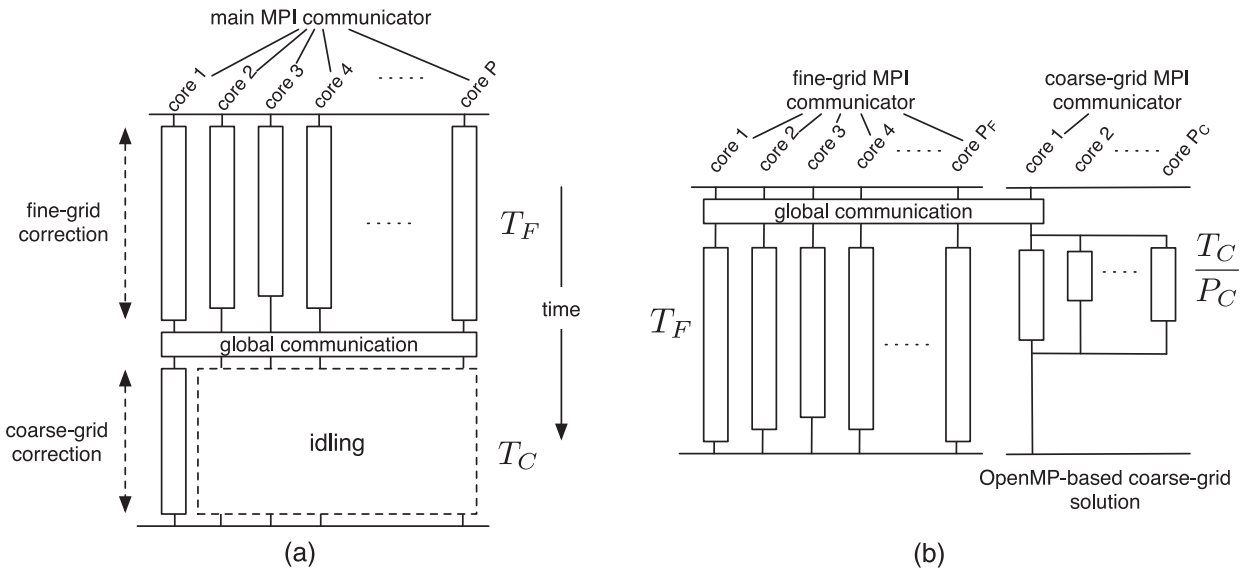

Fig. 1. Comparison of (a) the typical parallel distributed-memory implementation of Algorithms 1–3 and (b) the highly scalable one proposed in [12] implemented with multi-threading.

This scenario exacerbates the need of highly scalable algorithms and implementations. Only numerical algorithms with all their components scalable will efficiently run on extreme scale supercomputers. Extreme scale solvers should be developed under the assumption that local flops are cheap and communications expensive. On extreme core counts, it will be a must to reduce communication and synchronization among processors, and overlap communication with computation. At the largest scales, linear solvers are based on preconditioned Krylov subspace methods. Algorithmically scalable preconditioners include (algebraic) multigrid (MG) [3] and some domain decomposition (DD) algorithms [4]. However, this theoretical property is not enough for practical weak scalability, since the preconditioner itself must allow for a massively scalable implementation. Today's most scalable algorithms/implementations present practical limits of parallelism, e.g., due to the small, coarse problems to be solved in the hierarchical process for DD/AMG, and the loss of sparsity and denser communication patterns at coarser levels of AMG.

DD preconditioners make explicit use of the partition of the global mesh, e.g., for the finite element (FE) integration, into sub-meshes (subdomains) and provide a natural framework for the development of fast parallel solvers tailored for distributed-memory machines. One-level DD algorithms involve the solution of local problems and nearest-neighbors communications. A (second level) coarse correction (coupling all subdomains) is required to have algorithmic scalability, but it also harms the practical (CPU time) weak scalability. Two-level DD algorithms include the balancing Neumann–Neumann (BNN) preconditioner [5], the balancing DD by constraints preconditioner (BDDC) [6], and FETI-DP preconditioners [7]. In all these cases, for positive-definite matrices, a poly-logarithmic expression of the condition number of the preconditioned system $\kappa = 1 + \log^2(\frac{H}{h})$ can be proved, where h and H are the mesh and subdomain characteristic sizes, respectively, and d is the space dimension; $(\frac{H}{h})^d$ is the local problem size. Consequently, in weak scaling scenarios, i.e., increasing the linear system size and number of processors keeping $\frac{H}{h}$ fixed, the number of iterations of the preconditioned conjugate gradient (PCG) solver is (asymptotically) independent of the number of processors.

The practical scalability limits of a two-level DD implementation is determined by the coarse solver computation, whose size increases (at best) linearly with respect to the number of subdomains. The coarse problem rapidly becomes the bottleneck of the algorithm as we increase the number of processors, reducing weak scalability [8]. The coarse problem is several orders of magnitude smaller than the original global system, and only a very small portion of the computing cores can efficiently be exploited (assuming a parallel coarse solver). In typical DD implementations, this produces an unacceptable parallel efficiency loss, since all the cores not involved in the coarse solver computation are idling (see Fig. 1). One obvious strategy to improve scalability is to reduce the wall-clock time spent at the coarse solver by using, e.g., a MPI-distributed sparse direct solver like MUMPS [9] (see [10] for BDDC and [11] for FETI-DP). However, this approach only mitigates the problem.

2. Motivation

The BDDC preconditioner has some salient properties that permit to overcome this parallel overhead, making it an excellent candidate for extreme scale solver design:

(P1) It allows for a mathematically supported extremely aggressive coarsening. The ratio between the size of the global and coarse problem is of the order of the local problem size, i.e., $(\frac{H}{h})^d$. On memory-constrained supercomputers, it is in the order of 10^5 for sparse direct methods [12] and 10^6 for inexact solvers (see Section 6).