ARTICLE IN PRESS

Parallel Computing xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Parallel Computing

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

An efficient distributed randomized algorithm for solving large dense symmetric indefinite linear systems

Marc Baboulin [a,*], Dulceneia Becker [b], George Bosilca [b], Anthony Danalis [b], Jack Dongarra [b]

[a] Laboratoire de Recherche en Informatique, Inria/University Paris-Sud, Orsay, France
[b] Innovative Computing Laboratory, University of Tennessee, Knoxville, USA

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Randomized algorithms
Distributed linear algebra solvers
Symmetric indefinite systems
LDL^T factorization
PaRSEC runtime

ABSTRACT

Randomized algorithms are gaining ground in high-performance computing applications as they have the potential to outperform deterministic methods, while still providing accurate results. We propose a randomized solver for distributed multicore architectures to efficiently solve large dense symmetric indefinite linear systems that are encountered, for instance, in parameter estimation problems or electromagnetism simulations. The contribution of this paper is to propose efficient kernels for applying random butterfly transformations and a new distributed implementation combined with a runtime (*PaRSEC*) that automatically adjusts data structures, data mappings, and the scheduling as systems scale up. Both the parallel distributed solver and the supporting runtime environment are innovative. To our knowledge, the randomization approach associated with this solver has never been used in public domain software for symmetric indefinite systems. The underlying runtime framework allows seamless data mapping and task scheduling, mapping its capabilities to the underlying hardware features of heterogeneous distributed architectures. The performance of our software is similar to that obtained for symmetric positive definite systems, but requires only half the execution time and half the amount of data storage of a general dense solver.

1. Introduction

The last several years saw the development of randomized algorithms in high performance computing applications. This increased interest is motivated by the fact that the resulting algorithms are able to outperform deterministic methods while still providing very accurate results (*e.g.* random sampling algorithms that can be applied to least squares solutions or low-rank matrix approximation [1]). In addition to being easier to analyze, the main advantage of such algorithms is that they can lead to much faster solution by performing a smaller number of floating-point operations (*e.g.* [2]), or by involving less communication (*e.g.* [3]). As a result, they potentially allow domain scientists to address larger simulations.

However, to be of full interest for applications, these randomized algorithms must be able to exploit the computing capabilities of current highly distributed parallel systems, which can commonly achieve performance of more than one Tflop/s per node. Since randomized algorithms are supposed to be useful for very large problems, the main challenge for them is to exploit efficiently these distributed computing units and their associated memories. As a matter of fact, large-scale linear algebra solvers from standard parallel distributed libraries like ScaLAPACK [4] often suffer from expensive inter-node communication costs.

* Corresponding author. Tel.: +33 1 69 15 47 27.
E-mail addresses: marc.baboulin@inria.fr (M. Baboulin), dbecker7@eecs.utk.edu (D. Becker), bosilca@icl.utk.edu (G. Bosilca), danalis@icl.utk.edu (A. Danalis), dongarra@eecs.utk.edu (J. Dongarra).

0167-8191/$ - see front matter © 2013 Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.parco.2013.12.003

The advent of multicore processors has undermined the dominance of the SPMD programming style, reviving interest in more flexible approaches such as dataflow approaches. Indeed, several projects [5–9], mostly in the field of numerical linear algebra, revived the use of Directed Acyclic Graphs (DAG), as an approach to tackle the challenges of harnessing the power of multicore and hybrid platforms. In [10], an implementation of a tiled algorithm based on dynamic scheduling for the LU factorization on top of UPC (Unified Parallel C) is proposed. Gustavson et al. [11] uses a static scheduling for the Cholesky factorization on top of MPI to evaluate the impact of data representation structures. All these projects propose ad-hoc solutions to the challenging problem of harnessing all the computing capabilities available on today's heterogeneous platforms, solutions that do not expose enough flexibility to be generalized outside their original algorithmic design space. In the *PaRSEC* project (previously named *DAGuE* [12]) we address this problem in a novel way. Using a layered runtime system, we decouple the algorithm itself from the data distribution, as the algorithm is entirely expressed as flows of data, and from the underlying hardware, allowing the developer to focus solely on the algorithmic level without constraints regarding current hardware trends. The depiction of the algorithm uses a concise symbolic representation (similar to Parameterized Task Graph (*PTG*) proposed in [13]), requiring minimal memory for the DAG representation and providing extreme flexibility to quickly follow the flows of data starting from any task in the DAG, without having to unroll the entire DAG. As a result, the DAG unrolls on-demand, and each participant never evaluates parts of the DAG pertaining to tasks executing on other resources, thereby sparing memory and compute cycles. Additionally, the symbolic representation enables the runtime to dynamically discover the communications required to satisfy remote dependencies, on the fly, without a centralized coordination. Finally, the runtime provides a heterogeneous environment where tasks can be executed on hybrid resources based on their availability.

We illustrate in this paper how linear system solvers based on random butterfly transformations can exploit distributed parallel systems in order to address large-scale problems. In particular, we will show how our approach automatically adjusts data structures, mapping, and scheduling as systems scale up. The application we consider in this paper is the solution of large dense symmetric indefinite systems. Even though dense symmetric indefinite matrices are less common than sparse ones, they do arise in practice, for instance in parameter estimation problems, when considering the augmented system approach [14, p. 77], and in constrained optimization problems [15, p. 246].

For instance, the symmetric indefinite linear system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} \tag{1}$$

arises from the optimization problem

$$\min \frac{1}{2} \|r - b\|_2^2 \text{ subject to } A^T r = 0, \tag{2}$$

where x is the vector of Lagrange multipliers, $r = b - Ax$ is the residual, and I is the identity matrix.

Another class of applications is related to simulations in electromagnetics, when the 3-D Maxwell equations are solved in the frequency domain. The goal is to solve the Radar Cross Section (RCS) problem that computes the response of an object to the wave. In that case, the discretization by the Boundary Element Method (BEM) yields large dense symmetric complex systems which are non Hermitian [16].

Many of the existing methods do not exploit the symmetry of the system, which results in extra computation and extra storage. The problem size commonly encountered for these simulations is a few hundreds of thousands. When the matrix does not fit into the core memories of the computer, this may require the use of out-of-core methods that use disk space as auxiliary memory, which degrades performance (note that an option for very large size is iterative/multipole methods, for which the matrix does not need to be stored in memory). Our in-core solver will be an alternative to these existing methods by minimizing the number of arithmetical operations and data storage.

Symmetric indefinite systems are classically solved using a $\mathrm{LDL}^{\mathrm{T}}$ factorization

$$PAP^T = \mathrm{LDL}^{\mathrm{T}}, \tag{3}$$

where P is a permutation matrix, A is a symmetric square matrix, L is unit lower triangular, and D is block-diagonal, with blocks of size 1×1 or 2×2.

To our knowledge, there is no parallel distributed solver in public domain libraries (*e.g.* ScaLAPACK) for solving dense symmetric indefinite systems using factorization (3). This is why, in many situations, large dense symmetric indefinite systems are solved via LU factorization, resulting in an algorithm that requires twice as much, both arithmetical operations and storage than $\mathrm{LDL}^{\mathrm{T}}$. Moreover, as mentioned in [14], the pivoting strategy adopted in LAPACK [17] (based on the Bunch-Kaufman algorithm [18]) does not generally give a stable method for solving the problem (2), since the perturbations introduced by roundoff do not respect the structure of the system (1).

The approach we propose in this paper avoids pivoting thanks to a randomization technique initially described in [19] for the LU factorization. The main advantage of randomizing here is that we can avoid the communication overhead due to pivoting (this communication cost can represent up to 40% of the global factorization time depending on the architecture, and on the problem size [3]). The resulting distributed solver enables us to address the large dense symmetric indefinite systems

Please cite this article in press as: M. Baboulin et al., An efficient distributed randomized algorithm for solving large dense symmetric indefinite linear systems, Parallel Comput. (2014), http://dx.doi.org/10.1016/j.parco.2013.12.003