Computers and Mathematics with Applications 67 (2014) 199–209

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

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

Minimizing synchronizations in sparse iterative solvers for distributed supercomputers

Sheng-Xin Zhu [a,b,*], Tong-Xiang Gu [c], Xing-Ping Liu [c]

[a] Numerical Analysis Group, Mathematical Institute, Andrew Wiles Building, Oxford, OX2 6GG, UK

[b] Oxford Centre for Collaborative and Applied Mathematics, Mathematical Institute, Andrew Wiles Building, Oxford, OX2 6GG, UK

[c] Laboratory of Computational Physics, Institute of Applied Physics and Computational Mathematics, P.O. Box, 8009, Beijing 100088, PR China

ARTICLE INFO

Article history:
Received 18 April 2013
Received in revised form 11 October 2013
Accepted 8 November 2013

Keywords:
Minimizing communications
Parallel Krylov subspace methods
High performance computing
Distributed supercomputers

ABSTRACT

Eliminating synchronizations is one of the important techniques related to minimizing communications for modern high performance computing. This paper discusses principles of reducing communications due to global synchronizations in sparse iterative solvers on distributed supercomputers. We demonstrate how to minimize global synchronizations by rescheduling a typical Krylov subspace method. The benefit of minimizing synchronizations is shown in theoretical analysis and verified by numerical experiments. The experiments also show the local communications for some structured sparse matrix–vector multiplications and global communications in the underlying supercomputers increase in the order $P^{1/2.5}$ and $P^{4/5}$ respectively, where P is the number of processors.

1. Introduction

Minimizing communications in all kinds of algorithms for multi-core computing platforms is drawing increasing attention. On shared memory computers, reducing computations, exploring structured parallelism and increasing cache hit rate (or data reuse rate) are always given to priorities, while on distributed supercomputers, priority is often given to minimizing all kinds of communications [1]. In this paper, *communication* is referred to as the process of exchanging data between different computing nodes or processors via Message Passing Interface (MPI). Other efficient data movements in local shared memory parts without using MPI, which often require efficient data structure to increase the cache hit rate, go beyond our discussion. Communication time consists of two parts: the first part is *setup time*, which is used to prepare available computing resources (computing nodes or processors) or to wait for necessary information, it depends mainly on the *latency* of the underlying computer system and the number of *synchronizations*; whereas the second part depends on the ratio of the data volume to the bandwidth of the communication systems, or the so called *volume-to-surface ratio*. To reduce communications, one should consider either or both of these factors. This paper focuses on improving the performance of Krylov subspace methods on distributed supercomputers mainly by reducing the setup time via eliminating synchronizations.

The fundamental reasons for us to concentrate on reducing the setup time lies in that synchronizations are often unavoidable when computing reliable inner products on a distributed supercomputer, and the latency of an underlying distributed supercomputer is the hardest to be improved. Recent technology has shown that latency almost cannot be improved, with a tinny improvement, 5.5%/year, whereas bandwidth increases 26%/year and the speed of floating-point

* Corresponding author at: Oxford Centre for Collaborative and Applied Mathematics, Mathematical Institute, Andrew Wiles Building, Oxford, OX2 6GG, UK.

E-mail addresses: shengxin.zhu@maths.ox.ac.uk, zhus@maths.ox.ac.uk (S.-X. Zhu), txgu@iapcm.ac.cn (T.-X. Gu), lxp@iapcm.ac.cn (X.-P. Liu).

0898-1221/$ – see front matter
http://dx.doi.org/10.1016/j.camwa.2013.11.008

Table 1
Communication complexity of kernels of Krylov methods for structured sparse matrices.

Name	Operations	Computation time	Communication time
Matrix–vector multiplication	$y \leftarrow Ax$	$n_z N t_{fl}/P$	$\mu(t_s, t_w, P)$
One inner product	$dot \leftarrow x^T y$	$2Nt_{fl}/P$	$2(t_s + t_w)\omega(P)$
k inner products	–	$2kNt_{fl}/P$	$2(t_s + kt_w)\omega(P)$
Vector update	$y \leftarrow ax+y$	$2Nt_{fl}/P$	–

N: the number of unknowns; P: the number of processors; n_z: the average number of non-zeros per row of A; t_{fl}: the time per float point operation, t_s: the setup time, t_w: the time of passing per volume (unit) of the message. Usually, $t_s \gg t_w$.

operations increases 59%/year [2, p. 109] [3]. Secondly, there were relevant considerations of minimizing communications by analyzing the volume-to-surface ratio for kernels in Krylov iterative methods [4–10].

The remainder of this paper is organized as follows. In Section 2, we briefly discuss the communication cost of the computational kernels of Krylov subspace methods for some structured sparse matrices. Section 3 discusses principles to reduce communications. And then, a detailed case study is presented in Section 4, demonstrating how to reconstruct Krylov subspace methods. Section 5 compares the performance of the reconstructed methods with that of the corresponding serial versions. Finally we present some numerical results to verify our analysis and give some conclusions.

2. Communication cost

In this paper, P is the number of processors, N is the number of unknowns in an underlying linear system, t_{fl} is the time for one floating point operation and t_s is the average latency of an underlying system, and t_w is the volume-to-surface ratio, one unit volume message over the bandwidth of the underlying system. We approximate the total time for one iteration in a Krylov subspace methods as follows

$$T = \underbrace{\phi(N/P)t_{fl}}_{\text{computation time}} + \underbrace{\psi(t_s, t_w)\omega(P)}_{\text{global communication}} + \underbrace{\mu(t_s, t_w, P)}_{\text{local communication}} \tag{1}$$

where $\phi(N/P)t_{fl}$ is the computation time for the three basic computational kernels in classical Krylov subspace methods, namely, vector updates, matrix–vector multiplications and inner product computations. Table 1 shows their computation and communication complexity. Vector updates are parallel in nature and there is no need for communication. For general sparse matrices, efficient matrix–vector multiplications need sophisticated techniques [11], whereas for some structured sparse matrices, sparse matrix–vector multiplications usually only need *local communication*—only exchanging data with its neighbours, in particular the data structure of the underlying sparse matrices is well-organized. We denote the time for this part as $\mu(t_s, t_w, P)$. Inner product computation needs *global communications*, we denote the cost as $\omega(P)$. There are several theoretical models to describe the possible $\omega(P)$, see [12–14] for example. The terms $\mu(t_s, t_w, P)$ and $\omega(P)$ are left vague at the moment, and will be fitted and verified by real numerical results. The basic assumption is $\omega(P) \gg \mu(t_s, t_w, P)$ for large P.

Table 1 gives the computation and communications time of the three kernels needed on a distributed computer for structured sparse matrices.

3. Strategies for reducing communications

It should be mentioned that the fact that communication holds back the scalability of iterative Krylov subspace methods has been noticed since the 1980s. Pioneering researchers have proposed some original ideas to reduce communications [15–21]. However, due to the constraint on the memory size at that time, most of these pioneering works focused mainly on maximizing floating-point operations per unit memory. With the increase in development of hardware, more and more memory and processors have become available, and the gaps between the floating point operations, bandwidth and latency have increased dramatically. The importance of reducing communications is becoming more and more public. To measure the communication efficiency and further impact performance improvement, especially for sparse iterative methods, a new metric for ranking high performance computing systems has been introduced recently [22].

As discussed, both matrix–vector multiplications and inner product computation need communications. Avoiding communications requires us to reschedule these two computational kernels: minimizing communications in matrix–vector multiplications [4,5,9,6]; and minimizing global communications due to the inner product computations [23–26,21,27–33]. We focus on the latter one for reasons mentioned before.

Three strategies have been considered to minimize communications. Firstly, communications should be overlapped with computations as much as possible, this step is easy to be put into practice. Secondly, perhaps the most attractive one, is to remove inner production computations which need global communications, developing inner product free iterative solvers. For some symmetric linear systems, there are already some results. For example, the inner products in the conjugate gradient (CG) method can be replaced by solving a small linear system as described in the s-step methods [34–36,15,37,38] and multiple search direction conjugate gradient method [39,40]. The second remedy is to schedule more inner products to be computed at one synchronization point. In this way, the number of inner products usually increases but more inner