Scaling Iterative Solvers by Avoiding Latency Overhead of Parallel Sparse Matrix Vector Multiplication

R. Oguz Selvitopi[a], Mustafa Ozdal[b], Cevdet Aykanat[a]

[a] *Department of Computer Engineering, Bilkent University, Ankara 06800, Turkey*
[b] *Strategic CAD Labs of Intel Corporation, Hillsboro, OR, 97124, US*

Parallel iterative solvers are the most widely used methods for solving sparse linear systems of equations on parallel architectures. There are two basic types of kernels that are repeatedly computed in these solvers: Sparse-matrix vector multiply (SpMV) and linear vector operations. Since linear vector operations are performed on dense vectors, they are regular in nature and are easy to parallelize. Conversely, SpMV operations generally require specific methods and techniques for efficient parallelization due to irregular sparsity pattern of the coefficient matrix. In literature, several partitioning models and methods are proposed for efficient parallel computation of SpMV operations.

In a single iteration of the solver, SpMV operations cause irregular point-to-point (P2P) communication and inner product computations cause regular collective communication. The partitioning techniques proposed in the literature generally aim at reducing communication volume incurred in P2P communications, which loosely relates to latency overhead incurred in parallel SpMV operations. On current large-scale systems, the message latency overhead is at least as important as the message volume overhead, especially in the case of strong scaling in which average message sizes decrease with increasing number of processors. Our preliminary experiments on two large-scale systems (an IBM BlueGene/Q and a Cray XE6) demonstrate that the startup time is as high as transmitting four-to-eight kilobytes of data.

On the contrary to the studies that aim at hiding latency of collective communication operations [1] (by using nonblocking collective primitives and overlapping with computation), we propose a methodology to directly avoid *all* latency overhead associated with P2P messages of SpMV operations. Our methods rely on the observation that in most of the Krylov subspace methods, each SpMV computation is followed by an inner product computation which involves output vector of the SpMV. This introduces a write/read dependency on this vector between SpMV and inner product computational phases.

In [2], we propose a novel computational rearrangement method to resolve the above-mentioned computational dependency between these two computational phases. By doing so, we remove the communication dependencies between these two phases and enable P2P communications of SpMV and collective communications of inner products to be performed in a single communication phase. The computational rearrangement reduces the number of synchronization points for each SpMV and inner product computation pair by one, that is, the proposed scheme requires a single synchronization point in a typical CG implementation. Then, we realize this opportunity to propose a communication rearrangement method to avoid all latency overhead of P2P messages of SpMV operations. This is achieved by embedding P2P communications into collective communication operations. The proposed embedding scheme reduces both the average and the *maximum* number

of messages handled by a single processor to lg K in an iterative solver with K processors, regardless of the coefficient matrix being solved.

The downside, however, is that the embedding scheme causes extra communication volume due to forwarding of certain vector elements. To address this increase in message volume, two iterative-improvement-based algorithms are proposed. The basic idea of these heuristics is to place the processors that exchange high volume of data close to each other so that the store-and-forward scheme required by the embedding method causes less forwarding overhead. This is a preprocessing step as the partitioning itself and the running time of the described faster heuristic is lower than the partitioning time up to 2048 processors.

The mentioned methods and techniques are validated on Conjugate Gradient method. The 1D row-parallel algorithm is used for SpMV. We tested our methods on two large-scale high performance computing systems Cray XE6 and IBM BlueGene/Q up to 2048 processors with 16 test matrices from University of Florida Sparse Matrix Collection. With using proposed computational and communication rearrangement, we show that we obtain superior scalability performance on both architectures. Our findings indicate that the crucial factor to scale an iterative solver is to keep the message latency overhead low, which dominate the message volume overhead at high processor counts.

References

[1] P. Ghysels, W. Vanroose, Hiding global synchronization latency in the preconditioned conjugate gradient algorithm, Parallel Computing (0) (2013) –. doi:http://dx.doi.org/10.1016/j.parco.2013.06.001.
URL http://www.sciencedirect.com/science/article/pii/S0167819113000719

[2] O. Selvitopi, M. Ozdal, C. Aykanat, A novel method for scaling iterative solvers: Avoiding latency overhead of parallel sparse-matrix vector multiplies, Parallel and Distributed Systems, IEEE Transactions on PP (99) (2014) 1–1. doi:http://dx.doi.org/10.1109/TPDS.2014.2311804.
URL http://www.computer.org/csdl/trans/td/preprint/06766662-abs.html