Iterative Solver Selection Techniques for Sparse Linear Systems*

Extended Abstract†

Kanika Sood
University of Oregon
Eugene, OR, USA
kanikas@cs.uoregon.edu

Boyana Norris
University of Oregon
Eugene, USA
norris@cs.uoregon.edu

Elizabeth Jessup
University of Colorado Boulder
Boulder, USA
elizabeth.jessup@colorado.edu

ABSTRACT

Scientific and engineering applications often involve the solution of large sparse linear systems; hence, scalable preconditioned iterative methods are a popular choice. There are many software libraries that offer a variety of preconditioned Newton-Krylov methods for solving sparse problems. However, the selection of an optimal Krylov method remains to be user's responsibility. This document outlines the technique we propose for the optimal solver method suggestions based on the problem characteristics and the amount of communication involved in the Krylov methods.

KEYWORDS

linear systems; taxonomy; machine learning; communication model

ACM Reference Format:
Kanika Sood, Boyana Norris, and Elizabeth Jessup. 2018. Iterative Solver Selection Techniques for Sparse Linear Systems: Extended Abstract. In *Proceedings of International Conference on Parallel Processing (ICPP'18),* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 2 pages. https://doi.org/10.475/123_4

Problems from a variety of domains can be represented in the form of linear systems of equations. Such domains include astrophysics, computational fluid dynamics, thermodynamics among many others, e.g. [3, 6]. The solution of large linear systems is a fundamental component of many scientific and engineering applications, and high-performance numerical frameworks rely on advanced and optimized packages like PETSc [1] for the solution of such systems. For solving these equations, there are two main approaches: direct and iterative solver techniques. Direct solvers find a solution close to the exact solution of the problem by using a limited set of operations. Iterative methods provide an approximation of the solution by starting with an initial guess and updating the solution approximations over multiple iterations. Linear systems can be classified as dense or sparse, depending on the number of zero elements present in the coefficient matrix. Systems with a large number of zero-valued elements are referred to as sparse linear systems. The second class of linear systems are dense linear systems where majority of the elements have non-zero values.

*Produces the permission block, and copyright information
†The full version of the author's guide is available as `acmart.pdf` document

ICPP'18, August 2018, Eugene, OR USA

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

In many applications, the solution of large sparse linear systems is a key computation whose performance dominates the overall solution time. For example, applications that solve nonlinear partial differential equations through numerical approximations such as the Newton-Krylov family of methods, spend most of their time in the iterative linear system solution, which can be performed by any of the large number of preconditioned Krylov methods available. While their functionalities of these methods are equivalent, their performance (how fast a solution is found) varies greatly depending on the characteristics of the input linear system. This proliferation of available solution methods makes the task of selecting a specific algorithm *performs well* extremely challenging.

My research enables computer and computational scientists to choose a well-performing solution technique for their problems that can be represented as linear systems. While ideally we would like to find the optimal method, in practice, this is infeasible due to the problem size and complexity. Hence, I investigate machine learning and other modeling approaches to determine what solvers are likely to perform significantly better than the majority of solution methods for a given problem. We use PETSc, which offers a variety of scalable solvers and preconditioners, that can be used for solving scientific applications modeled by partial differential equations. For enabling well-performing solver selection, we have defined a technique for selecting solution methods with minimal or no input from the user. In this document, we illustrate how this approach is used in practice for solver selection.

1 APPROACH OVERVIEW

We model two aspects of the performance of Krylov methods: convergence behavior and communication overhead. Our ML model captures the convergence behavior of the Krylov methods via a supervised machine learning (ML) approach. Our analytical approach characterizes the communication performed in the Krylov methods to quantify their scalability. The convergence model (ML model) can be used as a standalone for getting solver suggestions at moderate scale. For larger processor counts, the ML model results can be improved by combining it with the communication model. In this section, we describe the two models and how to combine them.

1.1 Convergence Model (ML model)

We classify different Krylov methods based on their performance using several supervised ML techniques. For training the model, we use the University of Florida matrices [2] with the right hand side vector with all elements set to 1. We solve each linear system with multiple solver-preconditioner combinations and capture the solver timing. We compute various features of these systems such as the

trace, or row/column value variability. These properties along with the solver-preconditioner combination, together form the training set. Next, we build a binary classifier that labels solvers as "good" or "bad" based on their computation time. We apply different ML algorithms [4] to classify solvers and perform supervised learning. We validate the model by performing 10-fold cross validation and a 66 − 34% train-test validation. Finally, we select the algorithm that performs the best and classifies solver performance most effectively.

One of the most expensive steps in the convergence modeling is the feature collection. For reducing its cost, we identify the most relevant features, and rebuild the ML classifier using only the reduced feature sets. The output of the convergence model is a list of "good" solver recommendations. Figure 1 shows the different stages involved in the convergence model.

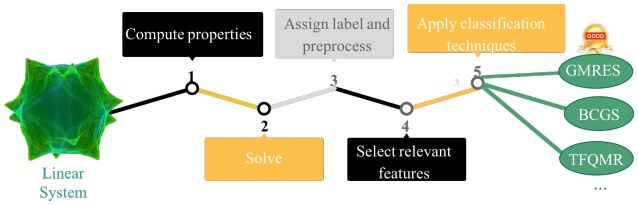

Figure 1: Convergence model for solver classification.

1.2 Communication Model

For problems that require high levels of parallelism (> 1000 cores), we propose a method for modeling the performance of parallel preconditioned Krylov methods. With this model, we analyze the solver and preconditioner algorithms provided by PETSc identifying the operations that perform inter-process communication for each iteration. Next, we count the number of times these operations are called for each algorithm. We only focus on the iteration and do not consider operations in setup or I/O functions which are outside of the Krylov solver iteration. In this model, we analyze communication alone, since solution time is modeled using the ML model. We exclude the operations that are common for all solvers and preconditioners.

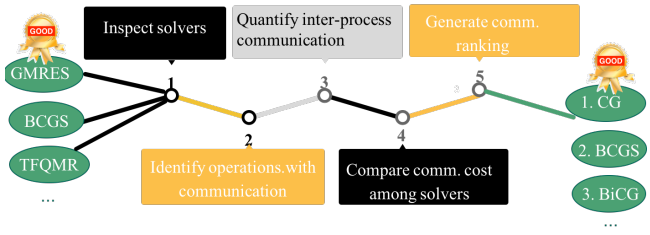

Figure 2: Communication model for solver ranking.

We use the analytical ranking of preconditioned Krylov methods to address the challenge of modeling the scalability of algorithms. We validate the accuracy of the model by applying it on a numerical simulation of driven fluid flow in a two dimensional cavity. Figure 2 shows the stages in the communication model. The result is a communication-wise ranking for the preconditioned and non-preconditioned cases. The solver ranking list obtained from this model, is combined with the ML-generated solver list to find the intersection, which generates a ranked list of "good" solvers.

1.3 Model Usage

We make a solver suggestion based on the ML approach and the communication based approach together, because either of the models as a standalone does not capture both the computation and communication aspects. For modeling the solvers scalability for large processor counts, collecting the training data for the convergence model is prohibitively expensive. In such cases, we model the computation aspect with the ML model and parallel overhead with the communication-based model. This is achieved by applying the ML-based approach in combination with the communication-based performance model to enable solver recommendations at different scales of parallelism.

2 MATRIX-FREE FEATURE COMPUTATION

An aspect of our research focuses on a matrix-free approach for selecting iterative Krylov methods. Sparse systems use less memory than dense systems because we store only the non-zero elements of the sparse systems. When all the matrix elements are stored and available at any given time, various features of the matrix can be computed, such as the number of rows, matrix diagonal, and others. For an extremely large matrix, storing it and performing matrix operations can be very expensive due to memory cost and computation time.

Krylov methods can, however, solve the linear system by approximating matrix-vector products without explicitly assembling the matrix [5], thus supporting even larger problems.

3 CONCLUSION

This research enables iterative solver recommendations for sparse linear systems by modeling the convergence behavior and the parallel overhead. When solving large sparse linear systems, users can rely on the the ML classifier recommendations for moderate scales of parallelism (hundreds of tasks). For cases where modeling both computation and communication is useful, we combine the ML suggestions by finding the top-ranked methods (in terms of communication overhead) within that solver set.

REFERENCES

[1] Satish Balay, Kris Buschelman, William D Gropp, Dinesh Kaushik, Matthew G Knepley, L Curfman McInnes, Barry F Smith, and Hong Zhang. 2001. PETSc web page.
[2] Timothy A Davis and Yifan Hu. 2011. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)* 38, 1 (2011), 1.
[3] Po Geng, J Tinsley Oden, and RA Van De Geijn. 1996. Massively parallel computation for acoustical scattering problems using boundary element methods. *Journal of Sound and Vibration* 191, 1 (1996), 145–165.
[4] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter* 11, 1 (2009), 10–18.
[5] Erich Kaltofen and Austin Lobo. 1999. Distributed matrix-free solution of large sparse linear systems over finite fields. *Algorithmica* 24, 3-4 (1999), 331–348.
[6] Florent Sourbier, Stéphane Operto, Jean Virieux, Patrick Amestoy, and Jean-Yves LâĂŹExcellent. 2009. FWT2D: A massively parallel program for frequency-domain full-waveform tomography of wide-aperture seismic dataâĂŤPart 1: Algorithm. *Computers & Geosciences* 35, 3 (2009), 487–495.