Sparse Matrix Partitioning for Parallel Eigenanalysis of Large Static and Dynamic Graphs

Michael M. Wolf* and Benjamin A. Miller
Lincoln Laboratory
Massachusetts Institute of Technology
Lexington, MA 02420
mmwolf@sandia.gov, bamiller@ll.mit.edu

***Abstract*—Numerous applications focus on the analysis of entities and the connections between them, and such data are naturally represented as graphs. In particular, the detection of a small subset of vertices with anomalous coordinated connectivity is of broad interest, for problems such as detecting strange traffic in a computer network or unknown communities in a social network. These problems become more difficult as the background graph grows larger and noisier and the coordination patterns become more subtle. In this paper, we discuss the computational challenges of a statistical framework designed to address this cross-mission challenge. The statistical framework is based on spectral analysis of the graph data, and three partitioning methods are evaluated for computing the principal eigenvector of the graph's residuals matrix. While a standard one-dimensional partitioning technique enables this computation for up to four billion vertices, the communication overhead prevents this method from being used for even larger graphs. Recent two-dimensional partitioning methods are shown to have much more favorable scaling properties. A data-dependent partitioning method, which has the best scaling performance, is also shown to improve computation time even as a graph changes over time, allowing amortization of the upfront cost.**

I. Introduction

Due to their utility in identifying subtle patterns of coordination, graph analytics have proven useful in several diverse application areas. Graph analysis—analysis of entities and the connections or interactions between them—has been used in ISR (Intelligence, Surveillance, and Reconnaissance, where graphs represent entities and relationships detected through multiple sources), social networks (where graphs represent relationships between individuals or document), and cyber security (where graphs may represent communication patterns of computers on a network). The objective of the graph analysis in each of these applications is to find anomalous subgraphs in the data, i.e., subsets of the entities with a statistically unlikely connection pattern. The application of graph analytics to these areas becomes harder as the patterns of coordination grow more subtle and the background graph grows larger and noisier.

A recent framework for anomalous subgraph detection focuses on spectral analysis of graph residuals [1]. Using a matrix representation of a graph, and an assumed model for expected behavior, the difference between the observed graph and its expected value can be computed, and this difference is projected into a low-dimensional space where subgraph detection and localization algorithms are run. This dimensionality reduction is typically done via a partial eigendecomposition, and is the main computational bottleneck of the processing chain.

Previous work focused on performing the partial eigendecomposition while scaling the size of the graph to over one billion vertices (entities) on commodity hardware [2]. In this paper, we consider several possible partitioning methods, evaluated on a high-performance supercomputer. While standard one-dimensional partitioning allows the computation of the principal eigenvector of the residuals matrix of a 4-billion-vertex graph in approximately 5 minutes on 16384 cores, recent two-dimensional partitioning techniques enable much more favorable scaling properties, as they are designed to limit the number of messages during inter-process communication. In addition to static graphs, we consider simple, dynamic graphs that add edges (connections) over time. One widely-used graph simulation model demonstrates a benefit in run time from a data-dependent partitioning algorithm, even when the partition is calculated when only 30% of the edges are present. This suggests that the data-dependent method is worth its additional computational overhead, as its cost can be amortized over time.

The remainder of this paper is organized as follows. In Section II, we briefly review the graph anomaly detection processing chain from [1]. Section III outlines implementation details of the eigensolver. In Section IV, we present the measured run times for several simulated graphs of a variety of sizes, using three partitioning algorithms to distribute the graphs among cores. Section V provides a summary and discussion.

This work is sponsored by the Intelligence Advanced Research Projects Activity (IARPA) under Air Force Contract FA8721-05-C-0002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Disclaimer: The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA or the U.S. Government.

*Currently with Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

978-1-4799-6233-4/14/$31.00 ©2014 IEEE

II. Signal Processing for Graphs

The statistical framework discussed here has been developed for the detection of subtle patterns in massive datasets as part of an effort called Signal Processing for Graphs (SPG). The goal of this project is to apply classical signal processing concepts to graph data in order to find these anomalies [1]. The SPG framework resolves a binary hypothesis test, determining statistically whether, for the observed data, an anomalous subgraph is present (rejection of the null hypothesis that there is no anomalous subgraph) or not (acceptance of the null hypothesis). An anomalous subgraph, in this context, is a subset of vertices among which far more edges occur than expected under the assumed model (i.e., a statistical outlier in terms of its connection density). The detection algorithms in the SPG framework are based on the concept of a residual graph (shown in Fig. 1), which is formed by subtracting the expected graph data from the observed graph data. The detection framework is designed to detect coordinated deviations from the expected graph topology (coordination in the residual graph). The SPG processing chain has several stages including temporal integration (where the graph is formed from multiple time steps), construction of the expected topology graph, dimensionality reduction, detection of anomalous subgraphs, and identification of those anomalous subgraphs (assuming such a subgraph is detected).

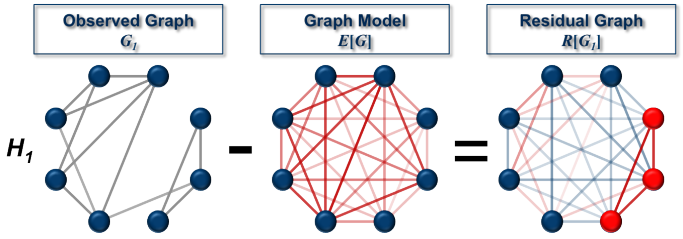

Fig. 1. Residual graph formation.

A common way of representing a graph is using an adjacency matrix. For an unweighted graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, an adjacency matrix $A = \{a_{ij}\}$ is 1 if there is an edge from vertex i to vertex j (where the vertices are given arbitrary numeric labels) and is 0 elsewhere. This matrix representation enables the use of classical linear algebra tools for dimensionality reduction.

Dimensionality reduction is a particularly important step in the processing chain, as it makes the detection and identification procedures feasible. Typically, this is done by decomposing the residual matrix through eigendecomposition (although a singular value decomposition would work as well). Although relatively strong signals can be detected with only one eigenvector, the more powerful detection methods that are needed to detect more subtle anomalies require hundreds of eigenvectors. As mentioned previously, this dimensionality reduction step dominates the computation in the graph analytic processing chain. Thus, our initial efforts have been to improve the performance of this dimensionality reduction step, focusing on speeding up the solution of the eigen system, $Bx_i = \lambda_i x_i$, where $B = A - \mathbb{E}[A]$ is the residual matrix. For the purpose of this work, we have focused on one particular residuals model, the modularity matrix [3]. The modularity matrix uses a rank one approximation to the observed data, $\mathbb{E}[A] = kk^T/(2|E|)$, where k is the degree vector, i.e., the vector of the number of edges connected to each vertex. The (orthogonal) principal eigenvectors of B are computed, to find the linear subspace in which the residuals are largest. This can also be done for a directed graph, where the adjacency matrix may not be symmetric. As in [2], [4], we use a "symmetrized" version of the modularity matrix, expressed as

$$\frac{1}{2}\left(A + A^T - \frac{k_{\text{out}}k_{\text{in}}^T}{|E|} - \frac{k_{\text{in}}k_{\text{out}}^T}{|E|}\right),$$

where k_{in} is the vector of inward edge counts and k_{out} is the corresponding vector for outward edges. The formulation uses the same rank-one assumption as for undirected graphs, but takes the average of the residuals matrix and its transpose to make the matrix symmetric, thus ensuring real-valued, orthogonal eigenvectors. Once the lower-dimensional residuals space is computed, a detection statistic is calculated to determine the presence of an anomalous subgraph. In the subsequent sections, we detail the parallel processing aspects of finding this subspace in a large graph.

III. Implementation

One popular approach to solving big data analytics problems is to use the MapReduce model of computation [5]. However, this is infeasible for many of our applications' throughput requirements. Thus, we turn to a more traditional high performance computing, sparse linear algebra approach. This approach allows us to find anomalies in very large graphs in a reasonable amount of time (e.g., latency of a few minutes for a graph with several billion vertices).

In previous work [2], SLEPc [6] was used to solve the modularity matrix based eigensystem resulting from the SPG framework. The authors solved eigensystems resulting from graphs with up to one billion vertices. On a small compute cluster, they used 64 processing cores to solve this problem in a little over an hour. We leverage the Anasazi eigensolver [7] from the Trilinos project [8] to greatly approve upon this result, solving the eigensystem resulting from the one billion vertex graph in under a minute. Anasazi was chosen since it allowed us some additional flexibility in our data partitioning that was unavailable with SLEPc. In the next section, we see that that additional flexibility was crucial to achieving good performance with the eigensolver as we increase the number of cores. In particular, we use the block Krylov-Schur method in Anasazi to find the eigenvectors corresponding to the eigenvalues with the largest real components (which correspond to highly coordinated residuals). We use 64 bit global ordinals (necessary for very large graphs) and user-defined operators to efficiently define our residual matrices without explicitly forming these potentially dense matrices. We have used Anasazi to find eigenpairs resulting from graphs with over four billion vertices.

IV. Results

We benchmarked the eigendecomposition of the modularity matrix (as previously described) on the Hopper supercomputer, managed by the National Energy Research Scientific Computing Center at Lawrence Berkeley National Laboratory. We timed this eigendecomposition for graphs at several different scales (from four million to four billion vertices), using R-Mat matrices [9] as surrogates for realistic network data. R-Mat matrices were a convenient choice in that they can be easily scaled to different problem sizes. For the R-Mat matrices in this work, we used the parameters a=0.5, b=0.125, c=0.125 and an average of 8 nonzeros per row. This setting creates greater community structure than the standard Graph 500 parameters, thus creating more realistic simulated networks. We ran numerical experiments on up to 16384 processing cores to solve these very large graph problems.

A. One-Dimensional Partitioning

Initially, we explored the performance of the eigensolver when using one-dimensional (1D) data partitioning of the sparse matrix to determine how the matrix nonzeros and vector entries in the eigensystem should be distributed across the processing cores. In 1D partitioning, each processing core is assigned all the non zeros in either a set of rows or columns. 1D distributions are the de facto standard for sparse matrices (being used nearly universally, either explicitly or implicitly) and tend to work well for more traditional computational science and engineering problems, such as structural mechanics and computational electromagnetics. For this initial set of numerical experiments, we use 1D random row-based partitioning, which assigns each core the non zeros for a random set of rows. Although a simple method, 1D random partitioning has the advantage of being inexpensive to compute and resulting in good load balance for the R-Mat matrices when the number of rows per core is sufficiently large.

Fig. 2 shows the run time to find the first eigenvalue for the eigensystem when we increase the problem size (keeping the number of vertices (or rows) per core constant at 2^{18}) while increasing the number of cores, also known as weak scaling. An ideal weak scaling result has no increase in run time as the number of cores increases, and for the first 2048 cores, we see only a modest increase. However, as we increase the number of cores further, the run time increases significantly. This is an indication of a performance problem and does not bode well for the effective use of many cores to find the eigenpairs quickly.

Fig. 3 shows the run time to find the first eigenvalue of a 2^{23}-vertex R-Mat matrix as we increase the number of cores (strong scaling). In this plot, the performance difficulty of solving this eigensystem when utilizing a simple 1D random distribution (blue curve) is even more apparent than with the weak scaling result. The scalability is clearly limited and the run time actually increases for more than 1024 cores.

As we see from the weak and strong scaling results, 1D distributions, although very effective for more traditional computational science and engineering problems, tend to be very ineffective for matrices derived from power-law graphs. One major problem with applying one-dimensional partitioning to these types of matrices is that the resulting sparse matrix-dense vector multiplication (SpMV), the computational workhorse for the eigensolver, requires an all-to-all communication operation. For very sparse matrices, this all-to-all communication may be too expensive since the computation costs quickly become relatively small (in comparison to the communication) as the number of cores is increased. Figs. 4 and 5 show the difference between the communication pattern from the sparse matrix-dense vector product of a more traditional computational science and engineering problem and power-law graph data for 64 cores. In these figures, the row represents the source process and the column represents the destination process. The color corresponds to the amount of data communicated (with the maximum value of the scale being the number of rows divided by the number of cores). In Fig. 4, we see the communication pattern for a 9-point 2D finite difference method. The 1D distribution works well for the finite difference matrix, with each process having to communicate with at most two other processes. In Fig. 5, we see the communication pattern for the R-Mat matrix.

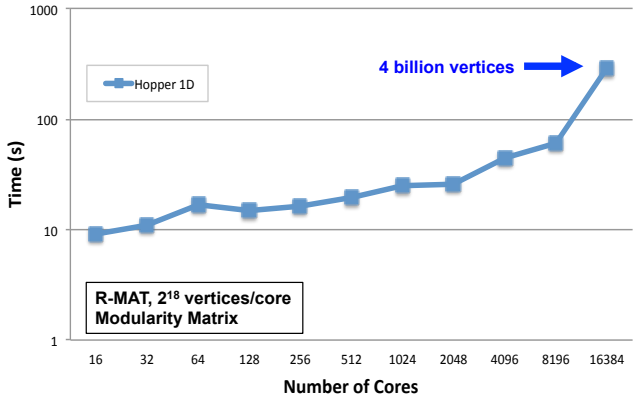

Fig. 2. Weak Scaling. Time (in seconds) to find one eigenvector using 1D random partitioning.

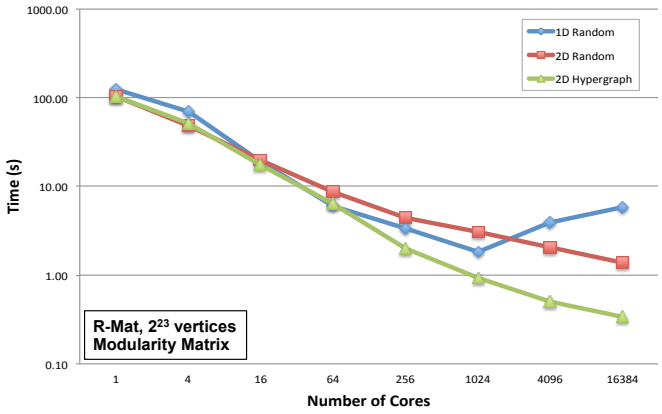

Fig. 3. Strong Scaling. Time (in seconds) to find one eigenvector when using different data distributions: 1D random, 2D Cartesian random, and 2D hypergraph random.

This communication pattern is undesirable, with each process having to communicate with all other processes.

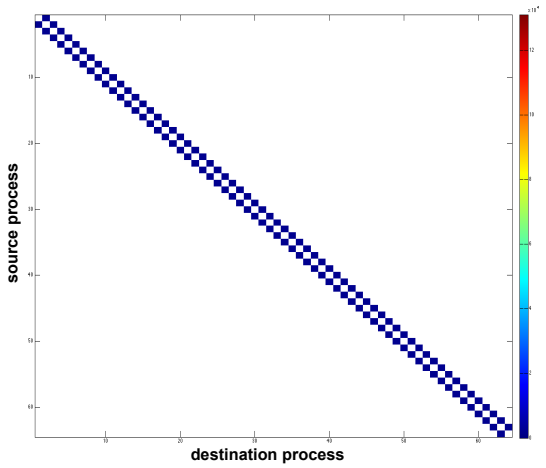

Fig. 4. Communication pattern for SpMV with 1D block distribution of 9-point 2D finite difference matrix over 64 cores. Row represents the source process/core and column represents the destination process/core. The color corresponds to the amount of data communicated (with the maximum value of the scale being the number of rows divided by the number of cores).

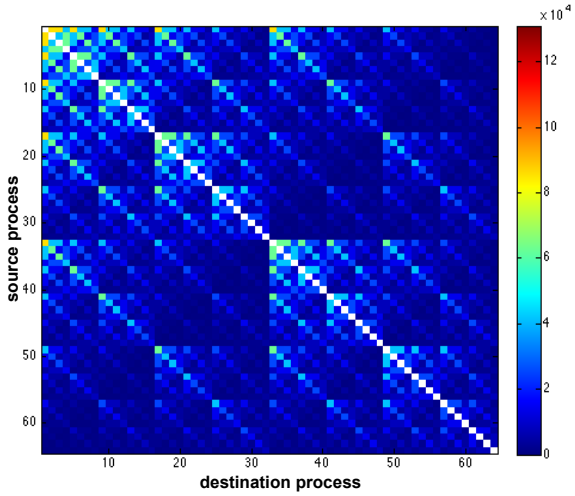

Fig. 5. Communication pattern for SpMV with 1D block distribution of R-Mat matrix over 64 cores. Row represents the source process/core and column represents the destination process/core. The color corresponds to the amount of data communicated (with the maximum value of the scale being the number of rows divided by the number of cores).

B. 2D Partitioning

With 1D distributions, solving the eigendecomposition problem for very large power-law graphs is infeasible for our run time requirements. Thus, we turn to 2D distributions for a solution, where the data is no longer partitioned by row or column, but more general sets of nonzeros are assigned to the different cores. 2D partitioning provides additional flexibility that is lacking in 1D partitioning. There has been recent work on using this additional flexibility of two-dimensional distributions to bound the number of messages to be communicated in the resulting SpMV operation [10], [11]. In Fig. 3, we show the improved results for two of these 2D distributions: one based on a random partitioning of rows/columns with an imposed 2D block Cartesian structure (2DR, red line, [10]) and one that uses hypergraph partitioning instead of random assignment of the rows/columns to improve the communication volume (2DH, green line, [11]). It is important to note that one nice property of the 2DH method is that the partitioning time is only slightly greater than the partitioning time of 1D hypergraph partitioning (2DH uses 1D hypergraph partitioning plus a small amount of additional work). Using these 2D methods, we are able to get more reasonable performance scaling and find eigenvectors for very large power-law graphs, with the 2DH method performing particularly well. One important factor contributing to these improved performance is that the sparse matrix-dense vector product no longer requires all-to-all communication. This allows the run time to continue to decrease with additional cores further than was seen for the 1D method. In Fig. 6, we see the communication pattern for one phase (2D partitioning results in two phases of communication) of the SpMV operation with an R-Mat matrix, when the 2DR method has been used to distribute the data. For this 2D method, each process has to communicate with at most $\sqrt{P}-1$ other processes, where P is the total number of processes.

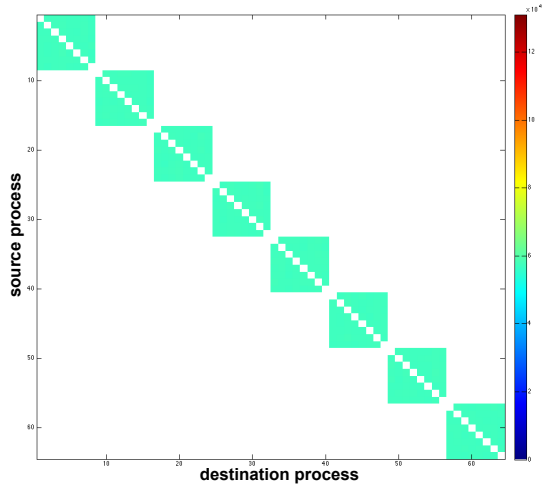

Fig. 6. Communication pattern for SpMV (first phase of communication) with 2D distribution of R-Mat matrix over 64 cores (2DR). Row represents the source process/core and column represents the destination process/core. The color corresponds to the amount of data communicated (with the maximum value of the scale being the number of rows divided by the number of cores).

C. 2D Hypergraph Partitioning and Dynamic Graphs

The challenge with the 2DH method is that it is relatively expensive to compute the distribution due to the expense of the hypergraph partitioning step. For our 8 million row R-Mat problem, approximately 40,000 SpMV operations are required to amortize the additional cost of calculating the 2DH distribution versus the 2DR distribution (as shown in Fig. 7). Since we typically need at most a few thousand SpMV

operations in our eigensolver, the 2DH distribution must be effective for multiple observed graphs to be useful.

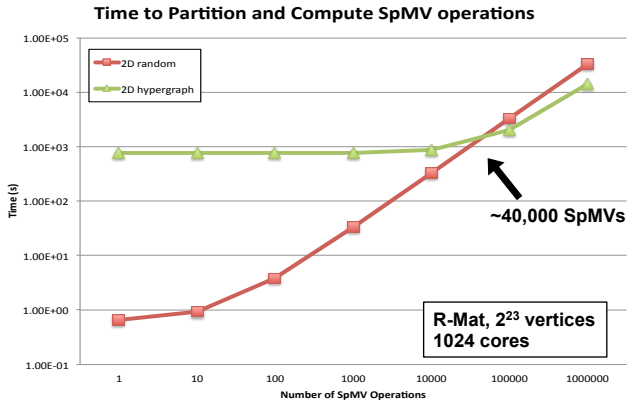

Fig. 7. Challenge with hypergraph partitioning. Time (in s) to partition once and compute multiple SpMV operations as graph evolves over time.

We explored this effectiveness using a simple dynamic graph model in which we partition an initial graph and use our R-Mat generator to add additional edges (using the defined distribution) to produce a sequence of additional graphs. Fig. 8 shows the run time (normalized by the number of edges) of the SpMV operations for both the 2DR and the 2DH distributions applied to several graphs that evolve from an initial graph containing 30% of the edges in the final graph. Although the 2DH distribution loses some of its advantage over the 2DR distribution, it still has a significant advantage at the final graph. Thus, it may be possible to effectively amortize the expensive 2DH distribution over several graphs and use this distribution to our computational advantage. We plan to explore this further for additional dynamic graph models.

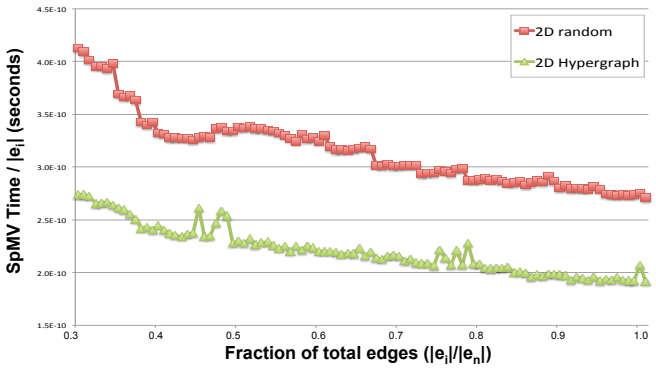

Fig. 8. Dynamic partitioning. SpMV time in seconds (normalized by number of nonzeros) as graph evolves.

These dynamic graph partitioning results also suggest that it may be possible to partition a simplified graph problem (one where edges have been removed) without significant loss of partitioning quality (i.e., without significant increase in SpMV execution time) when this partition is applied to the original graph. In preliminary results, this technique of partitioning a sampled graph is very promising. Using the hollywood movie actor network graph, hollywood-2009 [12], we see a great decrease in execution time when partitioning graphs that are formed by uniformly sampling the edges of the original graph (Fig. 9) with no significant increase in execution time of SpMV (Fig. 10) when these partitions are applied to the original graph. In the future, we plan to explore graph sampling to improve hypergraph partitioning times further.

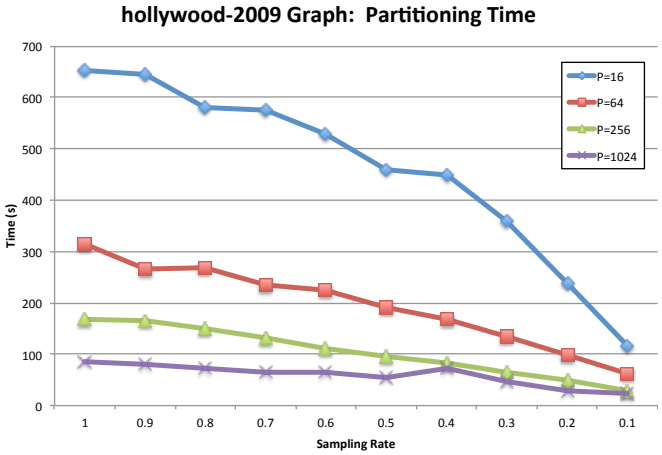

Fig. 9. Partitioning of sampled graph. Partitioning time in seconds for hollywood-2009 graph with the edges of the graph being sampled uniformly at different rates.

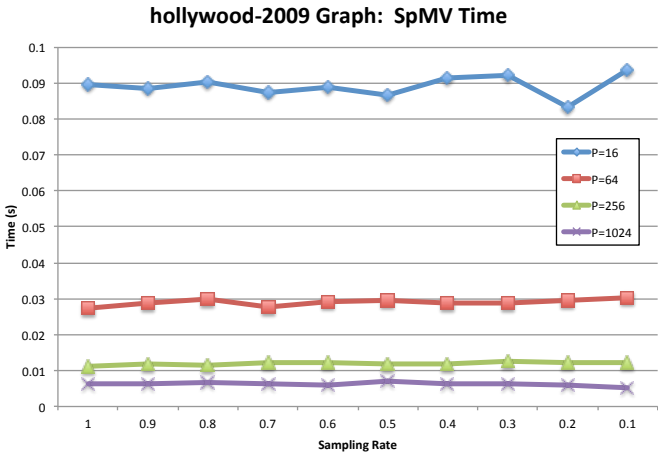

Fig. 10. Partitioning of sampled graph. SpMV time in seconds for hollywood-2009 graph when partitions of sampled graphs (with various sample rates) applied to original graph.

V. Summary/Conclusions

We have briefly described our signal processing framework for detecting subtle anomalies in noisy, very large data sets. The limiting step in this processing chain is solving very large eigensystems as part of dimensionality reduction. Using 2D distributions that bound the number of messages that are communicated, we are able to significantly improve the scalability of the eigensolver and find the desired eigenvectors in a reasonable amount of time. The hypergraph based partitioning method was found to be particularly effective in

reducing both the amount of communication and the runtime of the eigensolver. Unfortunately, this 2D hypergraph partitioning method is significantly more expensive than the random partitioning methods and this additional cost can be hard to amortize when solving an eigensystem resulting from a given graph. However, preliminary results show that it may be feasible to reuse a 2D hypergraph partition for multiple graphs for an over all gain. In future work, we plan to look into more realistic dynamic graph models to determine if this is a reasonable approach. We hypothesize that graph models where community structure remains similar over time will be able to reuse previous partitions effectively, since this would imply that ideal graph cuts computed by the partitioning algorithm would be consistent.

In summary, we have been successful in leveraging high performance eigensolvers and 2D partitioning methods to solve very large eigensystems. With these improvements to the performance of the eigensolver, we hope to be able to implement the full SPG processing chain, and detect anomalies in very large graphs (with billions of vertices) in the near future.

ACKNOWLEDGMENT

The empirical results in this paper were computed using resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The partitioning with sampling work and the subsequent hollywood2009 results were completed by Dr. Wolf after having transitioned to Sandia National Laboratories.

The authors wish to thank the LLGrid team at MITLL for their support in running many of our preliminary experiments, and Dr. Albert Reuther for his support of this work.

REFERENCES

[1] B. A. Miller, N. T. Bliss, P. J. Wolfe, and M. S. Beard, "Detection theory for graphs," *Lincoln Laboratory Journal*, vol. 20, no. 1, pp. 10–30, 2013.

[2] E. M. Rutledge, B. A. Miller, and M. S. Beard, "Benchmarking parallel eigen decomposition for residuals analysis of very large graphs," in *Proc. IEEE High Performance Extreme Computing Conf.*, 2012.

[3] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E*, vol. 74, no. 3, 2006.

[4] E. A. Leicht and M. E. J. Newman, "Community structure in directed networks," *Phys. Rev. Lett.*, vol. 100, no. 11, pp. 118 703–(1–4), Mar 2008.

[5] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[6] V. Hernandez, J. E. Roman, and V. Vidal, "SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems," *ACM Trans. Math. Software*, vol. 31, no. 3, pp. 351–362, 2005.

[7] C. G. Baker, U. L. Hetmaniuk, R. B. Lehoucq, and H. K. Thornquist, "Anasazi software for the numerical solution of large-scale eigenvalue problems," *ACM Trans. Math. Softw.*, vol. 36, no. 3, pp. 13:1–13:23, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1527286.1527287

[8] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, "An overview of the Trilinos project," *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 397–423, 2005.

[9] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-MAT: A recursive model for graph mining," in *Proc. of the 4th SIAM Conference on Data Mining*, 2004, pp. 442–446.

[10] A. Yoo, A. H. Baker, R. Pearce, and V. E. Henson, "A scalable eigensolver for large scale-free graphs using 2D graph partitioning," in *Proceedings of Supercomputing*. New York, NY, USA: ACM, 2011, pp. 63:1–63:11. [Online]. Available: http://doi.acm.org/10.1145/2063384.2063469

[11] E. G. Boman, K. D. Devine, and S. Rajamanickam, "Scalable matrix computations on large scale-free graphs using 2D graph partitioning," in *Proceedings of Supercomputing*. New York, NY, USA: ACM, 2013, pp. 50:1–50:12. [Online]. Available: http://doi.acm.org/10.1145/2503210.2503293

[12] T. A. Davis and Y. Hu, "The University of Florida sparse matrix collection," *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 1:1–1:25, Dec. 2011.