Final Report for

DEGAS: Dynamic Exascale Global Address Space Programming Environments

Sponsoring DOE Entity: Office of Advanced Scientific Computing Research (OASCR)

Grant DE-FC02-12ER26090/DE-SC0008700

Reporting Period 9/1/16 - 8/31/17

Grant Period 9/1/12-8/31/17

Principal Investigator: James Demmel

Research Organization involved: UC Berkeley

Abstract:

The Dynamic, Exascale Global Address Space programming environment (DEGAS) project will develop the next generation of programming models and runtime systems to meet the challenges of Exascale computing.

The Berkeley part of the project concentrated on communication-optimal code generation to optimize speed and energy efficiency by reducing data movement. Our work developed communication lower bounds, and/or communication avoiding algorithms (that either meet the lower bound, or do much less communication than their conventional counterparts) for a variety of algorithms, including linear algebra, machine learning and genomics.

Summary:

Runtime Data Layout Scheduling for Machine Learning Dataset
Yang You and James Demmel
ICPP'17 (Intern. Conf. Parallel Proc.)
Abstract:
Machine Learning (ML) approaches are widely-used classification/regression methods for data mining applications. However, the time-consuming training process greatly limits the efficiency of ML approaches. We use the example of SVM (traditional ML algorithm) and DNN (state-of-the-art ML algorithm) to illustrate the idea in this paper. For SVM, a major performance bottleneck of current tools is that they use a unified data storage format because the data formats can have a significant influence on the complexity of storage and computation, memory bandwidth, and the efficiency of parallel processing. To address the problem above, we study the factors influencing the algorithm's performance and conduct auto-tuning to speed up SVM training. DNN training is even slower than SVM. For example, using a 8-core CPUs to train AlexNet model by CIFAR-10 dataset costs 8.2 hours. CIFAR-10 is only 170 MB, which is not efficient for distributed processing. Moreover, due to the algorithm limitation, only a small batch of data can be processed at each iteration. We focus on finding the right algorithmic parameters and using auto-tuning techniques to make the algorithm run faster. For SVM training, our implementation achieves 1.7 – 16.3x speedup (6.8x on average) against the non-adaptive case (using the worst data format) for various datasets. For DNN training on CIFAR-10 dataset, we reduce the time from 8.2 hours to only roughly 1 minute. We use the benchmark of dollars per speedup to help the users to select the right deep learning hardware.

Performance Characterization of De Novo Genome Assembly on Leading Parallel Systems
M. Ellis, E. Georganas, R. Egan, S. Hofmeyr, A. Buluc, B. Cook, L. Oliker, K. Yelick
Euro-Par'17, August 2017
Abstract.
De novo genome assembly is one of the most important and challenging computational problems in modern genomics; further, it shares algorithms and communication patterns important to other graph analytic and irregular applications. Unlike simulations, it has no floating point arithmetic and is dominated by small memory transactions within and between computing nodes. In this work, we focus on the highly scalable HipMer assembler and identify the dominant algorithms and communication patterns, also using microbenchmarks to capture the workload. We evaluate HipMer on a variety of platforms from the latest HPC systems to ethernet clusters. HipMer performs well on all single node systems, including the Xeon Phi manycore architecture. Given large enough problems, it also demonstrates excellent scaling across nodes in an HPC system, but requires a high speed network with low overhead and high injection

rates. Our results shed light on the architectural features that are most important for achieving good parallel efficiency on this and related problems.

Matrix factorizations at scale: A comparison of scientific data analytics in Spark and C+MPI using three case studies
A.Gittens, A. Devarakonda, E. Racah, M. Ringenburg, L. Gerhardt, J. Kottalam, J. Liu, K. Maschhoff, S. Canon, J. Chhugani, P. Sharma, J. Yang, J. Demmel, J. Harrell, V. Krishnamurthy, M. Mahoney, Prabhat
2016 IEEE Intern. Conf on Big Data
Abstract:
We explore the trade-offs of performing linear algebra using Apache Spark, compared to traditional C and MPI implementations on HPC platforms. Spark is designed for data analytics on cluster computing platforms with access to local disks and is optimized for data-parallel tasks. We examine three widely-used and important matrix factorizations: NMF (for physical plausability), PCA (for its ubiquity) and CX (for data interpretability). We apply these methods to 1.6TB particle physics, 2.2TB and 16TB climate modeling and 1.1TB bioimaging data. The data matrices are tall-and-skinny, which enable the algorithms to map conveniently into Spark's data-parallel model. We perform scaling experiments on up to 1600 Cray XC40 nodes, describe the sources of slowdowns, and provide tuning guidance to obtain high performance.

Avoiding Communication in Primal and Dual Block Coordinate Descent Methods
A. Devarakonda, K. Fountoulakis, J. Demmel, M. Mahoney
Arxiv:1612.04003v2, May 2, 2017
Abstract:
Primal and dual block coordinate descent methods are iterative methods for solving regularized and unregularized optimization problems. Distributed-memory parallel implementations of these methods have become popular in analyzing large machine learning datasets. However, existing implementations communicate at every iteration, which on modern data center and super-computing architectures, often dominates the cost of floating-point computation. Recent results on communication-avoiding Krylov subspace methods suggest that large speedups are possible by reorganizing iterative algorithms to avoid communication. We show how applying similar algorithmic transformations can lead to primal and dual block coordinate descent methods that only communicate every s iterations–where s is a tuning parameter–instead of every iteration for the regularized least-squares problem. We show that the communication-avoiding variants reduce the number of synchronizations by a factor of s on distributed-memory parallel machines without altering the convergence rate and attains strong scaling speedups of up to 6.1× on a Cray XC30 supercomputer.

Exploiting Multiple Levels of Parallelism in Sparse Matrix-Matrix Multiplication
Azad, G. Ballard, A. Buluc, J. Demmel, L. Grigori, O. Schwartz, S. Toledo, S. Williams
SIAM J. Sci. Comp, 38(6), C624-C651, 2016
Abstract: Sparse matrix-matrix multiplication (or SpGEMM) is a key primitive for many high-performance graph algorithms as well as for some linear solvers, such as algebraic multigrid. The scaling of existing parallel implementations of SpGEMM is heavily bound by communication. Even though 3D (or 2.5D) algorithms have been proposed and theoretically analyzed in the flat MPI model on Erdös--Rényi matrices, those algorithms had not been implemented in practice and their complexities had not been analyzed for the general case. In this work, we present the first implementation of the 3D SpGEMM formulation that exploits multiple (intranode and internode) levels of parallelism, achieving significant speedups over the state-of-the-art publicly available codes at all levels of concurrencies. We extensively evaluate our implementation and identify bottlenecks that should be subject to further research.

Design and Implementation of a Communication-Optimal Classifier for Distributed Kernel Support Vector Machines
Y. You, J. Demmel, K. Czechowski, L. Song, R. Vuduc
IEEE Trans. On Parallel and Distributed Systems, 2016
(**Best Paper Award** for version appearing in IEEE Intern. Parallel and Distr. Processing Symp., 2015)
Abstract
We consider the problem of how to design and implement communication-efficient versions of parallel kernel support vector machines, a widely used classifier in statistical machine learning, for distributed memory clusters and supercomputers. The main computational bottleneck is the training phase, in which a statistical model is built from an input data set. Prior to our study, the parallel isoefficiency of a state-of-the-art implementation scaled as W = Omega(P^3), where W is the problem size and P the number of processors; this scaling is worse than even a one-dimensional block row dense matrix vector multiplication, which has W = Omega(P^2). This study considers a series of algorithmic refinements, leading ultimately to a Communication-Avoiding SVM method that improves the isoefficiency to nearly W = Omega(P). We evaluate these methods on 96 to 1536 processors, and show average speedups of 3 – 16x (7x on average) over Dis-SMO, and a 95% weak-scaling efficiency on six real-world datasets, with only modest losses in overall classification accuracy. The source code can be downloaded freely.

Asynchronous Parallel Greedy Coordinate Descent
Y. You, X. Lian, J. Liu, H.-F. Yu, I. Dhillon, J. Demmel, C.-J. Hsieh
Conference on Neural Information Processing Systems, 2016
Abstract:
In this paper, we propose and study an Asynchronous parallel Greedy Coordinate Descent (Asy-GCD) algorithm for minimizing a smooth function with bounded constraints. At each iteration, workers asynchronously conduct greedy coordinate descent updates on a block of variables. In the first part of the paper, we analyze the theoretical behavior of Asy-GCD and prove a linear convergence rate. In the second part, we develop an efficient kernel SVM solver based on Asy-GCD in the shared memory multi-core setting. Since our algorithm is fully asynchronous—each core does not need to idle and wait for the other cores—the resulting algorithm enjoys good speedup and outperforms existing multi-core kernel SVM solvers including asynchronous stochastic coordinate descent and multi-core LIBSVM.

Parallelpipeds obtaining HBL lower bounds
J. Demmel, A. Rusciano
https://arxiv.org/abs/1611.05944, 18 Nov 2016
Abstract:
This work studies the application of the discrete Holder-Brascamp-Lieb (HBL) inequalities to the design of communication optimal algorithms. In particular, it describes optimal tiling (blocking) strategies for nested loops that lack data dependencies and exhibit linear memory access patterns. We attain known lower bounds for communication costs by unraveling the relationship between the HBL linear program, its dual, and tile selection. The methods used are constructive and algorithmic. The case when all arrays have one index is explored in depth, as a useful example in which a particularly efficient tiling can be determined.

Scaling Deep Leaning on GPU and Knights Landing Clusters
Y. You, A. Buluc, J. Demmel
Supercomputing 17,
Abstract:
Training neural networks has become a big bottleneck. For example, training ImageNet dataset on one Nvidia K20 GPU needs 21 days. To speed up the training process, the current deep learning systems heavily rely on the hardware accelerators. However, these accelerators have limited on-chip memory compared with CPUs. We use both self-host Intel Knights Landing (KNL) clusters and multi-GPU clusters as our target platforms. From the algorithm aspect, we focus on Elastic Averaging SGD (EASGD) to design algorithms for HPC clusters. We redesign four efficient algorithms for HPC systems to improve EASGD’s poor scaling on clusters. Async EASGD, Async MEASGD, and Hogwild EASGD are faster than existing counterpart methods (Async SGD, Async MSGD, and Hogwild SGD) in all comparisons. Sync EASGD achieves 5.3X speedup over

original EASGD on the same platform. We achieve 91.5% weak scaling efficiency on 4253 KNL cores, which is higher than the state-of-the-art implementation.

ImageNet Training in Minutes
Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, K. Keutzer
arXiv:1709.05011, Nov 2017
Abstract:
Since its creation, the ImageNet-1k benchmark set has played a significant role as a benchmark for ascertaining the accuracy of different deep neural net (DNN) models on the classification problem. Moreover, in recent years it has also served as the principal benchmark for assessing different approaches to DNN training. Finishing a 90-epoch ImageNet-1k training with ResNet-50 on a NVIDIA M40 GPU takes 14 days. This training requires 10^18 single precision operations in total. On the other hand, the world's current fastest supercomputer can finish 2 x 10^17 single precision operations per second. If we can make full use of the computing capability of the fastest supercomputer for DNN training, we should be able to finish the 90-epoch ResNet-50 training in five seconds. Over the last two years, a number of researchers have focused on closing this significant performance gap through scaling DNN training to larger numbers of processors. Most successful approaches to scaling ImageNet training have used the synchronous stochastic gradient descent. However, to scale synchronous stochastic gradient descent one must also increase the batch size used in each iteration.
Thus, for many researchers, the focus on scaling DNN training has translated into a focus on developing training algorithms that enable increasing the batch size in data-parallel synchronous stochastic gradient descent without losing accuracy over a fixed number of epochs. As a result, we have seen the batch size and number of processors successfully utilized increase from 1K batch size on 128 processors to 8K batch size on 256 processors over the last two years. The recently published LARS algorithm increased batch size further to 32K for some DNN models. Following up on this work, we wished to confirm that LARS could be used to further scale the number of processors efficiently used in DNN training and, and as a result, further reduce the total training time. In this paper we present the results of this investigation: using LARS we efficiently utilized 1024 CPUs to finish the 100-epoch ImageNet training with AlexNet in 11 minutes with 58.6% accuracy (batch size = 32K), and we utilized 2048 KNLs to finish the 90-epoch ImageNet training with ResNet-50 in 20 minutes without losing accuracy (batch size = 32K). State-of-the-art ImageNet training speed with ResNet-50 is 74.9% top-1 test accuracy in 15 minutes (Akiba, Suzuki, and Fukuda 2017). We got 74.9% top-1 test accuracy in 64 epochs, which only needs 14 minutes. Furthermore, when the batch size is above 16K, our accuracy using LARS is much higher than Facebooks corresponding batch sizes (Figure 1). Our code is available upon request.

Communication-Avoiding Parallel Sparse-Dense Matrix-Matrix Multiplication
P. Koanantokool, A. Azad, A. Buluc, D. Morozov, S.-Y. Oh, L. Oliker, K. Yelick
Proc. 30th IEEE Intern. Parallel & Distr. Proc. Symp, May 2016
Abstract:
Multiplication of a sparse matrix with a dense matrix is a building block of an increasing number of applications in many areas such as machine learning and graph algorithms. However, most previous work on parallel matrix multiplication considered only both dense or both sparse matrix operands. This paper analyzes the communication lower bounds and compares the communication costs of various classic parallel algorithms in the context of sparse-dense matrix-matrix multiplication. We also present new communication-avoiding algorithms based on a 1D decomposition, called 1.5D, which — while suboptimal in dense-dense and sparse-sparse cases — outperform the 2D and 3D variants both theoretically and in practice for sparse- dense multiplication. Our analysis separates one-time costs from per iteration costs in an iterative machine learning context. Experiments demonstrate speedups up to 100x over a baseline 3D SUMMA implementation and show parallel scaling over 10 thousand cores.

Write-Avoiding Algorithms
E. Carson, J. Demmel, L. Grigori, N. Knight, P. Koanantakool, O. Schwartz, H. Simhadri
Proc. 30th IEEE Intern. Parallel & Distr. Proc. Symp, May 2016
Abstract:
Communication, i.e., moving data between levels of a memory hierarchy or between processors over a network, is much more expensive (in time or energy) than arithmetic. There has thus been a recent focus on designing algorithms that minimize communication and, when possible, attain lower bounds on the total number of reads and writes. However, most previous work does not distinguish between the costs of reads and writes. Writes can be much more expensive than reads in some current and emerging storage devices such as nonvolatile memories.
This motivates us to ask whether there are lower bounds on the number of writes that certain algorithms must perform, and whether these bounds are asymptotically smaller than bounds on the sum of reads and writes together. When these smaller lower bounds exist, we then ask when they are attainable; we call such algorithms “write-avoiding” (WA), to distinguish them from “communication-avoiding” (CA) algorithms, which only minimize the sum of reads and writes. We identify a number of cases in linear algebra and direct N-body methods where known CA algorithms are also WA (some are and some aren’t). We also identify classes of algorithms, including Strassen’s matrix multiplication, Cooley-Tukey FFT, and cache oblivious algorithms for classical linear algebra,

where a WA algorithm cannot exist: the number of writes is unavoidably within a constant factor of the total number of reads and writes. We explore the interaction of WA algorithms with cache replacement policies and argue that the Least Recently Used policy works well with the WA algorithms in this paper. We provide empirical hardware counter measurements from Intel's Nehalem-EX microarchitecture to validate our theory. In the parallel case, for classical linear algebra, we show that it is impossible to attain lower bounds both on interprocessor communication and on writes to local memory, but either one is attainable by itself. Finally, we discuss WA algorithms for sparse iterative linear algebra.

Communication-Avoiding Symmetric Definite Factorization
G. Ballard, D. Becker, J. Demmel, J. Dongarra, A. Druinsky, I. Peled, O. Schwartz, S. Toledo, I. Yamazaki
SIAM J. Matrix. Anal. Appl., V. 35, N. 4, 1364-1406, 2014
Abstract.
We describe and analyze a novel symmetric triangular factorization algorithm. The algorithm is essentially a block version of Aasen's triangular tridiagonalization. It factors a dense symmetric matrix A as the product A =P*L^T*L*P^T where P is a permutation matrix, L is lower triangular, and T is block tridiagonal and banded. The algorithm is the first symmetric indefinite communication-avoiding factorization: it performs an asymptotically optimal amount of communication in a two-level memory hierarchy for almost any cache-line size. Adaptations of the algorithm to parallel computers are likely to be communication efficient as well; one such adaptation has been recently published. The current paper describes the algorithm, proves that it is numerically stable, and proves that it is communication optimal.

Exploiting Data Sparsity in Parallel Matrix Powers Computations
N. Knight, E. Carson, J. Demmel,
Intern. Conf. Parallel Processing and Applied Math,
Lecture Notes in Computer Science, v. 8384, May 2014
Abstract:
We derive a new parallel communication-avoiding matrix powers algorithm for matrices of the form A=D+USV^H, where D is sparse and USV^H has low rank and is possibly dense. We demonstrate that, with respect to the cost of computing k sparse matrix-vector multiplications, our algorithm asymptotically reduces the parallel latency by a factor of O(k) for small additional bandwidth and computation costs. Using problems from real-world applications, our performance model predicts up to 13× speedups on petascale machines.

Contracting Symmetric Tensors using Fewer Multiplications
E. Solomonik, J. Demmel
ETH Technical Report, 2016
https://doi.org/10.3929/ethz-a-010345741
Abstract:
We present more computationally-efficient algorithms for contracting symmetric tensors. Tensor contractions are reducible to matrix multiplication, but permutational symmetries of the data, which are expressed by the tensor representation, provide an opportunity for more efficient algorithms. Previously known methods have exploited only tensor symmetries that yield identical computations that are directly evident in the contraction expression. We present a new 'symmetry preserving' algorithm that uses an algebraic reorganization in order to exploit considerably more symmetry in the computation of the contraction than the conventional approach. The new algorithm requires fewer multiplications but more additions per multiplication than previous approaches. The applications of this result include the capability to multiply a symmetric matrix by a vector, as well as compute the rank-2 symmetric vector outer product in half the number of scalar multiplications, albeit with more additions.
The symmetry preserving algorithm can also be adapted to perform the complex versions of these operations, namely the product of a Hermitian matrix and a vector and the rank-2 Hermitian vector outer product, in 3/4 of the overall operations. Consequently, the number of operations needed for the direct algorithm to compute the eigenvalues of a Hermitian matrix is reduced by the same factor.
Our symmetry preserving tensor contraction algorithm can also be adapted to the antisymmetric case and is therefore applicable to the tensor-contraction computations employed in quantum chemistry. For these applications, notably the coupled-cluster method, our algorithm yields the highest potential speed-ups, since in many higher-order contractions the reduction in the number of multiplications achieved by our algorithm enables an equivalent reduction in overall contraction cost. We highlight that for three typical coupled-cluster contractions taken from methods of three different orders, our algorithm achieves 2X, 4X, and 9X improvements in arithmetic cost over the standard approach.

Communication Lower Bounds for Tensor Contraction Algorithms
E. Solomonik, J. Demmel, T. Hoefler
ETH Technical Report
https://doi.org/10.3929/ethz-a-010350411
Abstract:
Contractions of nonsymmetric tensors are reducible to matrix multiplication, however, 'fully symmetric contractions' in which the tensors are symmetric and the result is symmetrized can be done with fewer operations. The 'direct evaluation algorithm' for fully symmetric contractions exploits equivalence between terms in the contraction equation to obtain a lower computation cost than the cost associated with nonsymmetric contractions. The 'symmetry

preserving algorithm' lowers the cost even further via an algebraic reorganization of the contraction equation. We derive vertical (between memory and cache) and horizontal (interprocessor) communication lower bounds for both of these algorithms. We demonstrate that any load balanced parallel schedule of the direct evaluation algorithm requires asymptotically more horizontal communication for some fully symmetric contractions than matrix multiplication for nonsymmetric contractions of the same size. Instances of such fully symmetric contractions arise in quantum chemistry calculations. Further, we prove that any schedule of the symmetry preserving algorithm requires asymptotically more vertical and horizontal communication than the direct evaluation algorithm for some fully symmetric contractions. However, for the instances of fully symmetric contractions that arise in quantum chemistry calculations, our lower bounds are asymptotically the same for both of these algorithms.

Avoiding Communication in Successive Band Reduction
G. Ballard, J. Demmel, N. Knight
ACM Trans. Parallel Computing, v. 1, i. 2, Jan 2015
Abstract:
The running time of an algorithm depends on both arithmetic and communication (i.e., data movement) costs, and the relative costs of communication are growing over time. In this work, we present sequential and distributed-memory parallel algorithms for tridiagonalizing full symmetric and symmetric band matrices that asymptotically reduce communication compared to previous approaches.
The tridiagonalization of a symmetric band matrix is a key kernel in solving the symmetric eigenvalue problem for both full and band matrices. In order to preserve structure, tridiagonalization routines use annihilate-and-chase procedures that previously have suffered from poor data locality and high parallel latency cost. We improve both by reorganizing the computation and obtain asymptotic improvements. We also propose new algorithms for reducing a full symmetric matrix to band form in a communication-efficient manner. In this article, we consider the cases of computing eigenvalues only and of computing eigenvalues and all eigenvectors.

Avoiding communication in the Lanczos bidiagonalizaiton routine and associated Least Squares QR solvers
E. Carson
UC Berkeley EECS Technical Report UCB/EECS-2015-15, Apr 12, 2015
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-15.html
Abstract:
Communication – the movement of data between levels of memory hierarchy or between processors over a network – is the most expensive operation in terms of both time and energy at all scales of computing. Achieving scalable performance in terms of time and energy thus requires a dramatic shift in the field of algorithmic design. Solvers for sparse linear algebra problems, ubiquitous

throughout scientific codes, are often the bottlenecks in application performance due to a low computation/communication ratio. In this paper we develop three potential implementations of communication-avoiding Lanczos bidiagonalization algorithms and discuss their different computational requirements. Based on these new algorithms, we also show how to obtain a communication-avoiding LSQR least squares solver.

Communication Avoiding Rank Revealing QR Factorization with Column Pivoting
J. Demmel, L. Grigori, M. Gu, H Xiang
SIAM J. Matrix Anal. Appl., V. 36, N. 1, pp 55-89
Abstract:
In this paper we introduce CARRQR, a communication avoiding rank revealing QR factorization with tournament pivoting. We show that CARRQR reveals the numerical rank of a matrix in an analogous way to QR factorization with column pivoting (QRCP). Although the upper bound of a quantity involved in the characterization of a rank revealing factorization is worse for CARRQR than for QRCP, our numerical experiments on a set of challenging matrices show that this upper bound is very pessimistic, and CARRQR is an effective tool in revealing the rank in practical problems. Our main motivation for introducing CARRQR is that it minimizes data transfer, modulo polylogarithmic factors, on both sequential and parallel machines, while previous factorizations as QRCP are communication suboptimal and require asymptotically more communication than CARRQR. Hence CARRQR is expected to have a better performance on current and future computers, where communication is a major bottleneck that highly impacts the performance of an algorithm.

Reconstructing Householder vectors from Tall-Skinny QR
G. Ballard, J. Demmel, L. Grigori, M. Jacquelin, N. Knight, H. D. Nguyen
J. Parallel and Distr. Computing, V. 85, Nov 2015
Abstract:
The Tall-Skinny QR (TSQR) algorithm is more communication efficient than the standard Householder algorithm for QR decomposition of matrices with many more rows than columns. However, TSQR produces a different representation of the orthogonal factor and therefore requires more software development to support the new representation. Further, implicitly applying the orthogonal factor to the trailing matrix in the context of factoring a square matrix is more complicated and costly than with the Householder representation.
We show how to perform TSQR and then reconstruct the Householder vector representation with the same asymptotic communication efficiency and little extra computational cost. We demonstrate the high performance and numerical stability of this algorithm both theoretically and empirically. The new Householder reconstruction algorithm allows us to design more efficient parallel QR algorithms, with significantly lower latency cost compared to Householder QR and lower bandwidth and latency costs compared with Communication-Avoiding

QR (CAQR) algorithm. Experiments on supercomputers demonstrate the benefits of the communication cost improvements: in particular, our experiments show substantial improvements over tuned library implementations for tall-and-skinny matrices. We also provide algorithmic improvements to the Householder QR and CAQR algorithms, and we investigate several alternatives to the Householder reconstruction algorithm that sacrifice guarantees on numerical stability in some cases in order to obtain higher performance.