Published in final edited form as:
IEEE Nucl Sci Symp Conf Rec (1997). 2014 November ; 2014: .

Handling Big Data in Medical Imaging: Iterative Reconstruction with Large-Scale Automated Parallel Computation

Jae H. Lee, Ph.D. [student],
University of North Carolina, Chapel Hill, NC 27599 USA

Yushu Yao,
National Energy Research Scientific Computing (NERSC) Center, Berkeley, CA 94704 USA

Uttam Shrestha,
Physics Research Laboratory, Department of Radiology and Biomedical Imaging, University of California, San Francisco, CA 94143 USA

Grant T. Gullberg [Fellow IEEE], and
Structural Biology and Imaging Department, Life Sciences Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94704 USA

Youngho Seo [Senior member IEEE]
Physics Research Laboratory, Department of Radiology and Biomedical Imaging, University of California, San Francisco, CA 94143 USA

Abstract

The primary goal of this project is to implement the iterative statistical image reconstruction algorithm, in this case maximum likelihood expectation maximum (MLEM) used for dynamic cardiac single photon emission computed tomography, on Spark/GraphX. This involves porting the algorithm to run on large-scale parallel computing systems. Spark is an easy-to- program software platform that can handle large amounts of data in parallel. GraphX is a graph analytic system running on top of Spark to handle graph and sparse linear algebra operations in parallel. The main advantage of implementing MLEM algorithm in Spark/GraphX is that it allows users to parallelize such computation without any expertise in parallel computing or prior knowledge in computer science. In this paper we demonstrate a successful implementation of MLEM in Spark/GraphX and present the performance gains with the goal to eventually make it useable in clinical setting.

I. Introduction

CURRENTLY Big Data refers to datasets that are so large and complex that it is too difficult to store, manage, analyze or visualize within commonly available computational architecture. For example, data produced by sequencing, mapping, and analyzing genomes may fall into this category. Similarly, processing and analyzing large volumes of *medical*

telephone: 818-590-7823, jaeholee@live.unc.edu. telephone: 510-486-4690, yyao@lbl.gov. uttam.shrestha@ucsf.edu. gtgullberg@lbl.gov. youngho.seo@ucsf.edu.

imaging data may challenge expeditious diagnosis. In biomedical image processing using transmission or emission tomography, a significant amount of computational time is required in order to reconstruct a diagnostic quality image. In myocardial imaging using radiolabeled tracers as in positron emission tomography (PET) or single photon emitted computed tomography (SPECT), patient motion and cardiac motion due to cardiac beating and respiration create unwanted artifacts in the reconstructed image. Solutions such as cardiac and respiratory gating, dynamic acquisition techniques, list-mode data acquisition, and reconstruction in higher dimensions have been proposed and show significant improvements over methods that ignore these types of motion. However, these techniques demand unprecedented computational time [3].

In this work, we implemented the standard MLEM algorithm for SPECT myocardial perfusion imaging in a largescale parallel computing software system at the National Energy Research Scientific Computing (NERSC) Center. This is a high performance supercomputing facility at Lawrence Berkeley Laboratory (LBNL), for the Department of Energy.

II. Method: Data Acquisition

The simulated SPECT dynamic myocardial perfusion imaging involved the camera head continuously rotating around a cardiac torso consisted of 128^3 voxels. In conventional 3D imaging (ignoring any motion), as shown in Fig. 3 projected images of 128×128 are acquired in 360 different views (or angles) over 360°.

The system matrix of the projections was represented by a large sparse matrix with dimension $128^2 \times 360$ by 128^3 that is used to perform the 3D spatial reconstruction. 4D reconstruction includes the reconstruction of spatial and temporal changes of the radiotracer in the organ tissues as a function of time. 5D reconstruction includes the additional reconstruction of cardiac deformation over the cardiac cycle and 6D includes the reconstruction of the additional organ motion such as the heart due to respiration. Due to these additional dimensions of motion throughout time, it takes a single core machine about 13 days for the full 6D reconstruction, thus the need for parallel computing and better computing power. In this work, we not only parallelize the reconstruction process, but also use an easier alternative to the Message Passage Interface (MPI) Standard, thus not requiring users to have expertise in computer science.

III. Method: Spark/GraphX

GraphX that runs on Spark was chosen as the large-scale parallel software system to be used in this simulation. Spark and GraphX were both developed by the UC Berkeley AMPLab and use Resilient Distributed Datasets (RDDs) as a distributed memory abstraction instead of Distributed Shared Memory (DSM). RDD is a fault-tolerant distributed inmemory abstraction, and that allows intermediate results to be reused more efficiently by allowing users to have control over storing intermediate results in memory to optimize data placement. Spark first “transforms” data into RDDs (e.g., using a *map* or *filter*) and uses “actions” (e.g., *count*, *collect*, and *save*) on RDDs. This model accelerates the computation speed because once any slow nodes are detected RDDs can create backup copies of slow tasks, which do not have to access the same memory as they do with DSM [6]. With such abstraction, Spark distributes the data over different working nodes that run computations in a parallel fashion by communicating with each other with intermediate results and combining them to get the final result.

GraphX is a large-scale graph-parallel system built on top of Spark. Graph is a representation of a set of objects, called *vertices*, where some pairs of objects are connected by links, called *edges* [4]. There is natural relation between a graph and a matrix, such as adjacency matrix of a graph. An adjacency matrix is a representation of which vertices are adjacent to which vertices to each by assigning the (i, j) entry of the adjacency matrix a value 1 if the vertices i and j are linked by and edge or a value 0 otherwise as shown in Fig. 4 [2].

GraphX also uses RDDs as a distributed memory abstraction, where vertices, edges, and graphs are transformed into RDDs. GraphX adopted and improved previously existing graph-parallel abstractions such as Pregel (which was modified in this project and made to compute matrix multiplications) using RDD abstraction [5].

There are different ways to parallelize computation, such as using MPI, but the biggest advantage of using Spark/GraphX is its usability, as only about 35 lines of code were needed to reproduce the MLEM algorithm. Due to the usability of Spark/GraphX, users with no expertise in computer science can easily implement different methods or algorithms that need to be parallelized.

In order to implement the iterative MLEM algorithm in GraphX, the reconstruction problem is represented as a sparse matrix equation: $Af = g$. Then the forward projection operator A, also called the system matrix, can be further represented in terms of graphs in GraphX, as there is a natural relation between matrices and graphs. In this case the system matrix A is a large sparse matrix, and it can be represented as a bipartite graph, where f and g are the two vertex sets and the values of A at corresponding indices are the edges (Fig. 5). A sparse matrix specifically has an advantage in using matrix operations represented in terms of graph-parallel operations because the zeroes of the matrix means that there is no edge between the vertices defined by the index of the matrix. Because a sparse matrix does not have these edges, the number of graph computations decreases with respect to the sparsity of the matrix.

Then sparse matrix operations are represented in terms of graph-parallel operations described in [5]. For example, each element of g can be represented as the weighted (by the corresponding edge value) sum of the value of each of its neighbor:

$$g_j = \Sigma_{f_i \in \{neighbors \quad of \quad g_j\}} A_{i,j} \cdot f_i \quad (1)$$

Spark/GraphX was used to test the computation time for reconstructing the 3D reconstruction problem. The test was performed by reconstructing a 128×128×128 spatially distributed image from 360 simulated 128×128 projections of the MCAT phantom acquired over 360°. The system matrix A for this problem was a large sparse matrix with a dimension of $128^2 \times 360 \times 128^3$. A program of 35 lines of code was implemented to determine the time per iteration for running on 4, 8, and 16 nodes of the computer Carver at NERSC Center. Carver has a CPU rate of 2.0 GHz. The results using Carver were compared with running the same problem on a desktop computer with a CPU rate of 2.8 GHz.

As shown in Fig. 6, Spark/GraphX improved the iteration time for the 3D reconstruction problem from 1 minute to 35 seconds by going from single processor desk-top computer to Spark/GraphX. From this result, we can expect the total reconstruction time for the 6D case to be reduced from about 13 days to approximately 8 days by using Spark/GraphX with just 16 nodes. Also, Spark/GraphX can be scaled depending on the need and the amount of data by simply changing the number of nodes and the amount of memory (Fig. 7). Whether the scalability is linear or not needs to be verified in further studies. This means that it is necessary to verify if the iteration time will decrease linearly respect to the increase in the number of nodes. Once it is proven that the reconstruction process can be linearly scaled, we can expect to reduce the total reconstruction time for the 6D reconstruction problem to the order of seconds using the computing power of a supercomputer.

V. Discussion

The results shown in this paper are still preliminary and require further improvement. Finding the optimal partition number and having a better understanding of the performance of Spark can better delineate the applicability of Spark/GraphX, and validation of the method with a model of real-data is also necessary. Spark is an easy-to-use framework to perform iterative algorithms in memory. UC Berkeley AMPLab is in the process of further developing the linear algebra library that will work more effectively for such iterative methods involving large matrix computations.

Carver, a liquid-cooled IBM iDataPlex system with 1202 compute nodes (9,984 cores), was chosen as the computational systems to test the experiments for this project. At the present the NERSC Center is pushing forward to support Spark/GraphX on Edison, which is the NERSC Center's newest supercomputer. Edison is a Cray XC30, with a peak performance of 2.57 petaflops per second, 133,824 compute cords, 257 terabytes of memory, and 7.56 petabytes of disk.

Big data analytics and parallel computing have been proven to bring advancements to the field of scientific research including medical imaging. Current methods of solving problems on supercomputers are very complicated and limited to people with expertise in computer science. However, with the usability of Spark/GraphX and the computing power of supercomputers at the NERSC Center, many challenging Big Data problems can be solved. A number of other science projects are already exploring Spark on the Edison supercomputer.

Acknowledgment

This work was supported in part by the U.S. Department of Energy, under the Science Undergraduate Laboratory Internship (SULI) program. We would like to thank Rostyslav Boutchko (LBNL), for giving valuable suggestion, and to Joey Gonzalez and Ankur Dave from UC Berkeley EECS/AMPLab for helping with GraphX. This research used 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-AC02-05-CH11231.

This work is supported in part by National Heart, Lung, and Blood Institute under grant #R01 HL050663 and by National Institute of Biomedical Imaging and Bioengineering under grant #R01 EB012965.

References

[1]. Bruyant PP. Analytic and iterative reconstruction algorithms in SPECT. J. Nucl. Med. 2002; 43:1343–1358. [PubMed: 12368373]

[2]. Godsil, C.; Royle, G. Algebraic Graph Theory. Springer; 2001.

[3]. Shepp LA, Vardi Y. Maximum likelihood reconstruction for emission tomography. IEEE Trans. Med. Imag. 1982; 1:113–122.

[4]. Trudeau, RJ. Introduction to Graph Theory. Dover Pub; New York: 1993. p. 19Corrected, enlarged republication. ed.ISBN 978-0-486-67870-2. Retrieved 8 August 2012. "A graph is an object consisting of two sets called its vertex set and its edge set

[5]. Xin, RS.; Crankshaw, D.; Dave, A.; Gonzalez, JE.; Franklin, MJ.; Stoica, I. GraphX: Unifying data-parallel and graph-parallel analytics. 2014. arXiv preprint arXiv:1402.2394

[6]. Zaharia, M.; Chowdhury, M.; Das, T.; Dave, A.; Ma, J.; McCauley, M.; Franklin, M.; Shenker, S.; Stoca, I. Electrical Engineering and Computer Sciences, University of California; Jul 19. 2011 Resilient Distributed Datasets: A Fault- Tolerant Abstraction for In-Memory Cluster Computing. Technical Report No. UCB/EECS-2011-82

[7]. Shrestha U, Seo Y, Botvinick EH, Gullberg GT. Image reconstruction in higher dimension: Myocardial perfusion imaging of tracer dynamics, cardiac and respiratory motion. IEEE Trans Med Imag. Submitted to.

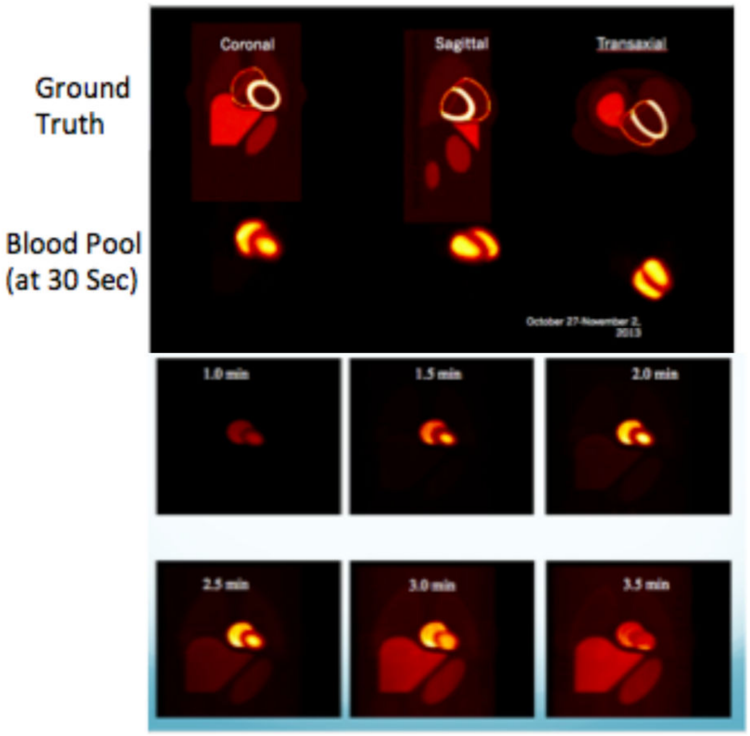


Fig. 1.

Example of the reconstruction at UCSF of simulated MCAT phantom dynamic data. (Top Row) The original anatomical phantom. (Second Row) The coronal view of the reconstruction of dynamic data 30 sec after the injection of ^{99m}Tc-tetrofosmin. The reconstruction of the gated cardiac data was performed using a 5D spatiotemporal reconstruction algorithm described in [7]. In this simulation, 6 B-spline basis functions were used for the temporal changes in the tracer concentration and 8 Gaussian basis functions were used for the cardiac deformation due to the heart beating. The number of unknowns estimated were 128×128×128×6×8 for the torso volume of dimensions 128×128×128. There were 120 views per camera rotation and 8 cardiac gates and the period of rotation was 15 sec corresponding to 15 cardiac cycles (15 views for each cardiac gate). (Third Row) The coronal view of the reconstruction of the dynamic data 1 min, 1.5 min, and 2.0 min after injection. (Bottom Row) Same as previous 2.5 min, 3.0 min, and 3.5 min after injection of ^{99m}Tc-tetrofosmin.

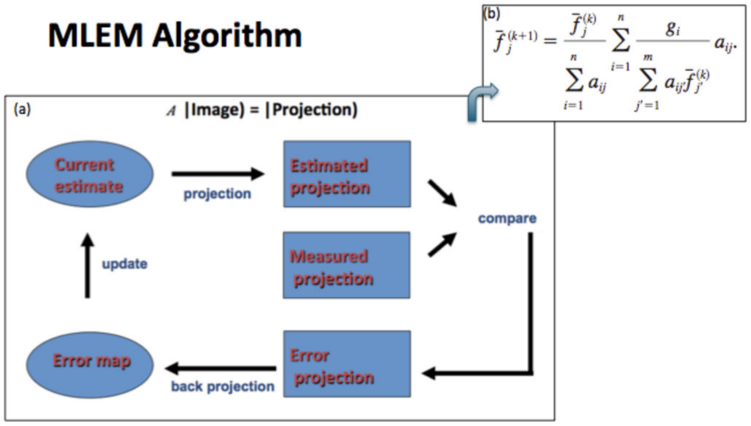

MLEM Algorithm
(a)
A |Image) = |Projection)
Current estimate
projection
Estimated projection
compare
Measured projection
update
Error map
back projection
Error projection
(b)
$\bar{f}_j^{(k+1)} = \frac{\bar{f}_j^{(k)}}{\sum_{i=1}^{n} a_{ij}} \sum_{i=1}^{n} \frac{g_i}{\sum_{j'=1}^{m} a_{ij'} \bar{f}_{j'}^{(k)}} a_{ij}.$

Fig. 2.

(a) Simple schematic diagram of the iterative MLEM algorithm (b) The update formula for the iterative MLEM algorithm.

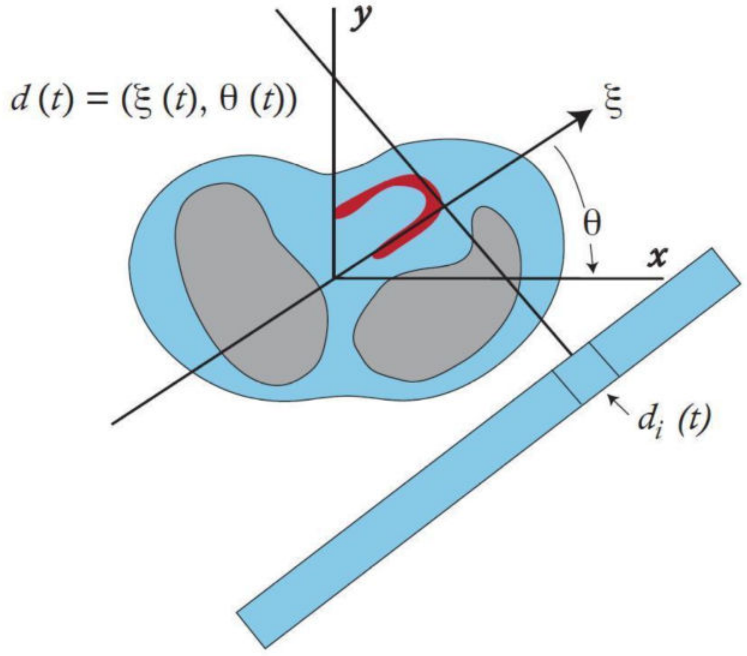


Fig. 3.
Depiction of the tomographic reconstruction process.

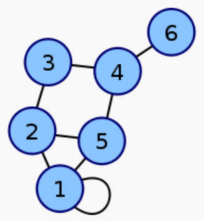
Labeled graph	Adjacency matrix
	$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$ Coordinates are 1-6.

Fig. 4.
The adjacency matrix representing an undirected graph.

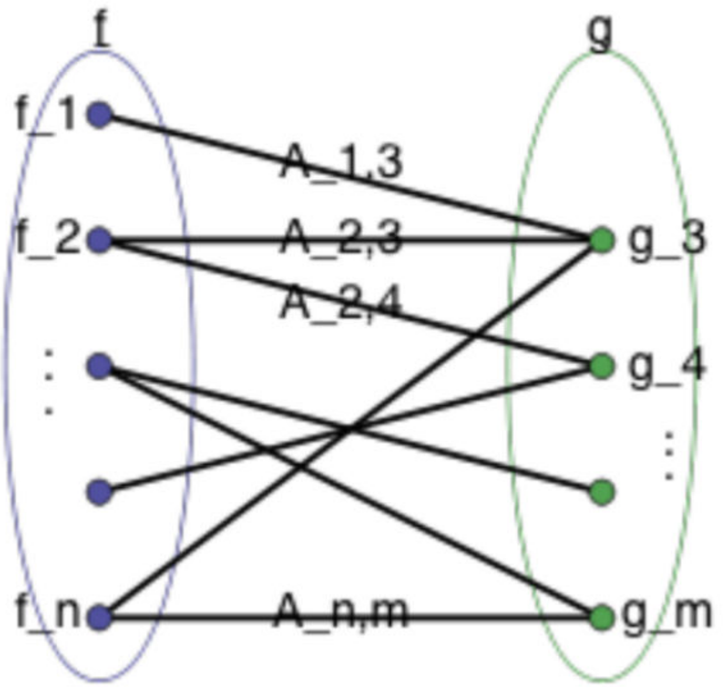


Fig. 5.
The system matrix, A, represented as a bipartite graph.

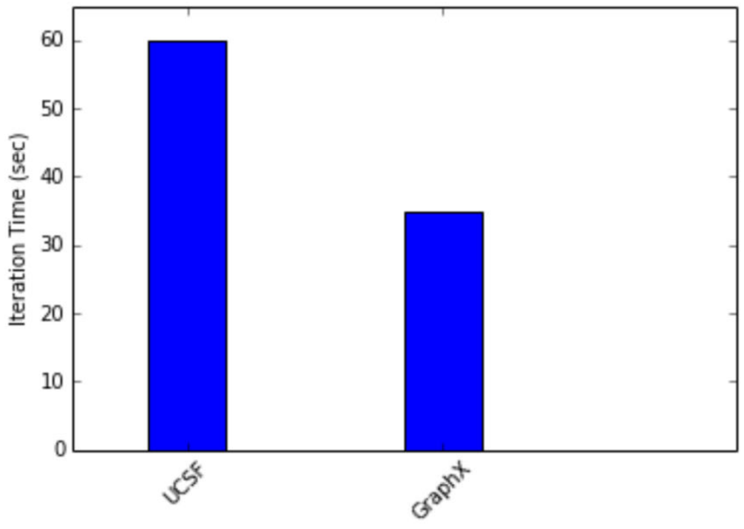


Fig. 6.
Comparing the iteration time of MLEM algorithm for typical 3D reconstruction with GraphX (16 nodes).

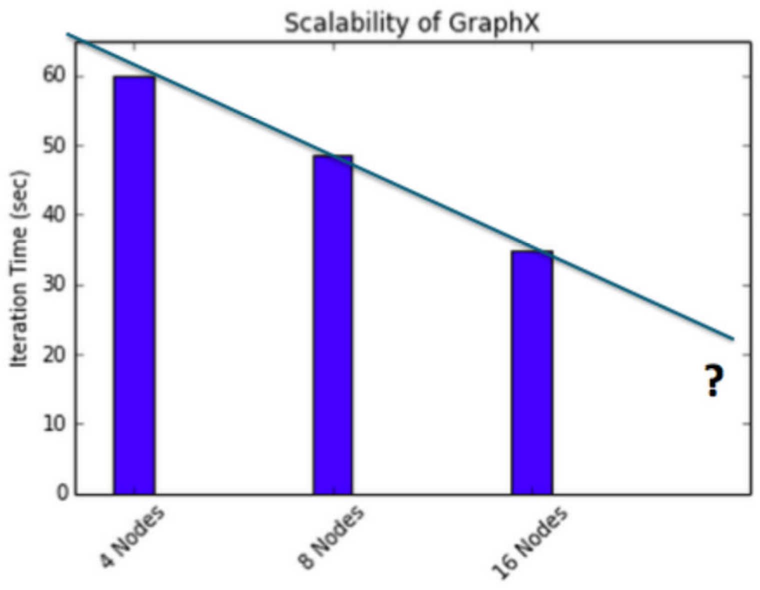


Fig. 7.
Iteration time of the MLEM algorithm with GraphX for different number of nodes.