Systems for High Performance Execution of Important Machine Learning Algorithms

The Big Data Analytics world has two big challenges: what kinds of analysis on what kind of data generates useful results, and how can we manage the logistics of big data and implement fast scalable frameworks for big data.   In this course we are primarily concerned with the latter questions.

In the current era of competing Big Data Frameworks, there are many frameworks publishing papers on their advantages, but few careful comparative studies. Lots of of open source, a number of available data sets or synthetic data set creation algorithms, and numerous frameworks makes it obvious that good science needs detailed comparisons.

In the first project idea for Machine Learning Systems, the specification is simple: construct a variety of fair comparisons between open source machine learning frameworks and deconstruct (figure out the root causes) of differences in performance.

Example frameworks:

- Mahout, https://mahout.apache.org
- Giraph, https://giraph.apache.org
- Hama, https://hama.apache.org
- GraphLab, PowerGraph, GraphChi, http://graphlab.org
- Spark, MLib, GraphX, https://spark.incubator.apache.org
- Naiad, http://research.microsoft.com/en-us/projects/naiad/
- CMU BigLearning team has an SSP, Petuum (http://petuum.org) and STRADS framework they might share too (Wei Dai**, wdai@cs.cmu.edu).**

Example problems:

- triangle counting
- pagerank
- LDA
- matrix factorization
- sparse regression
- graph coloring
- shortest path

Example hardware: NSF PRObE clusters: Marmot, Kodiak, Susitna, and AWS EC2.

A good instance of this project would use at least three frameworks and at least three problems with at least three different data sets. Doing this well will require becoming sufficiently knowledgeable about tuning and configuring each framework to show it at its best. It will also require instrumenting the frameworks so that you can measure internal parameters (numbers of events, durations of phases etc) in order to understand different behavior.

Done well, the world will be very interested, and some people will feel challenged. So the work needs to be transparent (well documented), repeatable, and unambiguous. If all this is achieved, a very good publication is possible.

It is certainly possible that a full comparison is too much for one person or small group, so we can break this up into multiple more in-depth comparisons. For example, Wei Dai is very interested in someone studying LDA in great detail:

Comparison of LDA on Yahoo!LDA, GraphLab, Google's PLDA, and Petuum (Dai Wei)

Latent Dirichlet Allocation (LDA) is a popular model used to discover latent topics from text documents. There has been much efforts in scaling up the sampling algorithm with better system implementations: Google's PLDA [3] represents one of the earliest implementation; Yahoo!LDA [1][2] is currently the most popular choice for large scale LDA; GraphLab [4] shows respectable performance given that it's a general purposed framework; and Petuum [5][6] which has demonstrated better performance than GraphLab on LDA. However, so far there has not been a comprehensive comparison among the 4 implementations, and some comparisons, like the one done by GraphLab authors [7], are outdated (they compare with the VLDB version of Yahoo!LDA (2010) instead of the WSDM version (2012).

A comprehensive comparison of these frameworks will provide LDA practitioners a good guide. An interesting question is why GraphLab is slower than Yahoo!LDA and Petuum (e.g., maybe GraphLab sends much more messages than XX or GraphLab synchronizes more often than XX.) You may explore how the graph abstraction of GraphLab has limited the optimization, using your experiences from faster frameworks. In other words, how would you change GraphLab's abstraction to allow optimizations that could remedy the problems you observe (eg., too many messages, synchronization). To answer this question you might need to look into how the LDA sampling algorithm is implemented on each framework's abstraction. Overall this project would be a good learning experience for students interested in learning state-of-the-art ML systems and algorithm beyond Map-Reduce.

All the implementations mentioned here are open-source so can be readily executed. While no in-depth knowledge of the sampling algorithm is needed, the student will still benefit from digging into the algorithm a bit, but this is not too necessary. It would also be interesting to see the performance on data set of various scales.

Contact Person: Dai Wei (wdai@cs.cmu.edu), Xun Zheng (xunzheng@cs.cmu.edu)

[1] Smola, Alexander and Narayanamurthy, Shravan. An architecture for parallel topic models, VLDB (2010).
[2] Amr Ahmed and Mohamed Aly and Joseph Gonzalez and Shravan Narayanamurthy

and Alexander J. Smola Scalable inference in latent variable models WSDM, 2012.
[3] Wang, Yi and Bai, Hongjie and Stanton, Matt and Chen, Wen-Yen and Chang, Edward Y., PLDA: Parallel Latent Dirichlet Allocation for Large-Scale Applications (2009)
[4] Yucheng Low and Joseph Gonzalez and Aapo Kyrola and Danny Bickson and Carlos Guestrin and Joseph M. Hellerstein. Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud. PVLDB (2012)
[5] Wei Dai, Jinliang Wei, Xun Zheng, Jin Kyu Kim, Seunghak Lee, Junming Yin, Qirong Ho, Eric P. Xing. Petuum: A Framework for Iterative-Convergent Distributed ML. arXiv: 1312.7651 (2013)
[6] http://petuum.org/
[7] slide 93 of http://www.cs.berkeley.edu/~jegonzal/talks/jegonzal_thesis_defense.pptx