15-740 Course Project: Large-Scale Sparse Matrix-Vector Multiplication

Dong Zhou, Mu Li

October 31, 2012

Group Info:

- Dong Zhou, dongz@cs.cmu.edu
- Mu Li, muli@cs.cmu.edu

Project Web Page: `http://www.cs.cmu.edu/~dongz/15-740`

Project Description: A large number of machine learning algorithms uses gradient descent like numerical method to obtain the optimal solution, such as page rank, linear classification, collaborative filtering. At its core, matrix vector multiplication is the essential computation. In the era of big data, the data matrix is usually of sheer volume and extremely sparse. Distributed computation framework, such as `MapReduce`, has been used to solve this problem. However, we need to pay for the extra costs of transferring data among machines. Since machine learning applications are usually data intensive, the huge traffic between machines occupies a large portion of running time. For example, if we can run a machine learning application on a single machine within 10 hours, then 100% of running time is local computation. If running it on 10 machines, we may reduce the local computation time to 1 hour, but another 1 hour may be needed for data transmission. When running on 100 machines, on the ideal case we finish local computation within 6 minutes but may spend extra 20 minutes on data transmission. Usually we measure the running cost by number of machine multiplies running time per machine, which is about the local computation time on a single machine plus the extra data transmission time. Definitely we want to use as less machines as possible to reduce the cost of data transmission.

In the project, we focus on optimization the local computation. That is, how to performance large scale sparse matrix vector multiplication on

the single machine efficiently to allow an application use fewer machines. We want to handle the case that the sparse matrix exceeds the capacity of the memory, also we would like to explore cache friendly, NUMA-aware implementation.

Our goal is:

75% Implement a multi-threaded, cache-friendly sparse matrix vector multiplication package.

100% Complete NUMA-aware implementation and support sparse matrix exceeding memory capacity.

125% Highly optimized version to maximize the utilization of resources: memory bandwidth, CPU cycles, etc.

Logistics

Plan and Schedule Our week-by-week schedule is:

- 10.28-11.3 Read and discuss related papers.
- 11.4-11.10 Implement multi-threaded, cache friendly version.
- 11.11-11.17 Optimize for NUMA, and add support of large matrix.
- 11.18-11.24 Tune performance.
- 11.25-12.1 Finish writeup.

Milestone We plan to complete 100% goal by November 20th, and use the last two weeks to do performance tuning.

Literature Search Following are the related papers we collected:

- *Optimization of sparse Matrix-Vector Multiplication on Emerging Multicore Platforms* by S. Williams et.al, SC 2007.
- *Implementing sparse matrix-vector multiplication on throghput-oriented processors* by Nathan Bell et.al, SC 2009.
- *Model-driven autotuing of sparse matrix-vector multiply on GPUS* by Jee W. Choi et.al, PPoPP 2010.

- *Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks* by Jeremy T. Fineman et.al, SPAA 2009.
- *GraphChi: Large-Scale Graph Computation on Just a PC* by Aapo Kryola et.al, OSDI 2012.

Resources Needed We have a Intel server with two CPU chips to use, so no extra resources are needed at present.

Getting Started We are now reading and discussing related papers.