Parallel Data Mining Tools
(Parallel support vector machine on GPU)

Feb 12, 2010 ICL-UTK

Prof. Geoffrey Fox, Judy Qiu, Seung-Hee Bae, Pragnesh Patel, Jaliya Ekanayake, Huapeng Yuan
Presenter: Pragnesh Patel

Contact: patel27@indiana.edu  www.infomall.org/salsa
Community Grids Laboratory, PTI
Indiana University
Collaboration in *SALSA* Project

- Community Grids Labs-Indiana University
- Microsoft Research
- Bioinformatics, IUB
- IU Medical School
- Cheminformatics, IUB
- Department of Physics, IUB
Outline

• Motivation
• Big picture of the project
• Graphics Processors
• CUDA architecture
• SVM basics
• SVM problems and solutions
• SMO algorithm
• MapReduce
• Implementation detail
• SVM classification
• Results and Comparison
• Future work
• Message
• Current projects
Motivation

• Today we are dealing with the large data set
• Areas: bioinformatics, physics, cheminformatics and medical
• Data analysis/Data mining is the key requirement in all these disciplines
• Developing Library of efficient data mining algorithms
• Choose algorithms that can be parallelized well
Motivation for GPGPU

• Kernel-based methods are computationally expensive
  - we often have more data than we can afford to process (we are all interested in scalability, over past several years single thread performance has not been increasing)

• Future performance will come through parallelism
  - single thread performance increases are tapped out

• Highly Parallel, general purpose processors are now becoming widely available GPUs are at the forefront of this trend

• Massive on-chip parallism can make it easier to parallelize algorithms
  - synchronization is cheaper

• Supercomputing at 1/3 cost ??
Here we are looking data analysis using cluster system, cloud computing and graphics processors

Note that focus on data analysis is relatively recent (e.g. in bioinformatics) and in era dominated by fast sequential computers

- Many key algorithms such as HMM, SVM, MDS, Gaussian Modeling, Clustering do not have good available parallel implementations/algorithms

The SALSA project is developing and applying parallel and distributed Cyber infrastructure to support large scale data analysis.

- Semiconductor companies provides Multicore, Many core and GPUs etc.
- New programming model and system software to bridge an application and architecture/hardware
- The exponentially growing volumes of data requires robust high performance tools.
Developing and applying parallel and distributed Cyberinfrastructure to support large scale data analysis.

- Childhood Obesity Studies (314,932 patient records/188 dimensions)
- Indiana census 2000 (65535 GIS records / 54 dimensions)
- Biology gene sequence alignments (640 million / 300 to 400 base pair)
- Particle physics (1 terabytes data that placed in IU Data Capacitor)
Component 2

- Data
  - Laptops
  - Desktops
  - Network Connection
  - HPC clusters
  - GPUs

- Application
  - Workstations

- Software
The exponentially growing volumes of data require robust high performance tools.

- Parallelization frameworks
  - MPI for high performance clusters of multicore systems
  - MapReduce for distributed systems (Hadoop, Dryad)
  - CUDA for GPU programming
- Data mining algorithms and tools
  - Deterministic Annealing Clustering (VDAC)
  - Pairwise Clustering
  - k-means, Nearest Neighbor, Naïve bayes
  - SVM
Component 4

Hardware

Software

Data

Data Intensive (Science) Applications
- Health
- Biology
- Chemistry
- Particle Physics
- GIS

Application
Classes of implementation

• Cluster system
• Cloud computing
• GPGPU
Cluster programming

- Cluster: A computer cluster is a group of linked computers, working together closely so that in many respects they form a single computer

- Tool: MPI

  - MPI is a widely-available communications library that enables parallel programs to be written in C, Fortran, Python, and many other programming languages.
Cloud Computing: Infrastructure and Runtimes

• Cloud infrastructure: outsourcing of servers, computing, data, file space, etc.
  – Handled through Web services that control virtual machine lifecycles.

• Cloud runtimes: tools (for using clouds) to do data-parallel computations.
  – Apache Hadoop, Google MapReduce, Microsoft Dryad, and others
  – Designed for information retrieval but are excellent for a wide range of science data analysis applications
  – Can also do much traditional parallel computing for data-mining
Cloud Related Technology Research

• MapReduce
  – Hadoop
  – Dryad (Microsoft) on Windows HPCS

• Azure Microsoft cloud

• FutureGrid
GPGPU

- GPGPU (General Purpose Graphic Processing Unit) computing is a technique to perform computation in applications traditionally handled by the CPU on GPU. It provides tremendous amount of processing power. GPUs are currently transitioning from their initial role as specialized to general purpose engines for high throughput floating-point computation.

- Additionally GPUs have much more aggressive memory subsystem, typically endowed with more than 10x higher memory bandwidth than a CPU.

- GPU architecture are specialized for computer intensive, highly-parallel computation, and therefore are designed such that more resources are devoted to data processing than caching and flow control.
## Graphics Processors

### Specification of GPU NVIDIA GeForce 8800

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Streaming Multiprocessors</td>
<td>16</td>
</tr>
<tr>
<td>Multiprocessor Width</td>
<td>8</td>
</tr>
<tr>
<td>Local Store Size</td>
<td>16 KB</td>
</tr>
<tr>
<td>Total number of Stream Processors</td>
<td>128</td>
</tr>
<tr>
<td>Peak SP Floating Point Rate</td>
<td>346 Gflops</td>
</tr>
<tr>
<td>Clock</td>
<td>1.35 GHz</td>
</tr>
<tr>
<td>Device Memory</td>
<td>768 MB</td>
</tr>
<tr>
<td>Peak Memory Bandwidth</td>
<td>86.4 GB/sec</td>
</tr>
<tr>
<td>Connection to Host CPU</td>
<td>PCI Express</td>
</tr>
<tr>
<td>CPU -&gt; GPU bandwidth</td>
<td>2.2 GB/sec</td>
</tr>
<tr>
<td>GPU -&gt; CPU bandwidth</td>
<td>1.7 GB/sec</td>
</tr>
</tbody>
</table>
GPU programming

• Programming is done through CUDA, an architecture developed by NVIDIA.
• Programmers organizes the computation into grids, which are organized as set of thread blocks.
• Programmer writes a single thread, designed to be launched in very large numbers (thousands to millions)
GPU memory model

- Each block can have up to 512 threads that synchronize
- Millions of blocks can be issued
CUDA
History of SVM

- SVM is a classifier derived from statistical learning theory by Vapnik
- SVM was first introduced in COLT-92
- SVM became famous when, using pixel maps as input, it gave accuracy comparable to NNs with hand-designed features in a handwriting recognition task
- SVM are a particular instance of Kernel Machines (large class of learning algorithms)
SVM

• A hugely popular machine learning technique for classification
• They have been applied to a wide range of applications, with good results
• Involves training stage, where machine learns patterns from the training data set
• Tries to find a hyperplane separating the different classes with “maximum margin”
• Non-linear surfaces can be generated through non-linear kernel functions
• Uses Quadratic Programming for training (specific set of constraints imply a wide variety of techniques for solving it)
Best Linear Separator?
Best Linear Separator: 

**Support Plane method**

\[ x \cdot w = \gamma + 1 \]
\[ x \cdot w = \gamma \]
\[ x \cdot w = \gamma - 1 \]

Maximize distance between two parallel support planes

Distance = “Margin” = \[ \frac{2}{||w||} \]
Maximize margin using quadratic program optimization

\[
\min_{(w, \gamma) \in \mathbb{R}^{n+1}} \frac{1}{2} \|w\|^2
\]

such that

\[
x_i \cdot w \geq \gamma + 1 \quad \text{if} \quad x_i \in \text{class } A_+
\]

\[
x_i \cdot w \leq \gamma - 1 \quad \text{if} \quad x_i \in \text{class } A_-
\]
Non-linear kernel

- In some cases, classification requires nonlinear separating surfaces.
- This can be done mapping data onto higher dimensional mathematical spaces.
- Mapped data are linearly separable (details omitted: see, e.g., N. Cristianini, *Support Vector and Kernel Machines*).
what about large datasets

• It is difficult to solve problem for larger data sets
• Requires a very large amount of computational time and memory
• The training time of a binary tasks composed of 100 points with tens of dimensions can often take on the order of hours serial execution.
• There were two approaches
  – Either solve the matrix problems in parallel or
  – Split up dataset and solve multiple sub problems
SVM problems and proposed solution

• SVM training is Quadratic programming optimization
• It does not perform well for larger data sets
• Solution proposed : SMO, Cascade SVM, Interior point method
• I implemented SMO on GPU
SMO algorithm

• The Sequential Minimal Optimization algorithm (Platt 1999) is an iterative solution method for the SVM training problem
• At each iteration, it adjusts only 2 of the variables
• The optimization step is then a trivial one dimensional problem
• Computing full kernel matrix Q not required
• Despite name, algorithm can be quite parallel
• Computation is dominated by KKT optimality condition updates
SVM Training algorithm: SMO

Input: training data $x_i$, labels $y_i$
Initialize: $\alpha_i = 0$, $f_i = -y_i$
Find maximally violating pair $I_{low}$, $I_{high}$
Optimize $\alpha_{i_{high}}$ and $\alpha_{i_{low}}$
repeat
    Map: update $f_i$, $\forall i \in \{1..l\}$
    Reduce: compute $b_{high}$, $I_{high}$, $b_{low}$, $I_{low}$
    Cleanup: Optimize $\alpha_{i_{high}}$ and $\alpha_{i_{low}}$
until $b_{low} \leq b_{high} + 2\xi$
SMO

• The goal is to find an optimal weight for each training point \( x \). The weights and the training set constitutes a classifier.
• SMO reduces SVM QP in each optimization step to its minimum form: updating two weights \( \alpha_i \).
• The bulk of computation is then to update the KKT optimality conditions for the remaining set of data points (map) and then find the two maximally violating weights (reduce), which are then optimized.
• when all points satisfy the optimality conditions to a given tolerance, the algorithm terminates.
Fitting SMO on a GPU

• Shared memory constraints on the GPU fits the algorithm as only two vectors need to be shared among all the threads

• Performance strongly dependent on the choice of the working set

• Several heuristics proposed – two are popular (1\textsuperscript{st} and 2\textsuperscript{nd} order)
First order selection heuristics

• The job of the variable selection heuristic is to choose the 2 variables which will be updated
• We use the maximal violating pair first order heuristic & KKT formulation proposed by (Keerthier al 2001)
Map Reduce

“MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key.”

MapReduce: Simplified Data Processing on Large Clusters
Jeffrey Dean and Sanjay Ghemawat

- Applicable to most loosely coupled data parallel applications
- The data is split into m parts and the map function is performed on each part of the data concurrently
- Each map function produces r number of results
- A hash function maps these r results to one or more reduce functions
- The reduce function collects all the results that maps to it and processes them
- A combine function may be necessary to combine all the outputs of the reduce functions together

map(key, value)

reduce(key, list<value>)

E.g. Word Count

map(String key, String value):
    // key: document name
    // value: document contents

reduce(String key, Iterator values):
    // key: a word
    // values: a list of counts
How does it work?

- The framework supports the splitting of data
- Outputs of the map functions are passed to the reduce functions
- The framework sorts the inputs to a particular reduce function based on the intermediate keys before passing them to the reduce function
- An additional step may be necessary to combine all the results of the reduce functions
Google’s Implementation

- Key Points
  - Data (Inputs) and the outputs are stored in the Google File System (GFS)
  - Intermediate results are stored on local discs
  - Framework, retrieves these local files and calls the reduce function
  - Framework handles the failures of map and reduce functions

E.g. Word Count

map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += parseInt(v);
  Emit(AsString(result));
SVM classification

• The SVM classification problem evaluate an unknown point $x$ with respect to the decision surface constructed in the training process, in order to classify $x$ into one of the two classes.
SVM: a *supervised* learning machine

1) training \[ \rightarrow \] find \((w, \gamma)\)
2) classification

\[ \text{if } z_i \cdot w > \gamma \text{ then } z_i \in A_+ \text{ unknown set} \]

\[ x \cdot w = \gamma \]

\[ \text{if } z_i \cdot w < \gamma \text{ then } z_i \in A_- \]
2) classification

\[ x \cdot w = \gamma \]
Results

- LIBSVM running on intel core 2 duo 2.00GHz
- Polynomial kernel used
- Achieved 5-30 x speedup
## Dataset information

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of instances</th>
<th>Number of attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOREST</td>
<td>50,000</td>
<td>54</td>
</tr>
<tr>
<td>MNIST</td>
<td>60,000</td>
<td>784</td>
</tr>
<tr>
<td>Adult</td>
<td>32,000</td>
<td>120</td>
</tr>
<tr>
<td>Web</td>
<td>49,000</td>
<td>300</td>
</tr>
<tr>
<td>USPS</td>
<td>7,200</td>
<td>256</td>
</tr>
</tbody>
</table>
# Speedup Comparison with LIBSVM

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Parallel SVM training time (sec)</th>
<th>LIBSVM on CPU training time (sec)</th>
<th>Speedup in (X) times compares with LIBSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOREST</td>
<td>16762.851</td>
<td>87329.277</td>
<td>5.21</td>
</tr>
<tr>
<td>MNIST</td>
<td>739.371</td>
<td>21345.657</td>
<td>28.87</td>
</tr>
<tr>
<td>Adult</td>
<td>47.936</td>
<td>689.325</td>
<td>14.38</td>
</tr>
<tr>
<td>Web</td>
<td>267.428</td>
<td>3425.762</td>
<td>12.81</td>
</tr>
<tr>
<td>USPS</td>
<td>4.266</td>
<td>49.325</td>
<td>11.56</td>
</tr>
</tbody>
</table>
## Accuracy Comparison with LIBSVM

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Parallel SVM Accuracy in %</th>
<th>LIBSVM Accuracy in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOREST</td>
<td>85.54 %</td>
<td>87.67 %</td>
</tr>
<tr>
<td>MNIST</td>
<td>93.01 %</td>
<td>96.24 %</td>
</tr>
<tr>
<td>Adult</td>
<td>97.91 %</td>
<td>98.65 %</td>
</tr>
<tr>
<td>Web</td>
<td>98.65 %</td>
<td>97.80 %</td>
</tr>
<tr>
<td>USPS</td>
<td>97.45 %</td>
<td>94.56 %</td>
</tr>
</tbody>
</table>
Conclusion

• Massively parallel processors provide useful speedup on SVM training and classification
• There are other sources of parallelism in SVM training that we have not exploited: multi class
• There is much interesting work to be done in finding massively parallel implementation of data mining algorithms
• Speedup depends on the dataset
• Data parallelism on GPUs or Task parallelism on CPUs
Future work

• Complete suite of useful data mining algorithms using GPUs.
• Comparison with cluster and cloud implementation
• Optimization techniques
• GPU cluster and MPI
References

- Fast training of support vector machines using sequential minimal optimization. J C Platt
- Improvement to Platt’s SMO algorithm for SVM classifier design. S.S. Keerthi et al
- Convergence of a Generalized SMO algorithm for SVM classifier design. S.S. Keerthi et al
- Fast support Vector Machine Training and Classification on Graphic Processors. Bryan et al
- Multiclass Classification using Support Vector Machine GPUs. Sergio et al
- Data Mining with Parallel Support Vector Machine for Classification. Tatjana et al
- Map-Reduce for Machine Learning on Multicore. Cheng et al
- Project website: www.infomall.org/salsa
Message from the talk

- GPGPU are useful approach
- Massive parallel processors provide useful speedup on SVM training step
- Data analysis/mining can use a lot of parallel hardware if software engineered properly
- Computational cost is less with higher accuracy.
Current projects

- OpenMP to OpenCL transformation framework
- Fusion of database management system and file system good features
Thank You !!!