Scalable I/O-Bound Parallel Incremental Gradient Descent for Big Data Analytics in GLADE

Chengjie Qin and Florin Rusu
University of California, Merced

June 23, 2013
Motivation

Bismarck [Feng, Kumar, Recht, and Re: Towards a Unified Architecture for in-RDBMS Analytics, SIGMOD 2012]

- User-Defined Aggregates (UDA) as general mechanism to represent analytics
- Shared memory
- **Scalability?**

<table>
<thead>
<tr>
<th>Task</th>
<th>Name</th>
<th>Dataset # examples</th>
<th>Size</th>
<th>Bismarck PostgreSQL</th>
<th>System DBMS A</th>
<th>System DBMS B</th>
<th>In-memory tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>Classify300M</td>
<td>300M</td>
<td>135GB</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SVM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LMF</td>
<td>Matrix5B</td>
<td>5B</td>
<td>100GB</td>
<td>Yes</td>
<td>N/A</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CRF</td>
<td>DBLP</td>
<td>2.3M</td>
<td>7.2GB</td>
<td>Yes</td>
<td>N/A</td>
<td>N/A</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 1: [Feng, Kumar, Recht, and Re: Towards a Unified Architecture for in-RDBMS Analytics, SIGMOD 2012]
k-means clustering (iterative processing): 1-node vs PostgreSQL: 4s vs 62s (15.5X); 9-node vs Hadoop: 4s vs 1,297s (324.25X)
Research Questions

- What does it take to implement parallel incremental gradient descent (IGD) in GLADE?
- What is the performance of IGD in GLADE?
- Why is GLADE relevant for the Cloud?

<table>
<thead>
<tr>
<th>Task</th>
<th>Dataset</th>
<th>System</th>
<th>GLADE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>Classify300M</td>
<td>300M</td>
<td>135GB</td>
</tr>
<tr>
<td>SVM</td>
<td>Yes</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>LMF</td>
<td>Matrix10B</td>
<td>10B</td>
<td>200GB</td>
</tr>
<tr>
<td>CRF</td>
<td>DBLP</td>
<td>2.3M</td>
<td>7.2GB</td>
</tr>
</tbody>
</table>
Agenda

• Predictive analytics
• Gradient descent
• Incremental (stochastic) gradient descent
• Parallel IGD
• GLADE
• Parallel IGD in GLADE
  – IGD as GLA
  – Iteration management
  – Randomization
  – Merging
• Answer to research questions
Predictive Analytics

- Massive amounts of example data, e.g., 10 billion
- Data are highly-dimensional, e.g., 50 to 600 million

<table>
<thead>
<tr>
<th>Analytics task</th>
<th>Objective function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Regression (LR)</td>
<td>$\sum_{(x_i, y_i) \in \text{data}} \log \left( 1 + e^{-y_i w^T x_i} \right) + \mu | \overrightarrow{w} |_1$</td>
</tr>
<tr>
<td>Classification (Support Vector Machines - SVM)</td>
<td>$\sum_{(x_i, y_i) \in \text{data}} \left(1 - y_i w^T x_i \right) + \mu | \overrightarrow{w} |_1$</td>
</tr>
<tr>
<td>Recommendation (Low-Rank Matrix Factorization - LMF)</td>
<td>$\sum_{(i,j) \in \Omega} \left(L_i^T R_j - M_{ij} \right)^2 + \mu | L, R |_F^2$</td>
</tr>
<tr>
<td>Labeling (Conditional Random Fields - CRF)</td>
<td>$\sum_{(x_i, y_i) \in \text{data}} \left[ \sum_j w_j F_j (y_i, x_i) - \log Z(x_i) \right]$</td>
</tr>
</tbody>
</table>
Gradient Descent

\[ \min_{w \in \mathbb{R}^d} \sum_{(x_i, y_i) \in \text{data}} f(w, x_i, y_i) \]

\[ w^{(k+1)} = w^{(k)} - \alpha_k \nabla f \left( w^{(k)} \right) \]

\( \nabla f \) is the gradient
\( \alpha_k \) is step size or learning rate
\( w^{(0)} \) is the starting point (random)

- Single step is taken for every iteration over data: slow convergence
- Step size is data-dependent: ping-pong effect; line search is expensive
- Convergence to minimum guaranteed for convex objective function
Incremental Gradient Descent (IGD)

**Gradient descent**

\[ w^{(k+1)} = w^{(k)} - \alpha_k \nabla f \left( w^{(k)} \right) \]

- Exact gradient computation
- Single step for one iteration
- Faster convergence close to minimum

**Incremental gradient descent**

\[ w^{(k+1)} = w^{(k)} - \beta_k \nabla f_{\eta(k)} \left( w^{(k)} \right) \]

- Approximate gradient at data point
- Take step for each data point
- Faster convergence far from minimum

Chengjie Qin and Florin Rusu – Parallel IGD in GLADE
Parallel IGD

- Assumption: IGD is commutative and algebraic

Distributed model

- Distribute model
- Merge partial models
- Scalable

Shared model

- Synchronize access to model
- Limited scalability
GLADE Execution Model

Chengjie Qin and Florin Rusu – Parallel IGD in GLADE
IGD as Generalized Linear Aggregate (GLA)

\[
\begin{align*}
\text{Chunk}_1 \quad \eta_{11} \quad w_{1i}^{(k+1)} = w_{1i}^{(k)} - \beta_k \Delta f \eta_1(w_{1i}^{(k)}) \\
\text{Chunk}_r \quad \eta_{1r} \quad w_{1i}^{(k+1)} = w_{1j}^{(k)} - \beta_k \Delta f \eta_1(w_{1j}^{(k)}) \\
\text{Node}_1 \quad \text{End Chunk} \quad w_{1i}^{(k)} + w_{1j}^{(k+1)} \\
\end{align*}
\]

\[
\begin{align*}
\text{Chunk}_1 \quad \eta_{n1} \quad w_{ni}^{(k+1)} = w_{ni}^{(k)} - \beta_k \Delta f \eta_n(w_{ni}^{(k)}) \\
\text{Chunk}_s \quad \eta_{ns} \quad w_{nj}^{(k+1)} = w_{nj}^{(k)} - \beta_k \Delta f \eta_n(w_{nj}^{(k)}) \\
\text{Node}_n \quad \text{End Chunk} \quad w_{ni}^{(k+1)} + w_{nj}^{(k+1)} \\
\end{align*}
\]

Chengjie Qin and Florin Rusu – Parallel IGD in GLADE
Iteration Management

- **Stop condition**
  - Compute error on consecutive models; fix number of iterations
- **Pass model between iterations**
  - File system (NFS); broadcast message
Data Randomization

- Randomize data during initial loading process: partition on random hash function

**Implicit randomization**

- Chunks are dropped non-deterministically
- Assignment of chunks to GLAs is non-deterministic
- No changes to GLADE

**Explicit randomization**

- Randomize scan order of chunks from disk: StorageManager
- Randomize scan order of tuples inside chunk: BeginChunk
Effect of randomization on convergence rate

- Log Likelihood

- LR on Classify300M
- Single node, single thread
Model Merging

- Strategies
  - (Weighted) average models component-wise
  - Ensembles
- LocalMerge and RemoteMerge provide support for any merging strategy, including hybrid
Effect of merging on convergence rate

-Log Likelihood vs # iterations for 1, 2, 4, 8, and 16 threads.

- LR on Classify300M
- Single node
Single Node Performance

Multi-threaded execution time

- Real
- Ideal CPU
- I/O Bound

Multi-threaded speedup

- Real
- Ideal CPU
- Ideal

- LMF on Matrix10B
- Single node: 16 cores @ 2GHz; 16GB RAM; 4 disks @ 110MB/s throughput/disk
Cluster Performance

Multiple nodes execution time

- Time per iteration (sec)

Multiple nodes speedup

- Speedup

- LMF on Matrix10B
- Single node: 16 cores @ 2GHz; 16GB RAM; 4 disks @ 110MB/s throughput/disk
- Cluster: 8 X worker + coordinator (9 nodes); Gigabit Ethernet; same rack
Answer to Research Questions

- What does it take to implement parallel IGD in GLADE?
  - Bismarck UDA-based implementation with conversion in data representation

- What is the performance of IGD in GLADE?
  - I/O-bound
  - Linear scalability on single-node (multi-threaded) and cluster

- Why is GLADE relevant for the Cloud?
  - Full resource utilization for your money
  - 324.25X on $30,000 9-node cluster vs Hadoop

<table>
<thead>
<tr>
<th>Task</th>
<th>Name</th>
<th>Dataset # examples</th>
<th>Size</th>
<th>Bismarck PostgreSQL</th>
<th>System GLADE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>Classify300M</td>
<td>300M</td>
<td>135GB</td>
<td>Yes</td>
<td>31.94 sec/iteration</td>
</tr>
<tr>
<td>SVM</td>
<td>LR</td>
<td></td>
<td></td>
<td>Yes</td>
<td>32.15 sec/iteration</td>
</tr>
<tr>
<td>SVM</td>
<td>SVM</td>
<td></td>
<td></td>
<td>Yes</td>
<td>80.04 sec/iteration</td>
</tr>
<tr>
<td>SVM</td>
<td>LMF</td>
<td>Matrix10B</td>
<td>200GB</td>
<td>Yes?</td>
<td>969.23 sec/iteration</td>
</tr>
<tr>
<td>SVM</td>
<td>CRF</td>
<td>DBLP</td>
<td>7.2GB</td>
<td>Yes</td>
<td>969.23 sec/iteration</td>
</tr>
</tbody>
</table>

Chengjie Qin and Florin Rusu – Parallel IGD in GLADE
Questions