Progressive & Algorithms & Systems

Florin Rusu

University of California Merced
Lawrence Berkeley National Laboratory
>>> select count(*) from candidates where mag > 10

>>> select count(*) from candidates where mag > 10 and mag < 100

>>> select count(*) from candidates where mag > 10 and mag < 200
How to derive confidence bounds that shrink progressively?
How to generate a meaningful estimate and confidence bounds as early as possible?
How to minimize the estimation overhead?
1. PF-OLA: Parallel Framework for Online Aggregation
2. OLA-GD: Online Aggregation for Gradient Descent Optimization
3. OLA-RAW: Online Aggregation over Raw Data
### PF-OLA API

<table>
<thead>
<tr>
<th>Method</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init ()</td>
<td>Basic interface</td>
</tr>
<tr>
<td>Accumulate (Item $d$)</td>
<td></td>
</tr>
<tr>
<td>Merge (UDA $input_1$, UDA $input_2$, UDA $output$) – local and remote</td>
<td></td>
</tr>
<tr>
<td>Terminate () – local and final</td>
<td></td>
</tr>
<tr>
<td>BeginChunk()</td>
<td>Chunk processing</td>
</tr>
<tr>
<td>EndChunk ()</td>
<td></td>
</tr>
<tr>
<td>Serialize ()</td>
<td>Transfer UDA across processes</td>
</tr>
<tr>
<td>Deserialize ()</td>
<td></td>
</tr>
<tr>
<td>EstimatorTerminate ()</td>
<td>Progressive computation</td>
</tr>
<tr>
<td>EstimatorMerge (UDA $input_1$, UDA $input_2$, UDA $output$)</td>
<td></td>
</tr>
<tr>
<td>Estimate ($estimator$, lower, upper, confidence)</td>
<td>OLA estimation</td>
</tr>
</tbody>
</table>

**Notes:**
- UDA: Unordered Data Array
Partial Aggregation

Execution Engine

Thread Pool

Thread
Thread
Thread

WorkUnit₁
WorkUnit₂

Processing

Waypoint

GLA List

积

Partial Result

Florin Rusu
Progressive & Algorithms & Systems
Parallel Sampling

- **Centralized random shuffling**
  - Permute data randomly at loading
  - Scan produces larger samples

- **Stratified sampling**
  - Permute data randomly in each partition
  - Direct extension of random shuffling to partitioned data

- **Global data randomization at loading**
  - Split data randomly at each node
  - Permute all received data randomly
  - Standard hash-based data partitioning
• Centralized
• Distributed tree
AGG $\leftarrow$ SELECT SUM($f(d)$)
FROM $D$
WHERE $P(d)$

- $S$ is simple random sample without replacement from $D$
- Estimator $X = \frac{|D|}{|S|} \sum_{s \in S, P(s)} f(s)$

$E[X] = AGG$

\[
\text{Var}[X] = \frac{|D| - |S|}{(|D| - 1)|S|} \left[ |D| \sum_{d \in D, P(d)} f^2(d) - \left( \sum_{d \in D, P(d)} f(d) \right)^2 \right]
\]

\[
\text{EstVar}[X] = \frac{|D|(|D| - |S|)}{|S|^2(|S| - 1)} \left[ |S| \sum_{s \in S, P(s)} f^2(s) - \left( \sum_{s \in S, P(s)} f(s) \right)^2 \right]
\]
Parallel Sampling Estimators

- Data are partitioned across \( N \) nodes: \( D = D_1 \cup D_2 \cup \cdots \cup D_N \)
- Take samples \( S_i, 1 \leq i \leq N \) independently at each node

**Single Estimator**

- Guarantee
  \[ S = S_1 \cup S_2 \cup \cdots \cup S_N \text{ is a sample from } D \]
- Synchronized estimator
  \[ \frac{S_i}{D_i} = k \text{ (const), } 1 \leq i \leq N \]
- Asynchronous estimator
  Global data randomization

**Multiple Estimators**

- Stratified sampling
- Build an estimator \( X_i \) for each partition \( D_i, 1 \leq i \leq N \):
  \[ X_i = \frac{|D_i|}{|S_i|} \sum_{s \in S_i, P(s)} f(s) \]
- \( X = \sum_{i=1}^{N} X_i \) is unbiased
- \( \text{Var} \left[ \sum_{i=1}^{N} X_i \right] = \sum_{i=1}^{N} \text{Var} [X_i] \)
SELECT n.name, SUM(l.extendprice*(1-l.discount)*(1+l.tax))
FROM lineitem, supplier, nation
WHERE l.shipdate = 1993-02-26 AND l.quantity = 1 AND
l.discount between [0.02,0.03] AND
l.suppkey = s.suppkey AND s.nationkey = n.nationkey
GROUP BY n.name

- TPC-H scale **8,000 (8TB)**
- Single node: 16 cores @ 2GHz; 16GB RAM; 4 disks @ 110MB/s throughput/disk
- Cluster: 8 X worker + coordinator (9 nodes); Gigabit Ethernet; same rack
## Estimation Overhead

<table>
<thead>
<tr>
<th>Query</th>
<th>Execution Time (seconds)</th>
<th>No estimation</th>
<th>OLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregate</td>
<td>222 222</td>
<td>222</td>
<td></td>
</tr>
<tr>
<td>Group$_{small}$</td>
<td>344 345</td>
<td>345</td>
<td></td>
</tr>
<tr>
<td>Group$_{large}$</td>
<td>404 407</td>
<td>407</td>
<td></td>
</tr>
<tr>
<td>Join</td>
<td>409 411</td>
<td>411</td>
<td></td>
</tr>
</tbody>
</table>

Chengjie Qin and Florin Rusu. *Parallel Online Aggregation in Action*. SSDBM 2013, pp. 383–386. [Demo]

1 PF-OLA: Parallel Framework for Online Aggregation
2 OLA-GD: Online Aggregation for Gradient Descent Optimization
3 OLA-RAW: Online Aggregation over Raw Data
Gradient Descent Optimization

\[
\min_{\vec{w} \in \mathbb{R}^d} \left\{ \Lambda(\vec{w}) = \sum_{(\vec{x}_i, y_i) \in \text{data}} f(\vec{w}, \vec{x}_i; y_i) \right\}
\]

\[
\vec{w}(k+1) = \vec{w}(k) - \alpha^{(k)} \nabla \Lambda \left( \vec{w}(k) \right)
\]

\(\Lambda(\vec{w})\) is the loss
\(\nabla \Lambda (\vec{w}) = \left[ \frac{\partial \Lambda(\vec{w})}{\partial w_1}, \ldots, \frac{\partial \Lambda(\vec{w})}{\partial w_d} \right]\) is the gradient
\(\alpha^{(k)}\) is step size or learning rate
\(\vec{w}(0)\) is the starting point (random)

- Convergence to minimum guaranteed for convex objective function
Gradient Descent as GLADE GLA

\[ w_{1i}^{(k+1)} = w_{1i}^{(k)} - \beta_k \Delta f_{\eta_1}(w_{1i}^{(k)}) \]

End Chunk

Local Term

\[ w_{1j}^{(k+1)} = w_{1j}^{(k)} - \beta_k \Delta f_{\eta_1}(w_{1j}^{(k)}) \]

End Chunk

\[ w_{1i}^{(k+1)} + w_{1j}^{(k+1)} \]

Chunk 1

Chunk r

Node 1

\[ w_{ni}^{(k+1)} = w_{ni}^{(k)} - \beta_k \Delta f_{\eta_n}(w_{ni}^{(k)}) \]

End Chunk

Local Term

\[ w_{nj}^{(k+1)} = w_{nj}^{(k)} - \beta_k \Delta f_{\eta_n}(w_{nj}^{(k)}) \]

End Chunk

\[ w_{ni}^{(k+1)} + w_{nj}^{(k+1)} \]

Chunk 1

Chunk s

Node n

\[ \sum w_n^{(k+1)} \]

Term

\[ w_{n1}^{(k+1)} + w_{n2}^{(k+1)} + \cdots + w_{n1}^{(k+1)} + w_{nj}^{(k+1)} + w_{ni}^{(k+1)} + \cdots + w_{nj}^{(k+1)} + w_n^{(k+1)} \]
Main idea: apply online aggregation (OLA) sampling to speed-up the execution of a speculative iteration

Approximate Gradient Descent

Speculative Gradient Descent

Approximate Gradient Descent
Approximate BGD

1: for each example \((\tilde{x}_{\eta(j)}, y_{\eta(j)})\) do
2:   for each active model \(\tilde{w}_i\) do in parallel
3:     Estimate gradient \(G_i[\text{est, std}]: \nabla \Lambda(\tilde{w}_i) \{ (\tilde{x}_{\eta(j)}, y_{\eta(j)}) \}\)
4:     Estimate loss \(L_i[\text{est, std}]: \Lambda(\tilde{w}_i, \tilde{x}_{\eta(j)}; y_{\eta(j)})\)
5:   end for
6: if estimators_convergence_check\((j)\) then
7:   Prune out models: \(\text{Stop Loss}\{L_i[\text{est, std}]\}_{i \leq s, \varepsilon_1}\)
8:   Let \(t\) be the number of remaining models, i.e., active
9:   if \(\text{Stop Gradient}\{G_{i/t}[\text{est, std}]\}_{t \leq d, \varepsilon_2}\) then break
10: Let \(s = t\)
11: end if
12: end for

OLA sampling

- no pre-determined sample size
- avoid correlation between estimators by random data shuffling (at runtime)
- continuous sampling at runtime until estimators converge

SELECT
\(\ldots, \text{SUM}(f_q), \ldots\)
FROM data AS D
SELECT
\(\ldots, Z_q = \text{SUM}(f_q), \ldots\)
\(\ldots, Z_q^2 = \text{SUM}(f_q^2), \ldots\)
FROM sample AS S

- estimator: \(Z_q \cdot \frac{|D|}{|S|}\)
- variance:
\[
\frac{|D|(|D| - |S|)}{|S|^2(|S| - 1)} \left(|S|Z_q^2 - (Z_q)^2\right)
\]
Approximate BGD

1. for each example \((\tilde{x}_i^{(j)}, y_i^{(j)})\) do
2.   for each active model \(\tilde{w}_i\) do in parallel
3.     Estimate gradient \(G_i[\text{est, std}]: \nabla \Lambda(\tilde{w}_i)\{(\tilde{x}_i^{(j)}, y_i^{(j)})\}\)
4.     Estimate loss \(L_i[\text{est, std}]: \Lambda(\tilde{w}_i, \tilde{x}_i^{(j)}; y_i^{(j)})\)
5.   end for
6. if estimators_convergence_check\((j)\) then
7.   Prune out models: \(\text{Stop Loss}(\{L_i[\text{est, std}]\}_{i \leq s}, \epsilon_1)\)
8.   Let \(t\) be the number of remaining models, i.e., active
9.   if \(\text{Stop Gradient}(\{G_{1i}[\text{est, std}]\}_{i \leq d}, \epsilon_2)\) then break
10.  Let \(s = t\)
11. end if
12. end for

- Identify step size with minimum loss based on estimate & confidence bounds
- Interacting estimators
- Exact pruning
- Approximate pruning
Approximate BGD

1: for each example \((\tilde{x}_n(i), y_n(i))\) do
2:     for each active model \(\tilde{w}_i\) do in parallel
3:         Estimate gradient \(G_i[\text{est, std}]: \nabla \Lambda(\tilde{w}_i)\{(\tilde{x}_n(i), y_n(i))\}\)
4:         Estimate loss \(L_i[\text{est, std}]: \Lambda(\tilde{w}_i, \tilde{x}_n(i); y_n(i))\)
5:     end for
6: if estimators_convergence_check\((j)\) then
7:     Prune out models: \(\text{Stop Loss}(\{L_i[\text{est, std}]\}_{i \leq s}, \varepsilon_1)\)
8:     Let \(t\) be the number of remaining models, i.e., active
9:     if \(\text{Stop Gradient}(\{G_{ii}[\text{est, std}]\}_{i \leq d}, \varepsilon_2)\) then break
10:    Let \(s = t\)
11: end if
12: end for

When to stop?

- all estimators converge (million dimensions)
- percentage of estimators converge, e.g., 90%
- unified convergence threshold, e.g., \(\varepsilon' = d \cdot \varepsilon\)
Train SVM Model

BGD OLA # step sizes

Loss per example vs Time (seconds)

- 50M examples, 200 dimensions, 136GB
50M examples, 200 dimensions, 136GB

Chengjie Qin and Florin Rusu. *Speculative Approximations for Terascale Distributed Gradient Descent Optimization*. DanaC@SIGMOD 2015.

Outline

1 PF-OLA: Parallel Framework for Online Aggregation

2 OLA-GD: Online Aggregation for Gradient Descent Optimization

3 OLA-RAW: Online Aggregation over Raw Data
Motivation & Approach

- Load
- Shuffle

Time:
- DBMS
- OLA
- RAW
- OLA-RAW

Elapsed time [sec]
- Error ratio

Florin Rusu
Progressive & Algorithms & Systems
**Parallel Bi-Level Sampling**

- **Exact computation**
  - \[ \begin{array}{cc}
   14 & 3.75 \\
   562 & 21.23 \\
   3216 & 42.836 \\
   72 & 0.1 \\
   
   \end{array} \]

- **Chunk-level sampling**
  - \[ \begin{array}{cc}
   4521 & 90.837 \\
   8425 & 0.526 \\
   3439 & 10.332 \\
   7294 & 9.236 \\
   
   \end{array} \]

- **Bi-level sampling**
  - \[ \begin{array}{cc}
   4521 & 90.837 \\
   8425 & 0.526 \\
   3439 & 10.332 \\
   7294 & 9.236 \\
   
   \end{array} \]

**Inspection paradox:** result order \(\neq\) random chunk order
Experimental Results

Selectivity = 100%

Selectivity = 50%

Yu Cheng, Weijie Zhao, and Florin Rusu. *Bi-Level Online Aggregation on Raw Data*. SSDBM 2017.
Thank you!

Questions?