Dot-Product Join: An Array-Relation Join Operator for Big Model Analytics
Chengjie Qin, Martin Torres, and Florin Rusu
University of California Merced

Background & Motivation
Example: Gradient in logistic regression where \( \hat{x} \) is the training example; \( \hat{w} \) is the model.
Model size grows beyond the memory of a single machine.

Dot-Product Join Algorithm

**Optimization Strategies**
Reordering strategies: LSH-based Nearest Neighbor, Radix Sort, K-Center Clustering

Parallel Dot-Product

Experimental Evaluation

Gradient Descent Integration

- **Required**:
  - \( U(\text{index INTEGER}, \text{value NUMERIC}), \text{tid INTEGER}) \)
  - \( V(\text{index INTEGER}, \text{value NUMERIC}) \)

- **Ensure**:
  - \( \text{DP}(\text{tid INTEGER}, \text{product NUMERIC}) \)
  - \( \text{memory budget M} \)

- **Optimization**
  1. For each page \( p_i \) do
  2. Reorder vectors \( u_i \) to cluster similar vectors together
  3. Group vectors \( u_i \) into batches \( B_i \) that access at most \( M \) pages from \( V \)

- **Batch Execution**
  1. For each batch \( B_i \) do
  2. Collect pages \( p_i \) from \( V \) accessed by vectors \( u_i \) in \( B_i \)
  3. Request access to pages in \( V \)

- **Dot-Product Computation**
  1. For each vector \( u_i \) in \( B_i \) do
  2. \( \text{dp} ← \) Dot-Product\((u_i, V)\)
  3. Append \((u_i, \text{tid}, \text{dp})\) to \( \text{DP} \)

4. Request access to pages in \( V \)

5. Return \( \text{DP} \)

**Execution Time (Full Dataset)**
**Relative Improvement over Basic LRU**

<table>
<thead>
<tr>
<th>Chunk</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>93.01</td>
<td>110.43</td>
<td>114.93</td>
<td>6752</td>
<td>65536</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>89.85</td>
<td>108.24</td>
<td>110.43</td>
<td>65536</td>
<td>65536</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>93.01</td>
<td>110.43</td>
<td>114.93</td>
<td>6752</td>
<td>65536</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>89.85</td>
<td>108.24</td>
<td>110.43</td>
<td>65536</td>
<td>65536</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>93.01</td>
<td>110.43</td>
<td>114.93</td>
<td>6752</td>
<td>65536</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>89.85</td>
<td>108.24</td>
<td>110.43</td>
<td>65536</td>
<td>65536</td>
<td></td>
</tr>
</tbody>
</table>

**Optimization Strategies**
- **Reordering heuristics execution time**
- **Relative improvement over basic LRU**

<table>
<thead>
<tr>
<th>Glade</th>
<th>RPG R (28 GB)</th>
<th>PG ARRAY RELATION (28 GB)</th>
<th>MovieDB R (28 GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>skewed</td>
<td>10000 [3.2 GB]</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>matrix</td>
<td>N/A [16 GB]</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>spliced</td>
<td>9564 [40 MB]</td>
<td>81054</td>
<td>N/A</td>
</tr>
<tr>
<td>MovieLens</td>
<td>955 [32 MB]</td>
<td>6648</td>
<td>72480</td>
</tr>
</tbody>
</table>

Dot-product join execution time (in seconds) with memory budget in GB. N/A stands for not finishing the execution in 24 hours.