SUDO: Optimizing Data Shuffling in Data-Parallel Computation by Understanding User-Defined Functions

Microsoft Research Asia
Microsoft Bing
Flow of Distributed Data Parallel Computation

Shuffling as the dominant cost:
200 PB for one-month trace in a production bed
58.6% of cross-pod traffic
Why Shuffling Stages Necessary?

DPP (Data-Partition Properties): Clustered (URL)

Hash By URL

bing, bing, live, live, msn, msn, gnib, gnib, evil, evil, nsm, nsm

Mapper: URL' = Reverse(URL)

Hash By URL'

evil, evil, gnib, gnib, nsm, nsm, ...
Unnecessary Shuffling Stages

DPP (Data-Partition Properties): Clustered (URL)

DPP: Clustered (URL’)

DPP: Clustered (URL’)

Hash By URL

Hash By URL’

Mapper: URL’=Reverse(URL)
Why Not Removed?

DPP (Data-Partition Properties): Clustered (URL)

Functional Property: One-to-One => None

DPP: Clustered (URL) => None (URL')

Hash By URL

Mapper: URL' = Reverse(URL)

Hash By URL'

DPP: Clustered (URL')
What is SUDO?

- Extract functional properties of the UDF
- Reasoning DPP across UDFs and Shuffling Stages
- Remove unnecessary shuffling steps
What’s next?

• DPP (Data-partition properties)
  – What are the DPP?
  – How DPP change across shuffling stages?

• Functional Properties
  – What are the functional properties?
  – How DPP change across UDFs?
  – How to identify the functional properties?
Data-partition Properties (DPP)

- **Weak**
  - None
  - AdHoc
  - Disjoint
  - Clustered

- **Strong**
  - GS0rted
  - PS0rted
  - LS0rted

- **Cross-Partition**
  - Ranged
    - 2,1,2
    - 4
  - Partitioned
    - 2,2
    - 4,1
  - None
    - 4,2
    - 2,1

- **Within-Partition**
  - None
    - 3
    - 2
    - 3
    - 1
  - Contiguous
    - 2
    - 3
    - 3
    - 1
  - Sorted
    - 1
    - 2
    - 3
    - 3
    - 3
DPP Lattice

- **Ranged**:
  - None
  - Cross-Partitioned

- **Partitioned**:
  - AdHoc
  - Clustered
  - PSorted
  - GSorted

- **Weak**:
  - None

- **Strong**:
  - Contiguous
  - Sorted

- **Within-Partition**:
  - None
Example: how DPP changes through shuffling steps

Local sort

Re-partition
- hash
- range

Merge sort

Hash By URL

Scheduler

Client

AdHoc
LSorted
Disjoint
PSorted
How DPP changes through shuffling steps?
(or how to achieve certain DPP via shuffling steps?)

Local sort

Re-partition
- Hash
- Range

Merge

Cross-Partition

Ranged

Partitioned

None

Disjoint

AdHoc

Clustered

GSorted

PSorted

LSorted

Within-Partition

Sorted

Contiguous
How DPP changes through shuffling steps? (or how to achieve certain DPP via shuffling steps?)

Local sort

Re-partition
- Hash
- Range

Merge
Functional Properties

Data-partition Properties

- GSorted
- Clustered
- LSorted

Sufficient and Necessary

Functional Properties

- Strictly-monotonic
- One-to-One
- Monotonic

Partitioning Properties

- Pass-Through
- Strictly-Increasing
- Strictly-Decreasing
- Increasing
- Decreasing
How DPP changes through UDFs?

Pass-through
Strictly-monotonic
One-to-one
Monotonic
Step 1: collect data-shuffling requirements based on given execution plan

Step 2: forward DPP propagation based on transition graph about DPP change across UDFs

Step 3: figure out shuffling ‘delta’ based on transition graph about DPP change across shuffling
Identify Functional Properties via Rule-based Deduction

Increase (day,hour) = _(y, t) :- ASSIGN y x, _(x, t)

Increase (tmp,hour) Floor

Increase(y, t) :- Floor y x, Increase(x, t)

Increase (tmp,hour) /

Increase(x, t) :- PassThrough(x, t)

Increase (tmp,hour) /

Increase(z, t) :- DIV z y x, Increase(y, t), Constant(x)

PassThrough (hour) 24

Constant

PassThrough(hour)

Constant(24)

UDF: day = Floor(hour/24)  

Deduction Rules
Implementation

• UDF analyzer to extract functional property
  – http://research.microsoft.com/Phoenix to extract AST with 8281 LOC (C#)
  – http://bddbdedb.sourceforge.net/ as deduction engine with ~100 Rules

• SUDO rewriter to do optimization
  – ~1316 LOC (C#)
### Coverage Study

Dataset: **236,457** UDFs in **10,099** jobs from production beds in 2010/2011.

<table>
<thead>
<tr>
<th>Property</th>
<th>UDF &lt;out-col, in-col&gt; #</th>
<th>Ratio %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pass-through</td>
<td>1,998,819</td>
<td>84.73</td>
</tr>
<tr>
<td>Strictly-increasing</td>
<td>147,820</td>
<td>6.27</td>
</tr>
<tr>
<td>Strictly-decreasing</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Increasing</td>
<td>138</td>
<td>0</td>
</tr>
<tr>
<td>Decreasing</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>One-to-one</td>
<td>1,758</td>
<td>0.08</td>
</tr>
<tr>
<td>Others</td>
<td>210,544</td>
<td>8.92</td>
</tr>
<tr>
<td><strong>Sum</strong></td>
<td><strong>2,359,079</strong></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>

Among **2,278 (22.6%)** eligible jobs in them, **17.5%** of them can be optimized by SUDO.

- Pass-through is the dominant functional property.
- 91.2% of the functional properties are identified.
- 17.5% of the eligible jobs can be optimized by SUDO.
## Effectiveness Study

<table>
<thead>
<tr>
<th>Case</th>
<th>Machine#</th>
<th>Native Shuffling IO (TB)</th>
<th>Native Latency (min)</th>
<th>Shuffling Stage# Change</th>
<th>Shuffling IO Reduction</th>
<th>Latency Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anchor Data Preprocessing</td>
<td>150</td>
<td>0.9</td>
<td>25</td>
<td>2 =&gt; 1</td>
<td>47%</td>
<td>40%</td>
</tr>
<tr>
<td>Trend Analysis</td>
<td>1,000</td>
<td>60</td>
<td>230</td>
<td>3 =&gt; 1</td>
<td>35%</td>
<td>45%</td>
</tr>
<tr>
<td>Query-Anchor Relevance</td>
<td>2,500</td>
<td>15</td>
<td>96</td>
<td>6 =&gt; 4</td>
<td>41%</td>
<td>-27%</td>
</tr>
</tbody>
</table>

Shuffling IO reduction is significant.

Latency reduction is introduced by data skew, which is rare case.
Related Work

- Data-partition property propagation to reduce shuffling stages
  - Incorporating partitioning and parallel plans into the SCOPE optimizer (ICDE’10)

- Apply program analysis to distributed data-parallel computation
  - Manimal (PVLDB’11)
An inter-disciplinary research area
A place where we leverage PL techniques to advance the state-of-the-art in system and database fields

SUDO is the beginning ...
Thanks!
Questions?

Microsoft Research Asia
Microsoft Bing