7 Things To Know

When Buying for an !

Alekh Jindal, Jorge Quiané, Jens Dittrich
What Shoes? Why Shoes?
Analyzing MR Jobs
(HadoopToSQL, Manimal)

Generating MR Jobs
(PigLatin, Hive)

Executing MR Jobs
(Hadoop++, epiC)

Data Layouts & Access Paths !!
Why Elephant Needs Different Shoes?
### Very Large Scale Storage & Execution

<table>
<thead>
<tr>
<th>DBMS</th>
<th>MapReduce</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="DBMS Diagram" /></td>
<td><img src="image2.png" alt="MapReduce Diagram" /></td>
</tr>
</tbody>
</table>
## Large Data Block Sizes

<table>
<thead>
<tr>
<th>DBMS</th>
<th>MapReduce</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 KB</td>
<td>1 GB</td>
</tr>
</tbody>
</table>
## Block Level Data Replication

<table>
<thead>
<tr>
<th>DBMS</th>
<th>MapReduce</th>
</tr>
</thead>
<tbody>
<tr>
<td>001 alex bsc</td>
<td>001 alex bsc</td>
</tr>
<tr>
<td>002 tim msc</td>
<td>002 tim msc</td>
</tr>
<tr>
<td>003 mat bsc</td>
<td>003 mat bsc</td>
</tr>
<tr>
<td>004 joel bsc</td>
<td>004 joel bsc</td>
</tr>
<tr>
<td>005 phil msc</td>
<td>005 phil msc</td>
</tr>
<tr>
<td>006 ron msc</td>
<td>006 ron msc</td>
</tr>
<tr>
<td>007 neo bsc</td>
<td>007 neo bsc</td>
</tr>
<tr>
<td>008 jack msc</td>
<td>008 jack msc</td>
</tr>
<tr>
<td>009 jens bsc</td>
<td>009 jens bsc</td>
</tr>
<tr>
<td>010 tom msc</td>
<td>010 tom msc</td>
</tr>
</tbody>
</table>
What’s Wrong with Old Shoes?
Current Data Layouts in Hadoop

Row (default)

Column*

PAX**

* A. Floratou et al. Column-Oriented Storage Techniques for MapReduce. PVLDB, April, 2011
** Y. He et al. RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems. ICDE, 2011
### Current Data Layouts in Hadoop

<table>
<thead>
<tr>
<th></th>
<th>Row</th>
<th>Column</th>
<th>PAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-required Reads</td>
<td>🟥</td>
<td>🟢</td>
<td>🟢</td>
</tr>
<tr>
<td>Network Costs</td>
<td>🟢</td>
<td>🟥</td>
<td>🟢</td>
</tr>
<tr>
<td>Data Block Placement</td>
<td>🟢</td>
<td>🟥</td>
<td>🟢</td>
</tr>
<tr>
<td>Tuple Reconstruction</td>
<td>🟢</td>
<td>🟥</td>
<td>🟢</td>
</tr>
</tbody>
</table>
## Current Data Layouts in Hadoop

<table>
<thead>
<tr>
<th>Data Access Cost [sec]</th>
<th>Number of Referenced Attributes (Out of 30)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trojan Layout</td>
<td></td>
</tr>
<tr>
<td>Row Layout</td>
<td></td>
</tr>
<tr>
<td>Column Layout</td>
<td></td>
</tr>
<tr>
<td>PAX Layout</td>
<td></td>
</tr>
<tr>
<td>Optimal Layout</td>
<td></td>
</tr>
</tbody>
</table>

The diagram shows the data access cost in seconds for different data layouts as a function of the number of referenced attributes. The layouts include Trojan, Row, Column, PAX, and Optimal. The graph illustrates how the data access cost increases with the number of referenced attributes, with the Optimal Layout generally having the lowest cost.
What Shoes do We Propose?
Trojan Data Layouts

Replica 1

Replica 2

Replica 3
### Trojan Data Layouts

<table>
<thead>
<tr>
<th></th>
<th>Row</th>
<th>Column</th>
<th>PAX</th>
<th>Trojan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-required Reads</td>
<td>Red</td>
<td>Green</td>
<td>Green</td>
<td>Green</td>
</tr>
<tr>
<td>Network Costs</td>
<td>Green</td>
<td>Red</td>
<td>Green</td>
<td>Green</td>
</tr>
<tr>
<td>Data Block Placement</td>
<td>Green</td>
<td>Red</td>
<td>Green</td>
<td>Green</td>
</tr>
<tr>
<td>Tuple Reconstruction</td>
<td>Green</td>
<td>Red</td>
<td>Red</td>
<td>Green</td>
</tr>
</tbody>
</table>
Challenges in Trojan Data Layouts

How do we design shoe for one leg?

How do we design shoes for all legs?

How do we make the shoes from the design?
How Do We Design the Shoes?
Single Replica

Column groups

Filter

Novel Column Group Interestingness

Interesting Column groups

Pack

Column Group Packing as 0-1 Knapsack

Complete & disjoint column groups

Single Replica
Multiple Replicas

Queries

Query groups

Filter

Interesting Query groups

Pack

Complete & disjoint query groups
Multiple Replicas

Replica 1

Replica 2

Replica 3

Column groups

Filter

Pack

Interesting Column groups

Complete & disjoint column groups

Filter

Pack

Complete & disjoint column groups

Filter

Pack

Complete & disjoint column groups
Multiple Replicas

TPC-H Customer

Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8

Filter

Pack

Q2, Q3, Q4

Replica 1

Columns

Custkey, Nationkey

Name, Address, Phone, AcctBal, Mktsegment, Comment

Filter

Q5

Replica 2

Columns

Name

Custkey

Mktsegment

Interesting Column groups

Q1, Q6, Q7, Q8

Replica 3

Columns

Name

Custkey

Mktsegment

Phone, AcctBal

Address, Nationkey, Comment

Filter

Pack
Trojan Layout Advantages

- Multiple layouts for a given workload
- Default row layout still available
- Specialized replicas for different query sub-class
- Divide and conquer layout computation
How do We Ride the Elephant?
Putting It All Together

**Load**
Create trojan layout configuration file in HDFS
dataset layout-1 layout-2 layout-3

**Query**
Supply referenced attributes in JobConf
itemize UDF to transparently read the referenced attributes

**Schedule**
Three Optimization Options:
- data locality (default)
- best layout
- best layout & locality
How were the Field Trials?
Setup

- **Datasets**
  TPC-H Lineitem, TPC-H Customer, SSB LineOrder, SDSS PhotoObj

- **Queries**
  First 8 queries from the respective benchmark for each table

- **Methodology**
  focus on scan and projection operators i.e. map-phase-only jobs
  improvement: record reader time (I/O and tuple reconstruction)

- **Hardware**
  50 virtual nodes in a 10 node cluster
Per-replica Trojan Layout Performance

TPC-H Lineitem

<table>
<thead>
<tr>
<th>TPC-H Queries</th>
<th>Improvement Factor over Hadoop-Row</th>
<th>Improvement Factor over Hadoop-PAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>4.3</td>
<td>5.6</td>
</tr>
<tr>
<td>Q2</td>
<td>3.8</td>
<td>5.2</td>
</tr>
<tr>
<td>Q3</td>
<td>1.2</td>
<td>3.5</td>
</tr>
<tr>
<td>Q4</td>
<td>3.5</td>
<td>4.8</td>
</tr>
<tr>
<td>Q5</td>
<td>4.8</td>
<td>5.2</td>
</tr>
<tr>
<td>Q6</td>
<td>3.8</td>
<td>4.6</td>
</tr>
<tr>
<td>Q7</td>
<td>3.2</td>
<td>3.8</td>
</tr>
<tr>
<td>Q8</td>
<td>3.5</td>
<td>3.8</td>
</tr>
</tbody>
</table>

The chart above illustrates the improvement in data access time when using per-replica Trojan Layouts over Hadoop-Row and Hadoop-PAX for TPC-H queries. The bars show the improvement factor, with blue bars indicating the performance over Hadoop-Row and green bars indicating the performance over Hadoop-PAX. The queries range from Q1 to Q8, with Q1 showing the highest improvement factor of 4.3 over Hadoop-Row. The results indicate that Trojan Layouts significantly reduce redundant attribute access and tuple reconstruction overhead, thereby improving performance.
## Layout Quality

<table>
<thead>
<tr>
<th></th>
<th>#Non-required Attributes Read</th>
<th>#Joins in Tuple Reconstruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP-ROW</td>
<td>525</td>
<td>0</td>
</tr>
<tr>
<td>HADOOP-PAX</td>
<td>0</td>
<td>139</td>
</tr>
<tr>
<td>HYRISE* Layout</td>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>Trojan Layout</td>
<td>14</td>
<td>20</td>
</tr>
</tbody>
</table>

>14% improvement over HYRISE

---

* M. Grund et al. HYRISE - A Main Memory Hybrid Storage Engine. PVLDB, November, 2010.
Scheduling Decisions

TPC-H Lineitem

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling Penalty</td>
<td>1.00</td>
<td>2.00</td>
<td>1.00</td>
<td>2.00</td>
<td>1.00</td>
<td>2.00</td>
<td>1.00</td>
<td>2.00</td>
</tr>
</tbody>
</table>

Legend:
- Green: Best-Layout & Locality
- Blue: Best-Layout
- Orange: Locality (default)
Summary

• Data layouts crucial to MR job performance
• Exploit default data block replication in MR
• Novel algorithm to compute per-replica layouts
• Improvement: \(4.8\times\) over Row, \(3.5\times\) over PAX
• Better than HYRISE; 14\% improvement