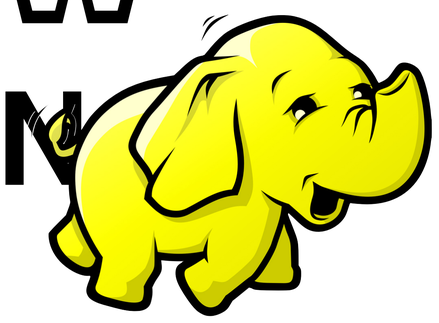
# 7 THINGS TO KNOW WHEN BUYING FOR AN

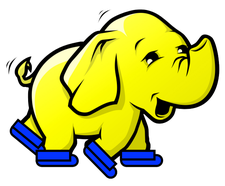
**Shoemakers:**
Alekh Jindal, Jorge Quiané, Jens Dittrich



## 1 WHY SHOES?

MapReduce has been subject of active research in many aspects:

- Analysis and optimization of MapReduce jobs
- High-level query languages
- Efficient execution of MapReduce jobs
- and many others...



However, **data layouts** have not been explored in depth.

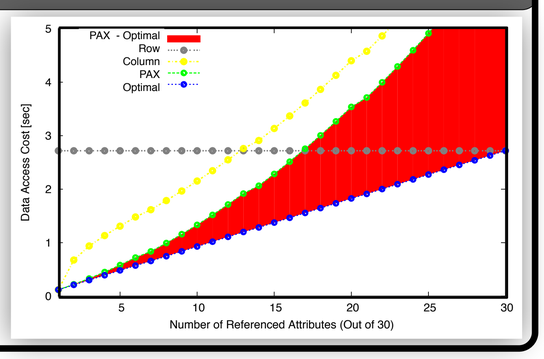
## 2 WHY OUR ELEPHANT NEEDS DIFFERENT SHOES?

| DBMS | MapReduce |
|---|---|
| •Typically deployed over a **small number** of nodes | •Typically deployed over **thousands** of nodes |
| •Usually use **8KB** data page size | •Data block size of **64MB** by default (up to **1GB**) |
| •Data is typically replicated at the **table**-level | •Data is replicated **3 times** by default at the **block**-level |

## 3 WHAT IS WRONG WITH OLD SHOES?

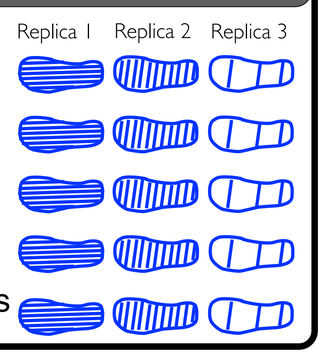
Row-, Column-, and PAX stores:

- Many redundant reads
- High network cost
- Complex data block placement
- High tuple reconstruction cost




## 4. WHAT SHOES DO WE PROPOSE?

- We keep the external-view of a data block intact:
  - Data blocks store the same data as before
  - Inside a block: any layout e.g. pax, column-group
- We exploit the default data block replication:
  - Different layouts for different data block replicas
  - Each replica optimized for different query sub-class




## 5 HOW DO WE DESIGN THE SHOES?

- **Interestingness function:** average normalized mutual information between any pair of attributes

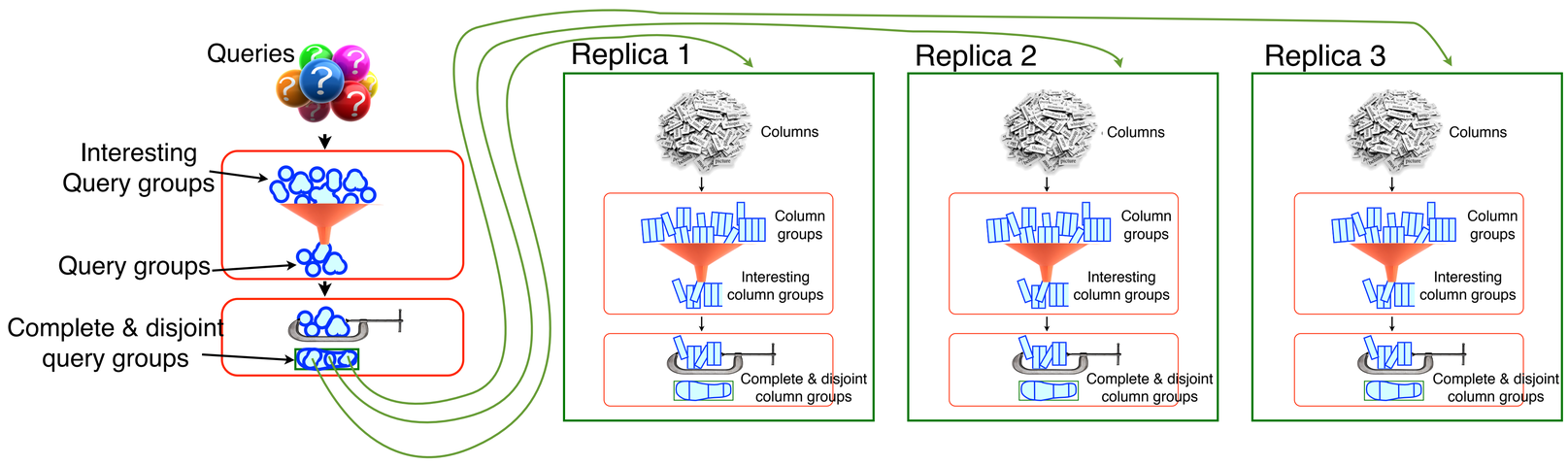
$$Intg(G) = \begin{cases} \frac{1}{\binom{|G|}{2}} \cdot \sum_{\{A,B\} \in G, A \neq B} nMI(A,B) & |G| > 1, \\ \frac{1}{|\mathbb{A}|-1} \cdot \sum_{A \in G, B \in \mathbb{A} \setminus G} 1 - nMI(A,B) & |G| = 1. \end{cases}$$

- We consider the column group packing as 0-1 Knapsack problem

$$max \sum_{i=1}^{m} Intg(G_i) \cdot x_i \quad \text{, subject to:}$$

$$\sum_{i=1}^{m} id(G_i) \cdot x_i \leq id(\mathbb{A})$$

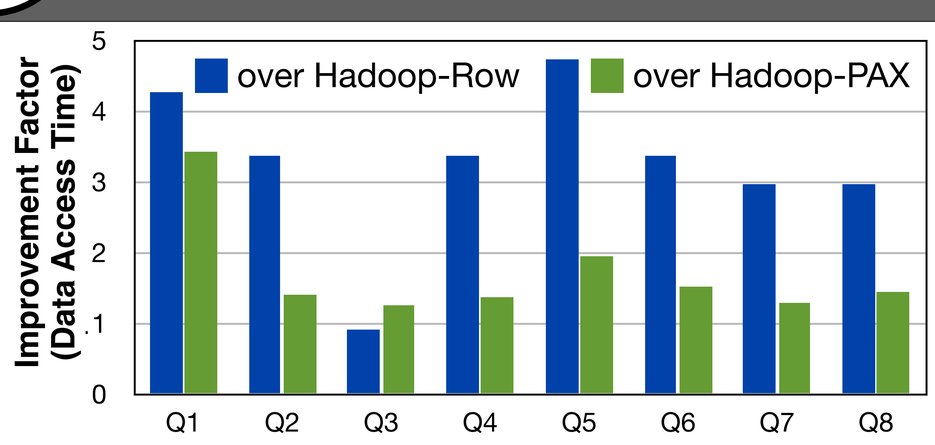
$$x_i + x_j \leq 1, \quad \forall i, j \text{ s.t. } i \neq j \wedge G_i \cap G_j \neq \emptyset.$$




## 6 HOW DO WE RIDE OUR ELEPHANT?

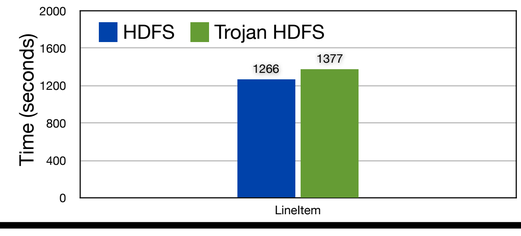
**(1)** Create the Trojan Layout configuration file in HDFS, e.g.
*MyDatasetName Layout-1 Layout-2 Layout-3*
**(2)** Upload input into HDFS as before
**(3)** Supply referenced attributes in the job configuration
**(4)** Supply the *itemize* UDF to transparently read Trojan Layouts
**(5)** Route map tasks to the best-layouts

## 7 HOW WERE THE FIELD TRIALS?




| | #Redundant Attributes Read | #Joins in Tuple Reconstruction |
|---|---|---|
| HADOOP-ROW | 525 | 0 |
| HADOOP-PAX | 0 | 139 |
| HYRISE* Layout | 2 | 64 |
| Trojan Layout | 14 | 20 |

**Upload Time**




## Further Information

**PAPER:** Trojan Data Layouts: Right Shoes for a Running Elephant. Alekh Jindal, Jorge Quiané, Jens Dittrich. SoCC 2011.

**TALK:** Big Data Session, Talk 3 (10:30-12:00).
Friday, 28th October.

**HADOOP++ PROJECT:** http://infosys.uni-saarland.de/hadoop++.php

**OCTOPUSDB PROJECT:** http://infosys.uni-saarland.de/octopusdb.php

SAARLAND UNIVERSITY
Information Systems Group