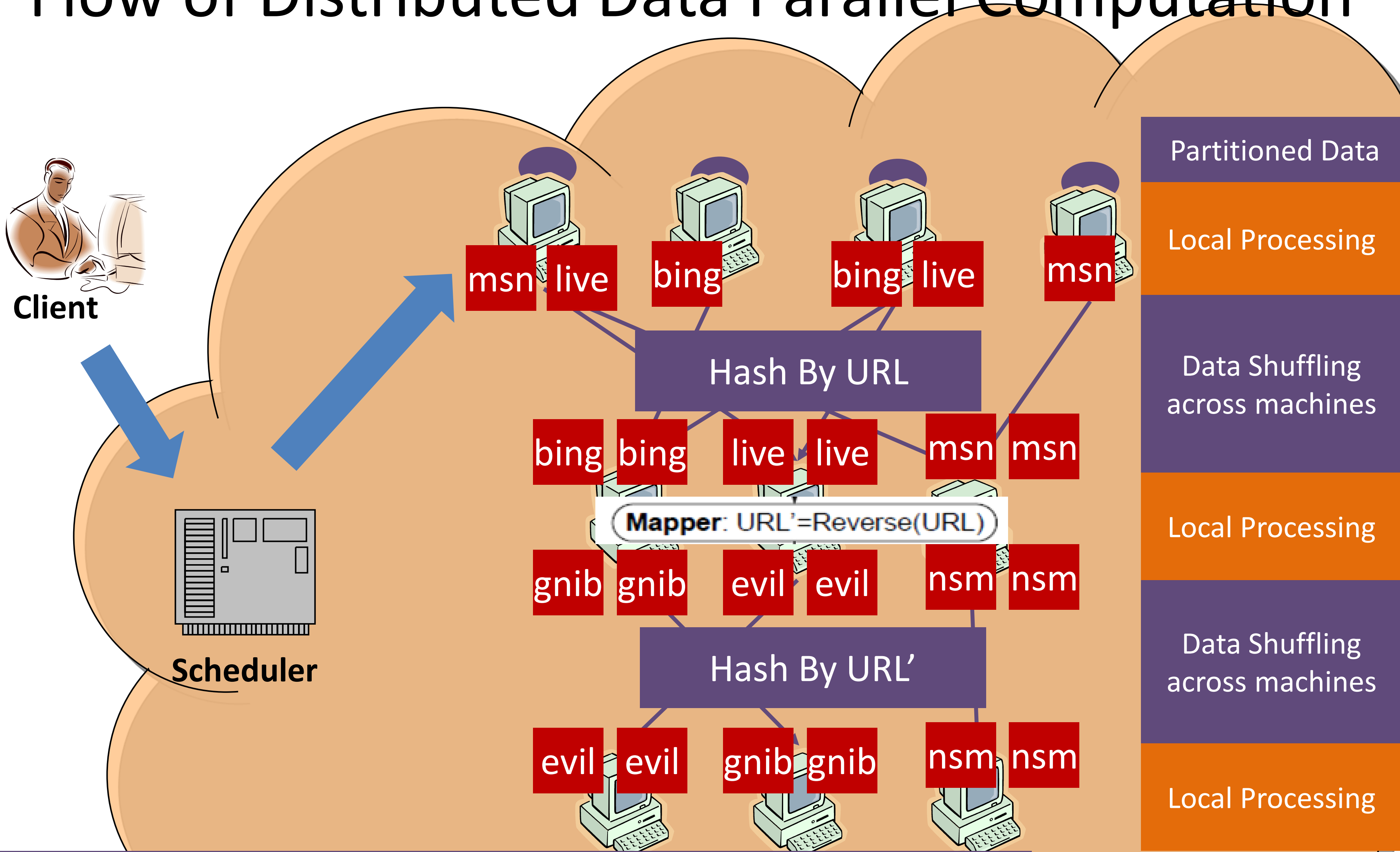
SUDO: Optimizing Data Shuffling in Data-Parallel Computation by Understanding User-Defined Functions

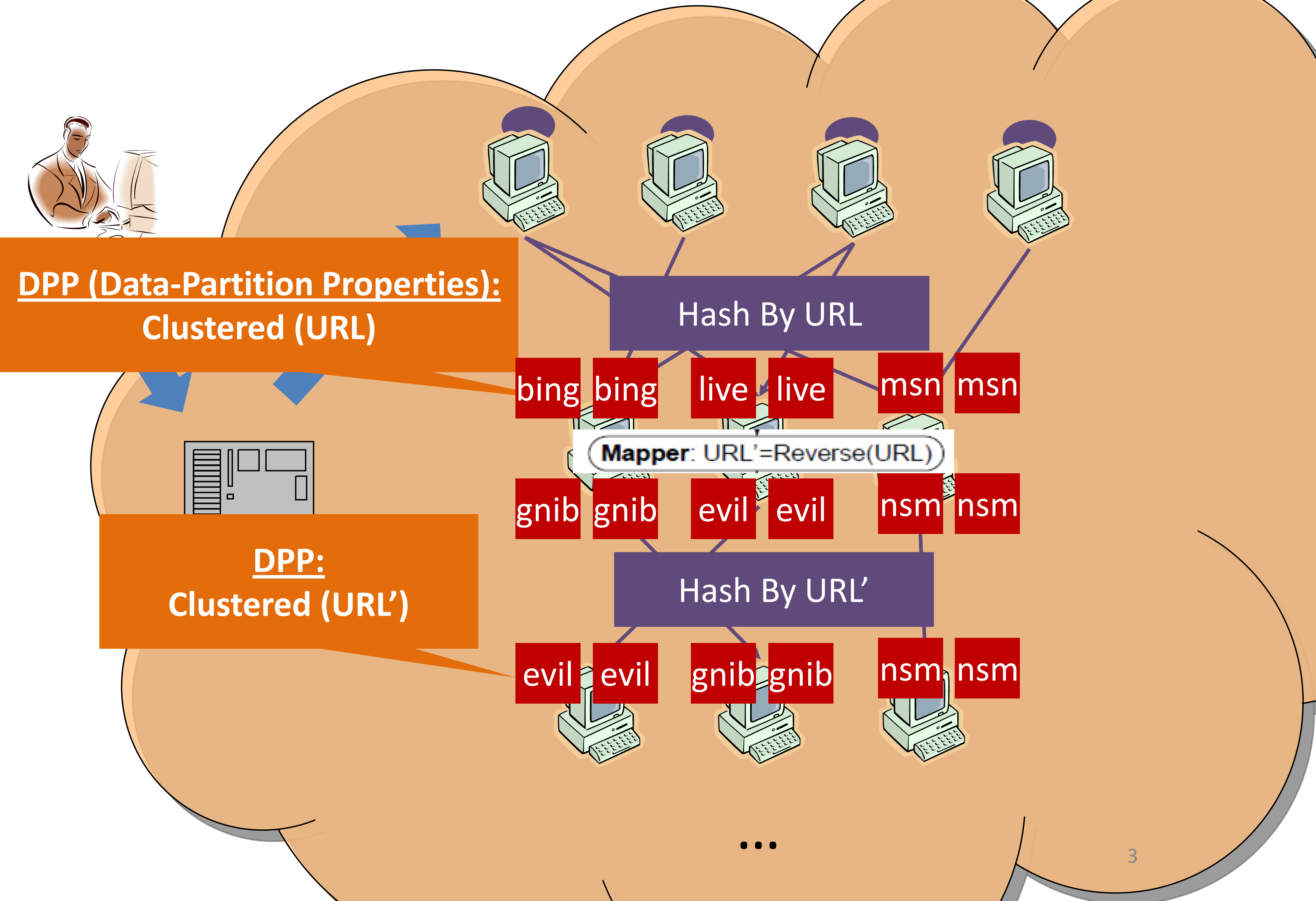
Jiaxing Zhang, Hucheng Zhou, Rishan Chen, Xuepeng Fan, Zhenyu Guo,
Haoxiang Lin, Jack Y. Li, Wei Lin, Jingren Zhou, Lidong Zhou
Microsoft Research Asia
Microsoft Bing

Flow of Distributed Data Parallel Computation

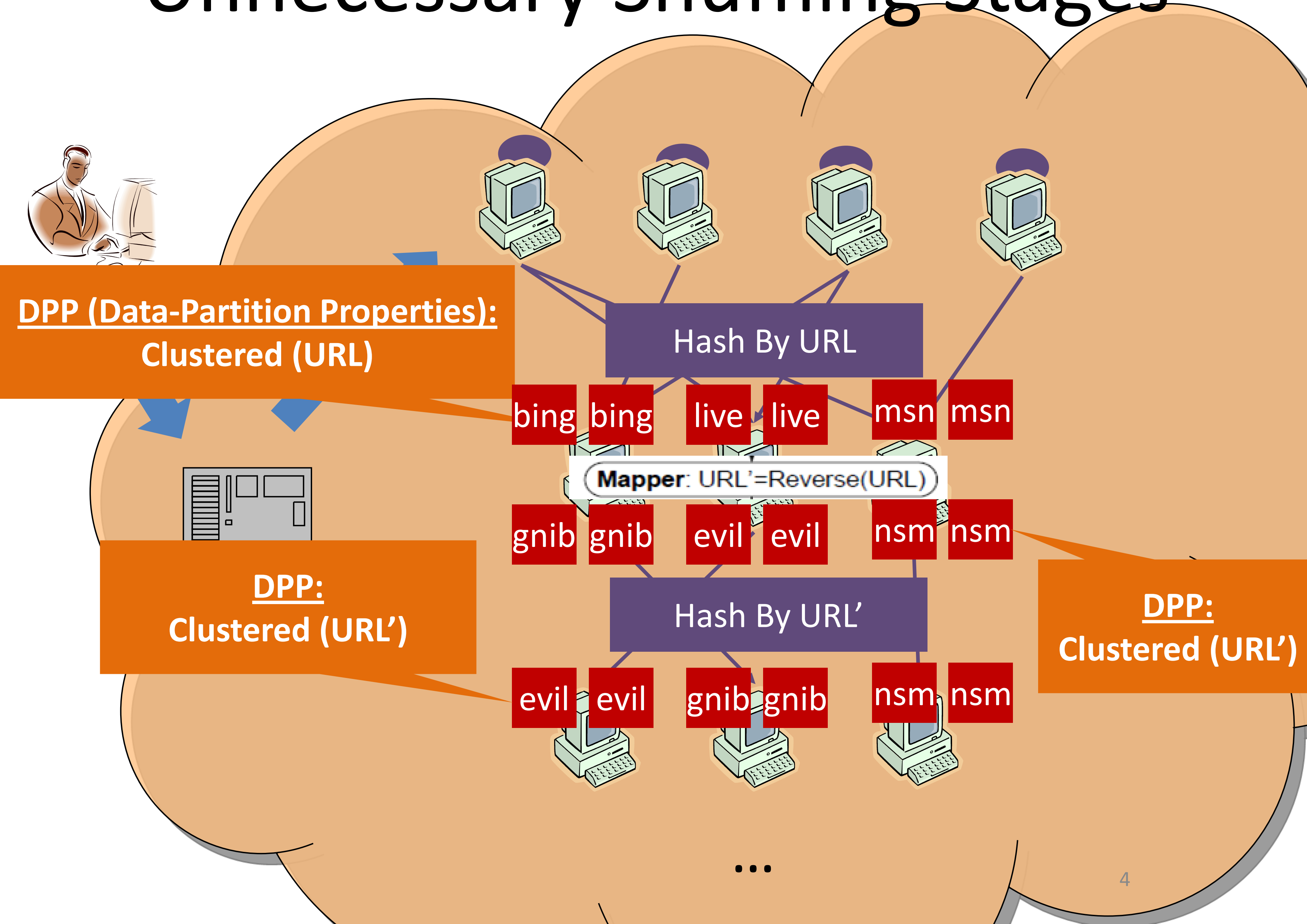


Shuffling as the dominant cost:
200 PB for one-month trace in a production bed
58.6% of cross-pod traffic

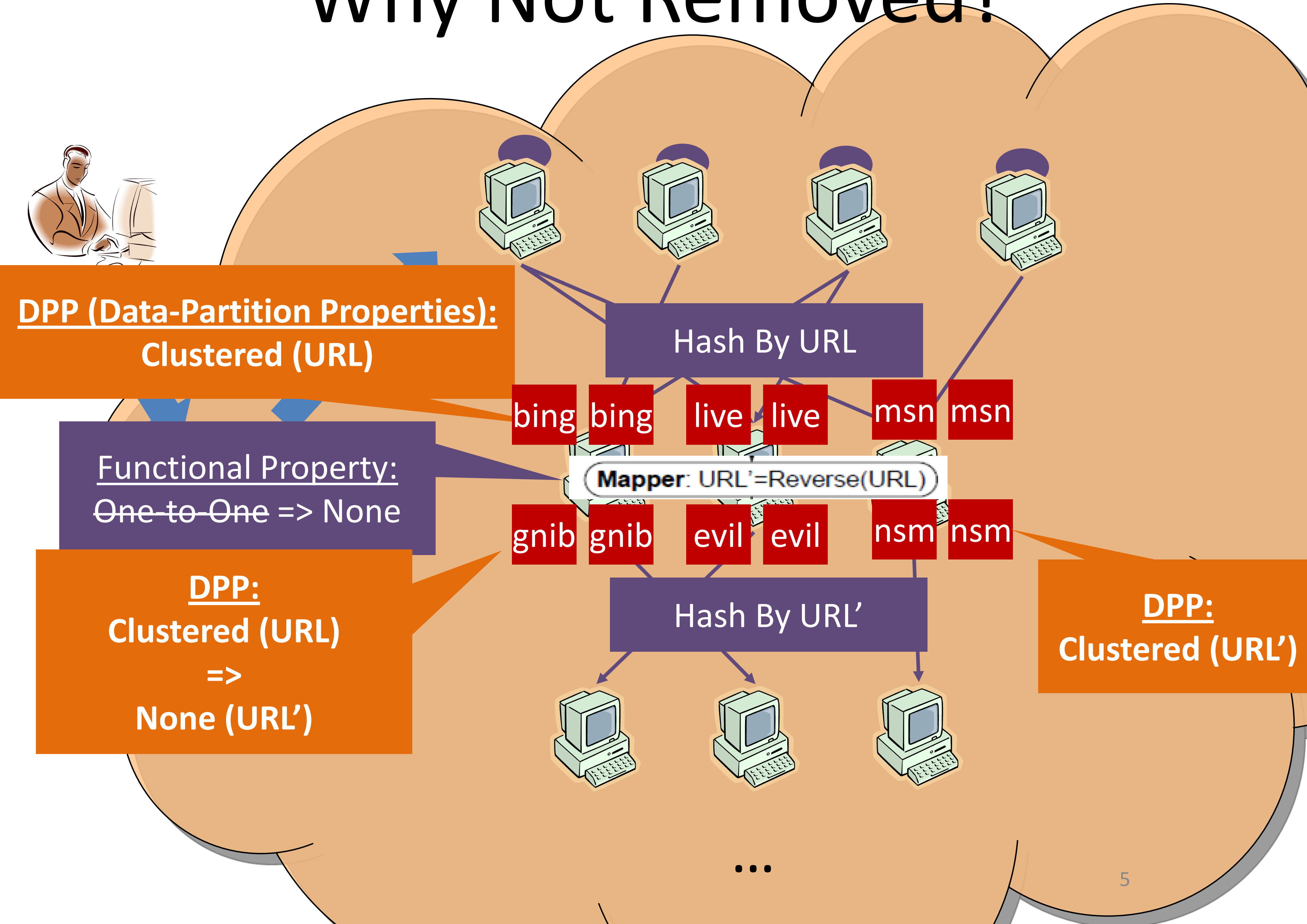
Why Shuffling Stages Necessary?



Unnecessary Shuffling Stages



Why Not Removed?



What is SUDO?

Extract functional properties of the UDF

↓

Reasoning DPP

across UDFs and Shuffling Stages

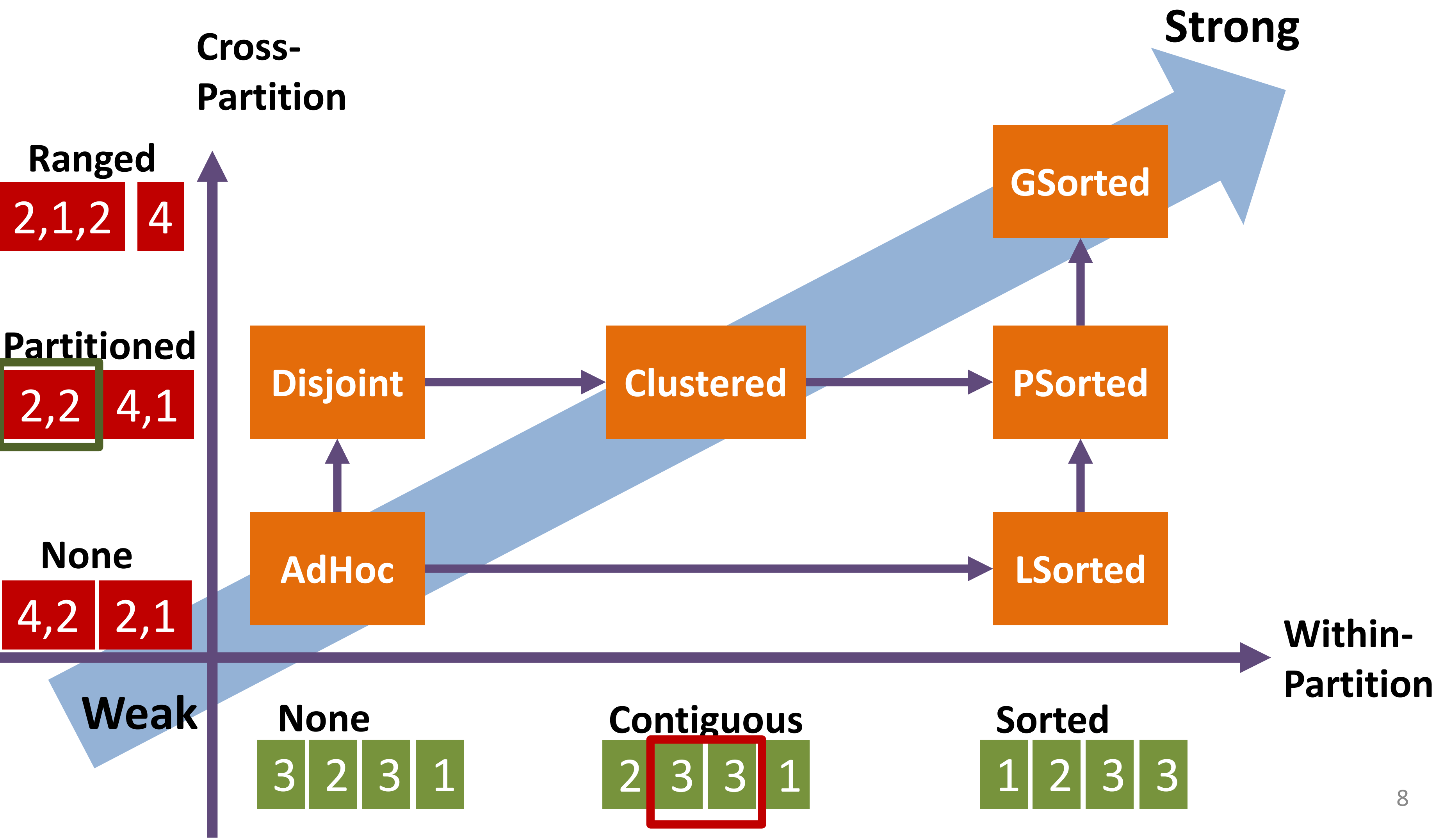
↓

Remove unnecessary shuffling steps

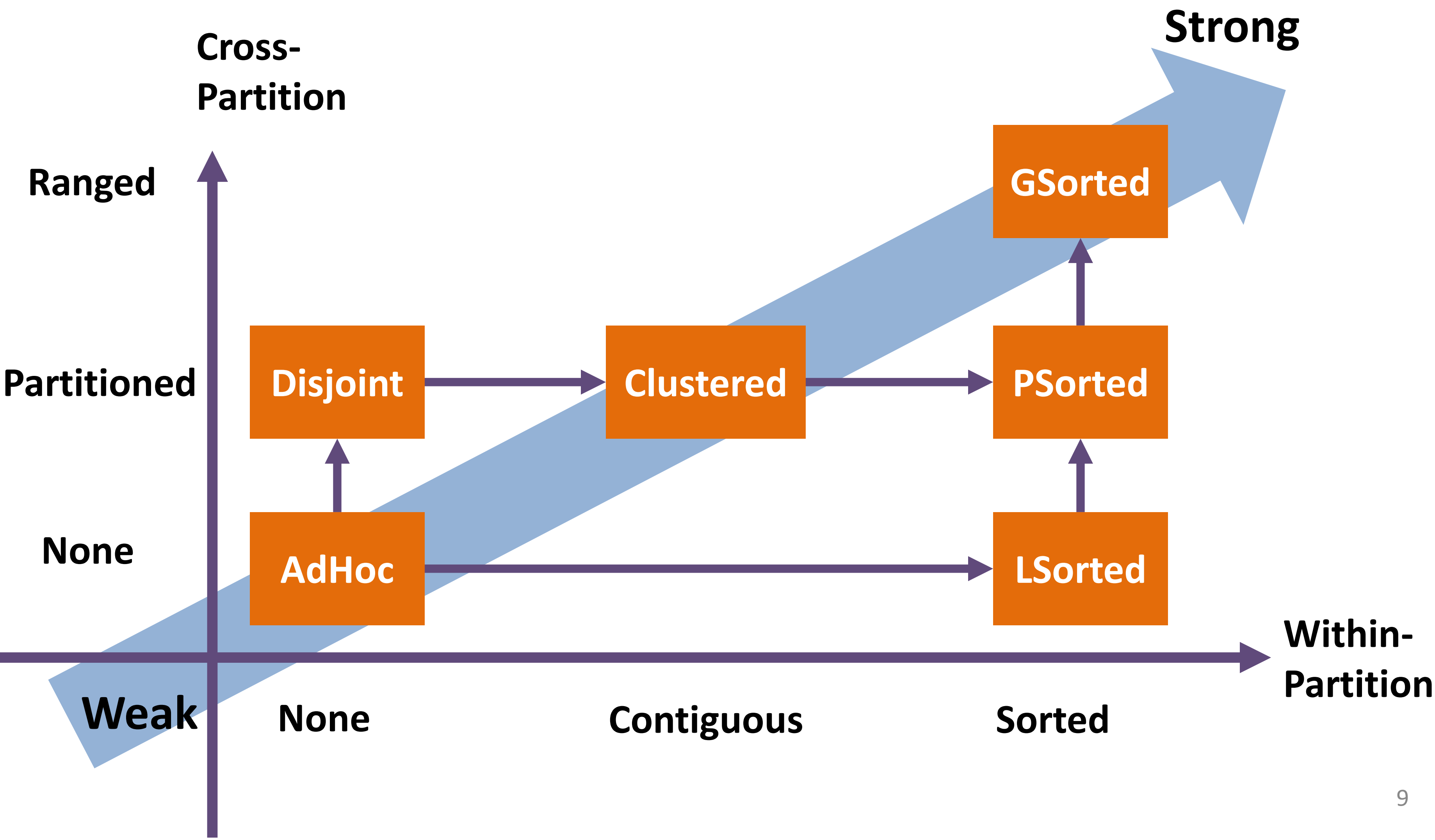
What's next?

- 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?

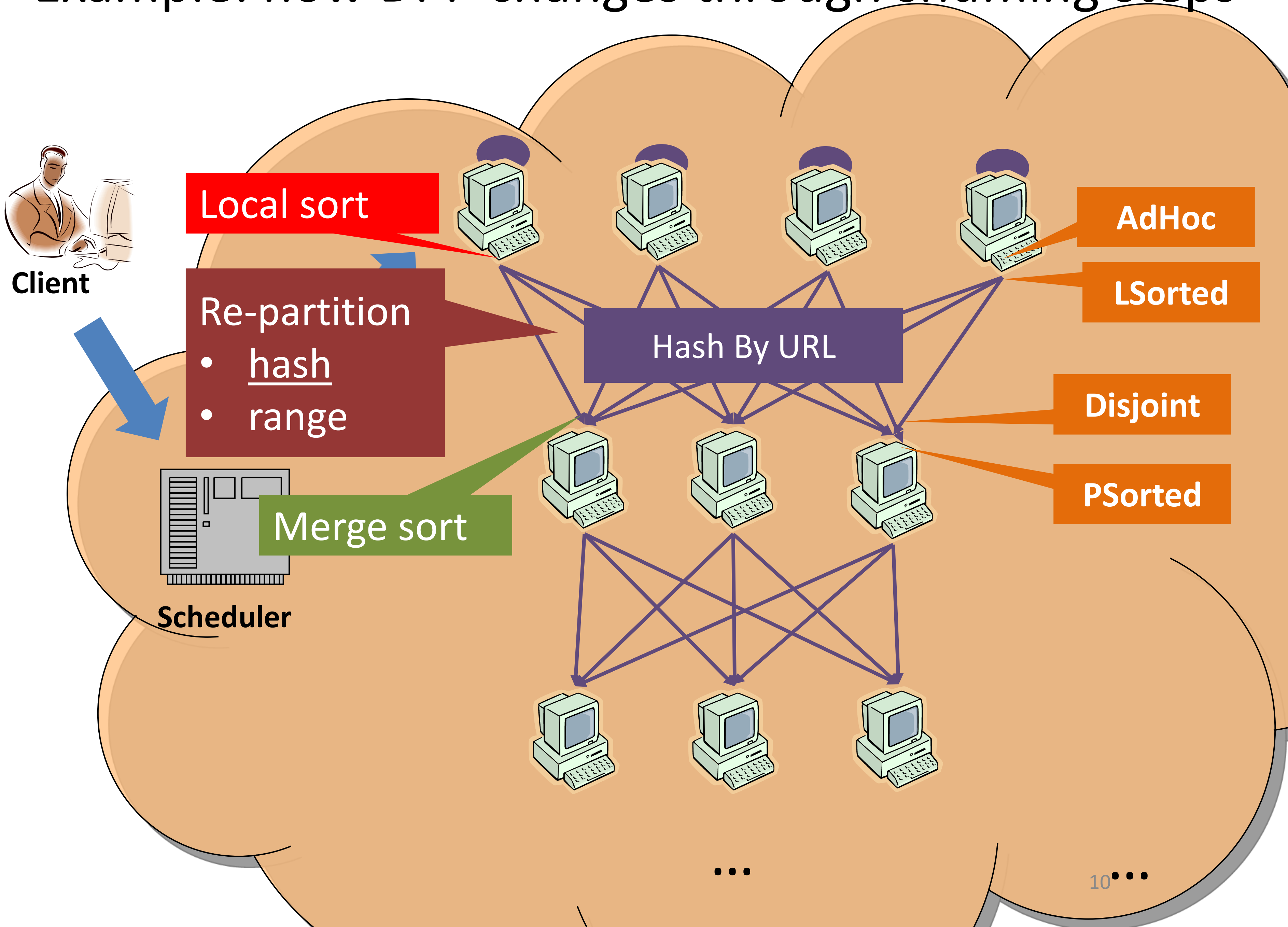
Data-partition Properties (DPP)



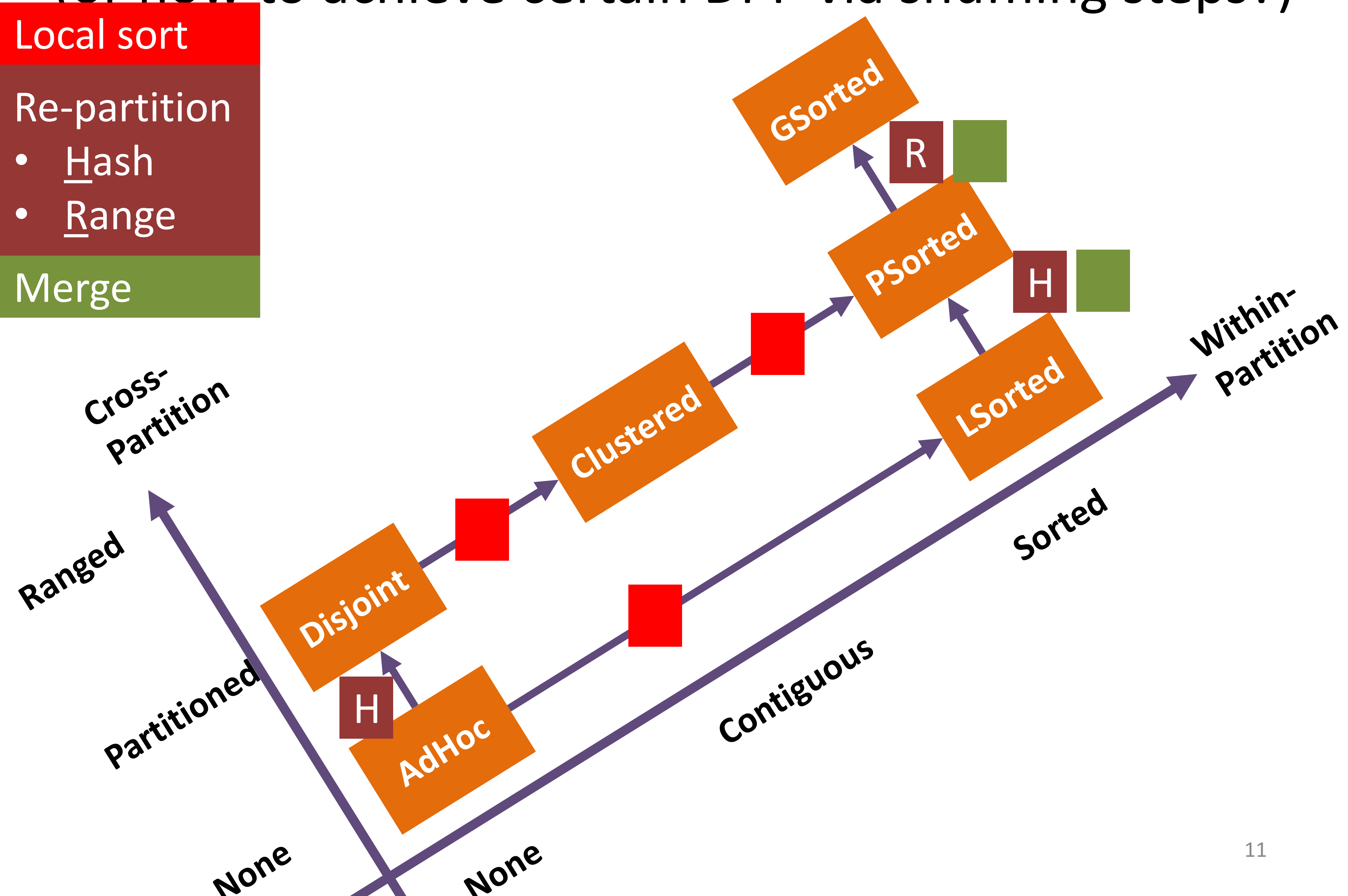
DPP Lattice


Strong
Cross-
Partition
Ranged
Partitioned
None
GSorted
Disjoint
Clustered
PSorted
AdHoc
LSorted
Within-
Partition
Weak
None
Contiguous
Sorted

Example: how DPP changes through shuffling steps



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

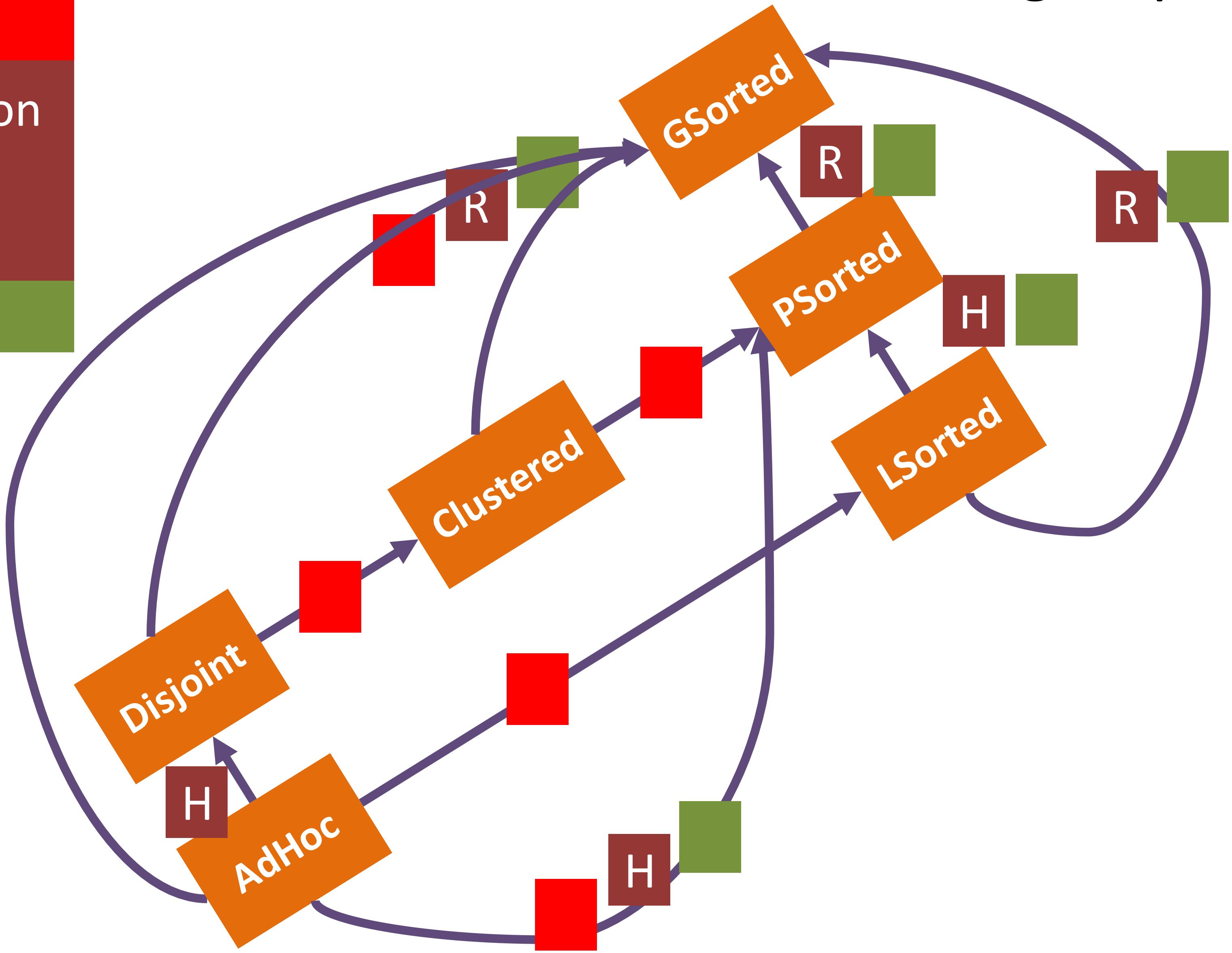


How DPP changes through shuffling steps?
(or how to achieve certain DPP via shuffling steps?)

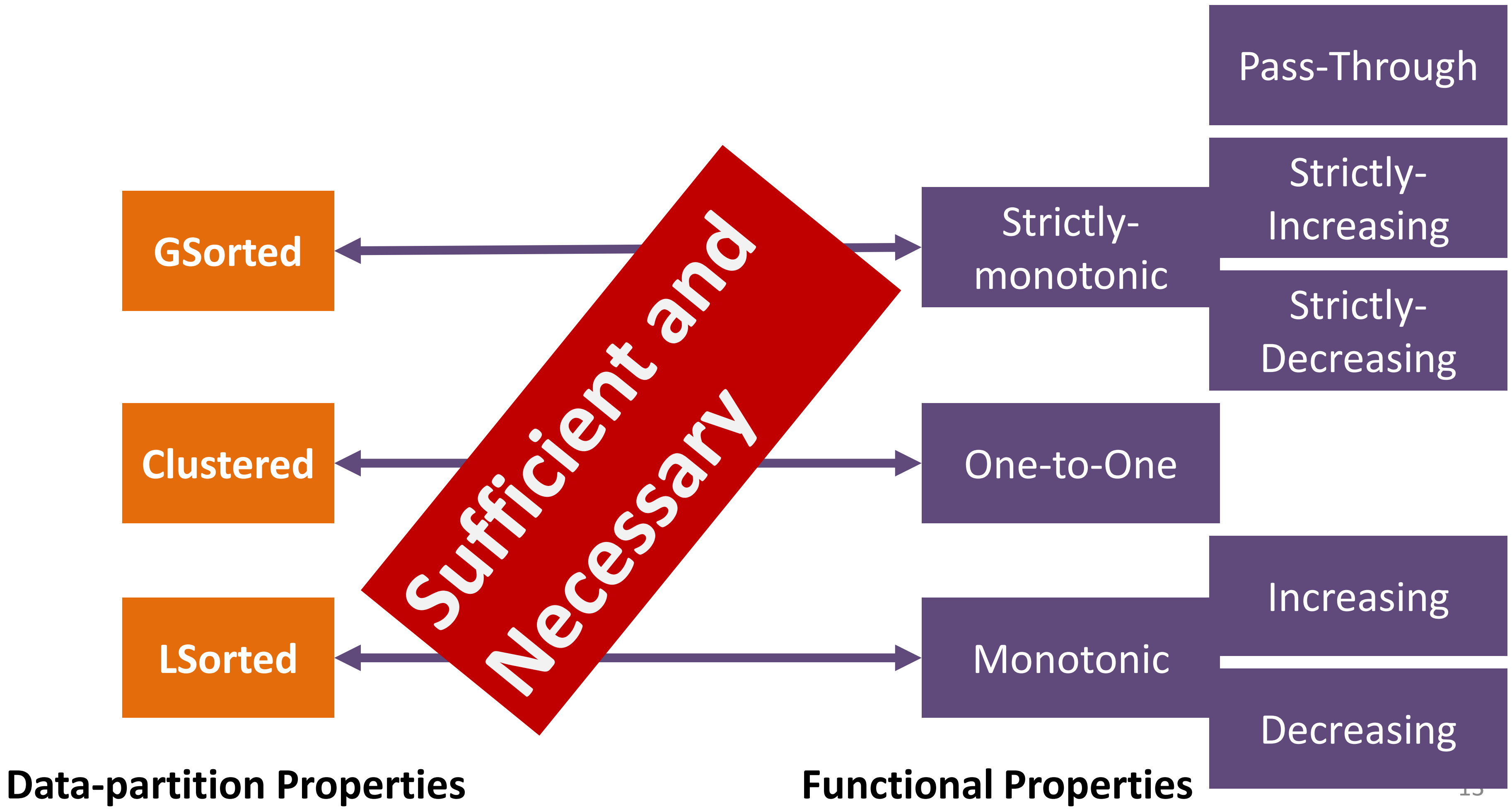
Local sort

Re-partition
- Hash
- Range

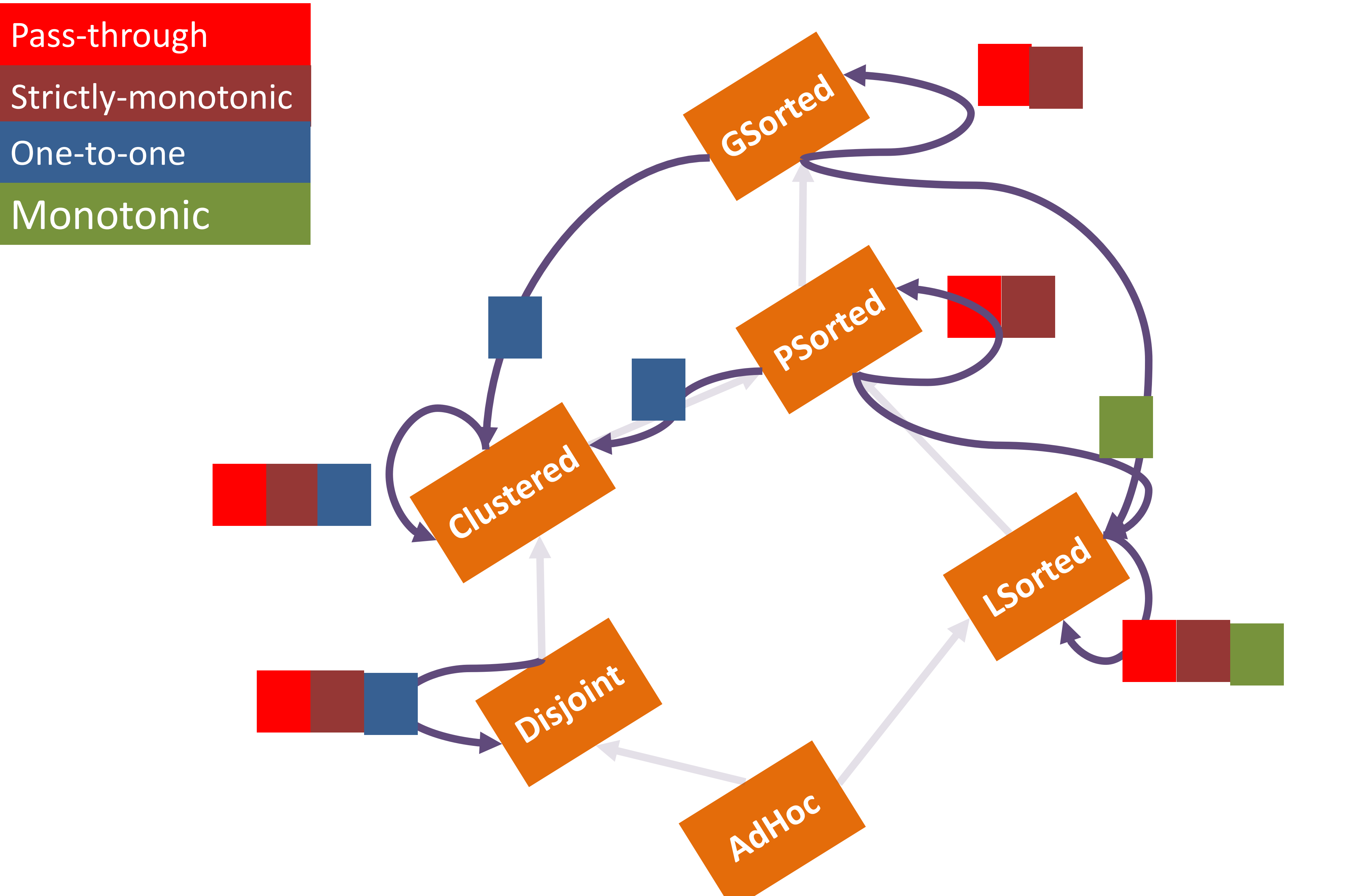
Merge



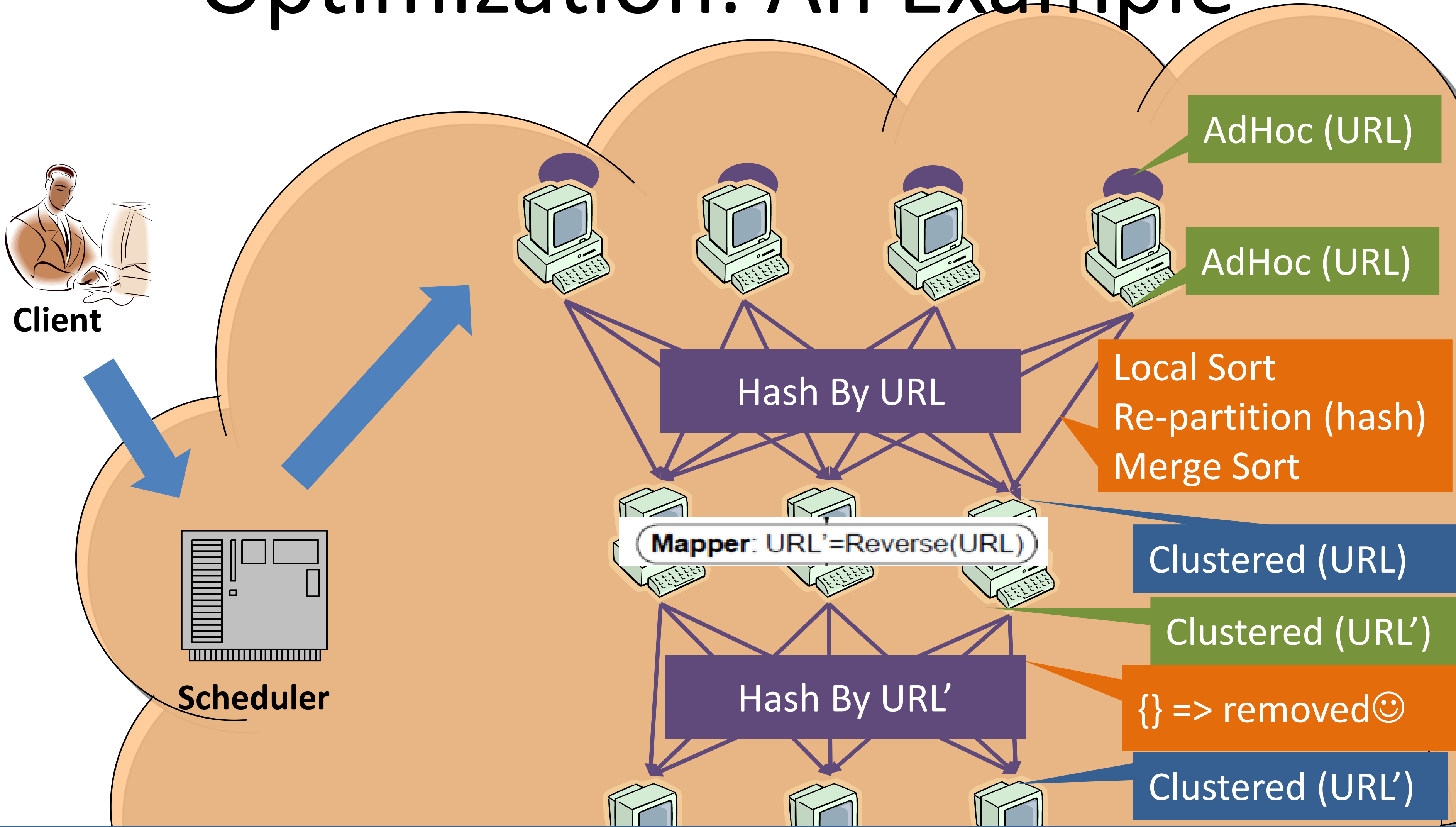
Functional Properties



How DPP changes through UDFs?



Optimization: An Example

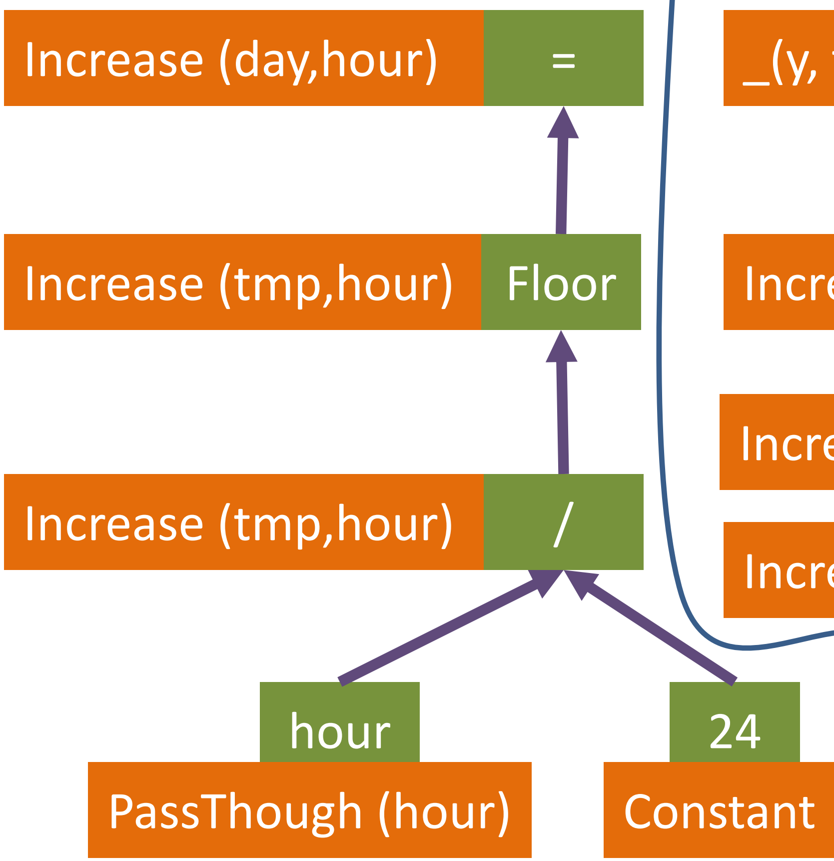


Step 1: collect data-shuffling requirements based on given execution plan

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



UDF: day = Floor(hour/24)

_(y, t) :- ASSIGN y x, _(x, t)

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

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

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

Constant(24)

PassThough(hour)

Deduction Rules

Implementation

- UDF analyzer to extract functional property
  - http://research.microsoft.com/Phoenix to extract AST with 8281 LOC (C#)
  - http://bddbddb.sourceforge.net/ as deduction engine with ~100 Rules
- SUDO rewriter to do optimization
  - ~1316 LOC (C#)

Coverage Study

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

Property	UDF <out-col, in-col> #	Ratio %
Pass-through	1,998,819	84.73
Strictly-increasing	147,820	6.27
Strictly-decreasing	0	0
Increasing	138	0
Decreasing	0	0
One-to-one	1,758	0.08
Others	210,544	8.92
Sum	***2,359,079***	***100***

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

Case	Machine#	Native Shuffling IO (TB)	Native Latency (min)	Shuffling Stage# Change	Shuffling IO Reduction	Latency Reduction
Anchor Data Preprocessing	150	0.9	25	2 => 1	47%	40%
Trend Analysis	1,000	60	230	3 => 1	35%	45%
Query-Anchor Relevance	2,500	15	96	6 => 4	41%	***-27%***

. Shuffling IO reduction is significant

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

Related Work

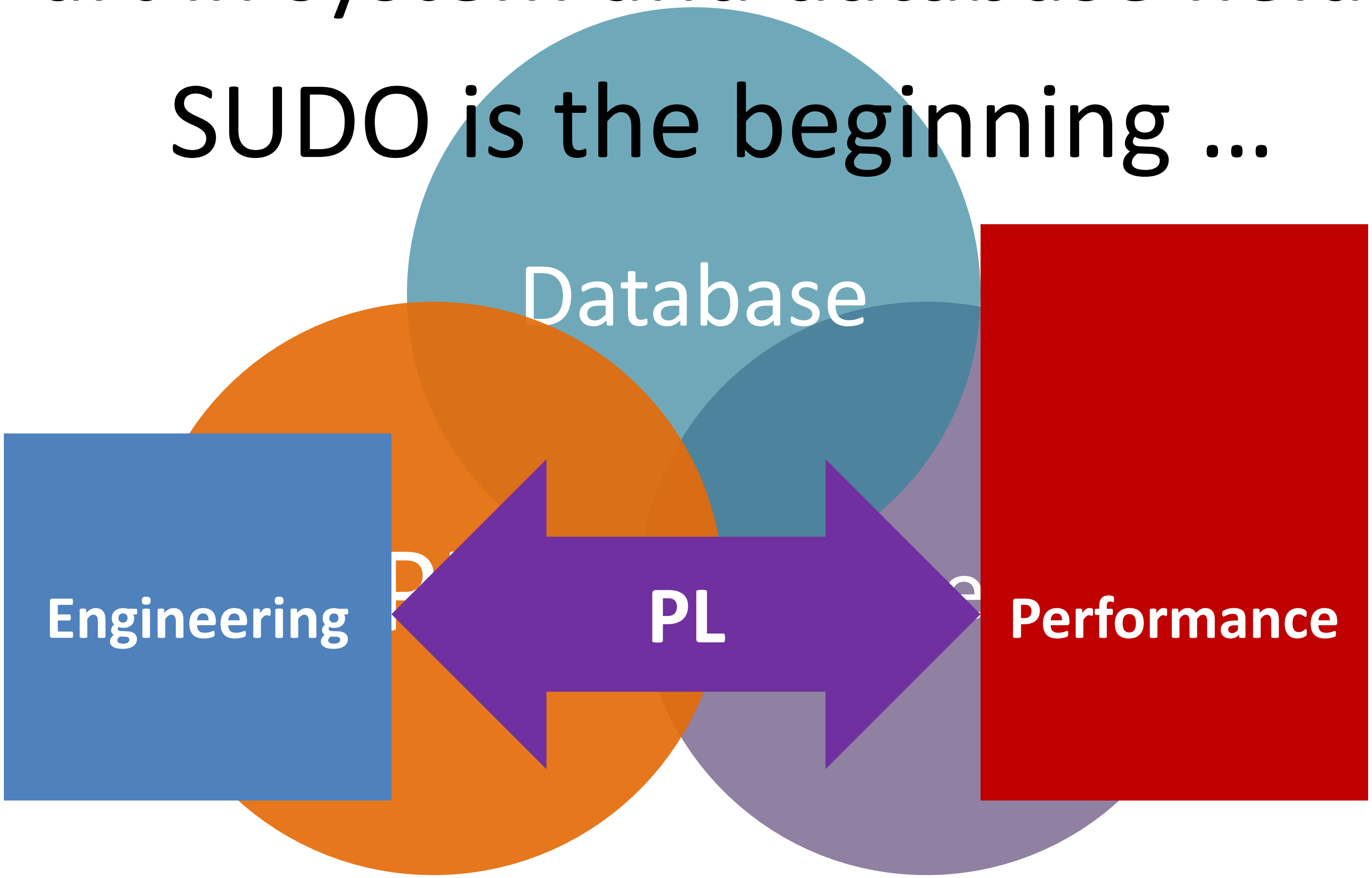
- Data-partition property propagation to reduce shuffling stages
  - Incorporating partitioning and parallel plans into the SCOPE optimizer (ICDE'10)

- 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