Class 6: Adaptive Hybrid Storage Layouts

By Sean Bready

Review of the Paper: **Bridging the Archipelago between Row-Stores and Column-Stores for Hybrid Workloads**

OLTP vs OLAP (vs HTAP)

Online Analytical Processing (OLAP)

Large Aggregate queries

Read only Queries

Complex- multi-step queries

Highly variable ad hoc queries

Which kind of physical storage layout would favor this kind of workload?

Examples of OLAP style Databases (Data Warehouses)

Amazon's RedShift

Biquery

Snowflake

Hive/HDFS/Spark

Online Transactional Processing (OLTP)

High Throughput of CRUD operations

Single object manipulation

Simple Queries

Which kind of physical storage layout would favor this kind of workload?

Examples of OLTP style Databases

Postgres

MySQL

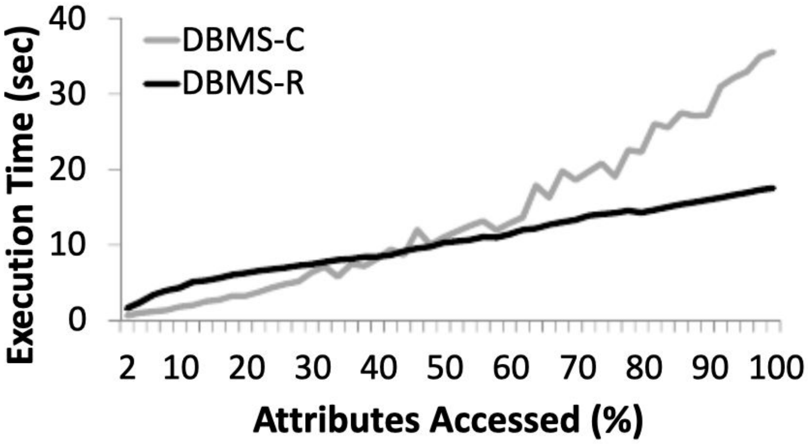
MS SQL Server

Oracle DB

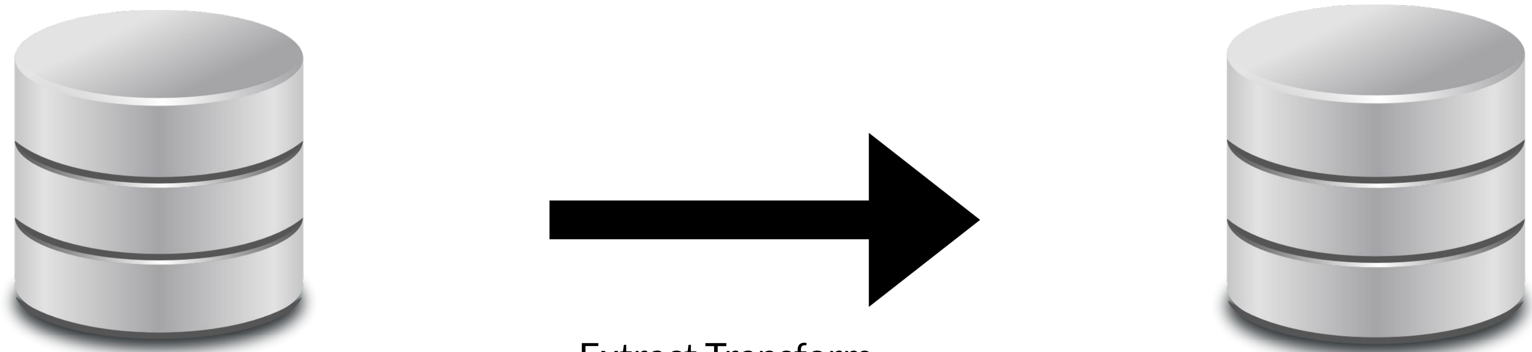
Real world...

One Size does not fit all workloads!

HTAP Systems



Solution: Two databases!



Problems?

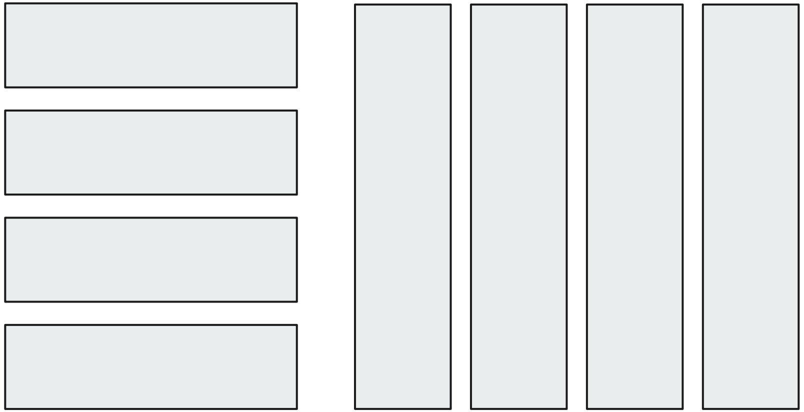
Problems

Double the work!

The OLAP DW is lagging behind OLTP DB

Hybrid Database model! Aka Hybrid Transactional Analytical Processing

One Database



Storing both Rows and Columns!

Ok but how?

The Flexible Storage Model (FSM)

ID	IMAGE-ID	NAME	PRICE	DATA
101	201	ITEM-101	10	DATA-101
102	202	ITEM-102	20	DATA-102
103	203	ITEM-103	30	DATA-103
104	204	ITEM-104	40	DATA-104

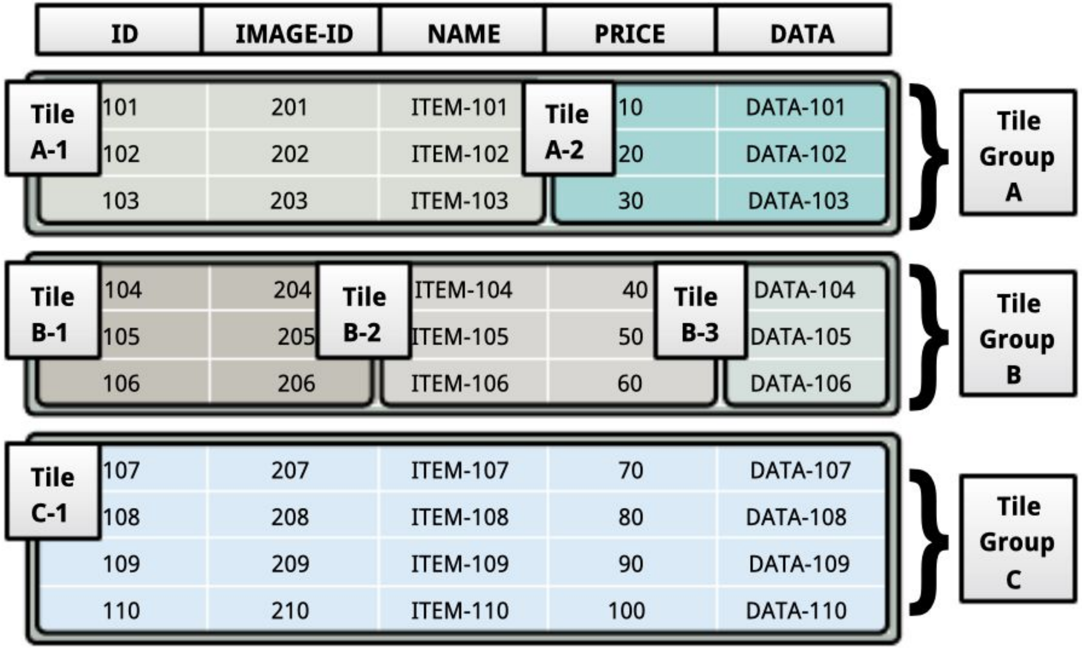
(a) OLTP-oriented N-ary Storage Model (NSM)

ID	IMAGE-ID	NAME	PRICE	DATA
101	201	ITEM-101	10	DATA-101
102	202	ITEM-102	20	DATA-102
103	203	ITEM-103	30	DATA-103
104	204	ITEM-104	40	DATA-104

(b) OLAP-oriented Decomposition Storage Model (DSM)

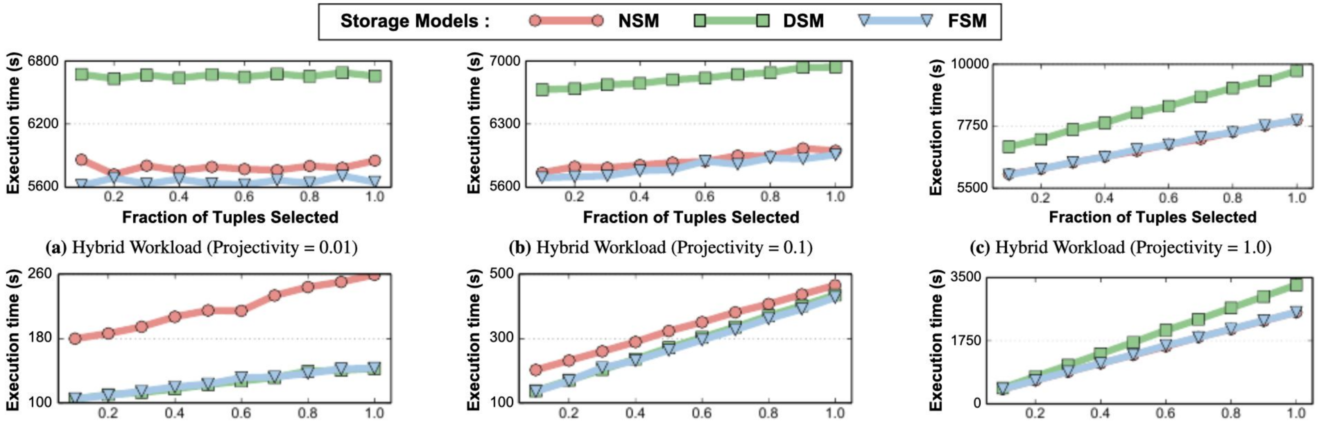
ID	IMAGE-ID	NAME	PRICE	DATA
101	201	ITEM-101	10	DATA-101
102	202	ITEM-102	20	DATA-102
103	203	ITEM-103	30	DATA-103
104	204	ITEM-104	40	DATA-104

(c) HTAP-oriented Flexible Storage Model (FSM)



ID
IMAGE-ID
NAME
PRICE
DATA
Tile A-1
Tile A-2
101 201 ITEM-101 10 DATA-101
102 202 ITEM-102 20 DATA-102
103 203 ITEM-103 30 DATA-103
Tile Group A
Tile B-1
Tile B-2
Tile B-3
104 204 ITEM-104 40 DATA-104
105 205 ITEM-105 50 DATA-105
106 206 ITEM-106 60 DATA-106
Tile Group B
Tile C-1
107 207 ITEM-107 70 DATA-107
108 208 ITEM-108 80 DATA-108
109 209 ITEM-109 90 DATA-109
110 210 ITEM-110 100 DATA-110
Tile Group C

Performance of FSM on Hybrid and Read only workloads



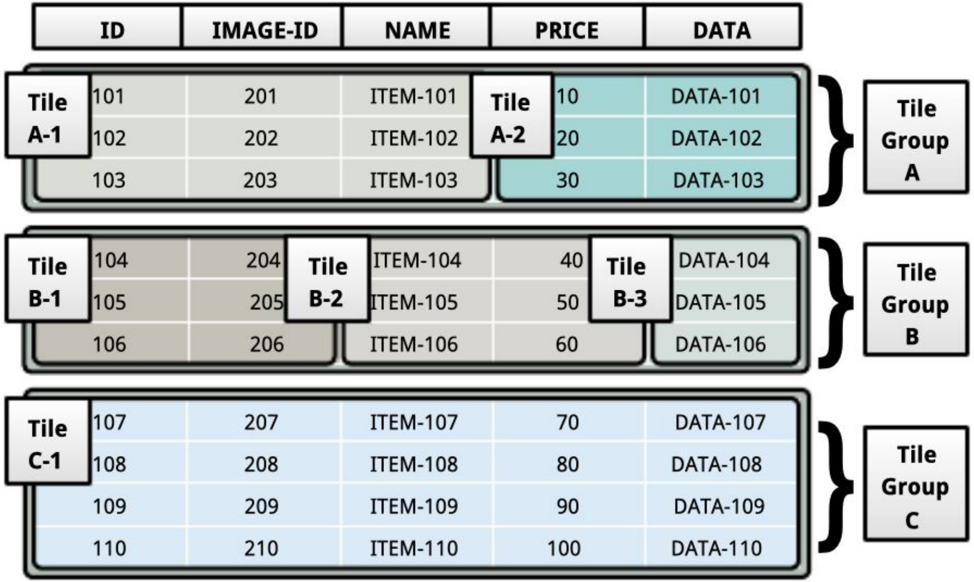
(a) Hybrid Workload (Projectivity = 0.01) (b) Hybrid Workload (Projectivity = 0.1) (c) Hybrid Workload (Projectivity = 1.0)

(d) Read-Only Workload (Projectivity = 0.01) (e) Read-Only Workload (Projectivity = 0.1) (f) Read-Only Workload (Projectivity = 1.0)

Advantage

1 Database; Two Systems!

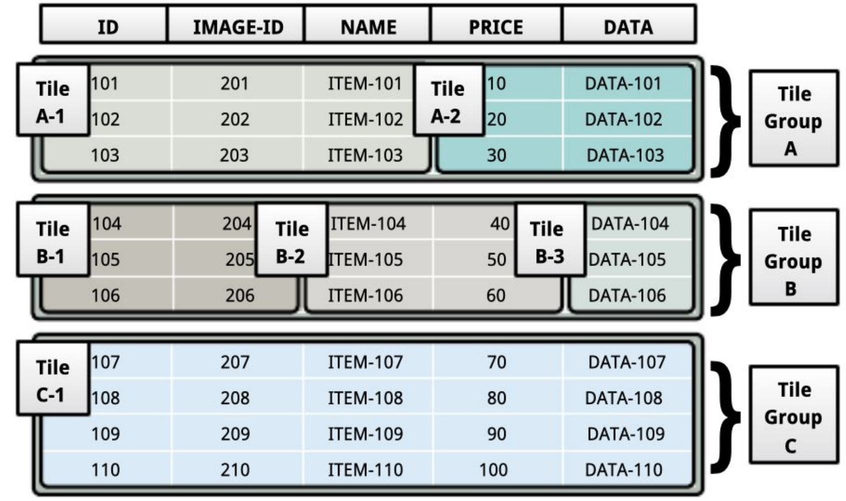
Everyone wins!



Problems?

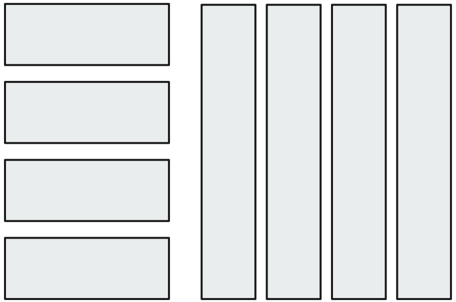
Problems part 2

One Database, several execution engines



Paper proposal: Adaptive HTAP!

One
Database



Storing both Rows and
Columns!

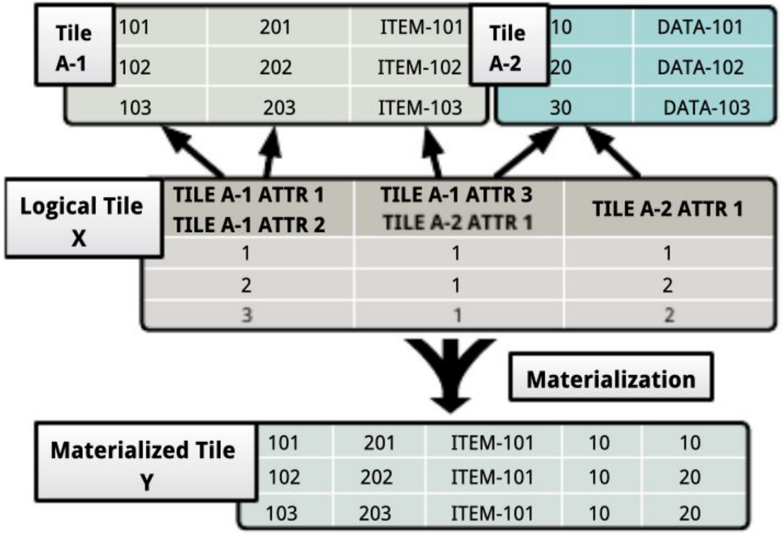
Plus….

Abstraction of Data storage
format

Adaptivity of storage
formats

Now we are getting
somewhere! But How?

Abstraction: Tile Based Architecture



Operators

Bridge Operators Convert Tiles to Physical or Logical

Mutators- Mutate Logical tiles

Metadata Operators - Mutate Logical tiles meta data

Pipeline Breakers - Force completion of themselves before query plan parent operation can begin.

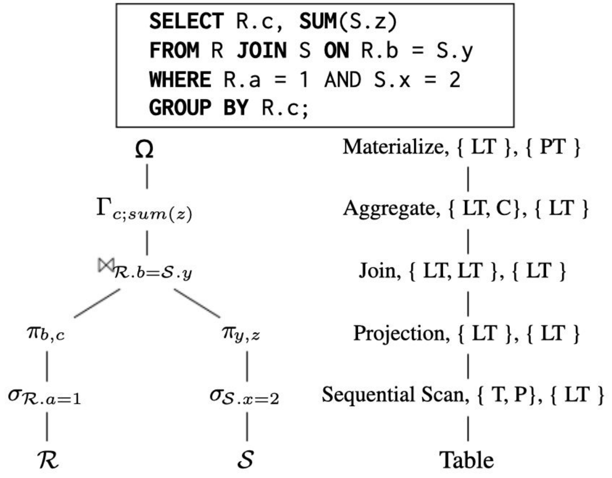


Figure 5: Sample SQL query and the associated plan tree for illustrating the operators of the logical tile algebra. We describe the name, inputs, and output of each operator in the tree. We denote the logical tile by LT, the physical tile by PT, the table by T, the attributes by C, and the predicate by P.

Benefits of the Tile Architecture

Vectorized Processing

Flexible Materialization

Caching Behavior

Adaptivity:Layout Reorganization

Query monitoring

Partitioning Algorithm

Background or on the spot reorganization

Query monitoring

Lightweight

Per Table

To optimize layout

- Where clause and Select Clause attributes

Why?

Partitioning Layouts

No good Algorithm...

Solution:

Clustering

Drifting

Clustering

Representative Query : r_j

New Query: Qi

Set of attributes accessed by a Query on a table : SetT (Qi)

Distance formula: Length (Set(Q1) UNION Set(Rj) - Set(Q1) INTERSECTION Set(Rj)) / Length (Set of attributes in T)

Drifting

Cluster drift with every new addition

Cj = representative Query (updated value)

w= forgetting factor of old tuples (Tunable)

s = number of query samples in the cluster

Co = representative Query (initial value)

Qi = current query being added to cluster

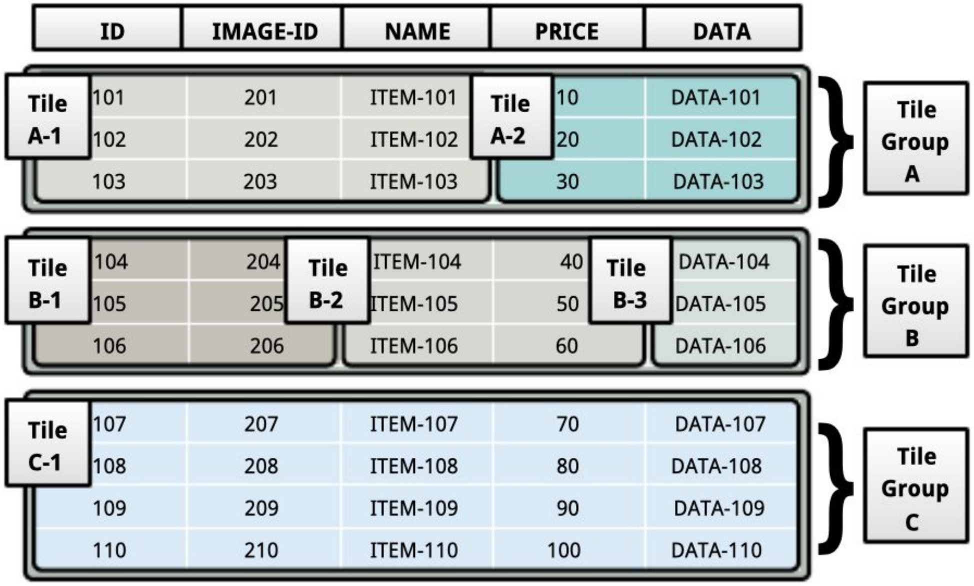
$$c_j = (1 - w)^s c_0 + w \sum_{i=1}^{s} (1 - w)^{s-i} Q_i$$

Reorganization!

Sort the Clusters by Query Plan cost

Highest Query plan cost, First served

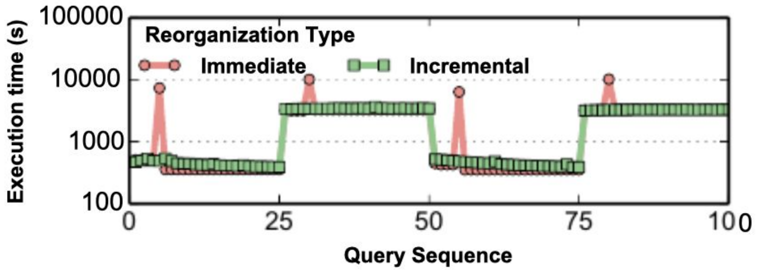
Workload changes so the Clusters change so the physical tiles change



When to Reorganize? pt.2

Amortized vs on the spot reorganization

Only cold tuples



Experiments

System

Adapt Benchmark

Results

System

DBMS: Peloton (with paper's additions)

Server: Dual Socket Intel Xeon E5-4620; Running Ubuntu 14.04 (64-bit)

CPUs in Sockets: Eight 2.6GHz cores

RAM: 128 GB of DRAM

L3 cache: 20 MB

ADAPT Benchmark

Novel Benchmark

Benchmark is made up of data, organized into tables, and workloads, represented by SQL

Adapt DB

Small table (50 attributes)

Wide table(500 attributes)

10m tuples each (200B, 2KB)

ADAPT Workload

Q_1: INSERT INTO R VALUES $(a_0, a_1, \ldots, a_p)$

Q_2: SELECT $a_1, a_2, \ldots, a_k$ FROM R WHERE $a_0 < \delta$

Q_3: SELECT MAX(a_1), ..., MAX(a_k) FROM R WHERE $a_0 < \delta$

Q_4: SELECT $a_1 + a_2 + \ldots + a_k$ FROM R WHERE $a_0 < \delta$

Q_5: SELECT X.a_1, ..., X.a_k, Y.a_1, ..., Y.a_k
FROM R AS X, R AS Y WHERE X.a_i < Y.a_j

Results (Projections Sanity test)

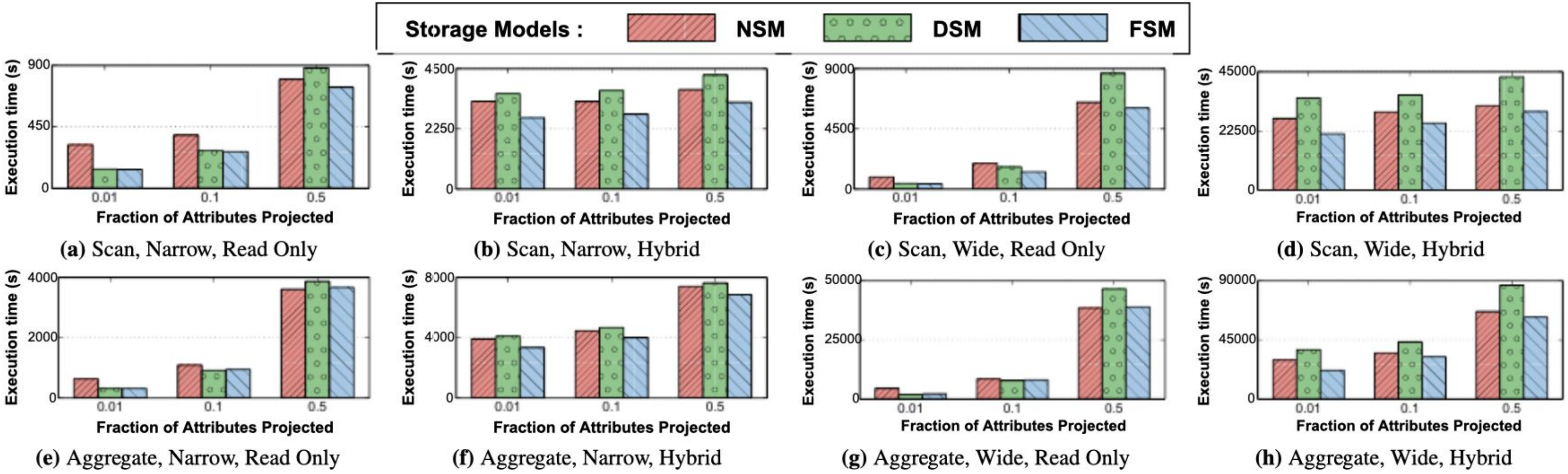


Figure 7: Projectivity Measurements– The impact of the storage layout on the query processing time under different projectivity settings. The execution engine runs the workload with different underlying storage managers on both the narrow and the wide table.

Results (Selectivity Sanity test)

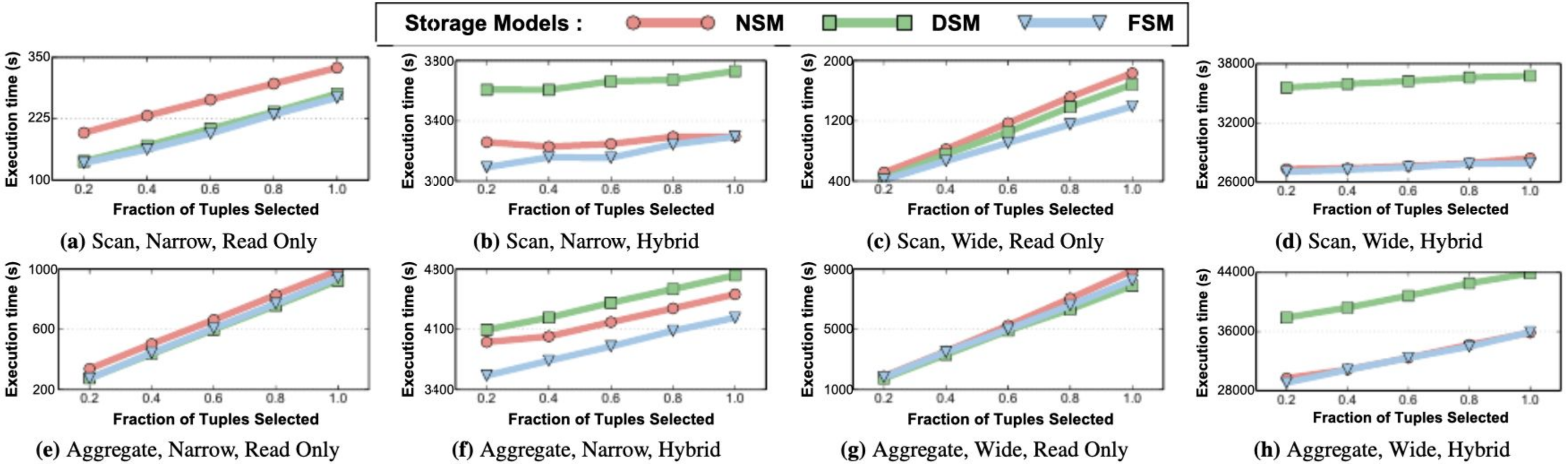
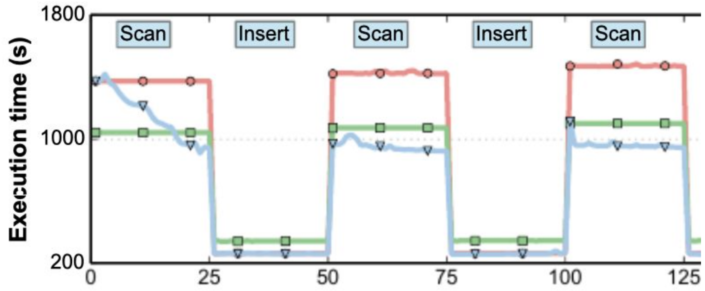


Figure 8: Selectivity Measurements – The impact of the storage layout on the query processing time under different selectivity settings. The execution engine runs the workload with different underlying storage managers on both the narrow and the wide table.

Results (Adaptivity Sanit



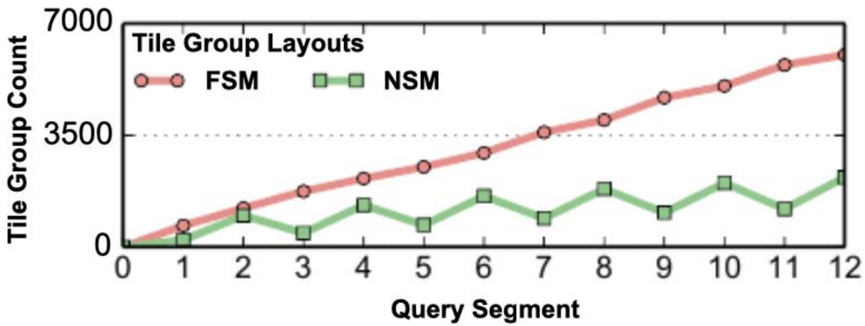


Figure 10: Layout Distribution – The variation in the distribution of tile group layouts of the table over time due to the data reorganization process.

Figure 9: Workload-Aware Adaptation – The impact of tile group layout adaption on the query processing performance in an evolving workload mixture from the ADAPT benchmark. This experiment also examines the behavior of different storage managers while serving different query types in the workload.

Conclusion; Why does this paper even matter?

Further optimized the way HTAP systems organize data

1.Added an abstraction to the Physical Storage

2.Added an subroutine reorganizes layout based on current workload

Much better Performance with a focus on extendability.

Further Work needed...

The tuning factor…

Research isn't settled on best Partitioning Algorithm

References

[1] *Joy Arulraj, Andrew Pavlo, and Prashanth Menon. 2016. Bridging the Archipelago between Row-Stores and Column-Stores for Hybrid Workloads. In Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16). Association for Computing Machinery, New York, NY, USA, 583–598. DOI:https://doi.org/10.1145/2882903.2915231*

[2] *Ioannis Alagiannis, Stratos Idreos, and Anastasia Ailamaki. 2014. H2O: a hands-free adaptive store. In Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14). Association for Computing Machinery, New York, NY, USA, 1103–1114. DOI:https://doi.org/10.1145/2588555.2610502*

Authors

Andrew Pavlo

Joy Arulraj

Prashanth Menon

Appendix A: Partitioning Algorithm

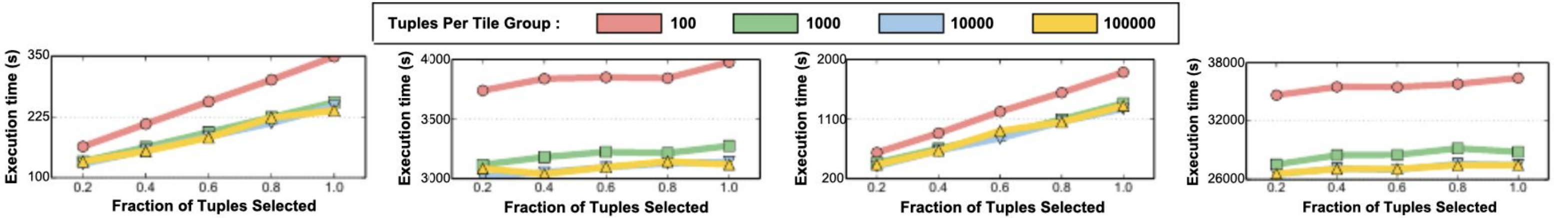
Calculating the optimized partition for a workload...

Greedy Algorithm must be used.

Algorithm 1 Vertical Partitioning Algorithm

Require: recent queries Q, table T, number of representative queries k
  function UPDATE-LAYOUT(Q, T, k)
    # Stage I : Clustering algorithm
    for all queries q appearing in Q **do**
      for all representative queries r_j associated with T **do**
        if r_j is closest to q **then**
          $r_j \leftarrow r_j + \text{w} \times (\text{q} - r_j)$
        end if
      end for
    end for
    # Stage II : Greedy algorithm
    Generate layout for T using r
  end function

Appendix B: Results (Horizontal Fragmentation)



(a) Scan, Narrow, Read Only (b) Scan, Narrow, Hybrid (c) Scan, Wide, Read Only (d) Scan, Wide, Hybrid

Figure 11: Horizontal Fragmentation – The impact of horizontal fragmentation on the DBMS's performance. We execute the read-only and hybrid workloads comprising of scan queries on the tables in the ADAPT benchmark under different fragmentation settings.

Appendix C: Multiversion Concurrency Control

The control metadata

Pipe Breakers and Metadata operations

Mutators

Bridge Operators

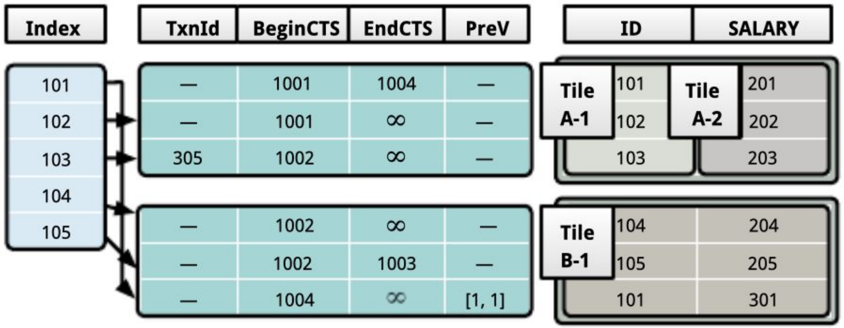


Figure 6: Concurrency Control – Versioning information that the DBMS records in the tile groups for MVCC.