H₂O: A Hands-free Adaptive Store

Ioannis Alagiannis*  Stratos Idreos‡  Anastasia Ailamaki*

*École Polytechnique Fédérale de Lausanne
‡Harvard University

SIGMOD 2014
One size does not fit all*

row-wise (NSM)

- row 1
- row 2
- row 3
- row 4
- row 5
- ...

tuple-at-a-time

hybrids‡

- a
- b
- c
- d
- e
- ...

column-wise (DSM)

- a
- b
- c
- d
- e
- ...

column-at-a-time

Once decided, the data layout will forever determine the query performance

- query performance
- data layout

depends on

determines
Fixed storage layout problem

Tuples: 50m Attrs: 250, memory resident
Queries: sel-proj-aggr (sel. 70%)

Need for dynamic data layouts
**Hands-free adaptive store**

**Hybrid storage layouts**

+ **Hybrid query plans**

+ **Ad-hoc Operators**

\[ \text{Hybrid storage layouts} + \text{Hybrid query plans} + \text{Ad-hoc Operators} = \text{H}_2\text{O: Adaptive Hybrid System} \]

- support different organizations
- combine different execution strategies
- generate code on-the-fly

**Dynamic layouts, plans, operators**
How $\text{H}_2\text{O}$ adapts to workloads

- Monitoring phase
- Collect statistics regarding the usage and access patterns

queries

Time
How $\text{H}_2\text{O}$ adapts to workloads

- Monitoring phase
  - Collect statistics regarding the usage and access patterns
- Time trigger
  - Examine alternative data layouts
- Adaptation mechanism
- Queries

Diagram: Queries lead to a trigger, which leads to a monitoring phase that triggers an adaptation mechanism. The adaptation mechanism collects statistics and examines alternative data layouts.
How $H_2O$ adapts to workloads

queries

Monitoring phase

Adaptation mechanism

Collect statistics regarding the usage and access patterns

Examine alternative data layouts

Lazy data layout generation

Combine data re-organization with query processing

Time
Data layouts & execution strategies

... both influence performance

- Upfront preparation is not possible
  - Numerous potential combinations
- Generic operators
  - High interpretation overhead
  - Data type and layout knowledge is lost

Just-in-time compilation to generate specialized code*

*Neumann [VLDB2011], Krikellas [ICDE2010]
Creating operators on-the-fly

```
select a + b + c from R
where d < v1 and e > v2
```

for each tuple

```
if(d < v1 && e > v2)
result[i] = (a + b + c);
```

Create layout-aware access operators

Combine layouts in the same operator
H₂O architecture

- Cost model:
  \[ q(L) = \sum_{i=1}^{\|L\|} \max(cost_{i}^{IO}, cost_{i}^{CPU}) \]

- Select access plan
- Generate code on-the-fly
- Propose alternative layouts
- Reorganize data
- Maintain different data layouts

- Queries
- Workload statistics
- Adaptation mechanism
- Data layout manager
- Query engine
- Operator generator
Adapting to workload changes

Tuples: 100m Attrs: 150 integers (in memory)
Queries access 10-30 attrs

- **Row-store**
- **Column-store**
- **Best-Hybrid**
- **H2O**

**Adapting phase**
- Assume column-major layout as input for H2O
- Adaptation phase starts
- The adaptation window is 20 queries
Adapting to workload changes

Tuples: 100m Attrs: 150 integers (in memory)
Queries access 10-30 attrs

Query Sequence

Query Response Time (sec)

Row-store  Column-store  Best-Hybrid  H2O

lazy data reorganization  new layouts

Adapting to workload changes

Tuples: 100m Attrs: 150 integers (in memory)
Queries access 10-30 attrs

Query Sequence

Query Response Time (sec)

Row-store  Column-store  Best-Hybrid  H2O

lazy data reorganization  new layouts
Adapting to workload changes

Tuples: 100m Attrs: 150 integers (in memory)
Queries access 10-30 attrs

H₂O outperforms static approaches
Conclusions

- Multiple data layouts can be beneficial for analytical ad-hoc workloads

- \( H_2O \) continuously adapts to the workload by
  - Generating data layouts on-the-fly
  - Choosing the appropriate execution strategy
  - Dynamically creating layout-aware operators
  - Without a priori workload knowledge

Thanks!