Proteus:
Autonomous Adaptive Storage
for Mixed Workloads

Michael Abebe  mtabebe@uwaterloo.ca
Horatiu Lazu
Khuzaima Daudjee
June 2022
tiny.cc/proteus
SELECT book, SUM(qnt) GROUP BY book

INSERT ( 104, LoTR, John, 2, $21 )
Mixed Workloads and Storage

Row Layout

Updates (OLTP)

INSERT
(104, LoTR, John, 2, $21)
Mixed Workloads and Storage

SELECT book, SUM(qnt) GROUP BY book
Performance Trade-Off

OLTP

OLAP

OLTP

OLAP
Performance Trade-Off
Keep Data in Both Layouts

Row Layout

Columnar Layout

Updates

Analytics
Keep Data in Both Layouts

Row Layout

Columnar Layout

Missing data
Keep Data in Both Layouts

Row Layout

Columnar Layout

Slow Updates
Keep Data in Both Layouts

Row Layout

Columnar Layout

Increased Storage Costs

Missing recent data
Which **data** in which layout?

**Row Layout**

**Columnar Layout**

```sql
SELECT book, SUM(qnt)
GROUP BY book
```
Which **data** in which layout?

<table>
<thead>
<tr>
<th>Row Layout</th>
<th>Columnar Layout</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SELECT**

book, **SUM**( qnt )

**GROUP BY** book
Which **data** in which layout?

Row Layout

Columnar Layout
Which data in which layout?

Row Layout

Columnar Layout
Which **data** in which layout?

**Row Layout**

**Columnar Layout**
Which **data** in which layout?

Row Layout

Columnar Layout

Need for autonomous adaptation
Proteus – Best of Both

![Graph showing performance comparison between Proteus, Row, and Col]

- **Proteus**
- **Row**
- **Col**
Proteus Goals

Selectively store data in different layouts

Leverage layout specific optimizations

System makes decisions based on workload

Scale-Out Distributed System
## Proteus Decisions

<table>
<thead>
<tr>
<th>Order</th>
<th>Book</th>
<th>User</th>
<th>Qnt</th>
<th>$</th>
</tr>
</thead>
</table>

### Storage layout
- Master/replica(s)

### Txn execution

### How to partition?

### When & what to change

---

Data partition
Proteus Architecture

Adaptive Storage Advisor (ASA)

Metadata
Cached Decisions
Execution Plan
Workload Model

Data Site 1
Storage aware ops.
Master copy (row)

Data Site 2
Sort & compressed
Replica (column)

Propagate updates
Executetxn & storage adaptations

Observed costs

Master copy (row)
Replica (column)
Efficient Execution

Adaptive Storage Advisor (ASA)

Metadata
Cached
Decisions
Execution
Plan
Workload
Model

Data Site 1
Storage aware ops.
Master copy (row)

Data Site 2
Sort & compressed
Replica (column)

Execute txn & storage adaptations
Propagate updates

Observed costs
## Storage-Aware Operators

<table>
<thead>
<tr>
<th>Order</th>
<th>Book</th>
<th>User</th>
<th>Qnt</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>HP</td>
<td>Alice</td>
<td>7</td>
<td>$15</td>
</tr>
<tr>
<td>101</td>
<td>Drac</td>
<td>Ted</td>
<td>90</td>
<td>$3</td>
</tr>
<tr>
<td>102</td>
<td>SQL</td>
<td>Geoff</td>
<td>1</td>
<td>$100</td>
</tr>
<tr>
<td>103</td>
<td>HP</td>
<td>Jean</td>
<td>3</td>
<td>$15</td>
</tr>
</tbody>
</table>

### SQL Command

```sql
SELECT $ 
WHERE 
book == ‘HP’ 
or QNT > 50
```
### Storage-Aware Operators

Use **column-specific** operators

Merge bitmaps for predicates, extract data

```
<table>
<thead>
<tr>
<th>Book</th>
<th>Qnt</th>
<th>Book == HP</th>
<th>Qnt &gt; 50</th>
<th>Merged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drac</td>
<td>90</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>HP</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>HP</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SQL</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

```
SELECT $ WHERE
book == 'HP'
or QNT > 50
```
Storage-Aware Operators

<table>
<thead>
<tr>
<th>Order</th>
<th>Book</th>
<th>User</th>
<th>Qnt</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>HP</td>
<td>Alice</td>
<td>7</td>
<td>$15</td>
</tr>
<tr>
<td>101</td>
<td>Drac</td>
<td>Ted</td>
<td>90</td>
<td>$3</td>
</tr>
<tr>
<td>102</td>
<td>SQL</td>
<td>Geoff</td>
<td>1</td>
<td>$100</td>
</tr>
<tr>
<td>103</td>
<td>HP</td>
<td>Jean</td>
<td>3</td>
<td>$15</td>
</tr>
</tbody>
</table>

```sql
SELECT book, SUM(qnt) GROUP BY book
```
## Storage-Aware Operators

### SQL Query

```sql
SELECT book, SUM(qnt) 
GROUP BY book
```

### Table

<table>
<thead>
<tr>
<th>Order</th>
<th>Book</th>
<th>User</th>
<th>Qnt</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>HP</td>
<td>Alice</td>
<td>7</td>
<td>$15</td>
</tr>
<tr>
<td>101</td>
<td>Drac</td>
<td>Ted</td>
<td>90</td>
<td>$3</td>
</tr>
<tr>
<td>102</td>
<td>SQL</td>
<td>Geoff</td>
<td>1</td>
<td>$100</td>
</tr>
<tr>
<td>103</td>
<td>HP</td>
<td>Jean</td>
<td>3</td>
<td>$15</td>
</tr>
</tbody>
</table>

### Results Table

<table>
<thead>
<tr>
<th>Book</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Storage-Aware Operators

<table>
<thead>
<tr>
<th>Order</th>
<th>Book</th>
<th>User</th>
<th>Qnt</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>HP</td>
<td>Alice</td>
<td>7</td>
<td>$15</td>
</tr>
<tr>
<td>101</td>
<td>Drac</td>
<td>Ted</td>
<td>90</td>
<td>$3</td>
</tr>
<tr>
<td>102</td>
<td>SQL</td>
<td>Geoff</td>
<td>1</td>
<td>$100</td>
</tr>
<tr>
<td>103</td>
<td>HP</td>
<td>Jean</td>
<td>3</td>
<td>$15</td>
</tr>
</tbody>
</table>

```sql
SELECT book, SUM(qnt) 
GROUP BY book
```

<table>
<thead>
<tr>
<th>Book</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP</td>
<td>7</td>
</tr>
<tr>
<td>Drac</td>
<td>90</td>
</tr>
</tbody>
</table>
## Storage-Aware Operators

### SQL Queries

#### SQL Query 1

```sql
SELECT book, SUM(qnt)
FROM orders
GROUP BY book
```

#### SQL Query 2

```sql
SELECT book, Qnt
FROM order_book
```

### Order Book Table

<table>
<thead>
<tr>
<th>Order</th>
<th>Book</th>
<th>User</th>
<th>Qnt</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>HP</td>
<td>Alice</td>
<td>7</td>
<td>$15</td>
</tr>
<tr>
<td>101</td>
<td>Drac</td>
<td>Ted</td>
<td>90</td>
<td>$3</td>
</tr>
<tr>
<td>102</td>
<td>SQL</td>
<td>Geoff</td>
<td>1</td>
<td>$100</td>
</tr>
<tr>
<td>103</td>
<td>HP</td>
<td>Jean</td>
<td>3</td>
<td>$15</td>
</tr>
</tbody>
</table>
### Storage-Aware Operators

<table>
<thead>
<tr>
<th>Order</th>
<th>Book</th>
<th>User</th>
<th>Qnt</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>HP</td>
<td>Alice</td>
<td>7</td>
<td>$15</td>
</tr>
<tr>
<td>101</td>
<td>Drac</td>
<td>Ted</td>
<td>90</td>
<td>$3</td>
</tr>
<tr>
<td>102</td>
<td>SQL</td>
<td>Geoff</td>
<td>1</td>
<td>$100</td>
</tr>
<tr>
<td>103</td>
<td>HP</td>
<td>Jean</td>
<td>3</td>
<td>$15</td>
</tr>
</tbody>
</table>

```sql
SELECT book, SUM(qnt) GROUP BY book
```

<table>
<thead>
<tr>
<th>Book</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP</td>
<td>10</td>
</tr>
<tr>
<td>SQL</td>
<td>1</td>
</tr>
<tr>
<td>Drac</td>
<td>90</td>
</tr>
</tbody>
</table>
Storage-Aware Operators

Use column-specific operators

```
SELECT book, SUM(qnt)
GROUP BY book
```

<table>
<thead>
<tr>
<th>Book</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drac</td>
<td>90</td>
</tr>
<tr>
<td>HP</td>
<td>7</td>
</tr>
<tr>
<td>HP</td>
<td>3</td>
</tr>
<tr>
<td>SQL</td>
<td>1</td>
</tr>
</tbody>
</table>
Storage-Aware Operators

Use column-specific operators

```
SELECT book, SUM(qnt)
GROUP BY book
```

<table>
<thead>
<tr>
<th>Book</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drac</td>
<td>90</td>
</tr>
<tr>
<td>HP</td>
<td>7</td>
</tr>
<tr>
<td>HP</td>
<td>3</td>
</tr>
<tr>
<td>SQL</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Book</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drac</td>
<td>90</td>
</tr>
</tbody>
</table>
Storage-Aware Operators

Use **column-specific** operators

```sql
SELECT book, SUM(qnt) GROUP BY book
```

<table>
<thead>
<tr>
<th>Book</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drac</td>
<td>90</td>
</tr>
<tr>
<td>HP</td>
<td>7</td>
</tr>
<tr>
<td>HP</td>
<td>3</td>
</tr>
<tr>
<td>SQL</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Book</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drac</td>
<td>90</td>
</tr>
<tr>
<td>HP</td>
<td>7</td>
</tr>
</tbody>
</table>


Storage-Aware Operators

Use **column-specific operators**

```
SELECT book, SUM(qnt) 
GROUP BY book
```

<table>
<thead>
<tr>
<th>Book</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drac</td>
<td>90</td>
</tr>
<tr>
<td>HP</td>
<td>7</td>
</tr>
<tr>
<td>HP</td>
<td>3</td>
</tr>
<tr>
<td>SQL</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Book</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drac</td>
<td>90</td>
</tr>
<tr>
<td>HP</td>
<td>10</td>
</tr>
</tbody>
</table>
Storage-Aware Operators

Use *column-specific* operators

```sql
SELECT book, SUM(qnt) 
GROUP BY book
```

Operate *directly* over *sorted* and *compressed* data

<table>
<thead>
<tr>
<th>Book</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drac</td>
<td>90</td>
</tr>
<tr>
<td>HP</td>
<td>7</td>
</tr>
<tr>
<td>HP</td>
<td>3</td>
</tr>
<tr>
<td>SQL</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Book</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drac</td>
<td>90</td>
</tr>
<tr>
<td>HP</td>
<td>10</td>
</tr>
<tr>
<td>SQL</td>
<td>1</td>
</tr>
</tbody>
</table>
Storage-Aware Operators

Use *column-specific* operators

Operate *directly* over *sorted* and *compressed* data

Per layout implementations of operators

- Update
- Read
- Scan
- Join
- Aggregate
### Zone Maps

<table>
<thead>
<tr>
<th>Order</th>
<th>Book</th>
<th>User</th>
<th>Qnt</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>HP</td>
<td>Alice</td>
<td>7</td>
<td>$15</td>
</tr>
<tr>
<td>101</td>
<td>Drac</td>
<td>Ted</td>
<td>90</td>
<td>$3</td>
</tr>
<tr>
<td>102</td>
<td>SQL</td>
<td>Geoff</td>
<td>1</td>
<td>$100</td>
</tr>
<tr>
<td>103</td>
<td>HP</td>
<td>Jean</td>
<td>3</td>
<td>$15</td>
</tr>
</tbody>
</table>

**SELECT** Book

**WHERE**

QNT > 100

<table>
<thead>
<tr>
<th>Min</th>
<th>Drac</th>
<th>Alice</th>
<th>1</th>
<th>$3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>SQL</td>
<td>Ted</td>
<td>90</td>
<td>$100</td>
</tr>
</tbody>
</table>
Zone Maps

Kept in memory and on a per partition basis

Allows data skipping, minimizing I/O

<table>
<thead>
<tr>
<th>Min</th>
<th>100</th>
<th>Drac</th>
<th>Alice</th>
<th>1</th>
<th>$3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>103</td>
<td>SQL</td>
<td>Ted</td>
<td>90</td>
<td>$100</td>
</tr>
</tbody>
</table>
Storage Adaptation Decisions

Adaptive Storage Advisor (ASA)

- Metadata
- Cached Decisions
- Execution Plan
- Workload Model

Data Site 1
- Storage aware ops.
- Master copy (row)

Data Site 2
- Sort & compressed
- Replica (column)

Executetxn & storage adaptations
Propagate updates
Observed costs
Adaptive Storage Advisor

Should we execute a proposed change?

What changes should we propose?
Adaptive Storage Advisor

Should we **execute** a proposed change?

What **changes** should we propose?
Quantify if a layout change is beneficial

<table>
<thead>
<tr>
<th>Order</th>
<th>Book</th>
<th>User</th>
<th>Qnt</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>HP</td>
<td>Alice</td>
<td>7</td>
<td>$15</td>
</tr>
<tr>
<td>101</td>
<td>Drac</td>
<td>Ted</td>
<td>90</td>
<td>$3</td>
</tr>
<tr>
<td>102</td>
<td>SQL</td>
<td>Geoff</td>
<td>1</td>
<td>$100</td>
</tr>
<tr>
<td>103</td>
<td>HP</td>
<td>Jean</td>
<td>3</td>
<td>$15</td>
</tr>
</tbody>
</table>

**SELECT** book, **SUM(qnt)**
**GROUP BY** book

Should Proteus use **column** storage **sorted** by **book**?
Quantifying Layout Decisions

**Quantify** if a layout change is **beneficial**

**Txn latency** with **current** vs **proposed** layouts

<table>
<thead>
<tr>
<th>Current</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>(row)</td>
<td>(Sorted col)</td>
</tr>
<tr>
<td>52 ms</td>
<td>15 ms</td>
</tr>
</tbody>
</table>

**SELECT** book, **SUM( qnt )**

**GROUP BY** book
Latency of a Transaction

**Breakdown** transaction into **physical** operators

```sql
SELECT book, SUM(qnt) GROUP BY book
```

**Row layout**

- **Row scan P1**
- **Hash aggregation** book, sum(qnt)

**Logical Plan**

- **Scan & Project** book, qnt
- **Aggregate** book, sum(qnt)

**Sorted column layout**

- **Sequential col scan P1**
- **Sorted col aggregation** book, sum(qnt)
Latency of a Transaction

Breakdown transaction into physical operators

```
SELECT book, SUM(qnt) GROUP BY book
```

Predictions

- 14 ms
- 1 ms

Sorted column layout

- Sequential col scan P1
- Sorted col aggregation book, sum(qnt)

Total predicted latency: 15 ms
Predicting Latency of a Transaction

**Breakdown** transaction into **physical** operators

Predict physical operator latency

Per layout with workload stats as parameters

Cardinality

*Data Width*  

*Est Selectivity*  

Seq col scan  

Predicted Latency
Predicting Latency of a Transaction

**Breakdown** transaction into physical operators

Predict physical operator latency

Per layout with workload stats as parameters

Cardinality

Data Width → Predicted Latency

Est Selectivity

Row scan
Quantifying Layout Decisions

**Quantify if a layout change is beneficial**

Transaction latency with current vs proposed layouts

<table>
<thead>
<tr>
<th>Current (row)</th>
<th>Proposed (Sorted col)</th>
<th>SELECT book, SUM(qnt)</th>
<th>GROUP BY book</th>
</tr>
</thead>
<tbody>
<tr>
<td>52 ms</td>
<td>15 ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 ms</td>
<td>28 ms</td>
<td>INSERT (104, LoTR, John, 2, $21 )</td>
<td></td>
</tr>
</tbody>
</table>

Weight by **likelihood** of transaction
Likelihood of a Transaction

Data accesses to storage often follow predictable pattern
Likelihood of a Transaction

Data accesses to storage often follow **predictable** pattern

![Graph showing periodicity and trend](image)

- **Periodicity**
- **Trend**
Quantifying Layout Decisions

Expected Benefit of Change

Txn *latency* with current vs proposed layouts

Weight by likelihood of txn

Cost incurred to perform change
Adaptive Storage Advisor

Should we execute a proposed change?

What changes should we propose?
Selecting What to Change

Based on **ongoing transaction**

Change high cost plan operators

<table>
<thead>
<tr>
<th>Storage Layout</th>
<th>Decisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>row</td>
</tr>
<tr>
<td>Tier</td>
<td>memory</td>
</tr>
<tr>
<td>Optimization</td>
<td>sort</td>
</tr>
<tr>
<td>Replication</td>
<td>split or merge</td>
</tr>
<tr>
<td>Mastership</td>
<td>Which site?</td>
</tr>
</tbody>
</table>

| column                  | disk                       |
| compress                | horizontal or vertical     |
Selecting What to Change

Based on *ongoing transaction*

- Change high cost plan operators

*Predictively* based on upcoming accesses

- Simulate transactions

Based on *storage constraints*

[tiny.cc/proteus](tiny.cc/proteus)
How well does **Proteus** work?

**CH BenCHmark (TPC-C & TPC-H)**

**Comparisons**

- Row Store (RS)
- Column Store (CS)

**Single Copy**

**Janus**

**Maintain row & column**

**TiDB**
CH BenCHmark

![Graph showing performance comparison between different database systems: TiDB, Row, Janus, Col, and Proteus. The chart indicates that Proteus has a performance of 5%, Janus has 8%, Col has 8%, Row has 5%, and TiDB has a lower performance.](image-url)
CH BenCHmark

![Bar chart showing workload completion times for Proteus, Row, Col, Janus, and TiDB. The chart indicates TiDB is 3.1x faster than Row and 1.5x faster than Col in certain workloads.]
Predictive Adaptation

![Graph showing OLAP Latency and OLTP Throughput over time](image-url)
More Details

More experimental workloads: Twitter, YCSB

Shifting workloads, Ablation studies

ML models (cost & access arrival)

Storage format and layout change execution

Efficient replication and concurrency control

tiny.cc/proteus
Proteus Takeaways

Adaptive Scale-Out Distributed DBMS

Selectively stores data in different formats

Proteus makes decisions based on workload

Superior performance on mixed workloads