Adaptive Replication and Partitioning in Data Systems

Brad Glasbergen, Michael Abebe, Khuzaima Daudjee

Middleware 2018

Data Systems Group
Single Node Architecture

Overloaded
Single Node Architecture

How to scale beyond a single node?

Replicate and partition
Single Node Architecture

How to scale beyond a single node?

Replicate and partition
Replicated Architecture

Handle more requests
Replicated Architecture

Cost of coordination
Replicated Architecture

How many replicas?
Replicated Architecture

W[A] -> A
S_1

W[A] -> A B
S_2

W[A] -> B
S_3
Replicated Architecture

Where to place replicas?
Replicated Architecture

How to propagate updates?

(A)synchronous Consistency
Replication Decisions

● How many replicas?
● Where to place replicas?
● How to propagate updates?
Single Node Architecture

How to scale beyond a single node?

Replicate and partition
Partitioned Architecture

Distributes requests
Partitioned Architecture

A
S₁

B
S₂
Partitioned Architecture

How to form partitions?
Partitioned Architecture

Where to place partitions?
Partitioned Architecture

How to execute multi-partition operations?
Partitioning Decisions

- How to form partitions?
- Where to place partitions?
- How to execute multi-partition operations?
Where to place partitions?

\[ S_1 \]

\[ A_1 \]

\[ S_3 \]

\[ A_2 \]

\[ B \]
Static Decisions

How to make a partitioning or replication decision when access patterns change?

Why do access patterns change?
Why do accesses change?

Humans have follow-the-sun cycles
Why do accesses change?

Load bursts
Why do accesses change?

Shifting hot-spots
Static Decisions

How to **make** a partitioning or replication **decision** when access patterns **change**?

**Adaptively** replicate and partition
Where to place partitions?
Where to place partitions?
How many replicas?
How many replicas?

$A_1 \quad B$

$S_1$

$A_2 \quad B$

$S_2$
Road Map

- Adaptive Replication
- Adaptive Partitioning
- Outlook
Adaptive Replication
Replication Decisions

● **How many** replicas?

● **Where to place** replicas?

● **How to propagate** updates?
Adaptive Replication

- Decentralized
- Geo-Distributed
- Caching
- Availability
Adaptive Replication (ADR)

(Wolfson et al., TODS 1997)
Adaptive Replication (ADR)

(Wolfson et al., TODS 1997)
Adaptive Replication (ADR)

S1  S2  S3

(Wolfson et al., TODS 1997)
Local Read

(Wolfson et al., TODS 1997)
Remote Read

(Wolfson et al., TODS 1997)
Remote Read

(Wolfson et al., TODS 1997)
Remote Read

(Wolfson et al., TODS 1997)
Replication for Reads

(Wolfson et al., TODS 1997)
No Free Lunch

(Wolfson et al., TODS 1997)
No Free Lunch

Reduces read cost,
Increases write cost

(Wolfson et al., TODS 1997)
When to Replicate?

(Wolfson et al., TODS 1997)
When to Replicate?

(Wolfson et al., TODS 1997)
When to Replicate?

(Wolfson et al., TODS 1997)

(Wolfson et al., TODS 1997)
When to Replicate?

(Wolfson et al., TODS 1997)

5 Fewer Messages!
When to Stop Replicating?

(Wolfson et al., TODS 1997)
When to Stop Replicating?

(Wolfson et al., TODS 1997)
When to Stop Replicating?

(Wolfson et al., TODS 1997)
Decentralized Decisions

(Wolfson et al., TODS 1997)
Adapts to Changing Workloads

(Wolfson et al., TODS 1997)
Extensions (Network Topology)

(Wolfson et al., TODS 1997)
Extensions (Network Topology)

(Wolfson et al., TODS 1997)
Extensions (Network Topology)

Form Tree

(Wolfson et al., TODS 1997)
Extensions (Blocks)

\[
\frac{1}{3} R[A] + \frac{1}{3} R[B] + \frac{1}{3} R[C] \quad \frac{1}{3} W[A] + \frac{1}{3} W[B] + \frac{1}{3} W[C]
\]

(Wolfson et al., TODS 1997)
Peer-to-Peer File Systems
Hash-Based P2P File System

h(s) = 275
Hash-Based P2P File System
Hash-Based P2P File System
Hash-Based P2P File System
Hash-Based P2P FS

Balanced load, but poor access locality!
Hierarchical P2P File Systems

The diagram illustrates a hierarchical file system structure. The root folder is labeled as `/`, with two subfolders `/b` and `/a`. `b` contains two files `/b/g1.txt` and `/b/g2.txt`, while `a` contains two files `/a/f2.txt` and `/a/f1.txt`.
Hierarchical P2P File Systems
Hierarchical P2P File Systems

\[
\text{S}_1 \quad /b \quad \text{S}_2 \quad /b/g1.txt \quad /b/g2.txt \quad \text{S}_3 \quad /a \quad /a/f2.txt \quad /a/f1.txt
\]
Hierarchical P2P File Systems

\[ S_1 \]

\[ S_2 \]

\[ S_3 \]

/\text{b}/g1\text{.txt} /\text{b}/g2\text{.txt} /\text{a}/f2\text{.txt} /\text{a}/f1\text{.txt}
Hierarchical P2P File System

Good locality, but poor balance
Locality and Load Balance

Moderate Load

Overloaded

Light Load

(Gopalakrishnan et al., ICDCS'04)
Locality and Load Balance

Moderate Load

Overloaded

Light Load

(Gopalakrishnan et al., ICDCS'04)
Locality and Load Balance

No replication

(Gopalakrishnan et al., ICDCS'04)
Locality and Load Balance

Replicate to balance load!

(Gopalakrishnan et al., ICDCS'04)
Locality and Load Balance

(Gopalakrishnan et al., ICDCS'04)
Locality **and** Load Balance

(Gopalakrishnan et al., ICDCS'04)

Replicate to balance load!
Locality and Load Balance

(Gopalakrishnan et al., ICDCS'04)
Decentralized Replicas

(Gopalakrishnan et al., ICDCS'04)
Replication Decisions

- **How many replicas?**
  Decentralized decision

- **Where to place replicas?**
  At the requester

- **How to propagate updates?**
  ADR: Synchronous, P2P: read-only

(Wolfson et al., TODS 1997) (Gopalakrishnan et al., ICDCS’04)
Global Scale Replication

Average Latency

31 ms
Global Scale Replication

Average Latency
113 ms
Global Scale Replication

Average Latency
95 ms
Global Scale Replication

Average Latency

64 ms
Global Scale Replication

Place data around the world

Minimize cost of access

Take me to your leader!

GPlacer
(Zakhary et al., EDBT 2018)

(Sharov et al., VLDB 2015)
Data Distribution Model

(Sharov et al., VLDB 2015)
Data Distribution Model

(Sharov et al., VLDB 2015)
Data Distribution Model

(Sharov et al., VLDB 2015)
Global Replication Problem

- Select replicas
- Assign replica roles (read or read-write)
- Assign leader

(Sharov et al., VLDB 2015)
Global Replication Problem

- Select replicas
- Assign replica roles (read or read-write)
- Assign leader

(Sharov et al., VLDB 2015)
Assign Leader

**Leader**: site that *minimizes access costs*

(Sharov et al., VLDB 2015)
Write transaction cost

\[ \text{cost} = \text{RTT}(S_3, S_1) + \text{median RTT}(S_1, \{S_1, S_2, S_3\}) \]

(Sharov et al., VLDB 2015)
Assign Leader

**Leader**: site that *minimizes access costs*

**Client cost**: $\text{RTT(client, replica)} + \text{cost( transaction )}$

(Sharov et al., VLDB 2015)
Weighting Client Cost

(Sharov et al., VLDB 2015)
Weighting Client Cost

(Sharov et al., VLDB 2015)
Assign Leader

**Leader**: site that minimizes access costs

**Client cost**: $\text{RTT(client, replica)} + \text{cost(transaction)}$

**Cost**: Weighted average of client costs

(Sharov et al., VLDB 2015)
Global Replication Problem

- Select replicas
- Assign replica roles (read or read-write)
- Assign leader

(Sharov et al., VLDB 2015)
Assign Replica Roles

**Leader**: minimizes median **RTT** to read-write replicas

**Read-write replicas**: 

(Sharov et al., VLDB 2015)
Write transaction cost

Quorum writes

Send write to leader

\[
\text{cost} = \text{RTT}(S_3, S_1) + \text{median} \text{ RTT}(S_1, \{S_1, S_2, S_3\})
\]

(Sharov et al., VLDB 2015)
Assign Replica Roles

**Leader**: minimizes median RTT to read-write replicas

**Read-write replicas**: Lowest RTT to leader

(Sharov et al., VLDB 2015)
Global Replication Problem

- Select replicas
- Assign replica roles (read or read-write)
- Assign leader

(Sharov et al., VLDB 2015)
Replica selection

Leader: *minimizes median RTT* to read-write replicas

Read-write replicas: Lowest RTT to leader

Read replicas:

(Sharov et al., VLDB 2015)
Replica selection

**Client cost**: $RTT(client, replica) + \text{cost( transaction )}$

(Sharov et al., VLDB 2015)
Replica selection

Client cost: $RTT(\text{client}, \text{replica}) + \text{cost( transaction )}$

Read replicas: Lowest RTT to clients

(Sharov et al., VLDB 2015)
Replica selection

**Leader:** minimizes median RTT to read-write replicas

**Read-write replicas:** Lowest RTT to leader

**Read replicas:** Lowest RTT to clients

(Sharov et al., VLDB 2015)
K-Means Replica selection

(Sharov et al., VLDB 2015)
K-Means Replica selection

Average Latency

55 ms

(Sharov et al., VLDB 2015)
K-Means Replica selection

**Select** replicas

**Assign** leader and read-write replicas

(Sharov et al., VLDB 2015)
Leaderless Protocols

Any quorum member can coordinate

(Zakhary et al., EDBT 2018)
Hinted Hand off

(Zakhary et al., EDBT 2018)
Hinted Hand off

(Zakhary et al., EDBT 2018)
Hinted Hand off

\[ \text{cost}( S ) = \text{cost of executing request at } S \]

Hand off request from \( S_1 \) to \( S_2 \) if:

\[ \text{cost}( S_1 ) > \text{RTT}( S_1, S_2 ) + \text{cost}( S_2 ) \]

(Zakhary et al., EDBT 2018)
Replication Decisions

- **How many replicas?**
  Centralized, given client workload

- **Where to place replicas?**
  Heuristic (clustering)

- **How to propagate updates?**
  Quorums / Leader-based (Sharov)

(Sharov et al., VLDB 2015) (Zakhary et al., EDBT 2018)
Intra-Region Latency

(Orasaram et al., WWW 2005)
Edge Nodes

Supports Static Data

Dynamic Data?

(Sivasubramanian et al., WWW 2005)
GlobeDB

(Sivasubramanian et al., WWW 2005)
Replication Granularity

(Sivasubramanian et al., WWW 2005)

<table>
<thead>
<tr>
<th>ID</th>
<th>ARTIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bryan Adams</td>
</tr>
<tr>
<td>2</td>
<td>Justin Bieber</td>
</tr>
<tr>
<td>3</td>
<td>Avril Lavigne</td>
</tr>
</tbody>
</table>
Replication Granularity

(Sivasubramaninan et al., WWW 2005)

<table>
<thead>
<tr>
<th>ID</th>
<th>ARTIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bryan Adams</td>
</tr>
<tr>
<td>2</td>
<td>Justin Bieber</td>
</tr>
<tr>
<td>3</td>
<td>Avril Lavigne</td>
</tr>
</tbody>
</table>

Per-Record?

High Overhead
Replication Granularity

(Sivasubramanian et al., WWW 2005)
Access-Driven Replicas

(Sivasubramanian et al., WWW 2005)

<table>
<thead>
<tr>
<th>ID</th>
<th>ARTIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bryan Adams</td>
</tr>
<tr>
<td>2</td>
<td>Justin Bieber</td>
</tr>
<tr>
<td>3</td>
<td>Avril Lavigne</td>
</tr>
</tbody>
</table>

When would these be replicated together?
Access-Driven Replicas

<table>
<thead>
<tr>
<th>ID</th>
<th>ARTIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bryan Adams</td>
</tr>
<tr>
<td>2</td>
<td>Justin Bieber</td>
</tr>
<tr>
<td>3</td>
<td>Avril Lavigne</td>
</tr>
</tbody>
</table>

\[ A_{\text{bryan}} = \langle r_1, \ldots, r_n, w_1, \ldots, w_n \rangle \]

(Sivasubramanian et al., WWW 2005)
Access-Driven Replicas

<table>
<thead>
<tr>
<th>ID</th>
<th>ARTIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bryan Adams</td>
</tr>
<tr>
<td>2</td>
<td>Justin Bieber</td>
</tr>
<tr>
<td>3</td>
<td>Avril Lavigne</td>
</tr>
</tbody>
</table>

\[ A_{\text{justin}} = \langle r_1, \ldots, r_n, w_1, \ldots, w_n \rangle \quad \text{Sim}(A_{\text{bryan}}, A_{\text{justin}}) \geq \tau ? \]

(Sivasubramanian et al., WWW 2005)
Access-Driven Replicas

### Shared Replication Scheme

<table>
<thead>
<tr>
<th>ID</th>
<th>ARTIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bryan Adams</td>
</tr>
<tr>
<td>2</td>
<td>Justin Bieber</td>
</tr>
<tr>
<td>3</td>
<td>Avril Lavigne</td>
</tr>
</tbody>
</table>

(Sivasubramanian et al., WWW 2005)
Access-Driven Replicas

\[ A_{p1} = \langle r_1, \ldots, r_n, w_1, \ldots, w_n \rangle \]

(Sivasubramanian et al., WWW 2005)
Access-Driven Replicas

\[ A_{\text{avril}} = \langle r_1, \ldots, r_n, w_1, \ldots, w_n \rangle \]

\[ \text{Sim}(A_{p1}, A_{\text{avril}}) \geq \tau? \]

(Sivasubramanian et al., WWW 2005)
## Access-Driven Replicas

(Sivasubramanian et al., WWW 2005)

<table>
<thead>
<tr>
<th>ID</th>
<th>ARTIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bryan Adams</td>
</tr>
<tr>
<td>2</td>
<td>Justin Bieber</td>
</tr>
<tr>
<td>3</td>
<td>Avril Lavigne</td>
</tr>
</tbody>
</table>
# Access-Driven Replicas

(Sivasubramanian et al., WWW 2005)

<table>
<thead>
<tr>
<th>ID</th>
<th>ARTIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bryan Adams</td>
</tr>
<tr>
<td>2</td>
<td>Justin Bieber</td>
</tr>
<tr>
<td>3</td>
<td>Avril Lavigne</td>
</tr>
<tr>
<td>4</td>
<td>Kanye West</td>
</tr>
<tr>
<td>5</td>
<td>Drake</td>
</tr>
<tr>
<td>6</td>
<td>David Guetta</td>
</tr>
<tr>
<td>7</td>
<td>Ed Sheeran</td>
</tr>
</tbody>
</table>
Transaction Processing

Origin Server:
Decide Partitions,
Place Replicas,
Place Master

(Sivasubramanian et al., WWW 2005)
Transaction Processing

(Sivasubramanian et al., WWW 2005)
Transaction Processing

(Sivasubramanian et al., WWW 2005)
Transaction Processing

(Sivasubramanian et al., WWW 2005)
Transaction Processing

(Sivasubramanian et al., WWW 2005)
Transaction Processing

(Sivasubramanian et al., WWW 2005)
Replica and Master Placement

(Sivasubramanian et al., WWW 2005)
Read Latency vs. Bandwidth

(Sivasubramanian et al., WWW 2005)
Read Latency vs. Bandwidth

(Sivasubramanian et al., WWW 2005)
Master Partition Placement

(Sivasubramanian et al., WWW 2005)
Master Partition Placement

(Sivasubramanian et al., WWW 2005)
Placement Heuristic

100% Threshold

(Sivasubramaninan et al., WWW 2005)
Placement Heuristic

95% Threshold

(Sivasubramanian et al., WWW 2005)
Placement Heuristic

(Sivasubramanian et al., WWW 2005)
Placement Heuristic

(Sivasubramanian et al., WWW 2005)
Placement Heuristic

Minimize:

\[ \alpha r + \beta b \]

Sivasubramanian et al., WWW 2005

0% Threshold
Replication Decisions

- **How many replicas?**
  Cost-based given requests

- **Where to place replicas?**
  Cost-based given requests

- **How to propagate updates?**
  Single-master, eventual consistency

(Sivasubramanian et al., WWW 2005)
Web Workload Characteristics

Read-heavy

Cache misses are very painful

(Glasbergen et al., EDBT 2018)
Web Workload Characteristics

(Glasbergen et al., EDBT 2018)
Web Workload Characteristics

(Glasbergen et al., EDBT 2018)
Web Workload Characteristics

(Glasbergen et al., EDBT 2018)
Predictive Caching

Q₁ → Forward(Q₁) → Result(Q₁) → Cache(Q₂) → Q₂

Predict(Q₂)

(Glasbergen et al., EDBT 2018)
Query Patterns (TPC-W)

```
SELECT C_ID FROM CUSTOMER WHERE C_UNAME = ? and C_PASSWORD = ?
SELECT MAX(O_ID) FROM ORDERS WHERE O_C_ID = ?
```

(Glasbergen et al., EDBT 2018)
SELECT `C_ID` FROM CUSTOMER WHERE `C_UNAME` = 'Alice' and `C_PASSWD` = 'pass'

SELECT MAX(O_ID) FROM ORDERS WHERE O_C_ID = 3

SELECT `C_ID` FROM CUSTOMER WHERE `C_UNAME` = ? and `C_PASSWD` = ?

SELECT MAX(O_ID) FROM ORDERS WHERE O_C_ID = ?
Building a Predictive Model

(Glasbergen et al., EDBT 2018)
Building a Predictive Model

(Glasbergen et al., EDBT 2018)
Building a Predictive Model

(Glasbergen et al., EDBT 2018)
Building a Predictive Model

All executed 5 times

100% probability $Q_2$ follows $Q_1$

20% probability $Q_1$ follows $Q_3$

(Glasbergen et al., EDBT 2018)
Finding Parameter Mappings

```
SELECT C_ID FROM CUSTOMER WHERE C_UNAME = ? and C_PASSWD = ?
SELECT MAX(O_ID) FROM ORDERS WHERE O_C_ID = ?
```
Finding Parameter Mappings

3

SELECT [C_ID] FROM CUSTOMER WHERE C_UNAME = ? and
C_PASSWD = ?

SELECT MAX(O_ID) FROM ORDERS WHERE O_C_ID = ?

(Glasbergen et al., EDBT 2018)
Finding Parameter Mappings

3

SELECT [C_ID] FROM CUSTOMER WHERE C_UNAME = ? and C_PASSWD = ?

SELECT MAX(O_ID) FROM ORDERS WHERE O_C_ID = ?

(Glasbergen et al., EDBT 2018)
Finding Parameter Mappings

SELECT `C_ID` FROM CUSTOMER WHERE C_UNAME = ? and C_PASSWD = ?
SELECT MAX(O_ID) FROM ORDERS WHERE O_C_ID = ?

(Glasbergen et al., EDBT 2018)
Predictive Caching

(Glasbergen et al., EDBT 2018)
Predictive Caching

Predictively Cache $Q_2$

(Glasbergen et al., EDBT 2018)
Predictive Caching

Predictively Cache $Q_3$

(Glasbergen et al., EDBT 2018)
Apollo Deployment

(Glasbergen et al., EDBT 2018)
Apollo Deployment

(Glasbergen et al., EDBT 2018)
Invalidations

(Glasbergen et al., EDBT 2018)
Invalidations

Invalidations Limit
Cache Effectiveness

(Glasbergen et al., EDBT 2018)
Session Semantics

(Glasbergen et al., EDBT 2018)
Session Semantics

Good fit for web data!

(Glasbergen et al., EDBT 2018)
Replication Decisions

- **How many replicas?**
  Predictively based on requests

- **Where to place replicas?**
  Client edge cache, predictively

- **How to propagate updates?**
  Cache updates with sessions

(Glasbergen et al., EDBT 2018)
Replication for Availability

Failures are common

Data systems must remain available
Replication for Availability

Tolerating \( r \) faults requires \( r + 1 \) replicas
Lower Overhead

XOR

\[ B_2 \ xor (B_1 \ xor B_2) = B_1 \]
Erasure Coding

Erasure coding tolerates $r$ faults and requires $(k+r)/k$ space.

$k$ partitions

$1$

$2$

$\ldots$

$k$

$r$ parity partitions

$k + 1$

$\ldots$

$k + r$

Tolerating $r$ faults requires $(k+r)/k$ space
Erasure Coding

$k = 2, \ r = 2$

Encode and store

$W [ A ]$

$A_1 \rightarrow S_1$

$A_2 \rightarrow S_2$

$A_3 \rightarrow S_3$

$A_4 \rightarrow S_4$

$S_5$
Erasure Coding

$k = 2, r = 2$

Read and decode

$A_1$  
$S_1$  
$A_2$  
$S_2$  
$A_3$  
$S_3$  
$A_4$  
$S_4$  
$S_5$
Erasure Coding

Reduces storage overhead

Requires parallel retrieval
Erasure Coded Storage

Where to place data

How to access data

EC-Store
(Abebe, ICDCS 2018)
EC-Store Data Access

\[ k = 2, r = 1 \]

Load aware

(Abebe et al., ICDCS 2018)
EC-Store Data Access

Access Strategy: Minimize cost of access

Cost of site access: load at site + I/O at site

(Abebe et al., ICDCS 2018)
EC-Store Data Movement

\(k = 2, r = 1\)

(Abebe et al., ICDCS 2018)
EC-Store Data Movement

Move data to minimize cost of future accesses and balance system load

Model access patterns to predict future accesses

(Abebe et al., ICDCS 2018)
Replication Decisions

- **How many replicas?**
  Fault tolerance requirements

- **Where to place replicas?**
  Dynamic movement, using access costs

- **How to propagate updates?**
  Synchronous updates

(Abebe et al., ICDCS 2018)
Road Map

- Adaptive Replication
- Adaptive Partitioning
- Outlook
Adaptive Partitioning
Partitioning Decisions

- How to form partitions?
- Where to place partitions?
- How to execute multi-partition operations?
Adaptive Partitioning

- Iterative improvements
- Partitioning per request
- Considering the overall workload
  - Heuristics
# Physical Database Design

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>
### Physical Database Design

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>

- **Column A:** 1, 2, 3
- **Column B:** 2, 4, 6
- **Column C:** 4, 6, 7
- **Column D:** 8, 3, 10
## Physical Database Design

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>
## Physical Database Design

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>
Physical Database Design

SELECT AVERAGE(C) FROM R WHERE R.D > 5;

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>
SELECT AVERAGE(C) FROM R WHERE R.D > 5;

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>
Analytic Database Design

```
SELECT AVERAGE(C) FROM R WHERE R.D > 5;
```

Need to know what to index upfront
Adaptive Range Indexing

SELECT ... FROM R WHERE R.D > 5;
SELECT ... FROM R WHERE R.D > 5 AND R.D < 10;
SELECT ... FROM R WHERE R.D > 10 AND R.D < 20;
Database Cracking

<table>
<thead>
<tr>
<th>Column D</th>
<th>Cracked Column D</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>21</td>
<td>21</td>
</tr>
</tbody>
</table>

(Idreos et al., CIDR 2007)
Indexes via Partitioning

(SELECT … WHERE R.D > 5)

(Idreos et al., CIDR 2007)
Indexes via Partitioning

SELECT ... WHERE R.D > 5

(Idreos et al., CIDR 2007)
Indexes via Partitioning

(Idreos et al., CIDR 2007)

SELECT ... WHERE R.D > 5
Indexes via Partitioning

SELECT ... WHERE
R.D > 5

(Idreos et al., CIDR 2007)
Indexes via Partitioning

SELECT ... WHERE R.D > 5

(Idreos et al., CIDR 2007)
Indexes via Partitioning

SELECT ... WHERE R.D > 5

(Cidreos et al., CIDR 2007)
Indexes via Partitioning

SELECT ... WHERE R.D > 5

(Idreos et al., CIDR 2007)
Indexes via Partitioning

Cracked Column D

SELECT ... WHERE R.D > 5

(Idreos et al., CIDR 2007)
Indexes via Partitioning

```
SELECT ... WHERE R.D > 5 AND R.D < 10
```

Cracked Column D

Only need to consider these

(Idreos et al., CIDR 2007)
Indexes via Partitioning

(SELECT … WHERE R.D > 5 AND R.D < 10)

(Idreos et al., CIDR 2007)
Indexes via Partitioning

### SELECT … WHERE

R.D > 10 AND R.D < 20

![Cracked Column D](image)

Only need to consider these

(Idreos et al., CIDR 2007)
Indexes via Partitioning

SELECT ... WHERE R.D > 10 AND R.D < 20

(Idreos et al., CIDR 2007)
Cracking: Extensions

Cracked Column D

Advanced Cracking Methods

Distribution

(Idreos et al., CIDR 2007)
Partitioning Decisions

- **How to form partitions?**
  Iteratively, based on queries

- **Where to place partitions?**
  Sorted in memory

- **How to execute multi-partition operations?**
  N/A

(Idreos et al., CIDR 2007)
Exploratory Workloads

App Usage Time
Exploratory Workloads

Usage By Device
Exploratory Workloads

Revenue By Device
Exploratory Workloads

Devices By Country

No upfront information, need generic partitioning!
Initial Partitioning (KD-Tree)

512 MB

256 MB

128 MB

64 MB

(Shanbhag et al., SoCC 2017)
Heterogeneous Tree

Contains more attributes!

(Shanbhag et al., SoCC 2017)
Building the Partitioning

A: 0.0
B: 0.0
C: 0.0
D: 0.0

(Shanbhag et al., SoCC 2017)
Building the Partitioning

A: 1.0
B: 0.0
C: 0.0
D: 0.0

(Shanbhag et al., SoCC 2017)
Building the Partitioning

A: 1.0
B: 0.5
C: 0.0
D: 0.0

(Shanbhag et al., SoCC 2017)
Building the Partitioning

(Shanbhag et al., SoCC 2017)
Building the Partitioning

A: 1.0
B: 0.5
C: 0.5
D: 0.5

(Shanbhag et al., SoCC 2017)
Building the Partitioning

A: 1.0
B: 0.75
C: 0.5
D: 0.5

(Shanbhag et al., SoCC 2017)
Adaptive Partitioning

\[ Q_1 = \sigma_{D\leq 45} \]

(Shanbhag et al., SoCC 2017)
Adaptive Partitioning

\[ Q_2 = \sigma_{A \geq 125} \]

Refine partitioning per the workload!

(Shanbhag et al., SoCC 2017)
Adaptive Partitioning: When?

Q₁', Q₂', Q₃', Q₁', ...

Q₁ = σ_{D≤45}

(Shanbhag et al., SoCC 2017)
Swap Operation

\[ Q_1, Q_2, Q_3, Q_1', \ldots \]

\[ Q_1 = \sigma_{D \leq 45} \]

Rewrite Tree

(Shanbhag et al., SoCC 2017)
Push Up Operation

$Q_1', Q_2', Q_3', Q_1', ...$

$Q_1 = \sigma_{D \leq 45}$

(Shanbhag et al., SoCC 2017)
Push Up Operation

\[ Q_1', Q_2', Q_3', Q_1', ... \]

\[ Q_1 = \sigma_{D \leq 45} \]

Logical Movement

(Shanbhag et al., SoCC 2017)
Divide and Conquer

\[ Q_1 = \sigma_{D \leq 45} \]

Get Best Subtree

(Shanbhag et al., SoCC 2017)
Partitioning Decisions

- **How to form** partitions?
  Upfront then iteratively, based on queries
- **Where to place** partitions?
  Rely on HDFS
- **How to execute** multi-partition operations?
  Rely on HDFS

(Shanbhag et al., SoCC 2017)
Exploiting Workloads

- Known ahead of time
- Parameterized
- Repetitive
Exploiting Workloads - OLTP

Warehouse

District

Customer
Partitioning OLTP

Write \([ W_1, D_1, C_1 ]\)

\(W_1\) \(D_1\) \(C_1\)

\(S_1\)

\(W_2\) \(D_2\) \(C_2\)

\(S_2\)
Partitioning OLTP

Write \([ W_1, D_1, C_2 ]\)

prepare to commit

Two phase commit
Partitioning OLTP

Workload based repartitioning

Per transaction partitioning

Later in the tutorial

G-Store
(Das et al., SoCC 2010)

L-Store
(Lin et al., SIGMOD 2016)
Key Grouping

Create group

Write $[W_1, D_1, C_2]$

(Das et al., SoCC 2010)
Key Grouping

Create group

Write[$W_1, D_1, C_2$]

Join request

(Das et al., SoCC 2010)
Key Grouping

Create group

Create

Write[ W₁, D₁, C₂ ]

Join request

Txn ops

W₁

D₁

C₁

S₁

W₂

D₂

C₂

S₂

Join

Propagate

(Das et al., SoCC 2010)
Key Grouping

Create group

Delete group

Write \([ W_1, D_1, C_2 ]\)

Join request

Propagate

Joined

(Das et al., SoCC 2010)
Create group

Key Grouping

Create group

Txn ops

Delete group

Write[ W₁, D₁, C₂ ]

Join request

Joined

Propagate

Free

(Das et al., SoCC 2010)
Key Grouping

Create group

Txn ops

Delete group

Write $[W_1, D_1, C_2]$
Key Grouping

On demand transactional partitioning

Works best when groups are small and transactions contain multiple operations

But groups are transient

(Das et al., SoCC 2010)
Localizing Execution

Repartition data via localization for single site execution

Dynamic partitioning based on transaction patterns

(Lin et al., SIGMOD 2016)
Localizing Execution

Ownership information

(Lin et al., SIGMOD 2016)
Localizing Execution

Ownership information

(Lin et al., SIGMOD 2016)
Localizing Execution

Write \([ W_1, D_1, C_1 ]\)

\(W_1\) | \(S_1\) | \(D_1\) | \(C_1\) | \(S_1\)
---|---|---|---|---
\(W_2\) | \(S_2\) | \(D_2\) | \(C_2\) | \(S_2\)

\(W_1\) \(D_1\) \(C_1\) \(S_1\)

\(W_2\) \(D_2\) \(C_2\) \(S_2\)

(Lin et al., SIGMOD 2016)
Localizing Execution

Owner request

Write \([ W_1, D_1, C_2 ]\)

(Lin et al., SIGMOD 2016)
Localizing Execution

Owner request

Write \([ W_1, D_1, C_2 ]\)

Transfer

(Lin et al., SIGMOD 2016)
Localizing Execution

Owner request

Write \([ W_1, D_1, C_2 ]\)

Txn ops

Transfer

Response

\(\text{(Lin et al., SIGMOD 2016)}\)
Localizing Execution

Write \([ W_1, D_1, C_2 ]\)

Txn ops

\[
\begin{array}{c|c|c|c}
W_1 & S_1 & W_1 & D_1 \\
W_2 & S_2 & C_1 & C_2 \\
D_1 & S_1 & S_1 & S_1 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
W_2 & D_2 & W_2 & D_2 \\
C_1 & S_1 & C_2 & S_1 \\
S_2 & S_2 & S_1 & S_1 \\
\end{array}
\]

(Lin et al., SIGMOD 2016)
Localizing Execution

Dynamic partitioning based on per transaction patterns

Does not consider workload overall

(Lin et al., SIGMOD 2016)
Partitioning Decisions

- **How to form partitions?**
  Transaction localization

- **Where to place partitions?**
  At requester

- **How to execute multi-partition operations?**
  L-Store protocol

(Lin et al., SIGMOD 2016)
Partitioning Decisions

- **How to form partitions?**
  Key groups, temporarily
- **Where to place partitions?**
  Key group leader
- **How to execute multi-partition operations?**
  Key group protocol

(Das et al., SoCC 2010)
Localizing Transactions

Commit Locally Without Synchronization!
Constructing the Graph

From a workload trace

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alice</td>
</tr>
<tr>
<td>B</td>
<td>Bob</td>
</tr>
<tr>
<td>C</td>
<td>Carol</td>
</tr>
</tbody>
</table>

(Curino et al., VLDB 2010)
Constructing the Graph

Add traced transactions: $R[A,B], 3x W[A,C]$

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alice</td>
</tr>
<tr>
<td>B</td>
<td>Bob</td>
</tr>
<tr>
<td>C</td>
<td>Carol</td>
</tr>
</tbody>
</table>

(Curino et al., VLDB 2010)
Constructing the Graph

Add node weights (size, load)

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alice</td>
</tr>
<tr>
<td>B</td>
<td>Bob</td>
</tr>
<tr>
<td>C</td>
<td>Carol</td>
</tr>
</tbody>
</table>

(Curino et al., VLDB 2010)
Constructing the Graph

Min-cut edges subject to weight imbalance

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alice</td>
</tr>
<tr>
<td>B</td>
<td>Bob</td>
</tr>
<tr>
<td>C</td>
<td>Carol</td>
</tr>
</tbody>
</table>

(Curino et al., VLDB 2010)
Constructing the Graph

Min-cut edges subject to weight imbalance

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alice</td>
</tr>
<tr>
<td>B</td>
<td>Bob</td>
</tr>
<tr>
<td>C</td>
<td>Carol</td>
</tr>
</tbody>
</table>

(Curino et al., VLDB 2010)
Constructing the Graph

Min-cut edges subject to weight imbalance

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alice</td>
</tr>
<tr>
<td>B</td>
<td>Bob</td>
</tr>
<tr>
<td>C</td>
<td>Carol</td>
</tr>
</tbody>
</table>

(Curino et al., VLDB 2010)
Adding Replica Support

\[ R[A,B] \ 3x \ W[A,C] \]

(Curino et al., VLDB 2010)
Adding Replica Support

R[A,B] 3x W[A,C]

Holistic Partitioning/Replication

Offline and Periodic

(Curino et al., VLDB 2010)
Access Patterns Change!

\[ W[A,B], 3x W[A,C], W[A,D], W[C,D] 3x R[B,C] \]

W(P1)=4, W(P2)=10, EC=8

(Nicoara et al., EDBT 2015)
Two Phases

(Nicoara et al., EDBT 2015)
Two Phases

(Nicoara et al., EDBT 2015)

**Phase 2**

**Rule:**
- Movement doesn’t overload
- Move best-gain candidates
- If overloaded, must move!

Logical Movement, then Migrate
Two Phases

(Nicoara et al., EDBT 2015)

W(P1)=4, W(P2)=10, EC=8, Bounds: (6,8)
Two Phases

Gain=0

W(P1)=4, W(P2)=10, EC=8, Bounds: (6,8)

(Nicoara et al., EDBT 2015)
Two Phases

W(P1)=5, W(P2)=9, EC=8, Bounds: (6,8)

(Nicoara et al., EDBT 2015)
Two Phases

Gain=-1

W(P1)=5, W(P2)=9, EC=8, Bounds: (6,8)

(Nicoara et al., EDBT 2015)
Convergence

W(P1)=6, W(P2)=8, EC=9, Bounds: (6,8)

(Nicoara et al., EDBT 2015)
Partitioning Decisions

- **How to form partitions?**
  - Graph partitioning

- **Where to place partitions?**
  - Based on partitioning

- **How to execute multi-partition operations?**
  - 2PC

(Curino et al., VLDB 2010) (Nicoara et al., EDBT 2015)
Graph Partitioning

Minimizes total number of distributed transactions

Ignores per node involvement
Adaptive Database Partitioning

Balance load and minimize distributed transactions

Database elasticity

E-Store
(Taft et al., VLDB 2014)

Clay
(Serafini et al., VLDB 2015)

P-Store
(Taft et al., SIGMOD 2018)
Considering Distributed Cost

**General Graph Partitioning:** minimize # edge cuts

such that: \( \text{load}(S_i) < (1 + \varepsilon) \text{avg load}(S) \)

load balanced

**General:** \( \text{load}(S_i) = \sum \text{w}(v) \quad (v \text{ at } S_i) \)

**Clay:** \( \text{load}(S_i) = \sum \text{w}(v) + k \sum \text{w}(uv) \quad (v \text{ at } S_i) \)

\( (u \text{ not at } S_i) \)

---

(Serafini et al., VLDB 2016)
Repartitioning Cost

General Graph Partitioning: \( \text{minimize} \# \text{ edge cuts} \)

Clay: \( \text{minimize} \# \text{ edge cuts} \) and \( \# \text{ of vertices mapped to new partitions} \)

\( \text{cost of repartitioning} \)

(Serafini et al., VLDB 2016)
Clumping

(Serafini et al., VLDB 2016)
Clumping

(Serafini et al., VLDB 2016)
Clumping

(Serafini et al., VLDB 2016)
Clumping

$S_1$

$S_2$

$S_3$

Increases cost

Migrate clump

(Serafini et al., VLDB 2016)
Clumping

(Serafini et al., VLDB 2016)
Clumping

(Serafini et al., VLDB 2016)
Clumping

(Serafini et al., VLDB 2016)

Migrate clump
Clumping

(Serafini et al., VLDB 2016)
Clumping

(Serafini et al., VLDB 2016)
Clumping

(Serafini et al., VLDB 2016)
Clumping

Expands clumps to \textit{frequently accessed neighbours}

Consider moving clump to \textit{lightly loaded sites}

Considers both \textit{re-partitioning and load costs}

(Serafini et al., VLDB 2016)
Elasticity

(Taft et al., VLDB 2015)
Elasticity

(Taft et al., VLDB 2015)
Elasticity

(Taft et al., VLDB 2015)
Elasticity

Repartition to *elastically* add or remove nodes

(Taft et al., VLDB 2015)
Elasticity

(Taft et al., VLDB 2015)
Elasticity

(S1, A, B) (S4)

Low Med High

(S2, E, F, C, D)

(Taft et al., VLDB 2015)
Elasticity Decisions

When the \textit{average load}: 

\textbf{increases:} add nodes \\
\textbf{decreases:} remove nodes

(Taft et al., VLDB 2015)
Two Tier Data Placement

Identify **hot** data

Evenly distribute **hot** data

Distribute cold data over remaining capacity

(Taft et al., VLDB 2015)
Identifying Hot Data

Monitor *partition level* access frequency

If *hot partition* enable *tuple level monitoring*

*Reacts* to changes in load

(Taft et al., VLDB 2015)
Reactive Elasticity

(Taft et al., VLDB 2015)
Reactive Elasticity

(Taft et al., VLDB 2015)
Reactive Elasticity

(Taft et al., VLDB 2015)
Reactive Elasticity

(Taft et al., VLDB 2015)
Reactive Elasticity

(Taft et al., VLDB 2015)
Reactive Elasticity

(Taft et al., VLDB 2015)
Reactive Elasticity

(Taft et al., VLDB 2015)
Ideal Elasticity

Load

Capacity

Time

predict the function

(Taft et al., SIGMOD 2018)
Periodic Workloads

Daily load variations

Seasonal load spikes

(Taft et al., SIGMOD 2018)
How to Predict Load

\[
\text{load}(t) = \text{avg}_\text{load}( t - p_i) + \text{change}_\text{in}_\text{load}( t - j_i)
\]

\[
\text{load}(t) = \text{avg}_\text{load}( t - p_i) + \text{change}_\text{in}_\text{load}( t - j_i)
\]

SPAR: Sparse Periodic Auto-Regression

(Taft et al., SIGMOD 2018)
Ideal Elasticity

Load

Capacity

Time

decide the

# of nodes

(Taft et al., SIGMOD 2018)
Number of Nodes

Assuming partitionable

\[
\text{# of nodes} = \frac{\text{Predicted Load}}{\text{Load per Server}}
\]

(Taft et al., SIGMOD 2018)
Partitioning Decisions

- **How to form partitions?**
  Heuristically (Clumping versus 2 Tier)

- **Where to place partitions?**
  React or predict based on load

- **How to execute multi-partition operations?**
  2PC

(Serafini et al., VLDB 2016)
(Taft et al., SIGMOD 2018)

(Taft et al., VLDB 2015)
Road Map

- Adaptive Replication
- Adaptive Partitioning
- Outlook
Outlook
Adaptive Systems

How to make a partitioning or replication decision when access patterns change?

Adaptively replicate and partition
Partitioning Decisions

- **How to form partitions?**
  - Iterative, Temporarily, Graph partitioning, Heuristic

- **Where to place partitions?**
  - Sorted, Leader, At requester, Graph partitioning, Reactively, Predictively

- **How to execute multi-partition operations?**
  - Novel protocols, 2PC
Replication Decisions

- **How many replicas?**
  Decentralized, Client workload, Cost-based, Predictive, Fault tolerance

- **Where to place replicas?**
  At requester, Heuristic, Cost-based, Predictive, Dynamic

- **How to propagate updates?**
  Synchronous, Quorums, Single-master, Cache
Decisions

- How many replicas? Predictively
- Where to place replicas? Predictively
- Where to place partitions? Predictively
Adaptive & Predictive Systems

How to make a partitioning or replication decision when access patterns change?

Adaptively and predictively replicate and partition
Predicting the Future

How can your system predict its future workload?

Apollo: Predict future queries (Markov Model)

P-Store: Predict future load (SPAR)
Predicting the Future QB5000

When, how many, and what queries will arrive?

SELECT * FROM C WHERE id = “C1”

SELECT * FROM C WHERE id = $

Pre-process: remove parameters, creating templates

(Ma et al., SIGMOD 2018)
Predicting the Future QB5000

When, how many, and what queries will arrive?

Cluster: group templates by arrival rate

(Ma et al., SIGMOD 2018)
Predicting the Future QB5000

When, how many, and what queries will arrive?

Forecast: Predict clusters arrival rate (Ensemble of RNN, LR, KR)

(Ma et al., SIGMOD 2018)
Predicting the Future QB5000

When, how many, and what queries will arrive?

Pre-process: remove parameters, creating templates

Cluster: group templates by arrival rate

Forecast: Predict clusters arrival rate
(Ensemble of RNN, LR, KR)

(Ma et al., SIGMOD 2018)
Predicting the Future

How can your system predict its future workload?

**Apollo**: Predict future queries (Markov Model)

**P-Store**: Predict future load (SPAR)

**QB5000**: Predict query workloads (Ensemble of RNN, LR, KR)
Predicting the Future

How can your system predict its future workload?

If your system knew the future workload, how could it partition and replicate data?