Improving Mobile Database Access Over Wide-Area Networks Without Degrading Consistency

Niraj Tolia

M. Satyanarayanan and Adam Wolbach
Carnegie Mellon University
Motivation

- Increased database use over wireless WANs
  - CRM, Sales, Insurance, etc…
- Providing performance and consistency is hard
  - Bandwidth problem acute for wireless networks
  - Previous solutions tend to use:
    - weaken consistency [Bayou, Mariposa, DBCache, Barbara99, …]
    - per-application consistency model [Gao03, Ganymed, …]
Cedar: Mobile Database Access
Cedar optimizes mobile database access
  • Improves performance under weak connectivity
  • Does not weaken consistency or modify application

Scope
  • Mobile users and wireless WANs
  • Assume at least some weak connectivity
  • OLTP-style applications
    • Query size smaller than results
    • Will show evaluation from a TPC-based benchmark
Key Insight

- A *stale database replica* on the client can be useful
  - Optimistic use to reduce data transmission volume
- Stale replicas can be used without compromises
- Tradeoff increased computation for network savings
High-level overview

SELECT query

Client

Time

Database Server (Master Copy)

SELECT query
High-level overview

Client

Database Server (Master Copy)

Query

SELECT query

Time
High-level overview

Client

Query

Query Exec.

SELECT query

Database Server (Master Copy)
High-level overview

Client

Database Server
(Master Copy)

Time

Query

Query Exec.

Result

SELECT query
High-level overview

Client

Database Server (Master Copy)

SELECT query
High-level overview

Client

Client Replica

Database Server (Master Copy)

Time

SELECT query
High-level overview

Client

Client Replica

Database Server (Master Copy)

Time

Query

SELECT query
High-level overview

Client

Client Replica

Database Server (Master Copy)

Time

Query

Query Exec.

SELECT query
High-level overview

SELECT query

Client

Client Replica

Database Server (Master Copy)

Time

Query

Query Exec.

Query + Recipe_{Result}

SELECT query
High-level overview

SELECT query

query

Client

Client Replica

Database Server (Master Copy)

SELECT query
High-level overview

Client

Client Replica

Database Server (Master Copy)

SELECT query

Time

Query

Query Exec.

Query + Recipe_{Result}

Query Exec. + Result Comparison

\Delta(\text{Result})
High-level overview

Client

Client Replica

Database Server (Master Copy)

Time

Query

Query Exec.

Query + Recipe_{Result}

Query Exec. + Result Comparison

Δ(Result)

SELECT query
High-level overview

SELECT query

UPDATE queries do directly to the server
**High-level overview**

**Client**

**Client Replica**

**Database Server (Master Copy)**

Query

Query Exec.

Query + Recipe\(_{Result}\)

Query Exec. + Result Comparison

RESULT

\(\Delta(\text{Result})\)

Result

**SELECT** query

**UPDATE** queries do directly to the server
High-level overview

SELECT query

UPDATE queries do directly to the server
Database Replicas

- Local storage and CPU allow for client replicas
- However, Cedar has to support partial replicas
  - Capacity, privacy, security, and regulatory concerns
- Replicas created using database *hoard profiles*
Database Replicas

- Local storage and CPU allow for client replicas
- However, Cedar has to support partial replicas
  - Capacity, privacy, security, and regulatory concerns
- Replicas created using database *hoard profiles*

```
<attr table="USERS" predicate="zip >= 15213 and zip <= 15295">
<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Address</th>
<th>Zip</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John Doe</td>
<td>412 Avenue</td>
<td>15213</td>
<td><a href="mailto:jd2@eg.com">jd2@eg.com</a></td>
</tr>
<tr>
<td>2</td>
<td>Mary Major</td>
<td>821 Lane</td>
<td>15232</td>
<td><a href="mailto:mm@eg.com">mm@eg.com</a></td>
</tr>
<tr>
<td>3</td>
<td>John Stiles</td>
<td>701 Street</td>
<td>00979</td>
<td><a href="mailto:js@eg.com">js@eg.com</a></td>
</tr>
</tbody>
</table>
```
Database Replicas

- Local storage and CPU allow for client replicas
- However, Cedar has to support partial replicas
  - Capacity, privacy, security, and regulatory concerns
- Replicas created using database *hoard profiles*

```xml
<attr table="USERS" predicate="zip >= 15213 and zip <= 15295">
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Address</th>
<th>Zip</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John Doe</td>
<td>412 Avenue</td>
<td>15213</td>
<td><a href="mailto:jd2@eg.com">jd2@eg.com</a></td>
</tr>
<tr>
<td>2</td>
<td>Mary Major</td>
<td>821 Lane</td>
<td>15232</td>
<td><a href="mailto:mm@eg.com">mm@eg.com</a></td>
</tr>
<tr>
<td>3</td>
<td>John Stiles</td>
<td>701 Street</td>
<td>00979</td>
<td><a href="mailto:js@eg.com">js@eg.com</a></td>
</tr>
</tbody>
</table>
```xml```
Database Replicas

- Local storage and CPU allow for client replicas
- However, Cedar has to support partial replicas
  - Capacity, privacy, security, and regulatory concerns
- Replicas created using database *hoard profiles*

```xml
<attr table="USERS" predicate="zip >= 15213 and zip <= 15295">

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Address</th>
<th>Zip</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John Doe</td>
<td>412 Avenue</td>
<td>15213</td>
<td><a href="mailto:jd2@eg.com">jd2@eg.com</a></td>
</tr>
<tr>
<td>2</td>
<td>Mary Major</td>
<td>821 Lane</td>
<td>15232</td>
<td><a href="mailto:mm@eg.com">mm@eg.com</a></td>
</tr>
<tr>
<td>3</td>
<td>John Stiles</td>
<td>701 Street</td>
<td>00979</td>
<td><a href="mailto:jsf@eg.com">jsf@eg.com</a></td>
</tr>
</tbody>
</table>
```

Database Replicas

- Local storage and CPU allow for client replicas
- However, Cedar has to support partial replicas
  - Capacity, privacy, security, and regulatory concerns
- Replicas created using database hoard profiles

<attr table="USERS" predicate="zip >= 15213 and zip <= 15295">

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Address</th>
<th>Zip</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John Doe</td>
<td>412 Avenue</td>
<td>15213</td>
<td><a href="mailto:jd2@eg.com">jd2@eg.com</a></td>
</tr>
<tr>
<td>2</td>
<td>Mary Major</td>
<td>821 Lane</td>
<td>15232</td>
<td><a href="mailto:mm@eg.com">mm@eg.com</a></td>
</tr>
<tr>
<td>3</td>
<td>John Stiles</td>
<td>701 Street</td>
<td>00979</td>
<td><a href="mailto:js@eg.com">js@eg.com</a></td>
</tr>
</tbody>
</table>

- Hoard profiles preserve schema of the original database
Using Replicas

- Replicas are standalone database engines
- Transparent to application and database server
Using Replicas

- Replicas are standalone database engines
- Transparent to application and database server

![Diagram showing the relationship between Application, JDBC Driver, Network, Cedar Proxy, and DB.]
Using Replicas

- Replicas are standalone database engines
- Transparent to application and database server
Using Replicas

- Replicas are standalone database engines
- Transparent to application and database server
Using Replicas

- Replicas are standalone database engines
- Transparent to application and database server
Using Replicas

- Replicas are standalone database engines
- Transparent to application and database server

- Assuming staleness simplifies replica control
  - No overhead of callback-based mechanisms
  - Replicas can be updated lazily at runtime
Commonality Detection

- Cedar uses Content Addressability

- Hash value is a globally unique identifier
  - Independent of any particular system
  - Infeasible to find another object with same hash
  - If hash values are equal, so are the source objects

- Any small change in source completely changes hash
Commonality Detection

- Exploit structure in results – they look like tables

```sql
SELECT name, address, zip, email FROM USERS
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Zip</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
<td>412 Avenue</td>
<td>15213</td>
<td><a href="mailto:jd2@eg.com">jd2@eg.com</a></td>
</tr>
<tr>
<td>Mary Major</td>
<td>821 Lane</td>
<td>15232</td>
<td><a href="mailto:mm@eg.com">mm@eg.com</a></td>
</tr>
<tr>
<td>John Stiles</td>
<td>701 Street</td>
<td>00979</td>
<td><a href="mailto:js@eg.com">js@eg.com</a></td>
</tr>
</tbody>
</table>

- Recipes are a compact representation of the result
  - A convenient method of comparing two results
Paring down the recipe

- Recipes can be large, esp. for results with small rows
  - Each hash is 20 bytes for SHA-1, 32 bytes for SHA-256

0x 2fd4e1c6 7a2d28fc ed849ee1 bb76e739 1b93eb12
0x de9f2c7f d25e1b3a fad3e85a 0bd17d9b 100db4b3
0x da39a3ee 5e6b4b0d 3255bffe 95601890 afd80709
Paring down the recipe

- Recipes can be large, esp. for results with small rows
  - Each hash is 20 bytes for SHA-1, 32 bytes for SHA-256

Solution: Only use 4 bytes as the row identifier

0x 2fd4e1c6
0x de9f2c7f
0x da39a3ee
Paring down the recipe

- Recipes can be large, esp. for results with small rows
  - Each hash is 20 bytes for SHA-1, 32 bytes for SHA-256

  \[\begin{array}{l}
  0x\ 2fd4e1c6 \\
  0x\ de9f2c7f \\
  0x\ da39a3ee
  \end{array}\]

  Solution: Only use 4 bytes as the row identifier

- How can this ensure correctness?
  - Only compares results from same query: reduced key space
  - Therefore, a reduced probability of hash collisions
  - With 4 bytes and 10,000 hash values, \( Pr\{\text{collision}\} = 2.3 \times 10^{-6} \)
  - However, much lower than SHA-1, where \( Pr\{\text{collision}\} = 2^{-160} \)
  - A hash over the entire result provides an end-to-end check
Putting it all together

Client

Database Server

Client Replica
Putting it all together

Client

Query

Client Replica

Database Server
Putting it all together

Query

Client

Tentative Result

Client Replica

Database Server
Putting it all together

Client

Query

Client Replica

0x2fd4e1c6
0xda39a3ee

Tentative Result

Database Server
Putting it all together

Client

Database Server

Client Replica

Query

Tentative Result

0x2fd4e1c6
0xda39a3ee

Query
Putting it all together

Client

Query

0x2fd4e1c6
0xda39a3ee

Client Replica

Tentative Result

Database Server
Putting it all together
Putting it all together
Putting it all together

Client

Query

Client Replica

Tentative Result

0x2fd4e1c6
0xda39a3ee

0x2fd4e1c6
0x8c3aea3b

Authoritative Result

Database Server

Query

0xda39a3ee

Query
Putting it all together

Client → Client Replica → Database Server

Query → Tentative Result → Authoritative Result
Putting it all together

Client

Query

Client Replica

Tentative Result

Database Server

Query

Authoritative Result

0x2fd4e1c6

0xda39a3ee

0x8c3aea3b

9
Putting it all together

Client

Query

Tentative Result

Client Replica

Database Server

Authoritative Result

Query

0x2fd4e1c6
0xda39a3ee
0x8c3aea3b

0x2fd4e1c6
0xda39a3ee

Putting it all together

Client

Query

Client Replica

Tentative Result

Database Server

Authoritative Result

0x2fd4e1c6
0xda39a3ee
0x8c3aea3b
0x31b3887a30f020

Query

0x2fd4e1c6
0xda39a3ee
Putting it all together

Client

Query

Tentative Result

Client Replica

Authoritative Result

Database Server

Query

0x2fd4e1c6
0xda39a3ee
0x8c3aea3b

0x2fd4e1c6
0x31b3887a30f020...

9
8
Putting it all together

Client

Query

Client Replica

Tentative Result

Database Server

Authoritative Result

Query

0x2fd4e1c6
0xda39a3ee

0x31b3887a30f020...

0x2fd4e1c6
0x8c3aea3b

0x2fd4e1c6
0xda39a3ee
Putting it all together

Client

Query

Tentative Result

0x2fd4e1c6
0xda39a3ee
0x8c3aea3b

0x2fd4e1c6
0x31b3887a30f020...

Database Server

Query

Authoritative Result
Putting it all together

Client

Query

Client Replica

Tentative Result

0x31b3887a30f020...

Database Server

Authoritative Result

Query

0x2fd4e1c6

0xda39a3ee

0x8c3aea3b

9

8
Putting it all together

Client

Query

Tentative Result

Client Replica

Database Server

Query

Authoritative Result

0x2fd4e1c6
0xda39a3ee

✓

0x8c3aea3b

0x2fd4e1c6
0xda39a3ee
Putting it all together

Client

Query

Client Replica

Tentative Result

Query

0x2fd4e1c6

✓

0xda39a3ee

Database Server

0x2fd4e1c6

0xda39a3ee

0x8c3aea3b

Authoritative Result
Putting it all together

Client

Query

Tentative Result

Client Replica

Database Server

Query

Authoritative Result

0x2fd4e1c6
0xda39a3ee
0x8c3aea3b

✅ 0x2fd4e1c6
❌ 0x8c3aea3b
Evaluation

- Microbenchmarks
- MobileSales
Microbenchmarks

- Single SQL Query microbenchmark
- Fetched approx. 0.1, 0.5, and 1 MB of data
  - Only presenting 0.5 MB + 1 Mbit/s @ 100 ms RTT
- Two configurations
  - Native – Benchmark uses vendor driver (MySQL)
  - Cedar – Benchmark uses Cedar driver
- Measured data transferred and response time
Data Transferred
1 Mbit/s – 100 ms RTT

Data Transferred (MB)

Replica Freshness (%)

Native

Cedar
Response Time

1 Mbit/s – 100 ms RTT

Native

Cedar

Time (seconds)

0 20 40 60 80 100

Replica Freshness (%)
MobileSales

- MobileSales (based on TPC-App)

- Two configurations
  - Native – Benchmark uses vendor driver (MySQL)
  - Cedar – Benchmark uses Cedar driver
- “Full” hoard profile used
MobileSales Benchmark

- Online distributor system (based on TPC-App)
  - Clients perform **Interactions**: add orders, add customers, view orders, list catalog, modify catalog...

- Performance metrics
  - Throughput (Total No. of Interactions)
  - Latency (Average Interaction completion time)

- Experimental configuration
  - 40:60 Read:Write Interaction ratio
  - Benchmark Length: 5 minutes
MobileSales – Throughput

Full Hoard Profile, 50 Background Clients
MobileSales – Latency

Full Hoard Profile, 50 Background Clients
Cedar Summary

- Cedar enables efficient mobile database access
  - In low bandwidth conditions
  - While preserving consistency
- Uses Content Addressability + Client Resources