Relational Cloud: a database service for the cloud

Carlo Curino,
Evan Jones, Raluca Ada Popa, Eugene Wu, Nirmesh Malviya
Sam Madden, Hari Balakrishnan, Nickolai Zeldovich
Relational Cloud: a database service for the cloud

Carlo Curino,
Evan Jones, Raluca Ada Popa, Eugene Wu, Nirmesh Malviya
Sam Madden, Hari Balakrishnan, Nickolai Zeldovich

CSAIL MIT
Cambridge, MA
Startup success: a story of DB drama

- the idea!
- buy 1st server
- install and configure MySQL

startup popularity
time

Monday, January 17, 2011
Startup success: a story of DB drama

the idea!

buy 1st server

install and configure MySQL

setup nightly backup

buy backup server

startup popularity

time

Monday, January 17, 2011
Startup success: a story of DB drama

- the idea!
- buy 1st server
- install and configure MySQL
- setup nightly backup
- hire DBA
- implement fail-over
- buy backup server
- buy 2nd server
- crash (16h downtime)

Dotted line represents startup popularity over time.
Startup success: a story of DB drama

Monday, January 17, 2011
Startup success: a story of DB drama

- **the idea!**
- **buy 1st server**
- **install and configure MySQL**
- **setup nightly backup**
- **hire DBA**
- **implement fail-over**
- **manual partitioning of DB**
- **implement home-made load balancing**
- **migrate to new DBMS**
- **buy backup server**
- **crash (16h downtime)**
- **buy 2nd server**
- **slashdotted (4h downtime)**
- **buy 4 more servers**
- **foreseen growth**
- **acquire commercial DB and + HW**
- **hire new DBA**

*Monday, January 17, 2011*
Startup success: a story of DB drama

- the idea!
- buy 1st server
- install and configure MySQL
- setup nightly backup
- hire DBA

- buy backup server
- crash (16h downtime)
- implement fail-over
- manual partitioning of DB
- implement home-made load balancing
- migrate to new DBMS

- buy 2nd server
- slashdotted (4h downtime)
- foresee growth
- hire new DBA
- set up DB snapshots and OLAP functionalities

- buy 4 more servers
- acquire commercial DB and + HW
- acquire analytical package + HW

- need analytics
- set up DB snapshots and OLAP functionalities

- startup popularity

Monday, January 17, 2011
WE SEE:
- Fun data management problems!

THEY SEE:
- stressful, non-core-business, technical challenges
- up-front costs and unpredictable results

Monday, January 17, 2011
Startup success: a story of DB drama

Monday, January 17, 2011
Startup success: a story of DB drama
WHAT IS WRONG WITH THIS PICTURE?

**HW resources are under utilized:**
- peak-provisioning
- HW for infrequent tasks
- low power-efficiency

**Same problems solved over and over:**
- hw/sw selection
- configuration and tuning
- scalability and load balancing
Database as a Service

- Transactional, Relational DB service
  - hide complexity
  - exploit resource pooling
  - increase automation
  - (both for private and public cloud)
• **Existing Commercial DB Services:**
  • Amazon RDS, SQL Azure (*and many others*)

• **What they got right:**
  • simplified provisioning/deployment
  • reduced administration/tuning headaches

• **What is still missing?**
  • workload placement (to reduce hw cost)
  • automatic partitioning (to scale out)
  • encryption (to achieve data privacy)
Relational Cloud: Key Contributions

Workload Placement [under submission]
- consolidation up to 17:1

Automatic Partitioning [pvldb2010]
- matches or outperforms manual sharding

Provable Data Privacy [under submission]
- run SQL over encrypted data
- low overhead (22.5% impact on TPC-C throughput)
Relational Cloud: Key Contributions

Workload Placement \textit{[under submission]}

\begin{itemize}
  \item consolidation up to 17:1
\end{itemize}

Automatic Partitioning \textit{[pvldb2010]}

\begin{itemize}
  \item matches or outperforms manual sharding
\end{itemize}

Provable Data Privacy \textit{[under submission]}

\begin{itemize}
  \item run SQL over encrypted data
  \item low overhead (22.5\% impact on TPC-C throughput)
\end{itemize}
Relational Cloud Architecture

Users
(and their workloads)
Relational Cloud Architecture

Unmodified DBMS backends

Postgres  MySQL  MySQL  Postgres  MySQL
Relational Cloud Architecture

monitoring & admin

workload-aware consolidation

Placement

Postgres  MySQL  MySQL  Postgres  MySQL
Relational Cloud Architecture

workload-aware automatic sharding

Partitioning
Placement

monitoring & admin
Relational Cloud Architecture

Partitioning & Placement

Postgres, MySQL, MySQL, Postgres, MySQL
Relational Cloud Architecture

monitoring & admin
Partitioning
Placement
Relational Cloud Architecture

distributed query/transactions high availability via replication
Relational Cloud Architecture

cloud run-time

SQL over encrypted data (adjustable encryption)

client run-time

CryptDB

monitoring & admin

en/decryption query rewriting

Placement

Postgres MySQL MySQL Postgres MySQL

Frontend

Monday, January 17, 2011
Relational Cloud Architecture

cloud run-time

client run-time

client run-time

client run-time

monitoring & admin

Partitioning

Placement

Frontend

CryptDB

CryptDB

CryptDB

Postgres

MySQL

MySQL

Postgres

MySQL
Workload Placement

Scenario:

- Each workload initially run on a dedicated server
- Consolidate DB machines

Problem Definition:
Allocate workloads to servers in a way that:
1) minimizes number of servers used
2) balances load across servers
3) maintains performance unchanged
Workload Placement

DBMSs tend to use all available resources
Workload Placement

measure resource utilization

estimate combined load

numerical models

resource non-linearities
Workload Placement

**measure resource utilization**

- W1
- W2
- W3

**estimate combined load**

- CPU
- RAM
- Disk I/O

**find optimal assignment**

- Numerical models
- Non-linear programming

\[
\text{minimize} \quad \sum_{i,j} (C_{1i} x_{ij} + C_{2i} z_{ij} + C_{3i} s_{ij}) \cdot \text{signum}(\sum_j x_{ij}) \\
\text{subject to} \quad \forall i \sum_j x_{ij} = R_i; \\
\forall j \text{ max}(\sum_i CPU_{k_i} x_{ij}) < \text{MaxCPU}_j; \\
\forall j \text{ max}(\sum_i MEM_{k_i} x_{ij}) < \text{MaxMEM}_j; \\
\forall j \text{ diskModel}(DISK_{k_i}, x_{ij}) < \text{MaxDISK}_j; \\
\ldots \\
\text{additional placement constraints} \\
\forall i, j x_{ij} \in N; 0 \leq x_{ij} \leq 1
\]

non-linear constraints and objective function
Non-Linear Integer Constraints:

- No overcommit of HW using:
  - historical resource time-series
  - combined resource estimation
- Each workload is assigned
- HA via replication (e.g., W2)
Non-Linear Integer Constraints:

- No overcommit of HW using:
  - historical resource time-series
  - combined resource estimation
- Each workload is assigned
- HA via replication (e.g., W2)
Workload Placement

Non-Linear Integer Constraints:

- No overcommit of HW using:
  - historical resource time-series
  - combined resource estimation
- Each workload is assigned
- HA via replication (e.g., W2)

<table>
<thead>
<tr>
<th>Workloads</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>W2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>W3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>W5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>W6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>W7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Workload Placement

Non-Linear Integer Constraints:

- No overcommit of HW using:
  - historical resource time-series
  - combined resource estimation
- Each workload is assigned
- HA via replication (e.g., W2)

<table>
<thead>
<tr>
<th>Workloads</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>W2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>W3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>W5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>W6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>W7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Objective function:

• minimize servers (use \textit{SIGNUM})

• maximize balance (use \textit{EXP})
• Validated our approach on synthetic data

• Estimated real-world impact:
  • Huge potential consolidation: 6:1 to 17:1
Partitioning

Why:
- scalability
- manageability

Problem Definition:
Partition the database into N chunks in a way that maximizes the workload performance
Partitioning

Why:

- scalability

**KEY TO SCALABILITY (OLTP/Web):**

- Limit percentage of distributed transaction

Problem Definition:

Partition the database into N chunks in a way that maximizes the workload performance

Monday, January 17, 2011
Graph-based Partitioning

**INPUT**
- Database
- Workload Trace

**GRAPH REPRESENTATION**
- Transaction edges
- **PARTITION 0**
- **PARTITION 1**

**EXPLANATION**
- P: 0  ID < 4
- P: 1  ID >= 4

**EXAMPLE**
- BEGIN UPDATE account SET bal=bal-1k WHERE id=1;
- BEGIN UPDATE account SET bal=bal+1k WHERE id=2;
- COMMIT
- BEGIN UPDATE account SET bal=bal+1k WHERE bal < 100k;
- COMMIT
- BEGIN SELECT * FROM account WHERE id IN {1,3};
- ABORT
- BEGIN UPDATE account SET bal=60k WHERE id=2;
- SELECT * FROM account WHERE id=5;
- COMMIT
- BEGIN UPDATE account SET bal=bal*1.1 WHERE id >= 4;
- COMMIT

Monday, January 17, 2011
Graph-based Partitioning

INPUT
- Database
- Workload Trace

WORKLOAD TRACING
BEGIN UPDATE account SET bal=bal-1k WHERE id=1; COMMIT
BEGIN UPDATE account SET bal=bal+1k WHERE id=2; COMMIT
BEGIN UPDATE account SET bal=bal+1k WHERE bal < 100k; COMMIT
BEGIN SELECT * FROM account WHERE id IN {1,3}; ABORT
BEGIN UPDATE account SET bal=60k WHERE id=2; SELECT * FROM account WHERE id=5; COMMIT
BEGIN UPDATE account SET bal=bal*1.1 WHERE id >= 4; COMMIT

EXPLANATION
- P: 0 ID < 4
- P: 1 ID >= 4

Database Input (logs+preprocessing)
Graph-based Partitioning

INPUT
Database
Workload Trace

GRAPH REPRESENTATION
Graph Partitioning (METIS)

EXPLANATION
P: 0  ID < 4
P: 1  ID >= 4

Database Graph Partitioning (METIS)

Monday, January 17, 2011
Graph-based Partitioning

INPUT
Database

Workload Trace

GRAPH REPRESENTATION

EXPLANATION

Classification (Decision Tree)

ID < 4
ID >= 4

Monday, January 17, 2011
Graph Partitioning Demo

- Example inspired by YCSB
- Single table, short scans
Partitioning Results

- SCHISM
- MANUAL
- REPLICATION
- HASHING

<table>
<thead>
<tr>
<th>Method</th>
<th>TPCC 50 Warehouses (10-way partitioning)</th>
<th>Epinions.com (2-way partitioning)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100%</td>
<td>62.1%</td>
</tr>
<tr>
<td>SCHISM</td>
<td>10.8%</td>
<td>4.5%</td>
</tr>
<tr>
<td>MANUAL</td>
<td>10.8%</td>
<td>6%</td>
</tr>
<tr>
<td>REPLICATION</td>
<td></td>
<td>8%</td>
</tr>
<tr>
<td>HASHING</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Monday, January 17, 2011
Conclusions

• Database as a Service has real potential

• Key Features to fully enable DBaaS
  • Workload Placement (up to 17:1 consolidation)
  • Automatic Partitioning (matches manual partitioning)
  • Provable Privacy (22.5% performance impact)

• What’s next?
  • Live Migration
  • Dynamic reallocation/repartitioning
Conclusions

• Database as a Service has real potential

• Key Features to fully enable DBaaS
  • Workload Placement (up to 17:1 consolidation)
  • Automatic Partitioning (matches manual partitioning)
  • Provable Privacy (22.5% performance impact)

• What’s next?
  • Live Migration
  • Dynamic reallocation/repartitioning

For follow-up comments and job-offers: curino@mit.edu