DynaMast

Adaptive Dynamic Mastering for Replicated Systems

Michael Abebe
Brad Glasbergen
Khuzaima Daudjee

mtabebe@uwaterloo.ca
ICDE (April 2020)
tiny.cc/dynamast
Single Database
Single Database

Single-site transactions
Single Database
Single Database
Single Database

Overloads site

C D
How to scale the database?

Single-master

Multi-master
How to scale the database?

Single-master

Multi-master
Single-Master

Writers

Master

Replica

Propagate
Single-Master

Writer

Readers

Single-site transactions

Master

Replica

A B

C D

a b

c d

Propagate
Single-Master

Writers

Master

A
B
C
D

Propagate

Replica

a
b
c
d

Readers
Single-Master

Writers

Replica

Master

Propagate

A B C D

a b c d
Single-Master

Writers

Overloads site

Readers

Master

Replica

Propagate

A B

C D

a b

c d
How to scale the database?

Single-master

Multi-master
Multi-Master

\[
\text{W[ A ]} \\
\text{A B c d}
\]

\[
\text{W[ D ]} \\
\text{a b C D}
\]

Site 1

Site 2
Multi-Master

Load distribution

Site 1
A B c d

Site 2
a b C D
Multi-Master

Site 1

A B c d

prepare

Site 2

a b C D

commit

W[ A, C ]

W[ C ]
Costly Coordination Protocol
Multi-Master

Load distribution

Costly Coordination Protocol

Single-Master

Single-site transactions

Overloads site
How to provide:

- Load distribution
- Single-site transactions

Dynamic Mastering
Dynamic Mastering

\[ W[ A, C ] \]

Site 1

\[ AB \]
\[ cd \]

Site 2

\[ \text{Remaster A} \]

\[ ab \]
\[ CD \]
Dynamic Mastering

Site 1

Site 2

Remaster A

W[ A, C ]
Dynamic Mastering

Single-site transactions

Site 1
a B c d

Site 2
A b C D

Remaster A

W[ A, C ]
Dynamic Mastering

Site 1

\[ W[ B ] \]

Site 2

\[ W[ A, C ] \]
Dynamic Mastering

Site 1
a B c d

Site 2
A b C D

W[ B ]
W[ A, C ]

Load distribution
Dynamic Mastering

Outside transaction boundaries

Site 1

Site 2

Remaster A

W[A, C]

W[C]
DynaMast

Distributed and replicated database system

Employs adaptive dynamic mastering

Provides both single-site transactions and load balance
Dynamic Mastering Challenges

How to perform remastering efficiently?

How to decide where to master data?
Dynamic Mastering Challenges

How to perform remastering efficiently?

How to decide where to master data?
Dynamic Mastering

\( W[ A, C ] \)

Site 1: A B c d

Site 2: a b C D

Remaster A
Dynamic Mastering

Exploit replicas

Site 1
a B c d

Site 2
A b C D

Remaster A

W[ A, C ]
Exploiting Replicas

\[ W[ A ] \]

Site 1

\[
\begin{array}{cc}
A & B \\
c & d \\
\end{array}
\]

Site 2

\[
\begin{array}{cc}
a & b \\
C & D \\
\end{array}
\]
Exploiting Replicas

$W[ A ]$

$W[ A, C ]$

Site 1

Remaster A

Site 2
Exploiting Replicas

Inconsistent!

Site 1

Site 2

Remaster A

W[ A, C ]
Ensuring Consistency

Old master must **not** allow new updates

**release** mastership

New master must have **all previous updates**

**grant** mastership
Ensuring Consistency

release

W[ A, C ]

Site 1

A B
c d

Site 2

a b
C D
Ensuring Consistency

Site 1

Site 2

release

grant

Propagate

W[ A, C ]

a B

c d

a b

C D

37
Ensuring Consistency

Site 1: a B c d
Propagate

W[ A, C ]

grant

Site 2: a b C D
Ensuring Consistency

Site 1

a  B

c  d

Site 2

A  b

C  D

W[ A, C ]

grant
Ensuring Consistency

New master must have all previous updates

New master was a lazy replica of old master

Little time spent waiting for updates

tiny.cc/dynamast
Dynamic Mastering Challenges

How to perform remastering efficiently?

How to decide where to master data?
Where to master?

\[ W[ A, C ] \]

Site 1

Site 2

A \quad B \quad C \quad D

a \quad b

c \quad d

Remaster A
Where to master?

Site 1
a B
c d

Site 2
A b
C D

W[ A, C ]
Where to master?

Site 1

Site 2

Remaster B

W[ A, B ]
Where to master?

Site 1

Site 2

Single-master!
Where to master?

W[ A, B ]

Remaster A

Site 1

Site 2
Where to master?

\[ W[ A, B ] \]

Site 1

A B c d

Site 2

a b C D

Remaster A
Where to master?

Ping-pong mastership

Site 1

Site 2

Remaster A

W[ A, C ]
Dynamic Mastering Strategies

Site Selector

Makes adaptive decisions

Site 1

A B
c d

Site 2

a b
C D
Dynamic Mastering Strategies

**Track:** data access patterns, site load

**Quantify benefit** of remastering to each site

- Load distribution
- Future remastering
- Update propagation

**Remaster** to site that maximizes benefit

[ tiny.cc/dynamast ]
How well does it work?

Workloads

**YCSB**  Scans & Multi-Key Read-Modify-Write

Uniform and Skew Access Patterns

**TPC-C**  Complex updates and reads
How well does it work?

Comparisons

- Single-Master
- Multi-Master
- Partition-Store
- LEAP

Replicated

Single Copy
YCSB with Skew - Throughput

- DynaMast
- LEAP
- Single-Master
- Partition-Store
- Multi-Master

![Graph showing throughput comparison](image)

- Avg. Throughput (txn/sec) vs. Clients
- DynaMast shows a 1.6x improvement compared to Single-Master.
- Multi-Master shows a 10x improvement compared to Single-Master.
YCSB with Skew - Routing

- DynaMast
- Single-Master
- Partition-Store
- Multi-Master
- LEAP

% Txns Routed to Site

Sites:
- Site 1
- Site 2
- Site 3
- Site 4

3.6x
TPC-C – New-Order Latency

DynaMast  Single-Master  Partition-Store  Multi-Master  LEAP

96% reduction

40%
DynaMast Takeaways

**Dynamic mastering** guarantees single-site transactions

Use replicas to remaster **efficiently** outside transaction boundaries

**Adaptive** site selector strategies **balance** load and **minimizes** future remastering

[tiny.cc/dynamast]
Extra slides
DynaMast Learns Over Time

The diagram shows the average throughput (txn/sec) over time in seconds into the workload. The throughput increases over time, indicating that DynaMast learns over time. The graph is annotated with an arrow indicating a 1.6x improvement.
TPC-C – New-Order Latency

- LEAP
- Partition-Store Multi-Master
- Single-Master DynaMast

![Graph showing TPC-C New-Order Latency with different systems and percent cross-warehouse transactions.](image)
Comparisons – Single-Master

Writers

Site 1

A B C D

Readers

Site 2

a b c d
Comparisons – Multi-Master

Site 1

A B
C d

Site 2

a b
C D

prepare
commit

W[ A, C ]
Comparisons – **Partition-Store**

![Diagram showing two sites, Site 1 and Site 2, with data elements A, B, C, and D. The diagram illustrates the process of preparing and committing data.]
Comparisons – LEAP

W[ A, C ]

request C

Site 1

Site 2

A
B

C
D
Comparisons – LEAP

W[ A, C ]

transfer C

Site 1

Site 2
Comparisons – LEAP

W[ A, C ]

Site 1
A B C

Site 2
D