

Database Scalability {Patterns}



 **OmniTI** / Robert Treat

robert treat
omniti
postgres
oracle - mysql
mssql - sqlite - nosql

What are “Database Scalability Patterns?”

Part Design Patterns

Part Application Life-Cycle

Part Design Patterns

Part Application Life-Cycle

Part Design Patterns

Part Application Life-Cycle

Part Design Patterns

Part Application Life-Cycle

MyFirstDatabase



Vertical Partitioning



Vertical Scaling



Vertical Scaling



Vertical Scaling



Federated Data Storage

“sharding”

Horizontal Partitioning

Read Slaves

Multi-Master

Horizontal Scaling

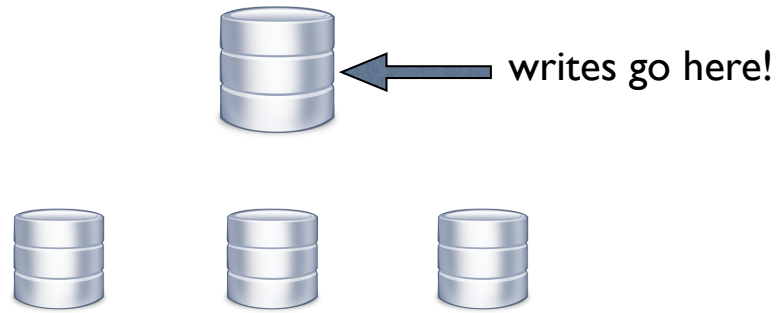
Read Slaves / Master - Slave

Scale Read Load



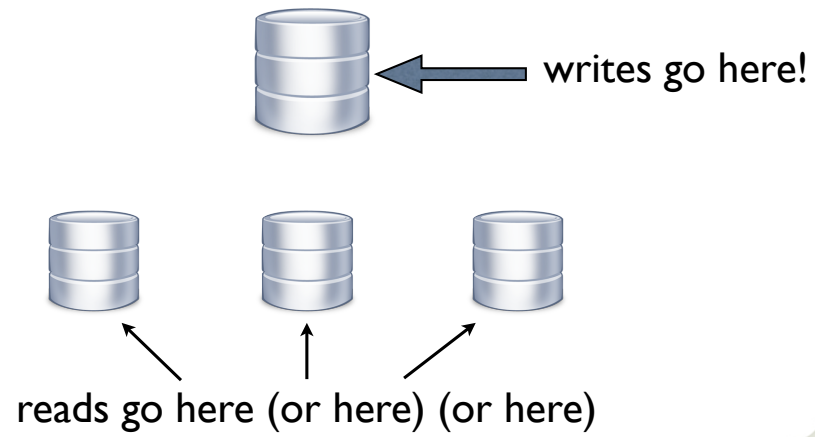
Read Slaves / Master - Slave

Scale Read Load



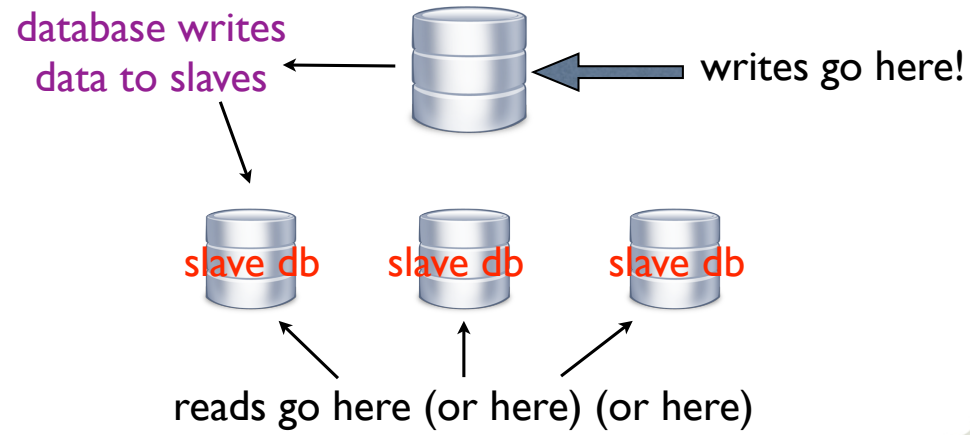
Read Slaves / Master - Slave

Scale Read Load



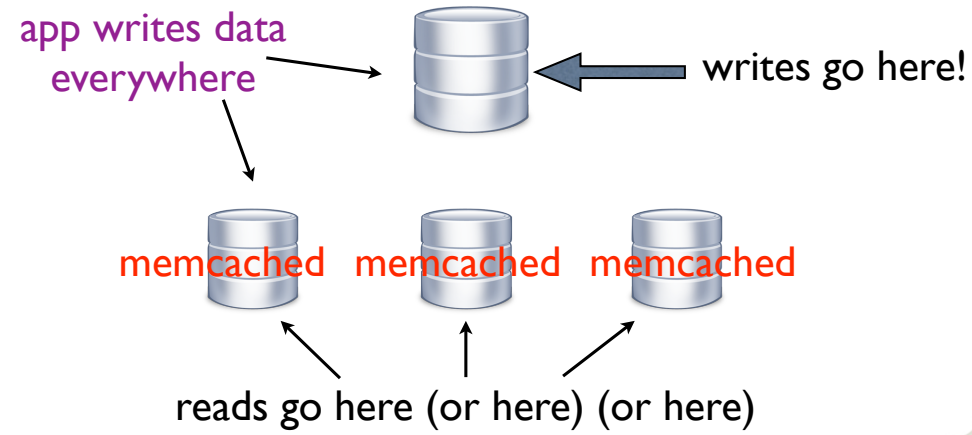
Read Slaves / Master - Slave

Scale Read Load



Read Slaves / Master - Slave

Scale Read Load



Read Slaves / Master - Slave

Scale Read Load

- Typically
 - Full Copy of Data
 - On Each Node
 - Asynchronous

Read Slaves / Master - Slave

Scale Read Load

- Typically
 - Full Copy of Data
 - On Each Node
 - Asynchronous
- Consider
 - Partial Copy
 - Synchronous
 - Don't use a RDBMS?

Read Slaves / Master - Slave

Scale Read Load

- Typically
 - Full Copy of Data
 - On Each Node
 - Asynchronous
- Consider
 - Partial Copy
 - Synchronous
 - Don't use a RDBMS?

Requires Application Changes

“easy”

Scaling Writes

“not easy”

Multi-Master

many different ways to implement this,
few that actually work in production



Multi-Master

many different ways to implement this,
few that actually work in production

write to any node, database syncs data

Multi-Master

many different ways to implement this,
few that actually work in production

write to any node, database syncs data
can reduce cpu, doesn't reduce i/o

Multi-Master

many different ways to implement this,
few that actually work in production

write to any node, database syncs data
can reduce cpu, doesn't reduce i/o

failover solution
not a scalability solution

Horizontal Partitioning

“sharding”

“SOA”

Horizontal Partitioning

~~“sharding”~~

“SOA”

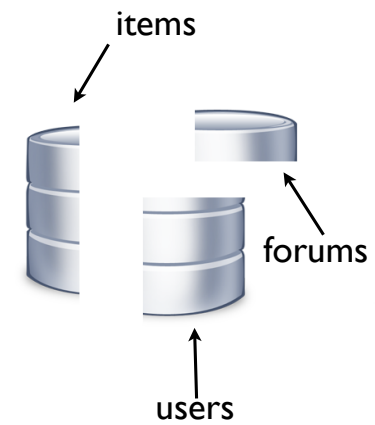
Horizontal Partitioning

- Divide schema by job operations
- Move each piece to own server
- Duplicate some data as needed



Horizontal Partitioning

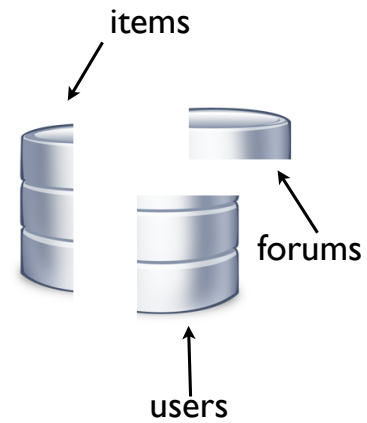
- Divide schema by job operations
- Move each piece to own server
- Duplicate some data as needed



Horizontal Partitioning

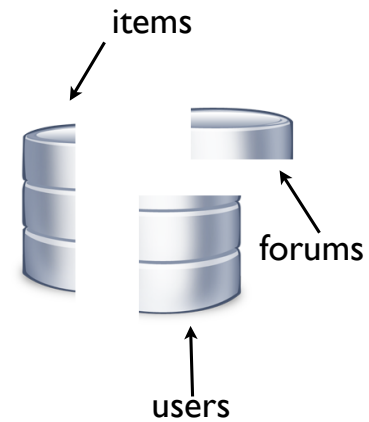
- Divide schema by job operations
- Move each piece to own server
- Duplicate some data as needed

- You must separate dependencies in the app code first!



Horizontal Partitioning

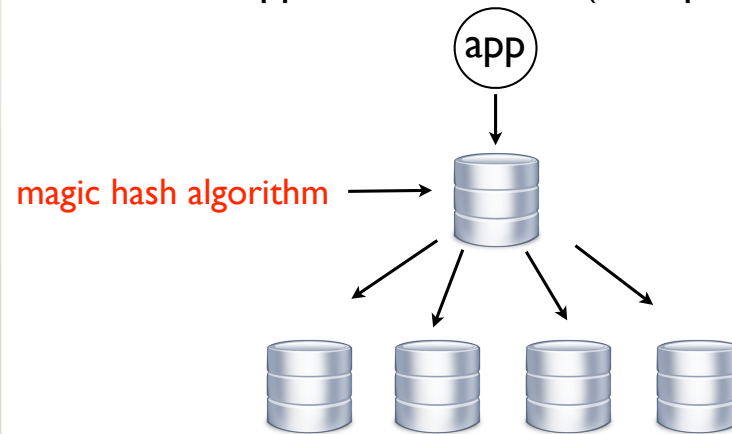
- Divide schema by job operations
 - Move each piece to own server
 - Duplicate some data as needed
-
- You must separate dependencies in the app code first!



Each node is a new instance of vertical scaling

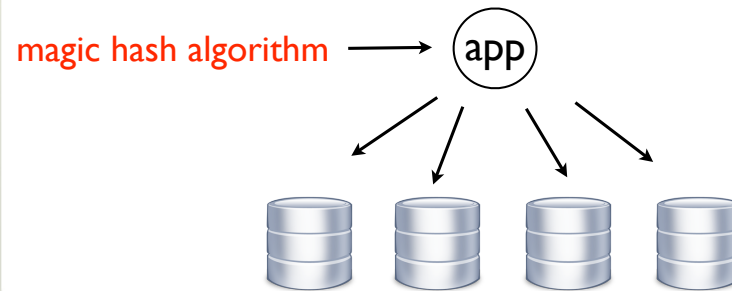
Horizontal Scaling

- data split across servers based on algorithm
- data dropped into buckets (multiple?)



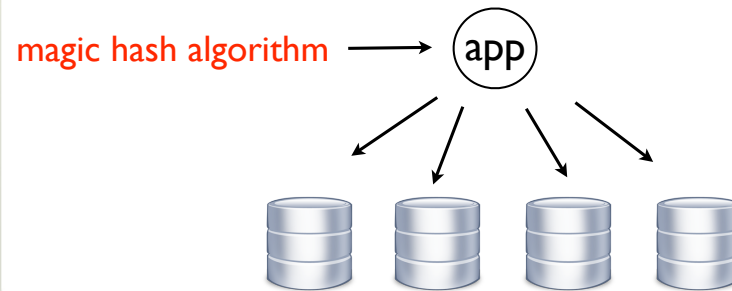
Horizontal Scaling

- data split across servers based on algorithm
- data dropped into buckets (multiple?)



Horizontal Scaling

- data split across servers based on algorithm
- data dropped into buckets (multiple?)
- someone must keep track of data, and provide lookup services



Universal Truths of Scaling Databases

Vertical Scalability is Helpful for Every Pattern

Even in a horizontally scaled, fully distributed database, the number of nodes needed is affected by vertical scalability

Universal Truths of Scaling Databases

New Nodes Are Never Free

- Add points of failure
- Add management costs
- Add complexity to architecture
- Add complexity to your app code

MyFirstDB



Vertical Partitioning



Vertical Scaling



Read Slaves



Horizontal Partitioning

tips

plan for layered data sources

read / write connections in code

use schemas to separate services

THANKS!

more:
xzilla.net
@robtreat2

omniti.com/surge
(scalability conference)