

# CourseRank Backend

Benjamin Bercovitz  
CourseRank Operations Lead  
[benjamin@courserank.com](mailto:benjamin@courserank.com)



# We live in EC2

- Rent-a-datacenter
- Rent VMs, configure firewalls, set up SAN...
- No way we could do without it
- Traffic is bursty: you need 100x the server power when big schools have registration versus off-peak operations
- We obviously aren't going to buy 100 servers and have 99 of them idle...

# We love EC2

- Made us shift our administration paradigm
- Realizes some of the ideas Paul Borrill presented in EE380 lecture last quarter (“Rethinking time in distributed systems”)
- Principle: do not log into servers and administer them. People needed  $O(n \log n)$
- Think about server as a huge shell script. The server is like a process with limited ability to modify while running

# Is EC2 safe?

- If you are paranoid enough, no shared computing resources is safe
- Datacenter network and infrastructure is probably *physically* pretty secure
- Can encrypt disks, can use IPSEC
- Hypervisor, shared network increases attack surface vs. dedicated datacenter

# Database

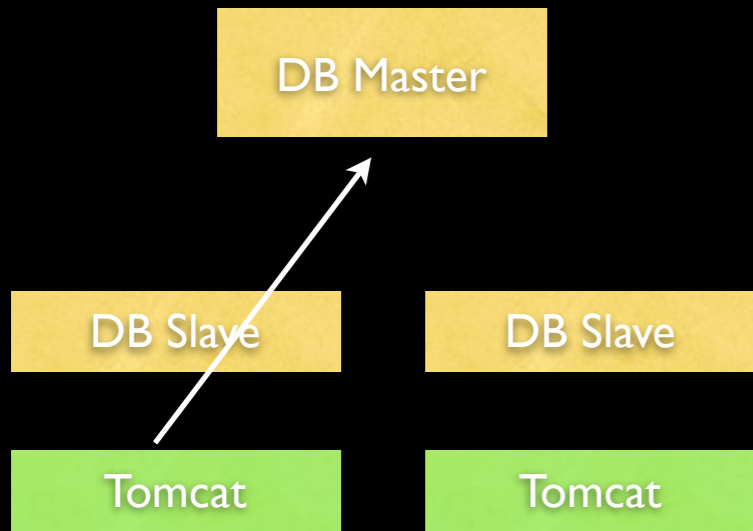
- Reads sent to local database replica
  - scales with number of webapp nodes
- Writes sent to master
  - scales with number of clusters
- (our common and expensive queries are all reads)

# Database Details

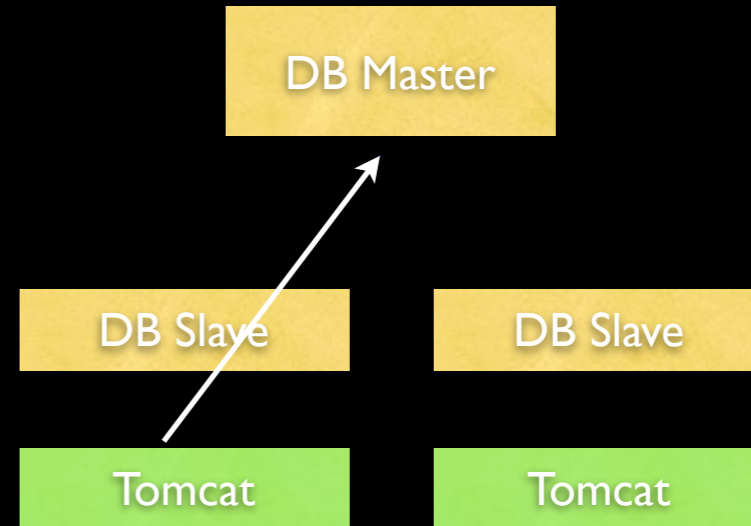
- Implementation notes:
- Two connection pools, reads use one, writes use the other
- Found reads to localhost significantly faster
- How do you handle consistency?

# Single node Read-after-write consistency

- When a query is executed on the master, a local AtomicInteger is incremented
- Update a special table row on the db master with this number
- Run reads in a loop on the local replica, wait for it to increase to the value you updated

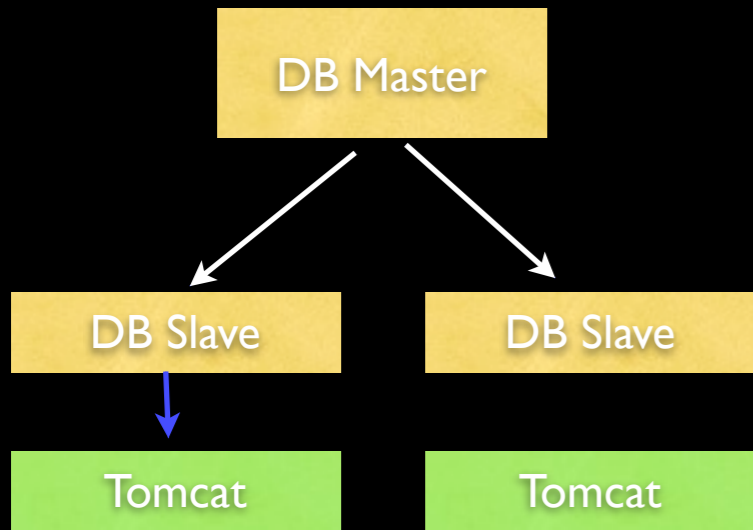


UPDATE table1 SET colA=2

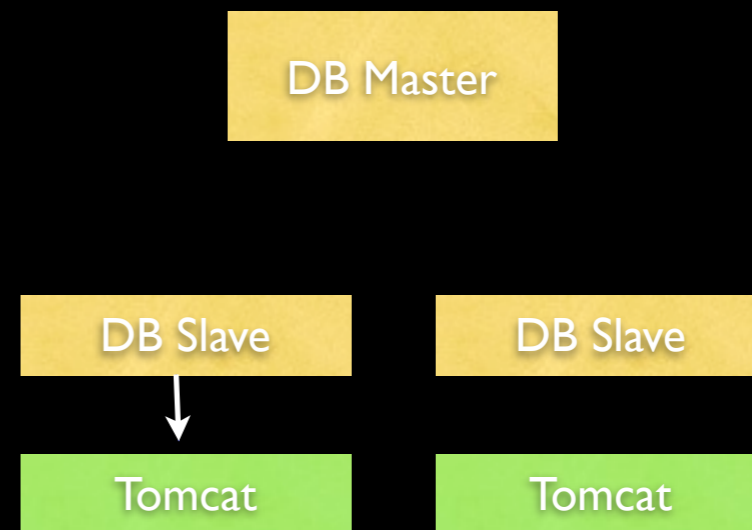


UPDATE write SET gen=12123  
WHERE node='node1'

until gen >= 12123



SELECT gen FROM write  
WHERE node='node1'



SELECT colA FROM table1

Still have a race

# Async queries

- (no longer supported, no one used it!)
- Supported putting low priority writes (e.g. logging) on a queue. Request only takes latency hit = enqueue time
- Supported running a potentially long read query. If it returns quickly, the result is included in the page output. Else, a stub is included that will ajax load the result.

# Memcached

- Two pools: local and global
- local contains objects that cannot change
- global contains objects that can change
- global acts as write-back (cache read from and written to first)
- localhost memcached is WAY faster than over LAN network

# Sessions

- We don't have them! (yet...)
- Makes requests easier to handle
- Store authentication in browser cookie, use SHA256-HMAC
- Probably will eventually store them in a memcached instance

# ExploreCourses

# System architecture

- Everything is in Lucene index
- Serialized form of data objects stored as binary field
- Database is barely used by requests
- Updates rebuild index
- ~40x faster than storing objects with well-optimized JPA

# Shout-outs

- Hector Garcia-Molina (Stanford infoLab)
- Steve Souders (Google) Front-end performance tips
- Tom Black (Stanford's Registrar)

# How can I get?

- A Google Books T-shirt with Very Hungry Caterpillar eating through the logo