

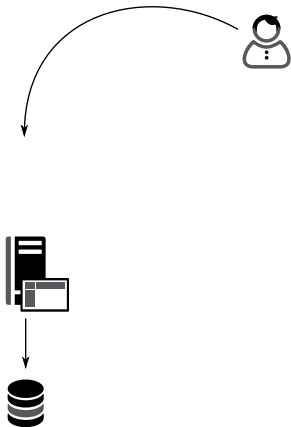
Evolution of Web Application Architecture

International PHP Conference

Kore Nordmann / @koredn / <kore@qafoo.com>
June 9th, 2015



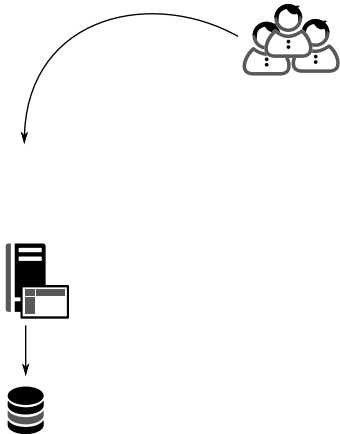
Evolution



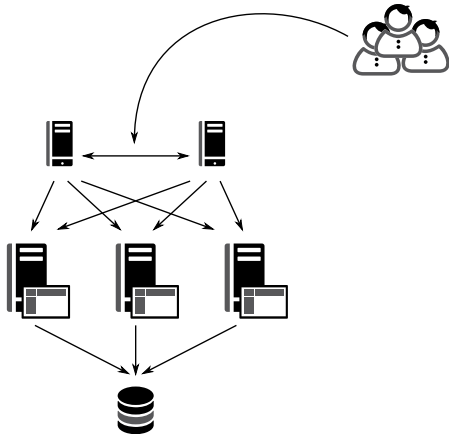
Too many visitors



Evolution



Evolution



Lessons Learned: Load Balancing

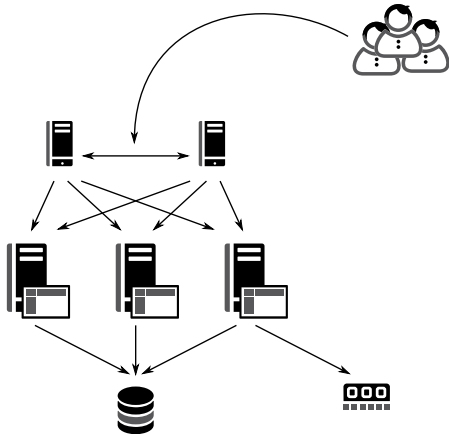
- ▶ Works because of HTTP & PHP
 - ▶ HTTP is LCoDC\$SS
 - ▶ PHP is build for shared-nothing
- ▶ Round Robin works best
 - ▶ Sticky sessions will overload certain servers



Non-sticky session – how?



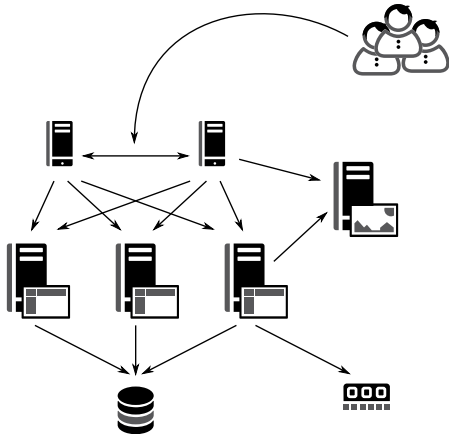
Evolution



Where to put the static data?



Evolution



Lessons Learned: Static Files

- ▶ NFS will eventually lead to dead locks
 - ▶ ... still seems the most popular solution around.
- ▶ Multiple domains can hurt performance (TCP slow start)
- ▶ Using dedicated CDN providers can help
 - ▶ Content locality

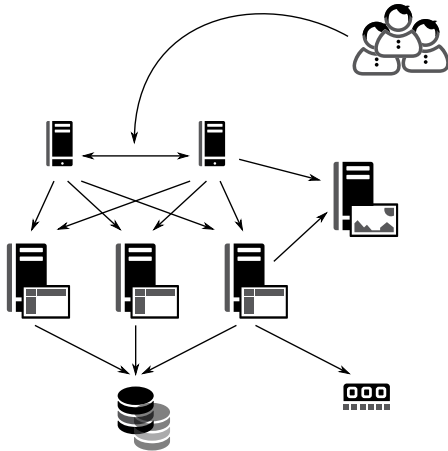


Problem

DB server too slow



Evolution



Lessons Learned: Replicate Database

- ▶ Master Slave Replication is fairly easy to set up
 - ▶ Obviously only scales READs
 - ▶ WRITEs are usually not your first problem

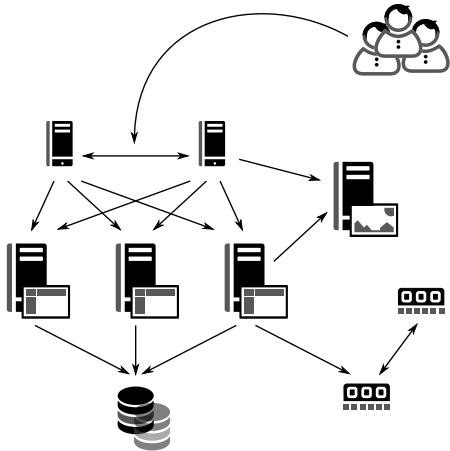


Problem

DB servers are too expensive



Evolution



Lessons Learned: Cache With Memcache

- ▶ Cache all the things in *memory*
 - ▶ Cache entities
 - ▶ Cache collections
 - ▶ Full page cache
- ▶ Cache invalidation

*There are three hard things in Computer Science:
Cache invalidation and off by one errors.*

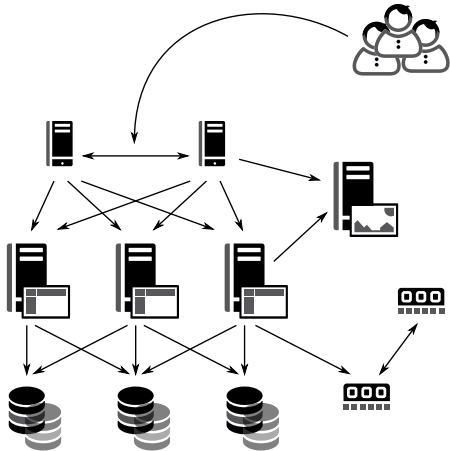
- ▶ Cache dependency calculation
- ▶ The $n + 1$ problem



Too many writes



Evolution



Lessons Learned: Sharding

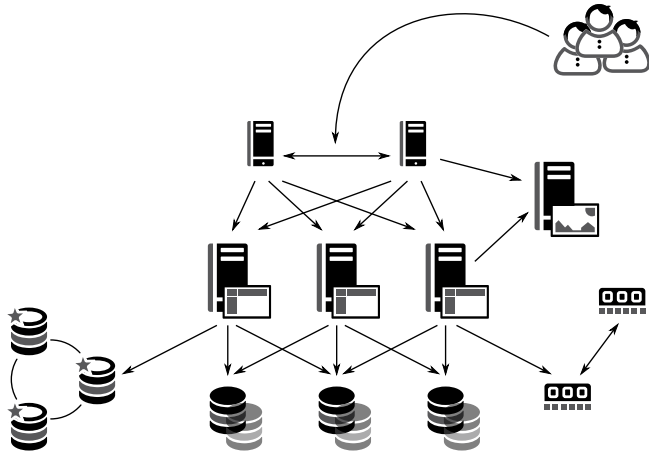
- ▶ Shard by table
 - ▶ ... or even shard by consistent hash per entity
 - ▶ No referential integrity checking
- ▶ Queries are limited to sharding solution
- ▶ Schema updates across multiple shards are *fun*



Database setup too complex



Evolution



Lessons Learned: NoSQL

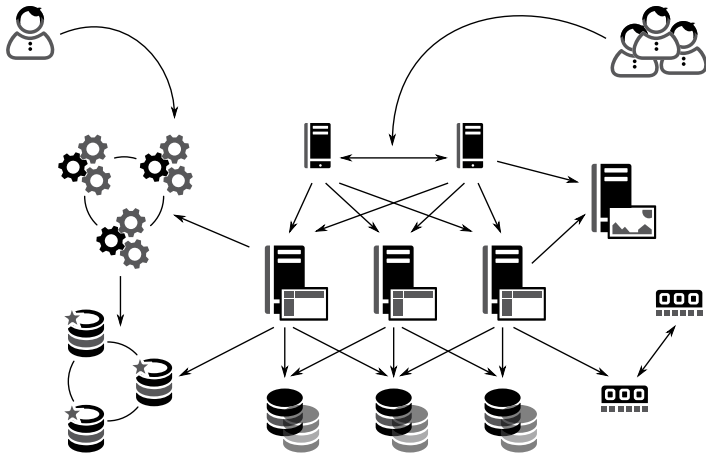
- ▶ Usually solves one problem really well:
 - ▶ Sharding
 - ▶ Multi-Master-Replication
 - ▶ Cross-shard queries
- ▶ ... we lost all relevant features from Relational Database Management Systems anyways...



Business wants to query data



Evolution



Lessons Learned: Map-Reduce

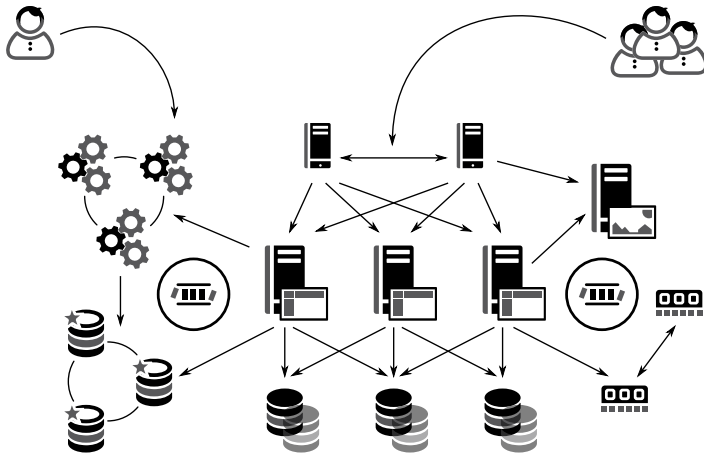
- ▶ Execute queries on distributed databases
- ▶ New query language to learn
 - ▶ Your developers write analysis scripts, instead of the business analysts writing SQL slow queries



How to orchestrate?



Evolution



Lessons Learned: Queues

- ▶ Queues can ensure data is processed asynchronously
 - ▶ Following the data flow of an action can be “tricky”
- ▶ Used to distribute data between systems



Evolution



Microservices

- ▶ Microservices *can* simplify things:
 - ▶ Separate services by concern or team
 - ▶ Decide on technology stack per service
- ▶ Microservices will also complicate things:
 - ▶ Deployment automation is a must
 - ▶ Service orchestration is still a problem
 - ▶ Service downtimes and latency must be handled gracefully
- ▶ Big Data™ will stay a problem

Lessons Learned (subjective)

- ▶ Boring technology choices will often work best
 - ▶ Just start & stay with LAMP?
- ▶ Only bring in shiny new technologies with care
 - ▶ There are enough reasons to eventually do that, though



There is no conclusion

Do not jump on every bandwagon – this includes
microservices





THANK YOU

Rent a quality expert
qafoo.com