



A Global In-memory Data System for MySQL

Daniel Austin, PayPal Technical Staff

Percona Live!

London, Oct. 25, 2011

v1.1

AGENDA

Intro: Globalizing NDB

Proposed Architecture

What We Learned

Q&A

Our Mission

"UserBase: Develop a globally distributed DB For User-related data"

- Must Not Fail (99.999%)
- Must Not Lose Data. Period.
- Must Support Transactions
- Must Be FAST
- Must Support (some) SQL
- May Be Buzzword-compliant (RFC 2119)

THE FUNDAMENTAL PROBLEM IN DISTRIBUTED DATA SYSTEMS

“How Do We Manage Reliable Distribution of Data Across Geographical Distances?”



To SQL or Not to SQL, 'Tis the Query

- Modern NoSQL Systems as solutions
 - Hive, Cassandra, Mongo, 120+ more
 - Mostly designed for 'Big Data' problems
 - Low levels of maturity
- Trade-offs:
 - Consistency vs. Availability, Fault Tolerance (dreaded CAP theorem)
 - Relational Model & X-actions dropped

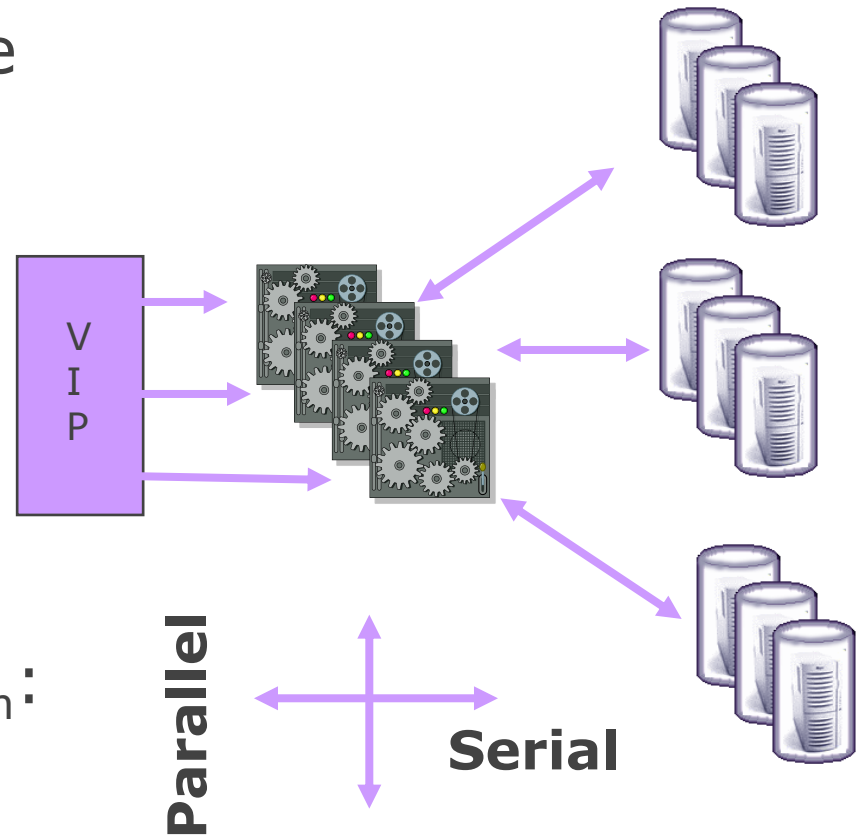
SYSTEM AVAILABILITY DEFINED

- Availability of the entire system:

$$A_{\text{sys}} = 1 - \prod_{i=1}^n (1 - \prod_{j=1}^m r_i)_j$$

- Number of Parallel Components Needed to Achieve Availability A_{min} :

$$N_{\text{min}} = \lceil \ln(1 - A_{\text{min}}) / \ln(1 - r) \rceil$$

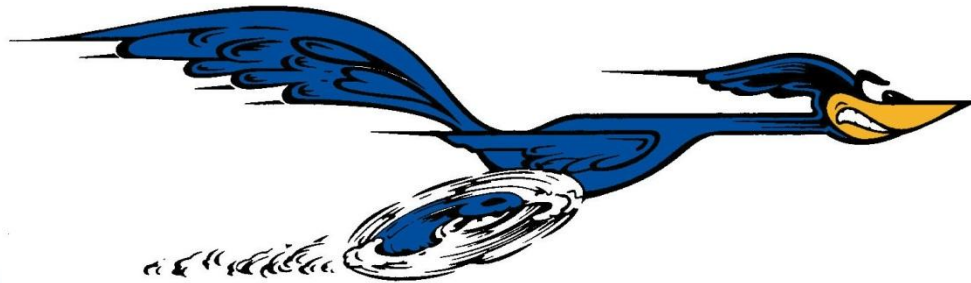


What about "High Performance"?

- Maximum lightspeed distance on Earth's Surface: ~**67** ms
- Target: data available worldwide in < 1000 ms

Sound Easy?

Think Again!



Intro: Globalizing NDB

Proposed Architecture

What We Learned

Q&A

WHY NDB?

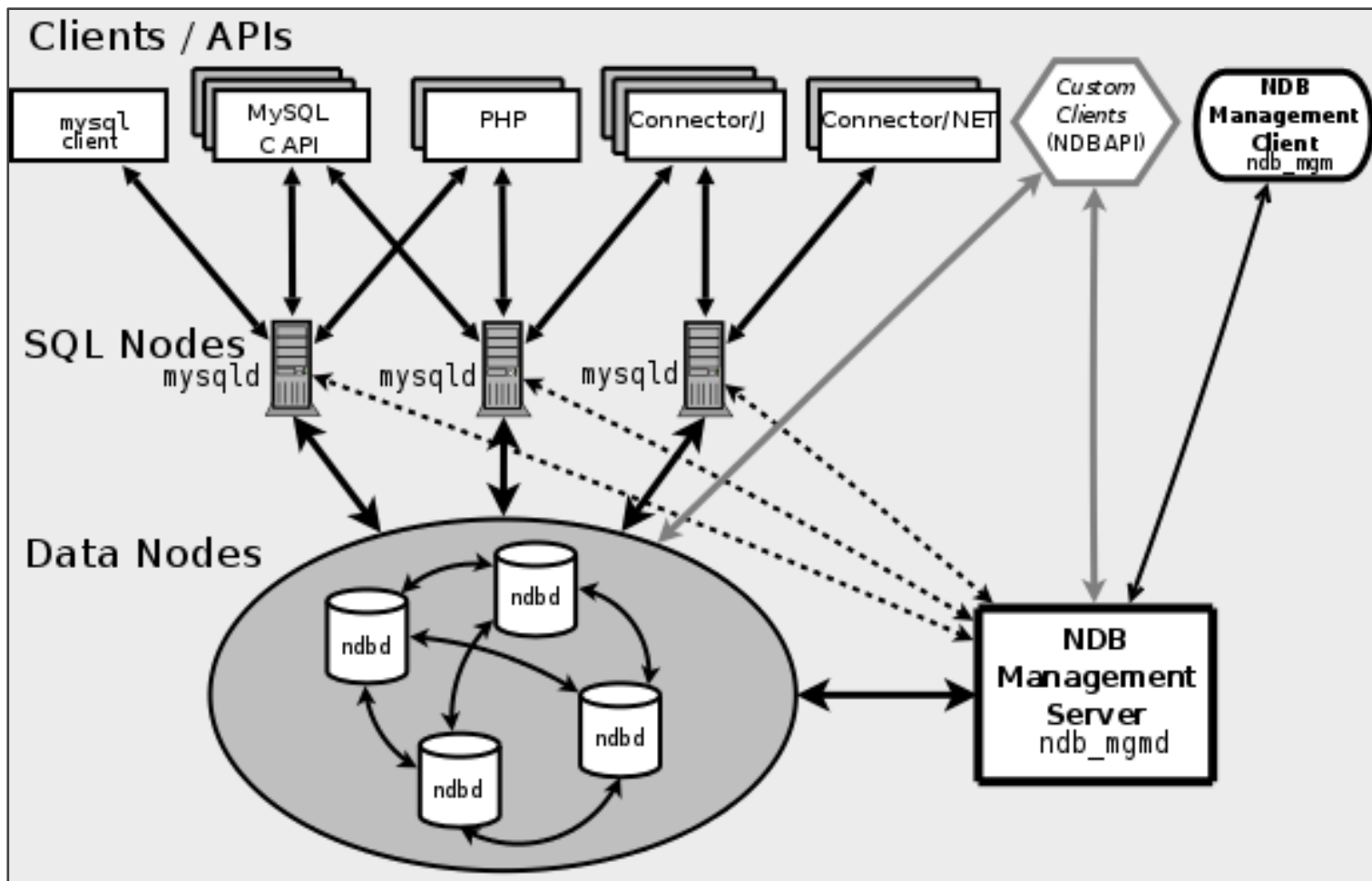
Pro

- True HA by design
 - Fast recovery
- Supports (some) X-actions
- Relational Model
- In-memory architecture = high performance
- Disk storage for non-indexed data (since 5.1)
- APIs, APIs, APIs

Con

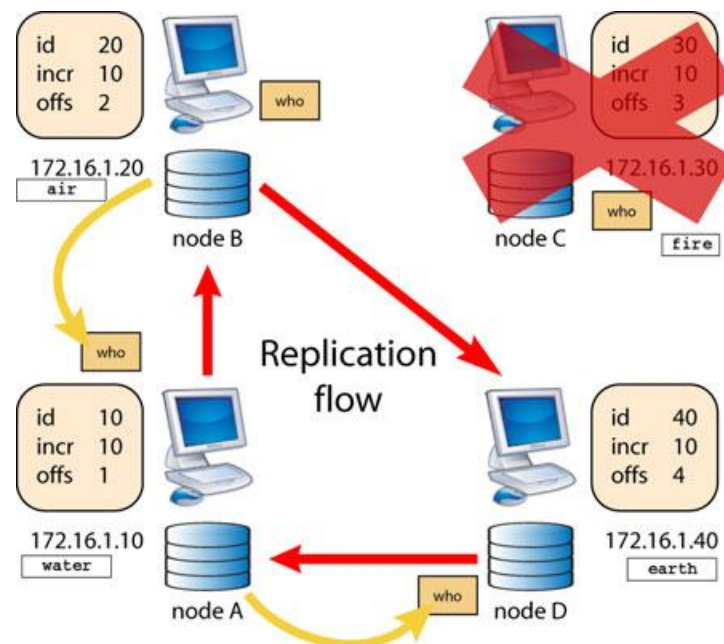
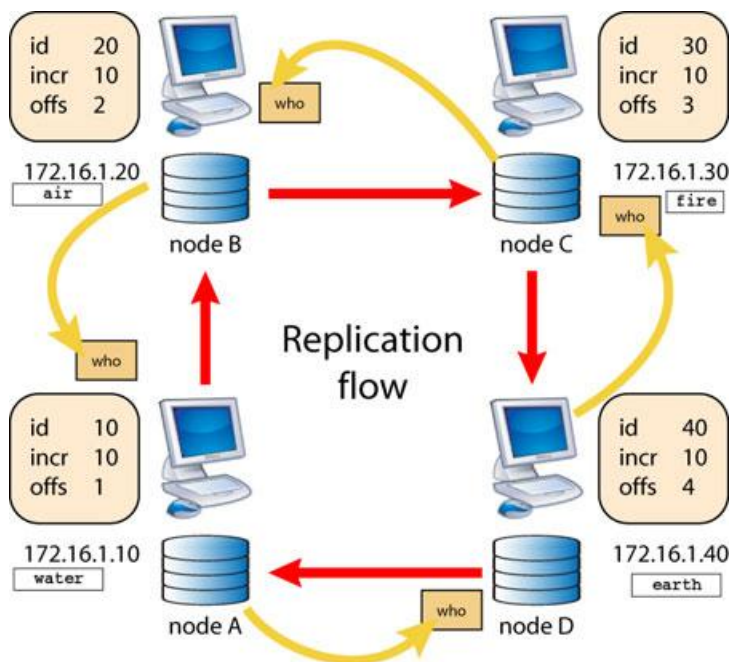
- Some semantic limitations on fields
- Size constraints (2 TB?)
 - Hardware limits also
- Higher cost/byte
- Requires reasonable data partitioning
- Higher complexity

How NDB Works in One Slide



Graphics courtesy dev.mysql.com

CIRCULAR REPLICATION/FAILOVER



Graphics courtesy O'Reilly OnLamp.com

AWS Meets NDB

- Why AWS?
 - Cheap and easy infrastructure-in-a-box
(Or so we thought! Ha!)
- Services Used:
 - EC2 (Centos 5.3, small instances for mgm & query nodes, XL for data)
 - Elastic IPs/ELB
 - EBS Volumes
 - S3
 - Cloudwatch

ARCHITECTURAL TILES

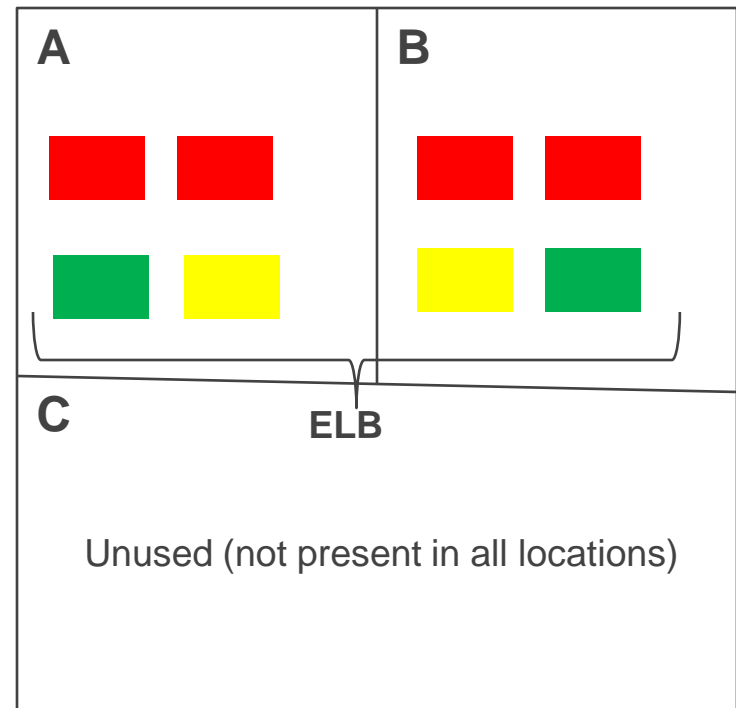
Tiling Rules

- Never separate NDB & SQL
- Ndb:2-SQL:1-MGM:1
- Scale by adding more tiles
- Failover 1st to nearest AZ
- Then to nearest DC
- At least 1 replica/AZ
- Don't share nodes
- Mgmt nodes are redundant

Limitations

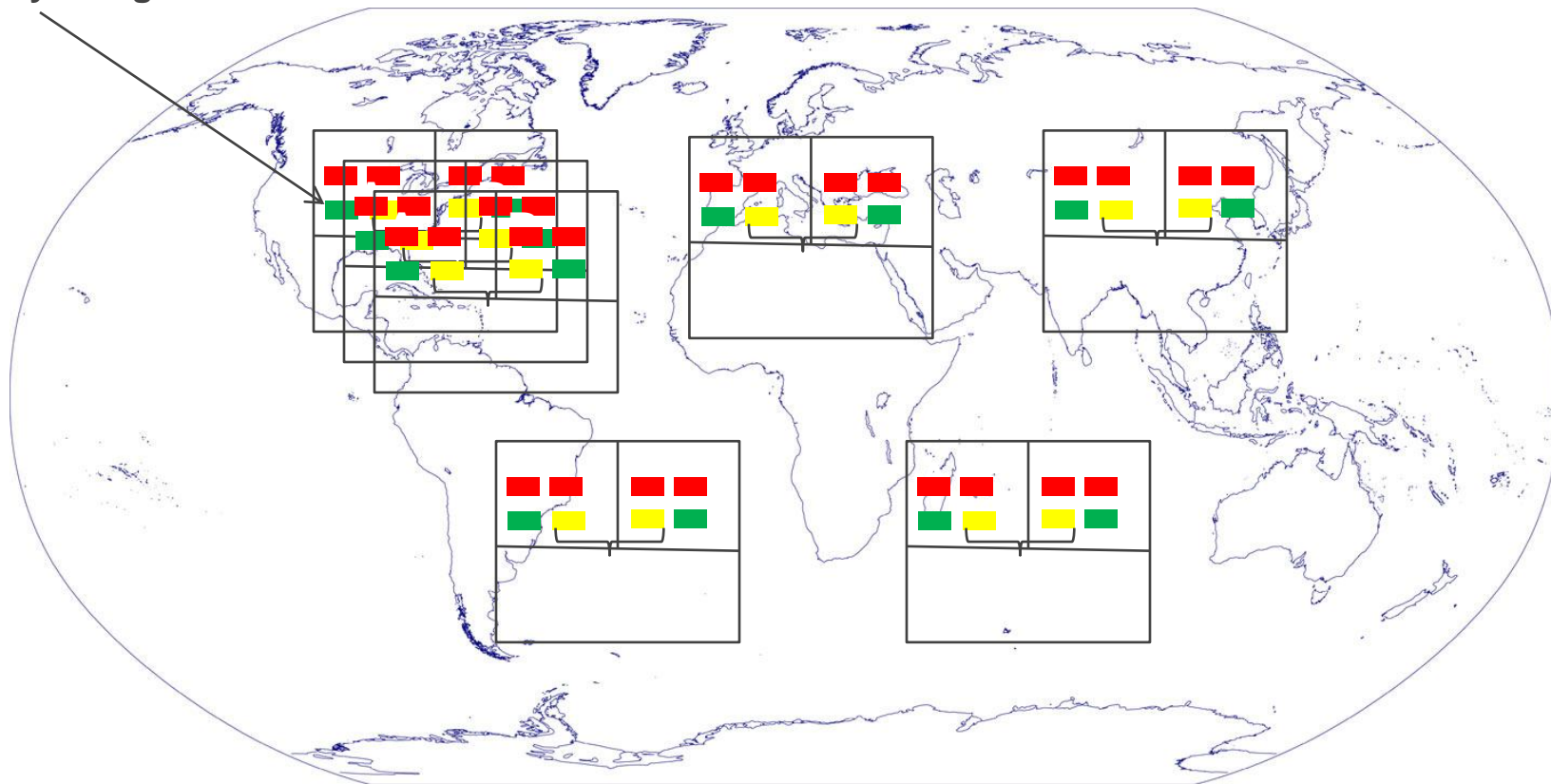
- AWS is network-bound @ 250 MBPS – ouch!
- Need specific ACL across AZ boundaries
- AZs not uniform!
- No GSLB
- Dynamic IPs
- ELB sticky sessions !reliable

AWS Availability Zones



Architecture Stack

Scale by Tiling



**5 AWS Data Centers:
US-E, US-W, TK, EU, AS**

Other Technologies Considered

- Paxos
 - Elegant-but-complex consensus-based messaging protocol
 - Used in Google Megastore, Bing metadata
- Java Query Caching
 - Queries as serialized objects
 - Not yet working
- Multiple Ring Architectures
 - Even more complicated = no way

Intro: Globalizing NDB

Proposed Architecture

What We Learned

Q&A

SYSTEM READ/WRITE PERFORMANCE (!)

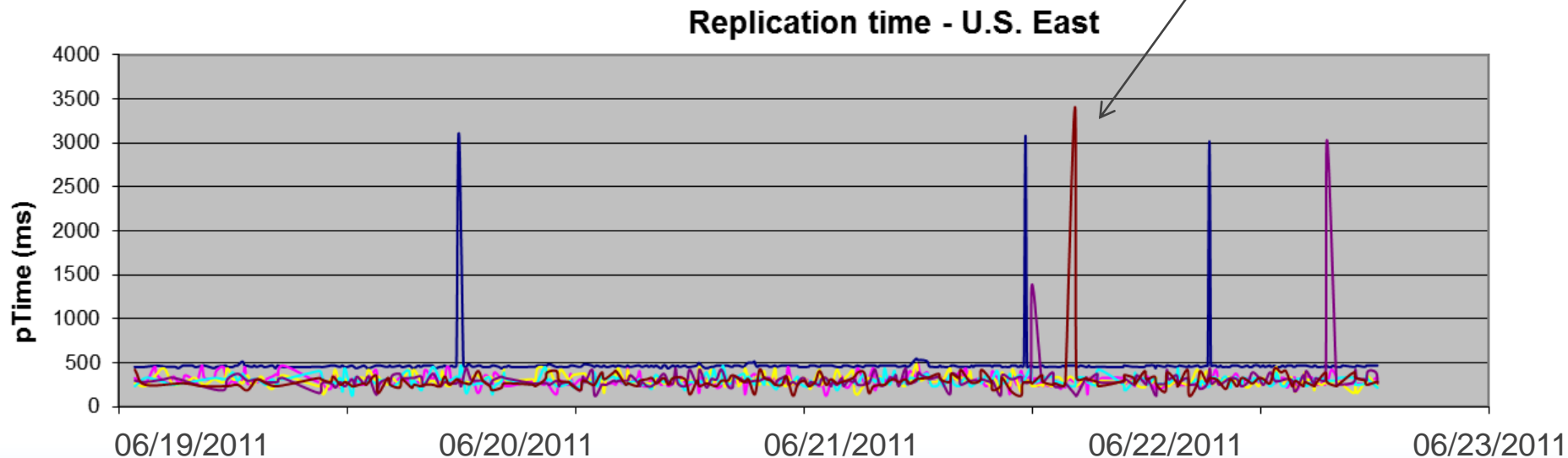
What we tested:

- 32 & 256 byte char fields
- Reads, writes, query speed vs. volume
- Data replication speeds

Results:

- Global replication < 350 ms
- 256 byte read < 10ms worldwide

In-region replication tests



Data Models and Query Optimization for NDB

- Network Latency is an obvious issue
- Data model requires all segments present in each geo-region
- Parameterized (Linked) Joins
 - SPJ technique from Clustra (see Clement Frazer's blog for details)

Commit Ordering

- Why does commit ordering matter?
- Write operators are non-commutative

$[W(d,t1),W(d,t2)] \neq 0$ unless $t1=t2$

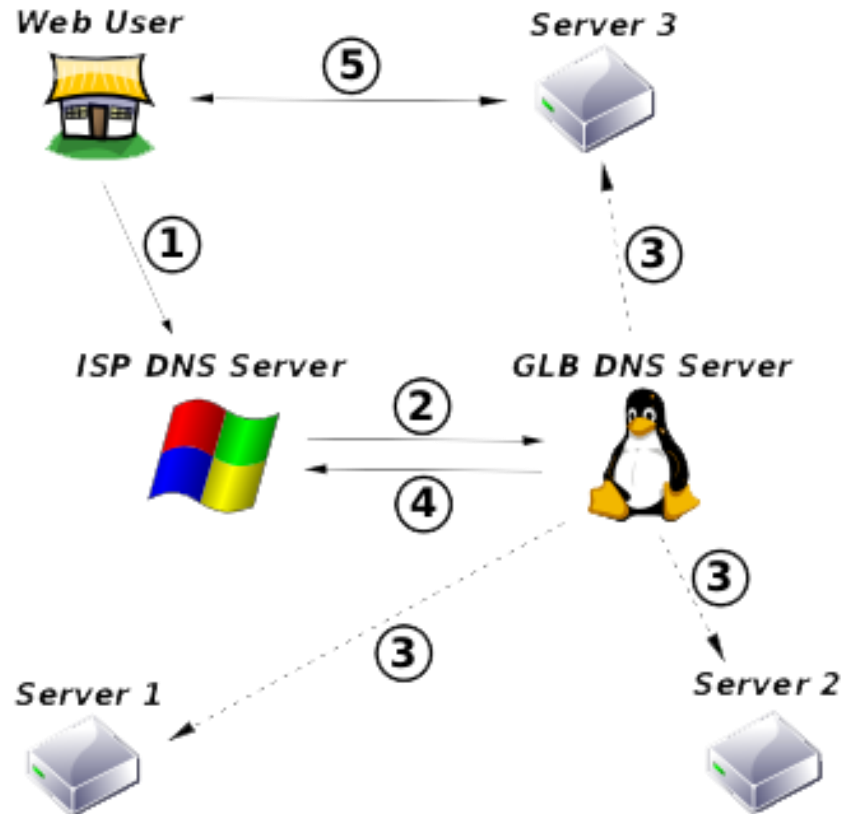
- Can lead to inconsistency
- Can lead to timestamp corruption
- Forcing sequential writes defeats Amdahl's rule
- Can show up in GSLB scenarios

Dark Side of AWS

- Deploying NDB at scale on AWS is hard
 - Dynamic IPs (use hostfile)
 - DNS issues
 - Security groups (ec2-authorize)
 - Inconsistent EC2 deployments
 - Are availability zones independent network segments?
 - No GSLB (!) (rent or buy)
- Be Prepared to struggle a bit!

NOTES ON GLOBAL LOAD BALANCING

- Don't try this at home
 - Rent or buy a solution
 - BGP-based solutions are best
- Deployment and config for AWS are tough
- We tried both Zeus and Dyn
 - Liked Dyn better, \$\$ & ease of use
- Absolutely crucial part of infrastructure!



Graphics courtesy <http://www.oes.co.th>

Hard Lessons, Shared

- Be Careful...
 - With “Eventual Consistency”-related concepts
 - ACID, CAP are not really as well-defined as we’d like considering how often we invoke them
- NDB is a good solution
 - Real HA, real SQL
 - Notable limitations around fields, datatypes
 - Successfully competes with NoSQL systems for most use cases – better in many cases
- NoSQL Systems
 - All have relatively low levels of maturity
 - More suitable for simple key-value models
 - Better for very high volumes

Summing Up on v0.7

- It works!
- Very fast, very reliable
- Very complicated!
- AWS poses challenges that private data centers may not experience
- You can achieve high performance and availability without giving up relational models and read consistency!

Future Directions

- Alternate solution using Pacemaker, Heartbeat
 - From Yves Trudeau @ Percona
 - Uses InnoDB, not NDB
- Implement Memcached plugin
 - To test NoSQL functionality, APIs
- Add simple connection-based persistence to preserve connections during failover
- Better data node distribution
- Better testing & monitoring



"In the long run, we are all ~~dead~~ eventually consistent."

Maynard Keynes on NoSQL Databases

Twitter: @daniel_b_austin
Emai: daaustin@paypal.com

With apologies and thanks to the real DB experts, Andrew Goodman, Yves Trudeau, Clement Frazer, Daniel Abadi, and everyone else!