ORACLE®

MySQL Cluster – What are we working on

Mario Beck – Principal Consultant

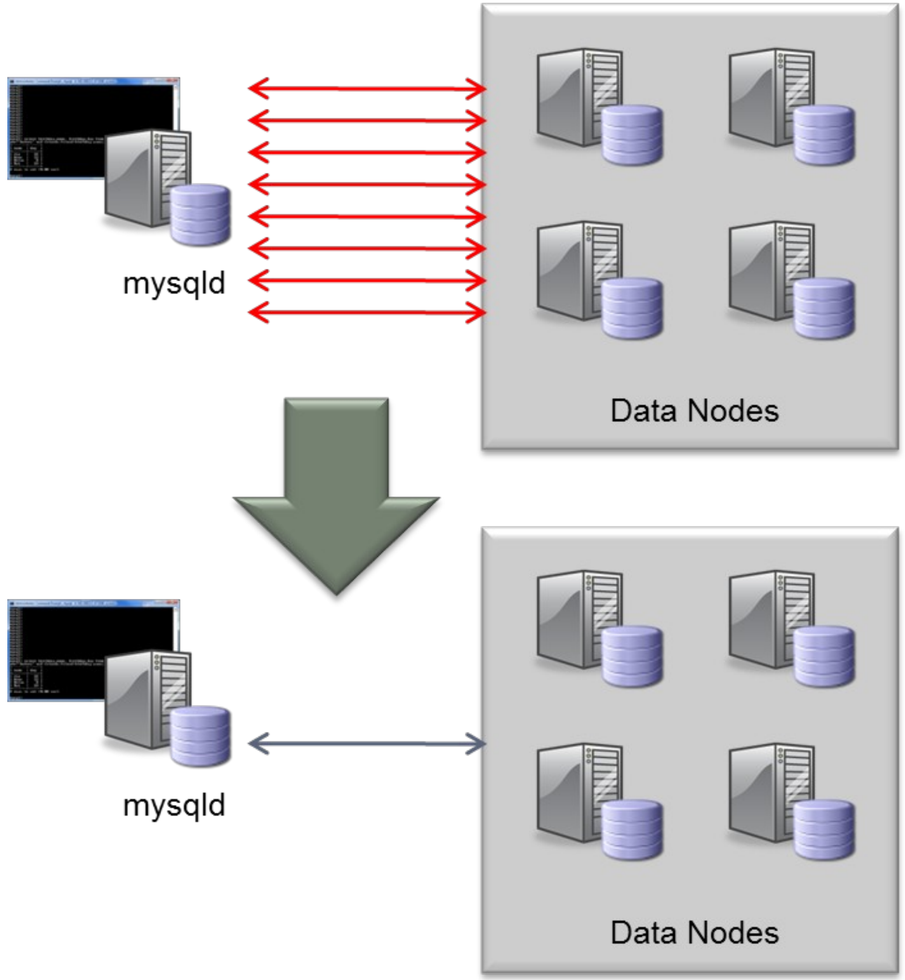
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decision. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

MySQL Cluster 7.2 Development Milestone Release & Beyond

MySQL Cluster 7.2 Development Release

- Adaptive Query Localization:
  - Large, multi-way joins slow on MySQL Cluster
  - Certain types of joins now executed within the data nodes
  - Performance gains:
    - DocQ results: 2.11 sec query -> 0.1 sec
    - 50x improvement for some queries
- User tables stored in Cluster
- Increase from 128 to 512 columns per row
  - Can be added on-line

MySQL Cluster 7.2 Development Release: Adaptive Query Localization



- A linked operation is formed by the MySQL Server from the SQL query and sent to the data nodes
- Linked operations should include only 1 scan together with primary key lookups
- More complex queries could be sent as multiple linked operations
- Does not apply to all queries; application changes still required
  - JOINed columns must have the same data type
  - Queries referencing BLOBs are not supported
  - Explicit locking is not supported
  - Only supports fully or partially qualified primary keys or plain indexes as access method for child tables

MySQL Cluster 7.2 Development Release

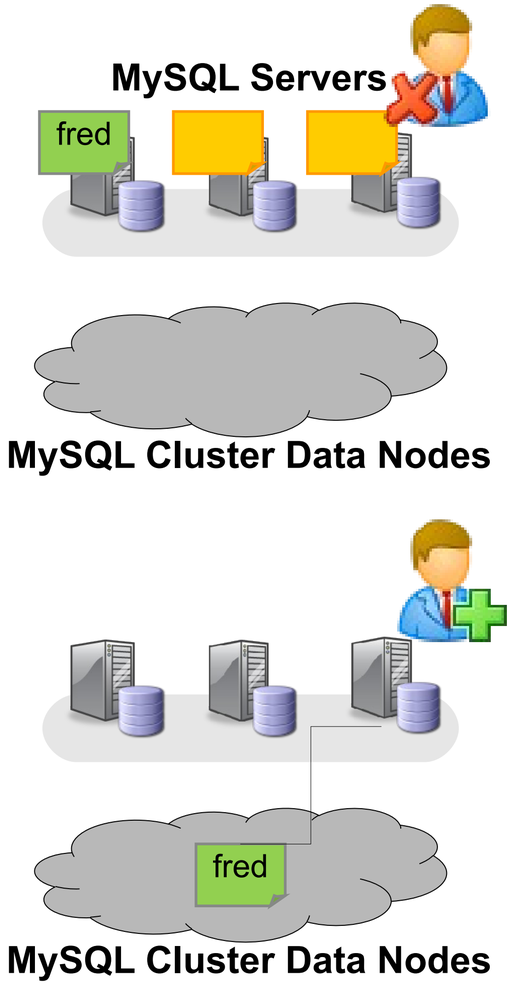
DR

"Docudesk relies on MySQL Cluster to support our DocQ SaaS offering which demands high update rates, low latency and continuous availability from the database. Testing of Adaptive Query Localization has yielded over 20x higher performance on complex queries within our application, enabling Docudesk to expand our use of MySQL Cluster into a broader range of highly dynamic web services."

Casey Brown

Manager, Development & DBA Services, Docudesk

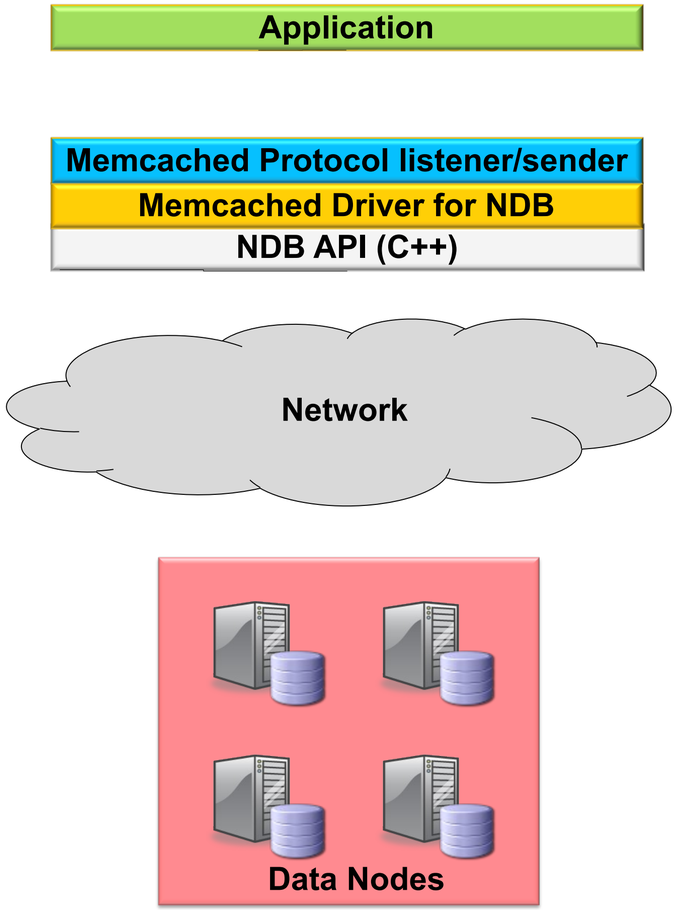
MySQL Cluster 7.2 Development Release: System data stored in Cluster



- System tables (e.g. users) currently use MyISAM storage engine
  - Local to each MySQL Server
  - Information must be managed on each server
  - Wastes time
  - Prone to human error
- Allows user privilege tables to be stored in the MySQL Cluster storage engine
  - Data held in the data nodes
  - Every MySQL server sees and acts on the exact same data
  - Only need to add/change/delete data in one MySQL Server

NoSQL with Memcached

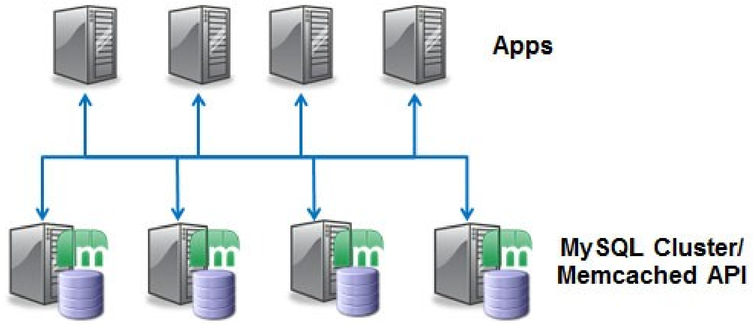
Pre-GA version available from labs.mysql.com

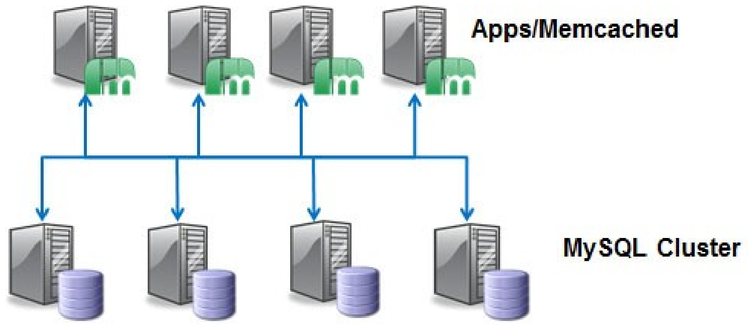


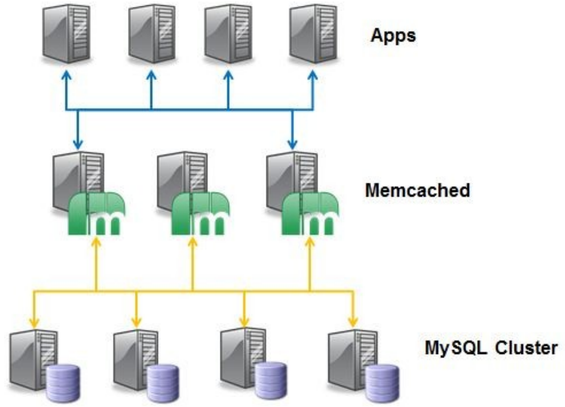
- Memcached is a distributed memory based hash-key/value store with no persistence to disk
- NoSQL, simple API, popular with developers
- MySQL Cluster already provides scalable, in-memory performance with NoSQL (hashed) access as well as persistence
  - Provide the Memcached API but map to NDB API calls
- Writes-in-place, so no need to invalidate cache
- Simplifies architecture as caching & database integrated into 1 tier
- Access data from existing relational tables

NoSQL with Memcached

Pre-GA version available from labs.mysql.com







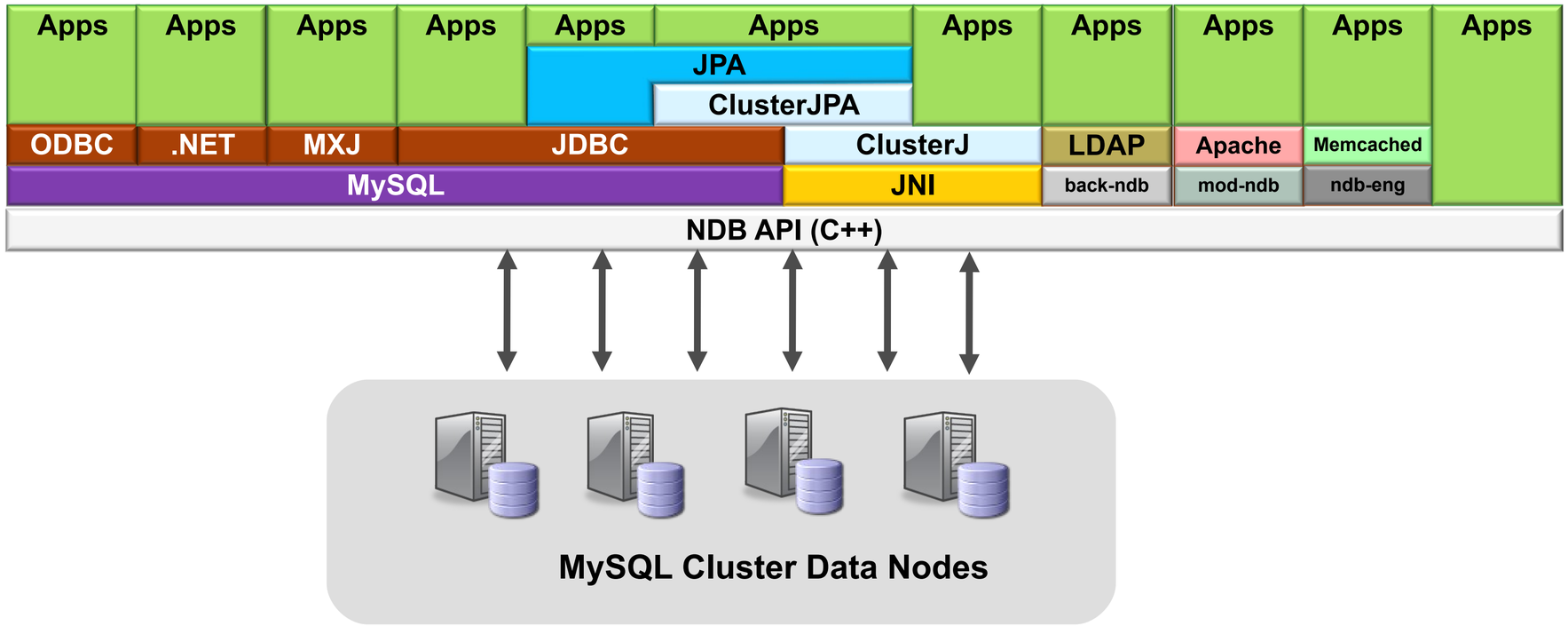
- Flexible:
  - Deployment options
  - Multiple Clusters
  - Simultaneous SQL Access
  - Can still cache in Memcached server
  - Flat key-value store or map to multiple tables/columns

- Simple:

```
set maidenhead 0 0 3
SL6
    STORED

get maidenhead
    VALUE maidenhead 0 3
    SL6
    END
```

NoSQL(s) and MySQL Combined

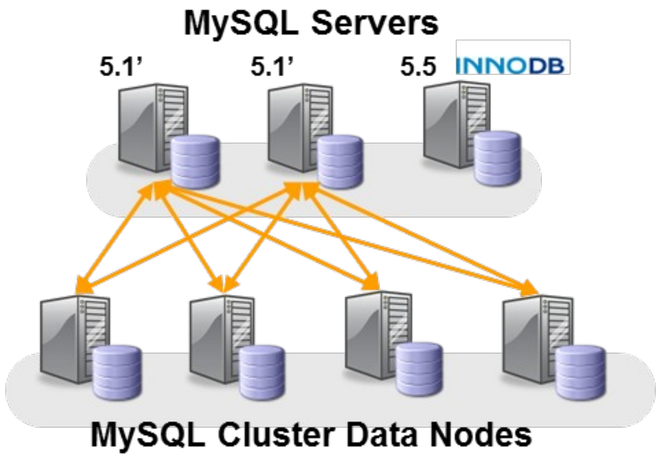


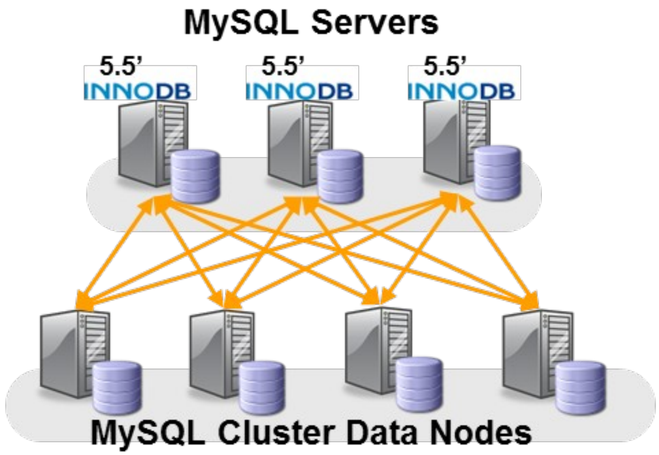
SQL and NoSQL Access Methods

	SQL	NoSQL				
	SQL, JDBC, ODBC…	C++ (NDB API)	Java (ClusterJ/JPA)	Memcached	LDAP	mod-ndb
Description	Provided by MySQL Server, access to all MySQL connectors	Native API directly into the data nodes	Java ORM that accesses the NDB API directly	Presents popular Memcached API with direct access to NBD API	Plugin to allow direct access from LDAP server to NDB API	Apache module that provides a REST web services API for MySQL Cluster
Use cases	Applications wanting the simplicity and richness of SQL.	Where the lowest, most predictable latency & highest throughput is needed.	Java applications wanting > simplicity and > performance than JDBC	Scalable, persistent key-value store with simple API	LDAP based applications looking for high write performance & scalability	Web applications wanting fast, access to Highly Available data using REST API
Support	**GA. Supported by Oracle**	**GA. Supported by Oracle**	**GA. Supported by Oracle**	**Being developed by Oracle**	**Supported by Symas**	**Community**
Schema changes	**Yes (online)**	**Through SQL (online)**	**Yes (online), if using JPA option**	**Key-value store**	**Relational schema built from LDAP schema**	**Use MySQL Server (online)**
Read/Write Performance	**High and scalable**	**Extreme and scalable**	**Very high and scalable using ClusterJ**	**Very high and scalable**	**Very High and scalable**	**Very high and scalable**
Ease of use	**Simple**	**Complex**	**Very simple to Java developers**	**Very simple**	**Very simple to LDAP developers**	**Simple**
Language	**Regular SQL, common to MySQL & other RDMS**	**Unique to Cluster**	**ClusterJ – unique to Cluster JPA – open & common**	**Memcached API is commonly used – esp in web**	**LDAP is a common directory access protocol**	**Very common: HTTP, XML, JSON, HTML**
JOINS	**Yes (getting faster)**	**Programmatically**	**Using JPA**	**No**	**No**	**Programmatically**

Note: Can simultaneously mix & match these methods

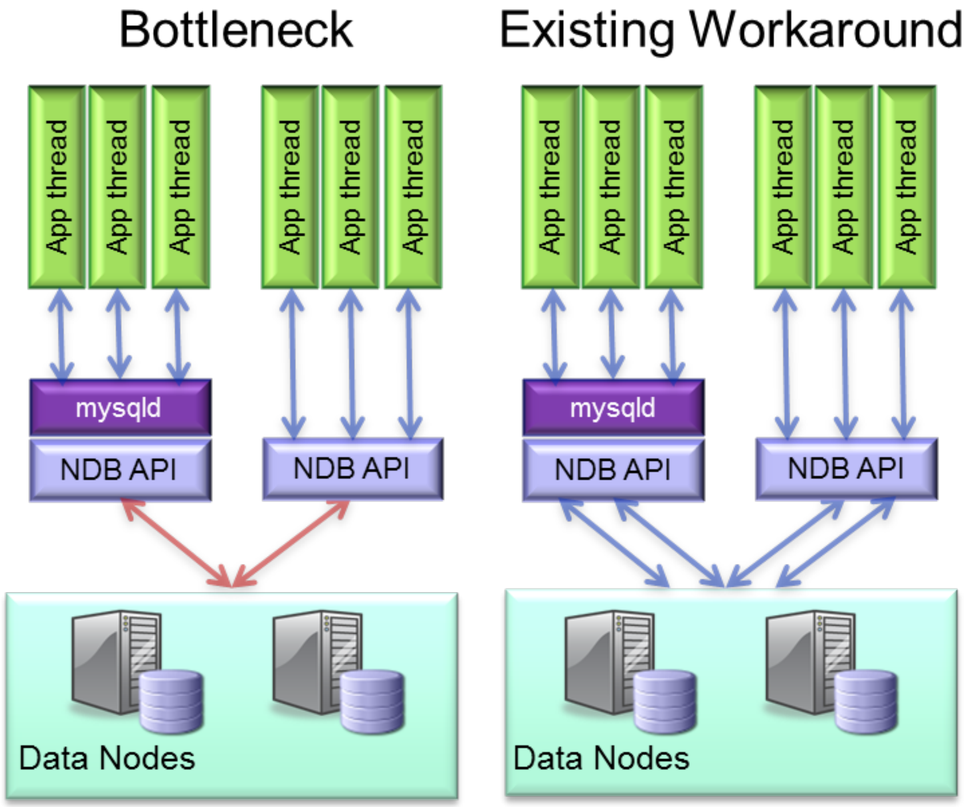
Current Enhancement Plans:
Rebase Server onto MySQL 5.5





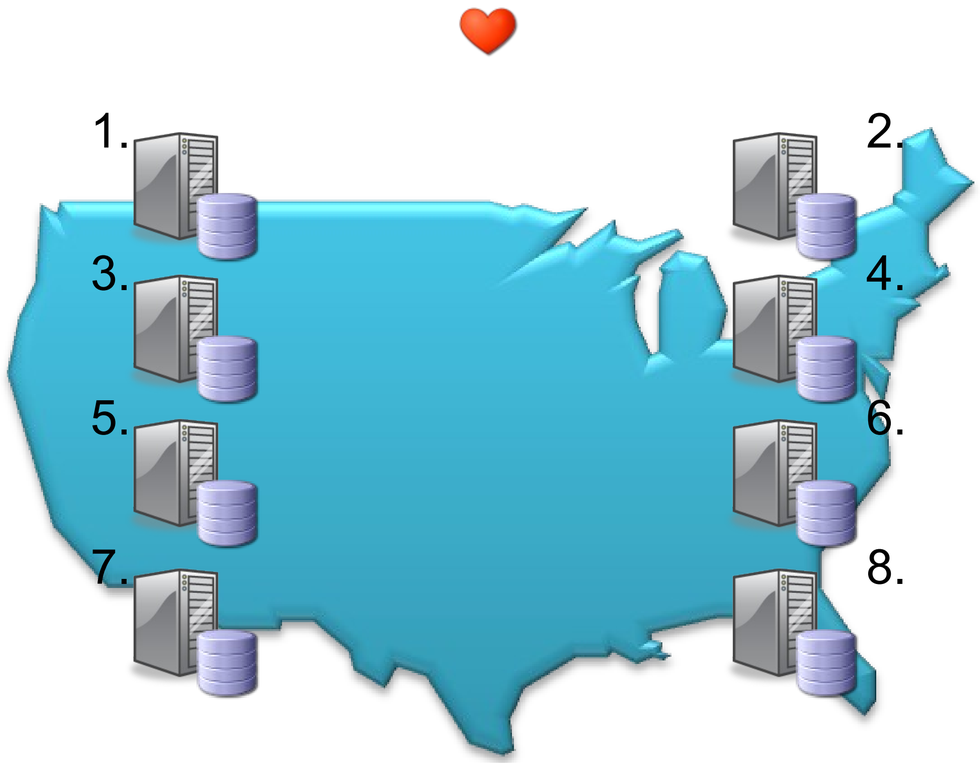
- Use case:
  - Users wanting to use the same MySQL Server for Cluster and InnoDB tables
  - MySQL 5.5 yields many benefits for the InnoDB tables
  - User must currently use different MySQL Server instances to get the best of both worlds

Current Enhancement Plans:
NDB API Thread Contention



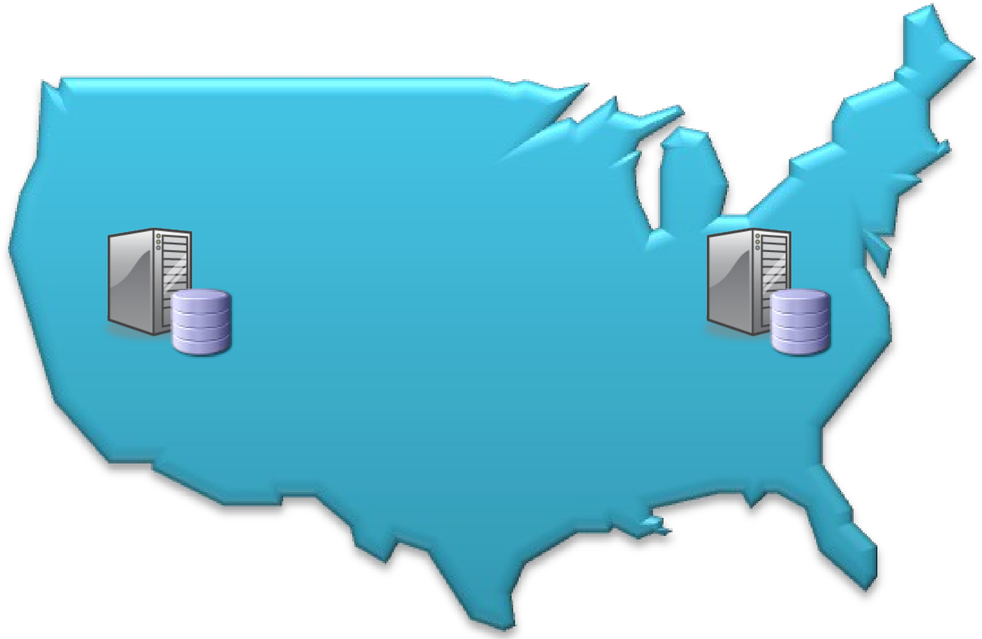
- The problem:
  - When multiple application threads are using a single NDB-API connection (either directly or through MySQL Server), that connection becomes a bottleneck
- Current workaround is to have multiple NDB-API connections for each app/sql process
- Drawbacks of workaround
  - Extra complexity & management
  - Each connection consumes a node-id from the pool of 255 ids
  - Extra memory consumption (send buffers)
- PoC: 1 optimised connection beats 8 existing connections
- Great for dynamic web apps (PHP requires a mysqld connection per session)

Current Enhancement Plans:
Heartbeat enhancements for multi-site Clusters



- Currently do not support splitting data nodes over a WAN
- Heartbeat enhancements designed to more reliably enforce network partitioning in event of WAN latency spikes
- Users would be able to split data nodes if application can tolerate WAN latency

Current Enhancement Plans:
Enhanced Active-Active Geographic Replication



- Enhancements to conflict detection/resolution:
  - Remove need for schema & application changes
  - Better handling of deletes
  - Transaction-level roll-back

Resources to Get Started

- Download the software:
  - 7.1 GA: www.mysql.com/downloads/cluster/
  - 7.2 DMR: dev.mysql.com/downloads/cluster/
  - Memcached: labs.mysql.com
- Blogs on latest developments: clusterdb.com
- MySQL Cluster Quick Start Guides
  - www.mysql.com/products/database/cluster/get-started.html#quickstart
- MySQL Cluster 7.1, Architecture and New Features
  - www.mysql.com/why-mysql/white-papers/mysql_wp_cluster7_architecture.php
- MySQL Cluster on the Web
  - www.mysql.com/products/database/cluster