



**ENGINEERED
FOR INNOVATION**

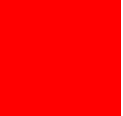
**ORACLE
OPEN
WORLD**

ORACLE®

NoSQL Access to MySQL: The Best of Both Worlds

Andrew Morgan

MySQL Product Management



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 decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Session Agenda



- NoSQL – What are people looking for?
- RDBMS – What advantages do they still have?
- How MySQL Delivers the Best of Both Worlds
 - MySQL Cluster
 - NoSQL attributes: Scale-out, performance, ease-of-use, schema flexibility, on-line operations
 - NoSQL APIs
 - Key-Value store access to InnoDB (Memcached)

NoSQL – Why is it needed?



- Web applications demanding:
 - Development velocity: Simplicity & flexibility of data model & APIs
 - Scalability & performance: high write throughput and Key/Value access
 - Support for “Big Data”

What NoSQL must deliver

- Massive scalability
 - No application-level sharding
- Performance
- High Availability/Fault Tolerance
- Ease of use
 - Simple operations/administration
 - Simple APIs
 - Quickly evolve application & schema

Scalability	
Performance	
HA	
Ease of use	

Types of NoSQL stores

Key-Value

- Cassandra
- Memcached
- BigTable
- Hadoop
- Voldermort

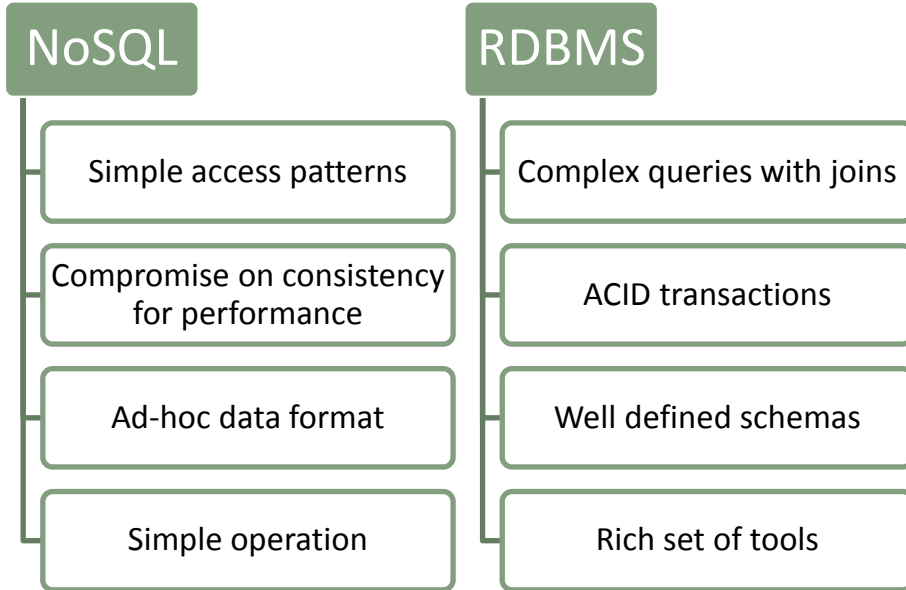
Document

- MongoDB
- CouchDB

Graph

- Neo4J
- FlockDB

Still a role for the RDBMS?



- No best single solution fits all
- Mix and match

Scalability	
Performance	
HA	
Ease of use	
SQL/Joins	
ACID Transactions	

MySQL Cluster introduction

Carrier Grade Database

- Shared-nothing in-memory parallel database server
- ACID compliant relational database

Highly Available

- Five nines (99.999%) availability
- Self-healing, sub-second failover

Real-time Performance

- High load, real-time performance
- Predictable low latency, bounded access times

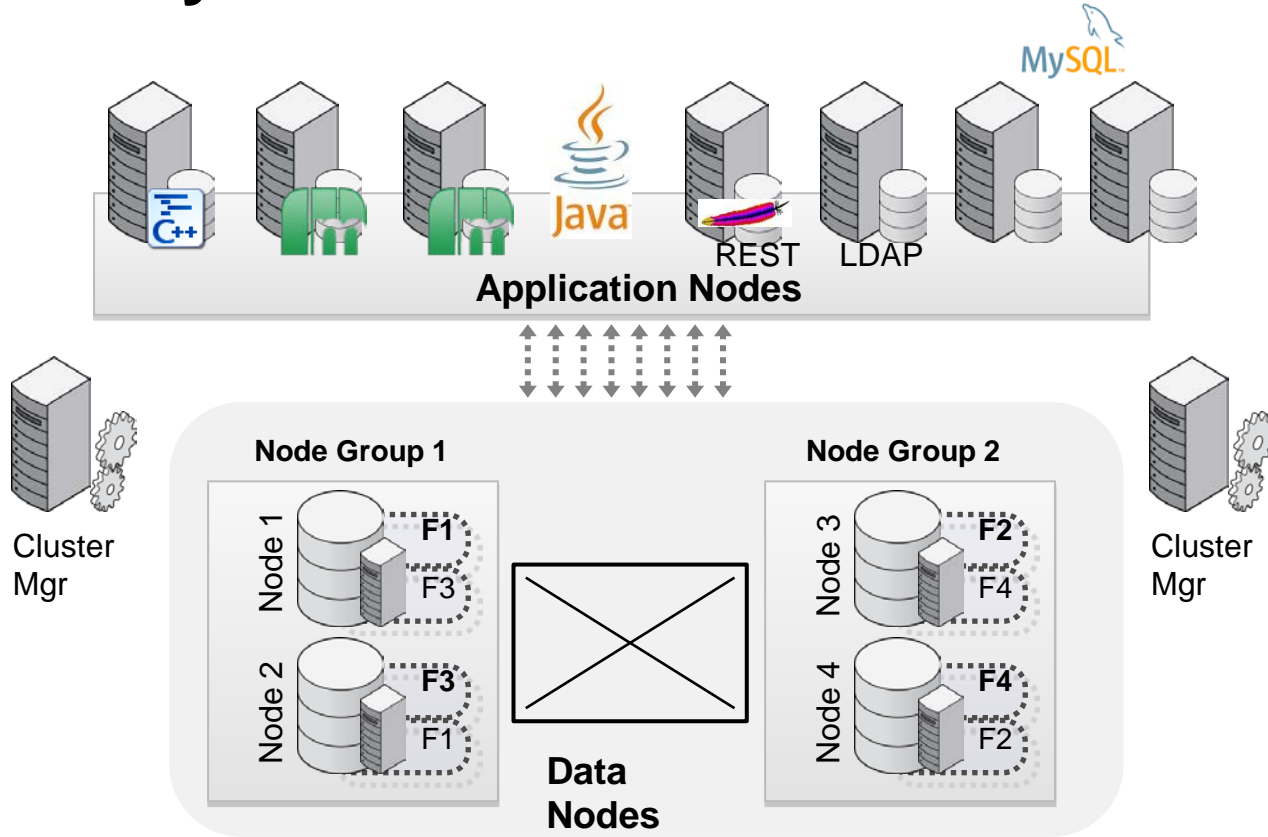
Linearly Scalable

- Incrementally grow out with application demands
- Linearly scale with distribution awareness

Open Development

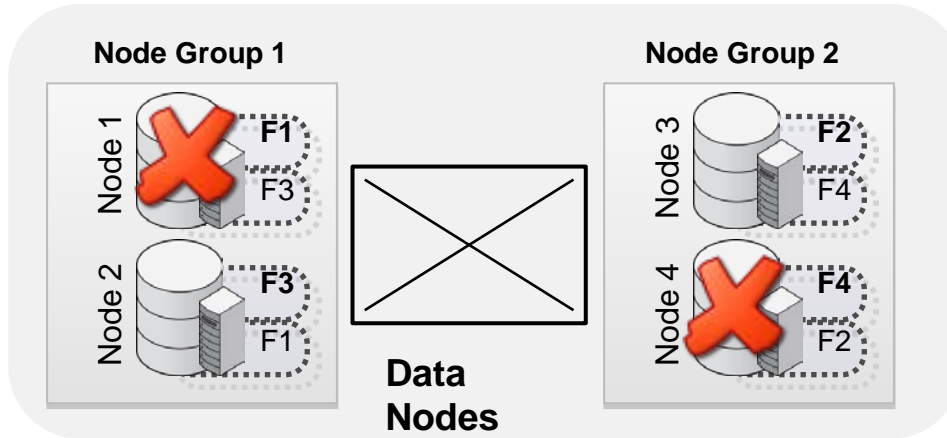
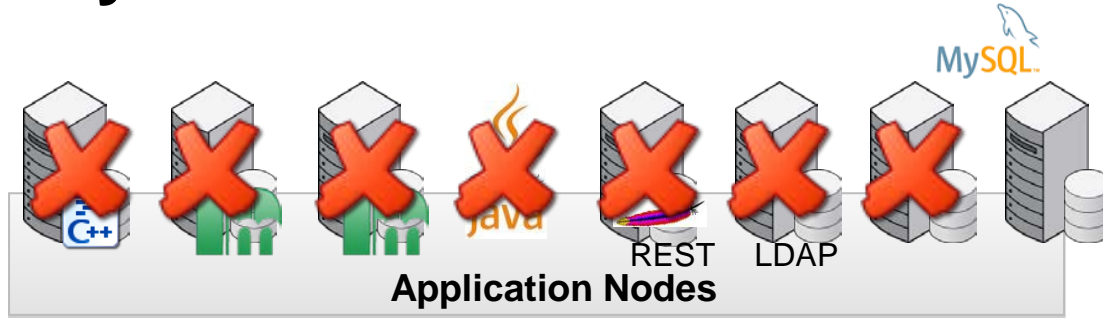
- Open Source, multiple data access
- High performance APIs (C++, Java, SQL, LDAP)

MySQL Cluster Architecture



Scalability	
Performance	
HA	
Ease of use	
SQL/Joins	✓
ACID Transactions	✓

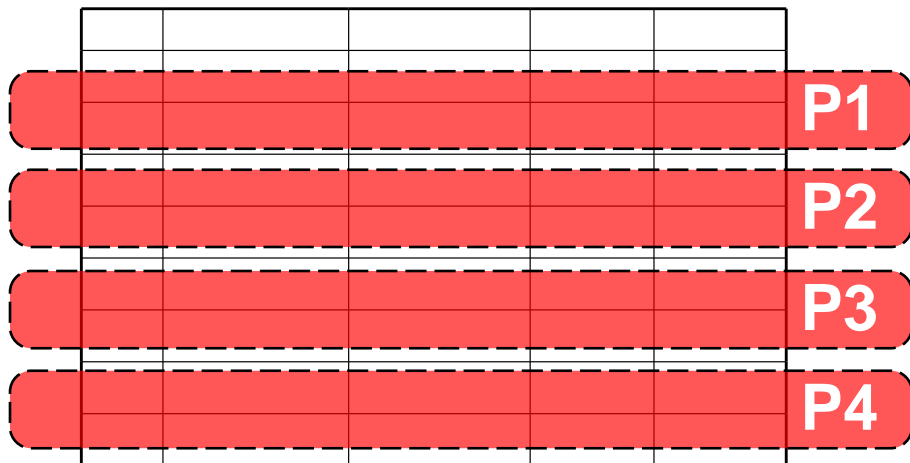
MySQL Cluster Architecture



Scalability	
Performance	
HA	✓
Ease of use	
SQL/Joins	✓
ACID Transactions	✓

Scale-Out: Auto-Partitioning

Table T1



Data Node 1



Data Node 2



Data Node 3

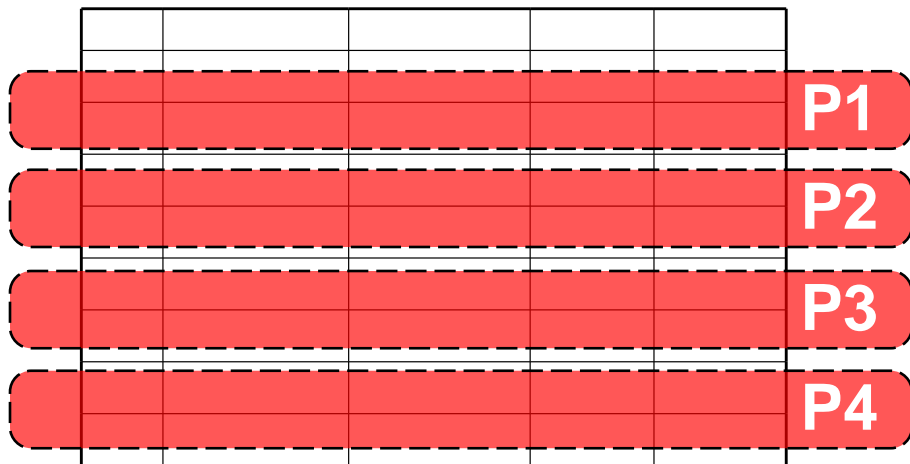


Data Node 4



Scale-Out: Auto-Partitioning

Table T1



Data Node 1



Data Node 2



Data Node 3

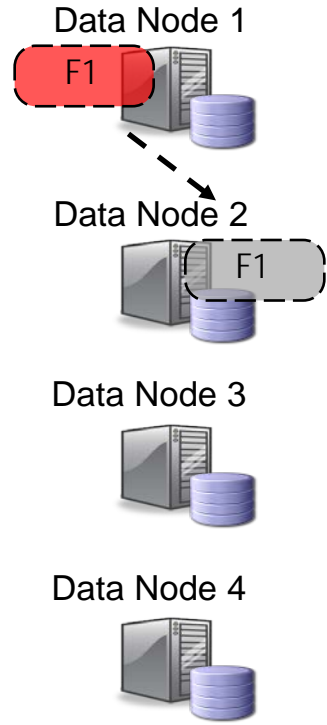
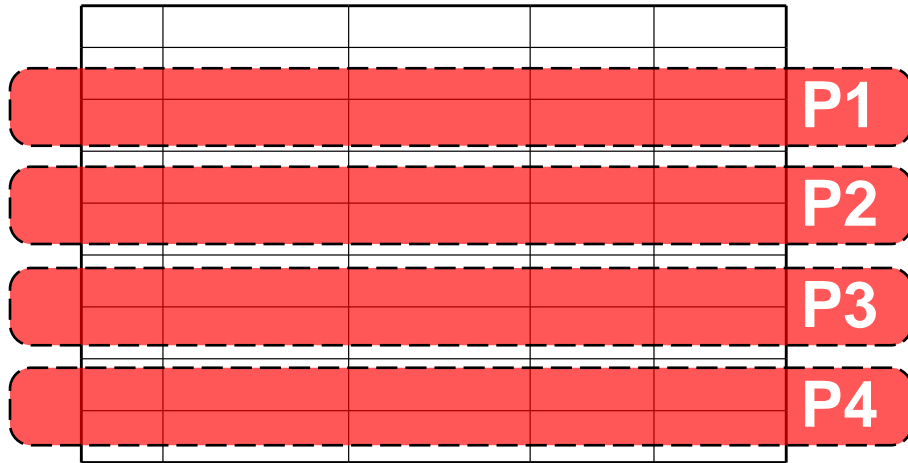


Data Node 4



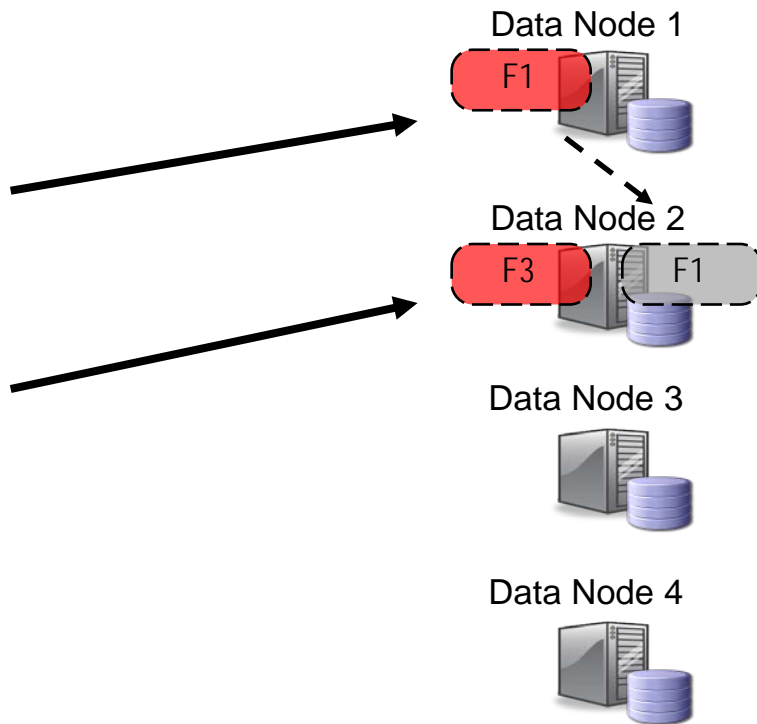
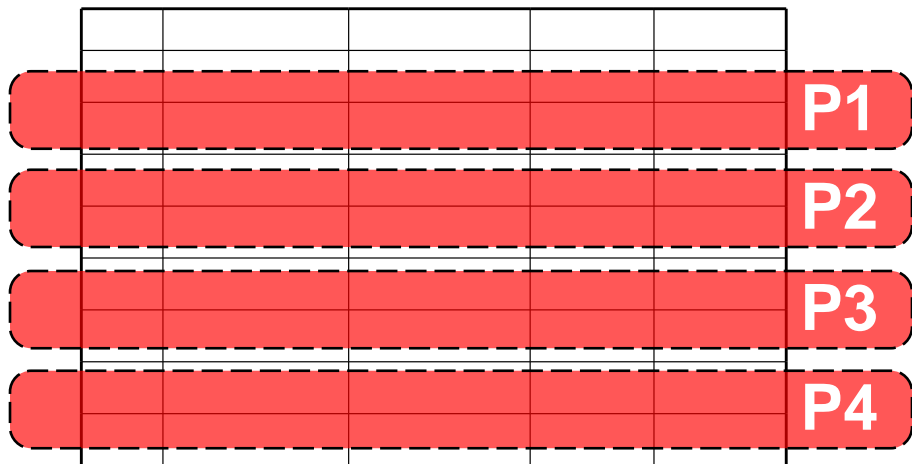
Scale-Out: Auto-Partitioning

Table T1



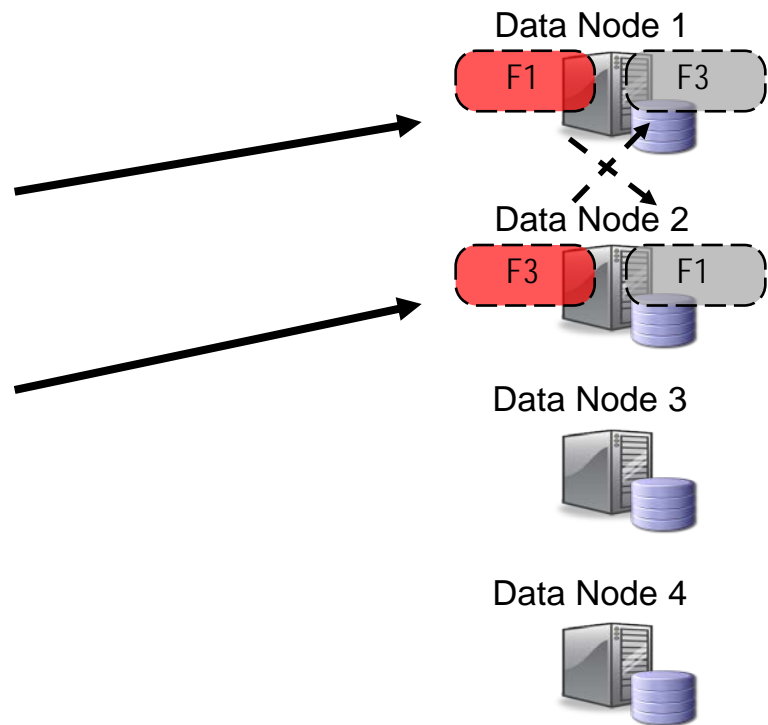
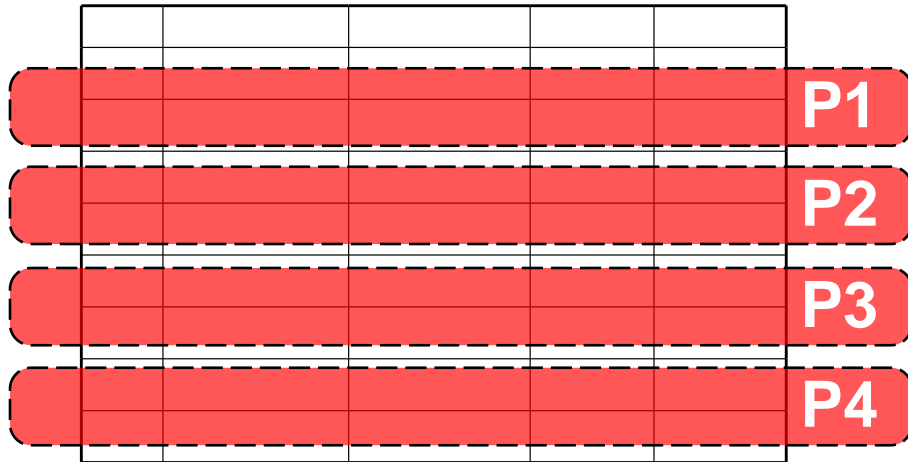
Scale-Out: Auto-Partitioning

Table T1



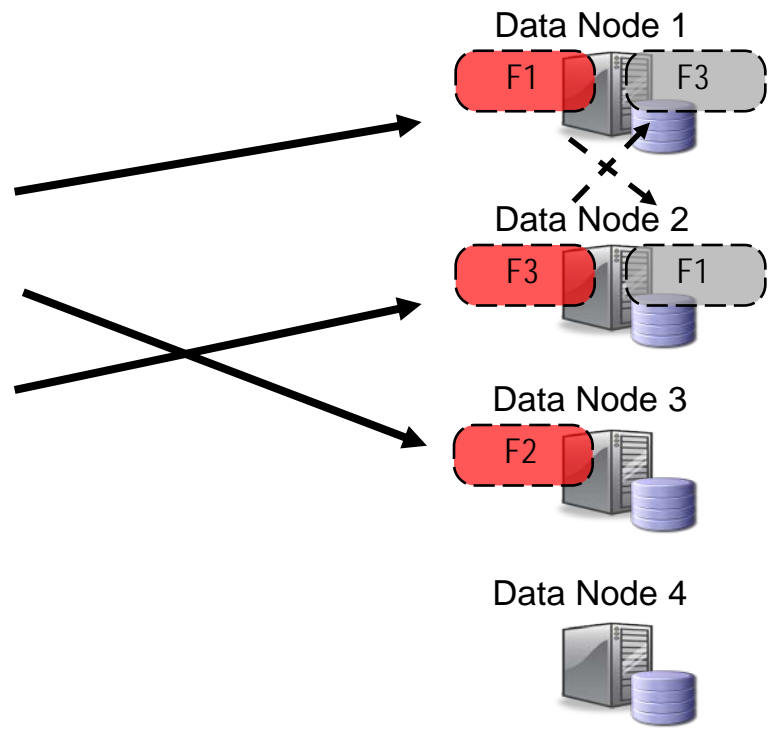
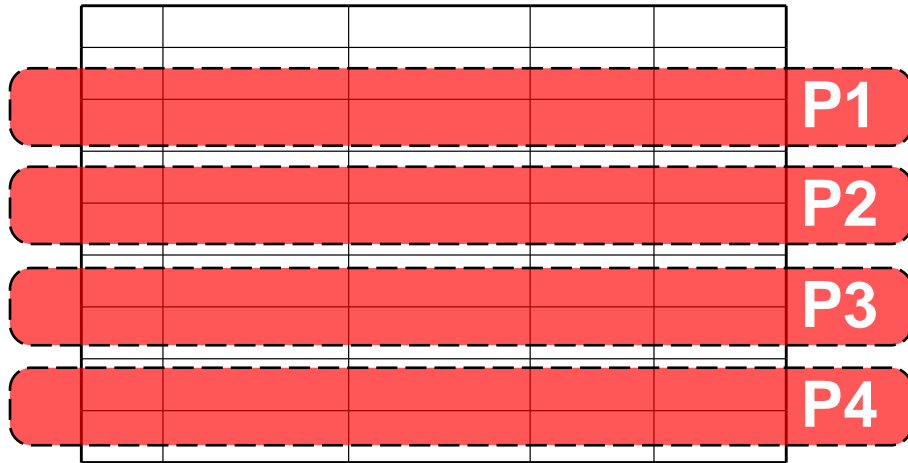
Scale-Out: Auto-Partitioning

Table T1



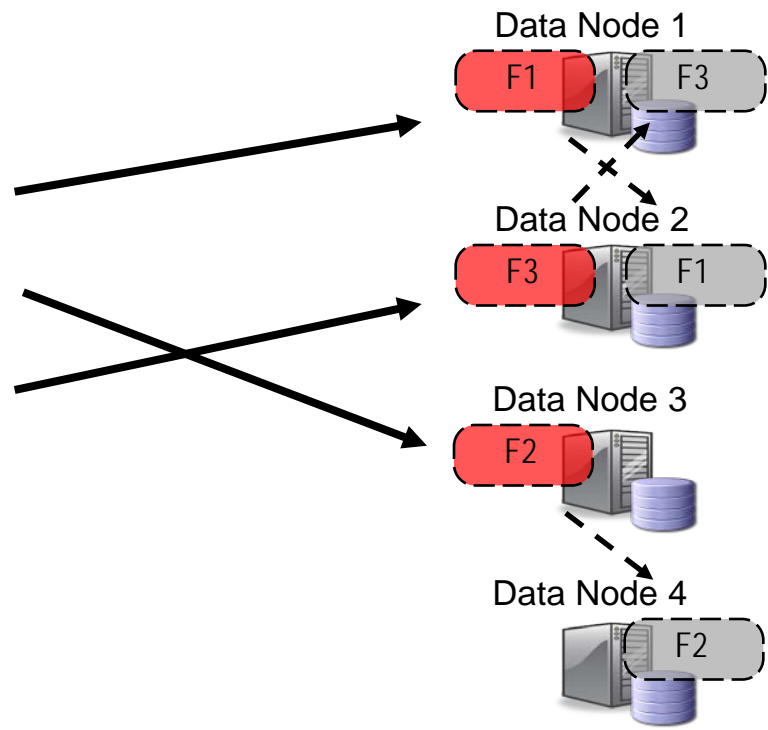
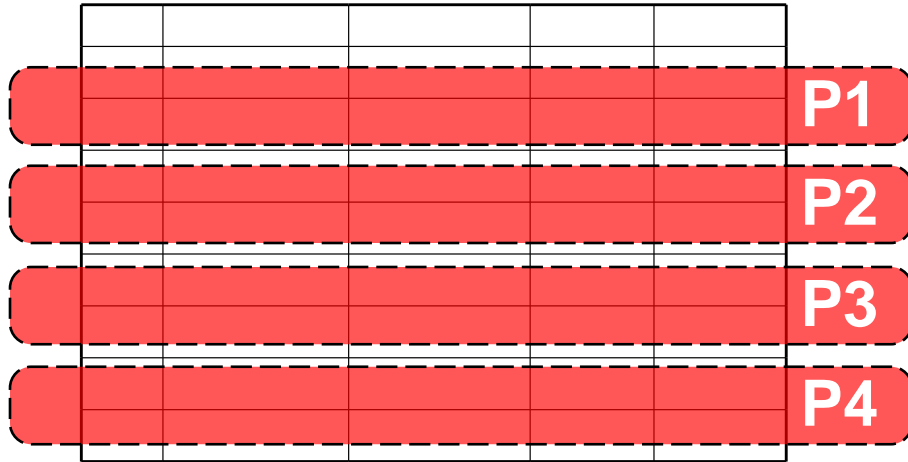
Scale-Out: Auto-Partitioning

Table T1



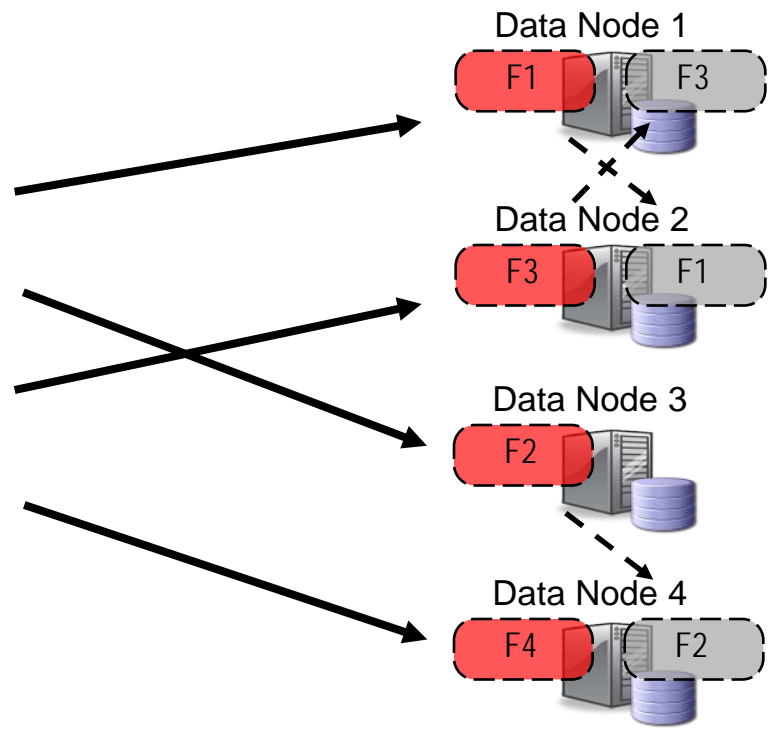
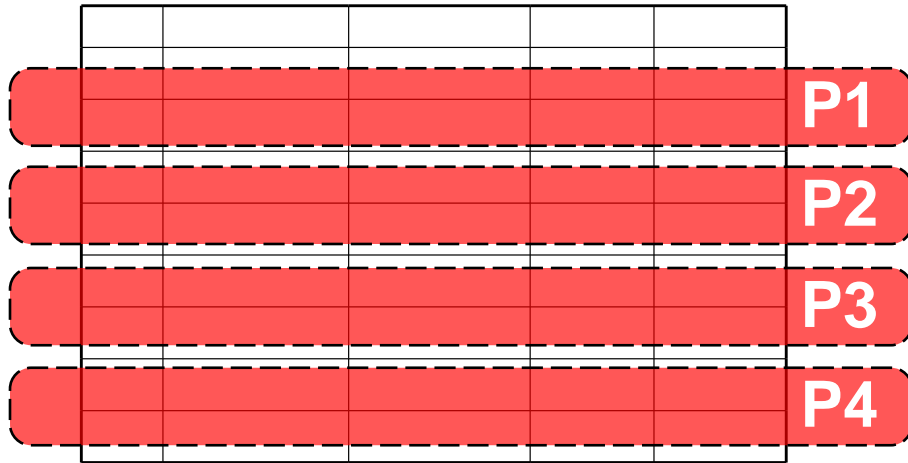
Scale-Out: Auto-Partitioning

Table T1



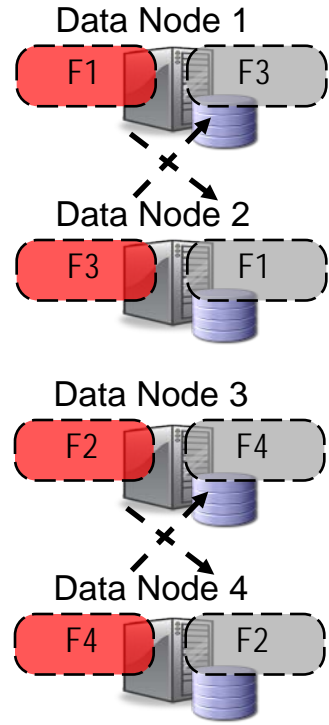
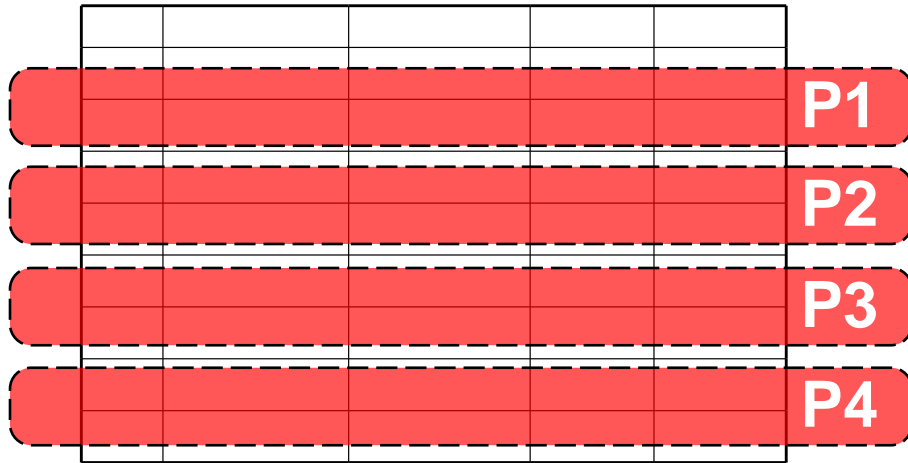
Scale-Out: Auto-Partitioning

Table T1



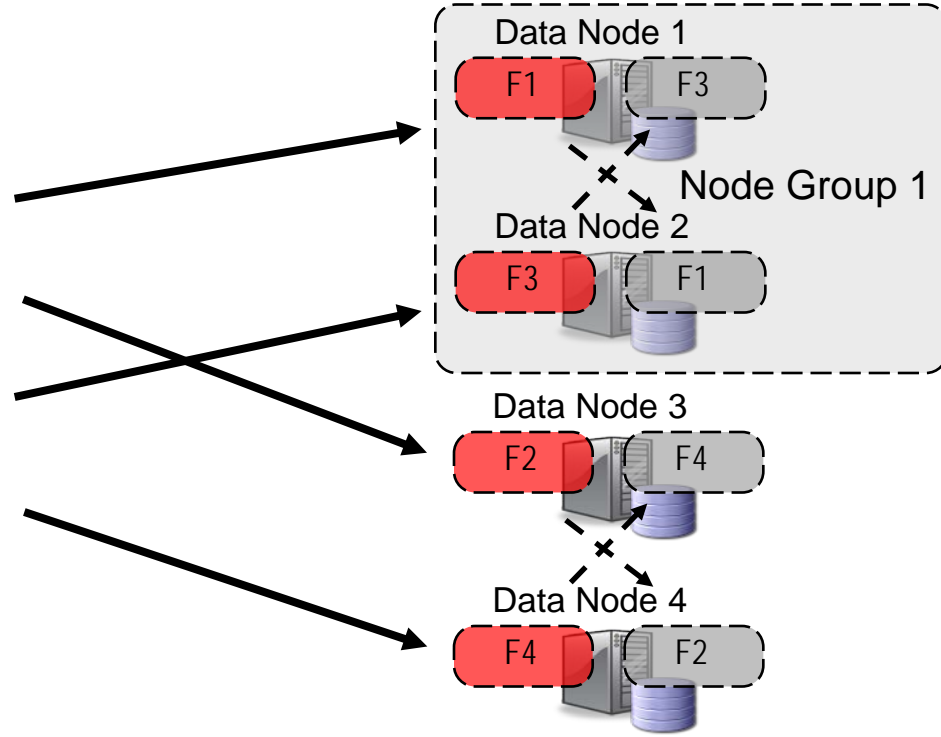
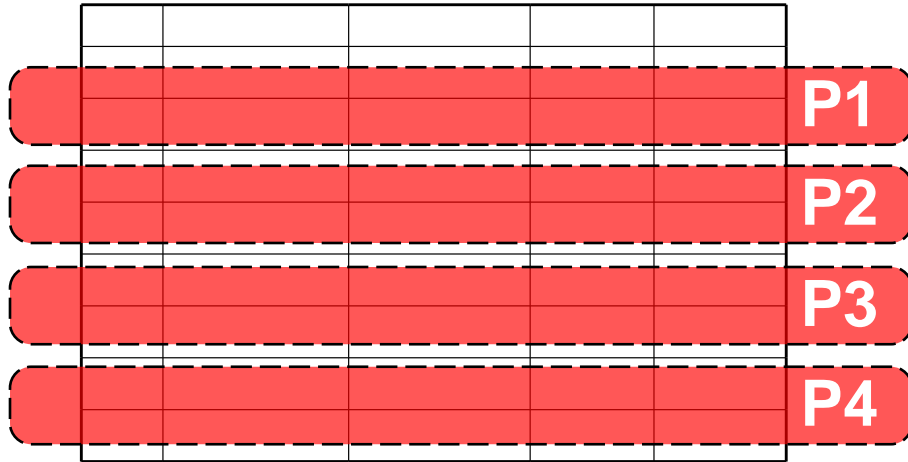
Scale-Out: Auto-Partitioning

Table T1



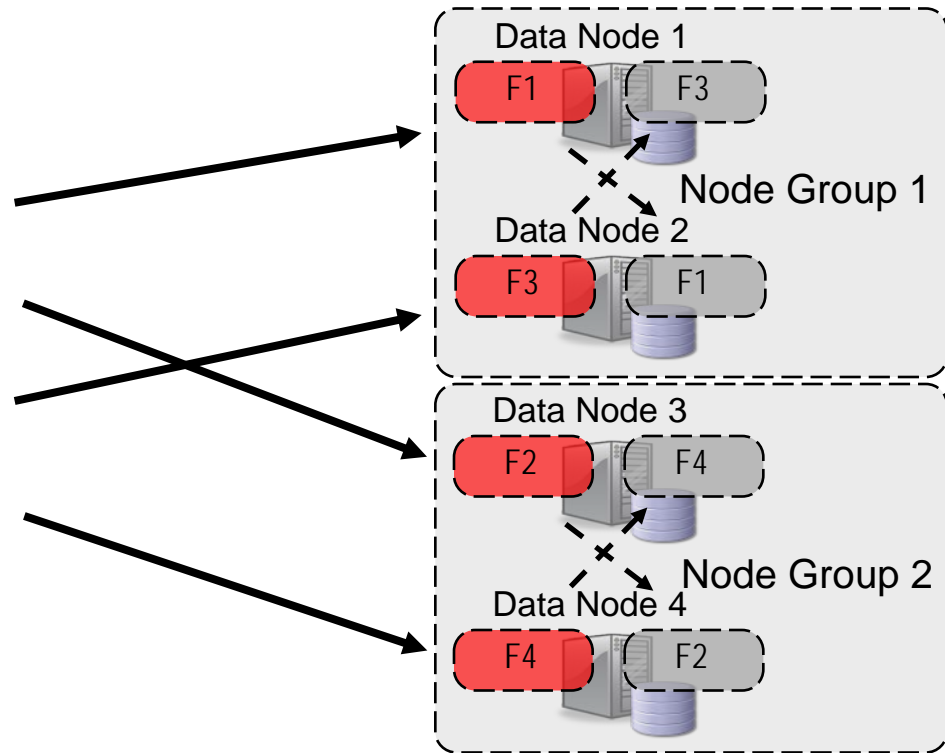
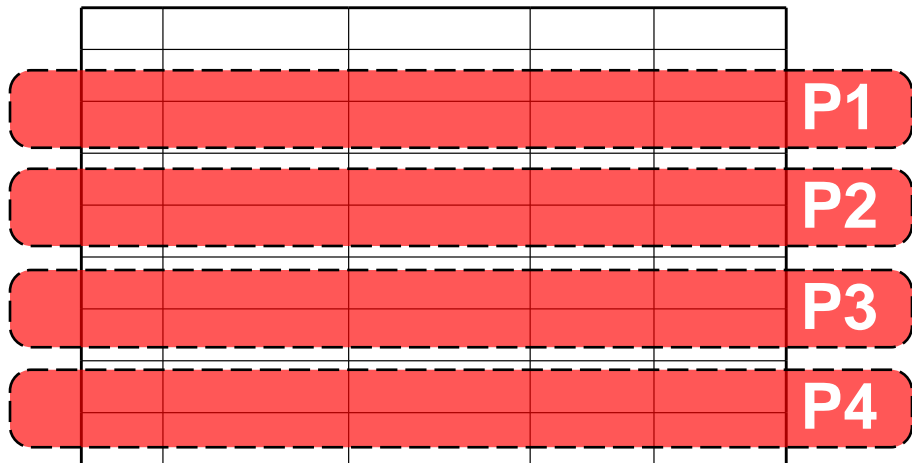
Scale-Out: Auto-Partitioning

Table T1



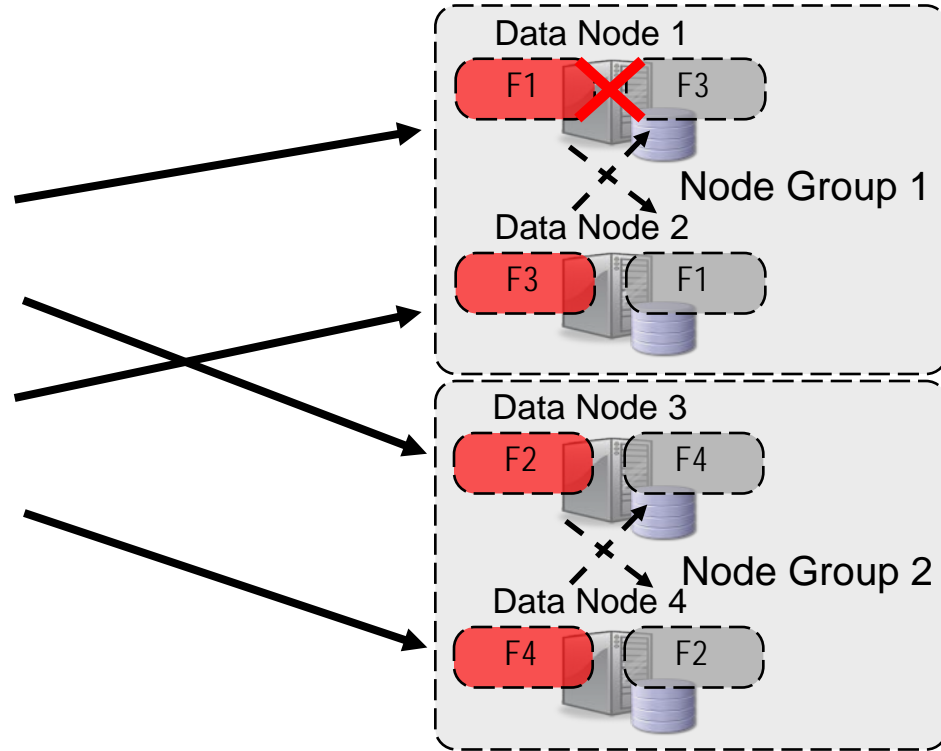
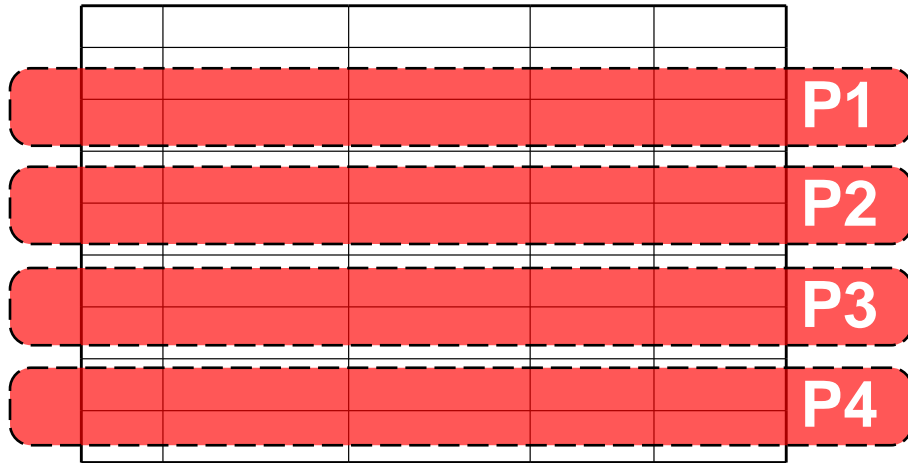
Scale-Out: Auto-Partitioning

Table T1



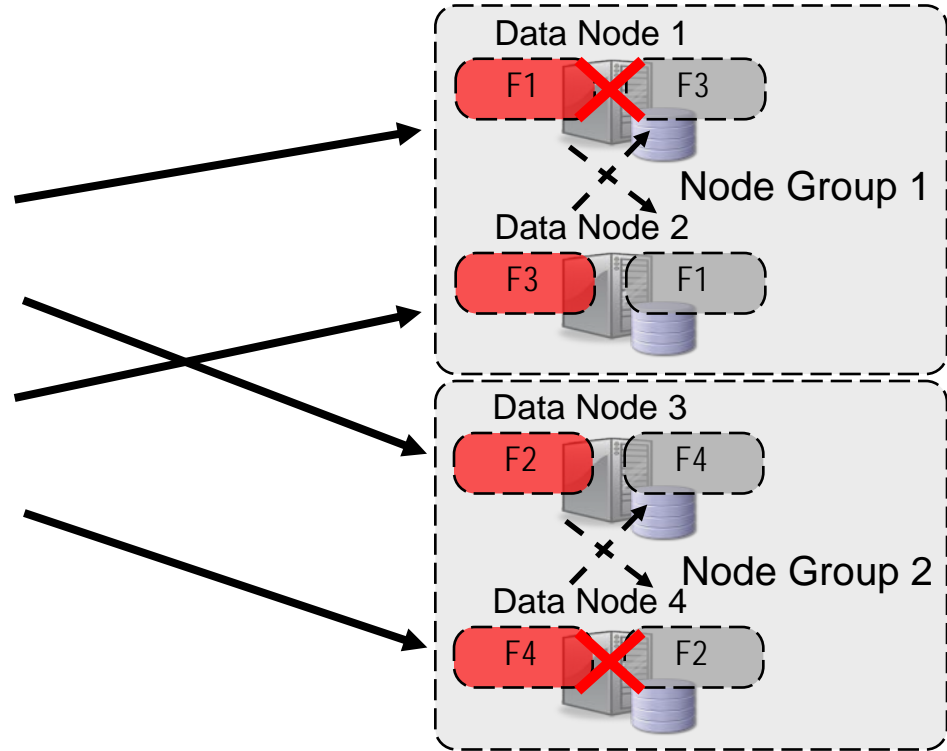
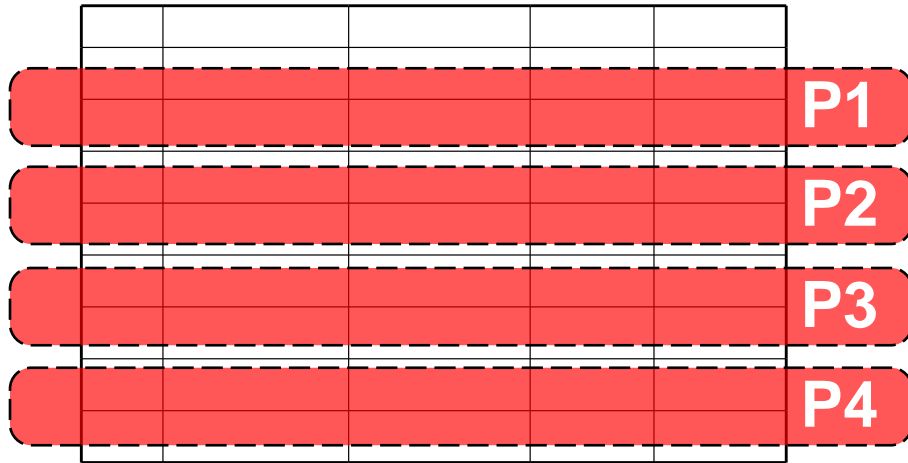
Scale-Out: Auto-Partitioning

Table T1



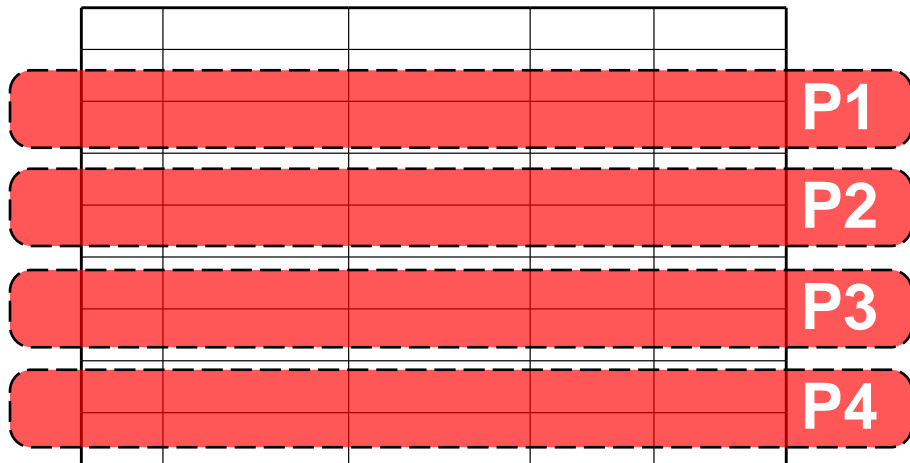
Scale-Out: Auto-Partitioning

Table T1



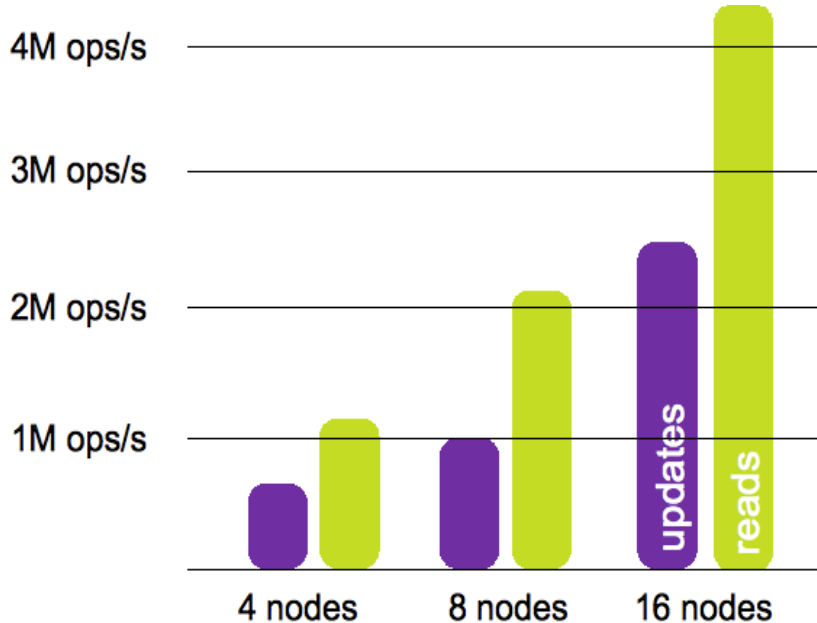
Scale-Out: Auto-Partitioning

Table T1



Scalability	✓
Performance	
HA	✓
Ease of use	
SQL/Joins	✓
ACID Transactions	✓

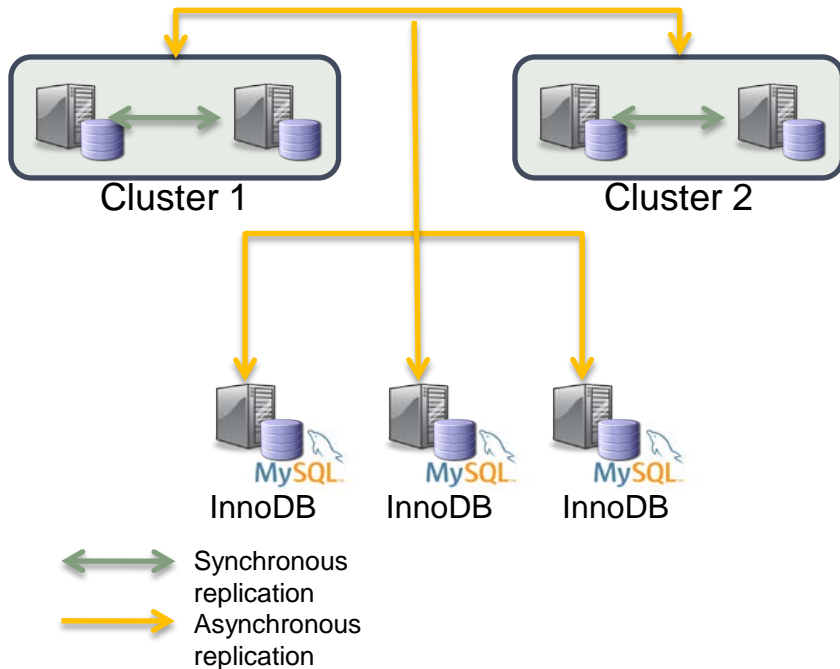
Scale-Out Reads & Writes on Commodity Hardware



- NDB API Performance 4.33 M Queries per second!
- 8 Intel servers, dual-6-core CPUs @2.93 GHz, 24GB RAM
- 2 Data Nodes per server
- flexAsync benchmark
 - 16 parallel threads, each issuing 256 simultaneous transactions
 - Read / Write 100 byte attribute

Scalability	✓
Performance	✓
HA	✓
Ease of use	
SQL/Joins	✓
ACID Transactions	✓

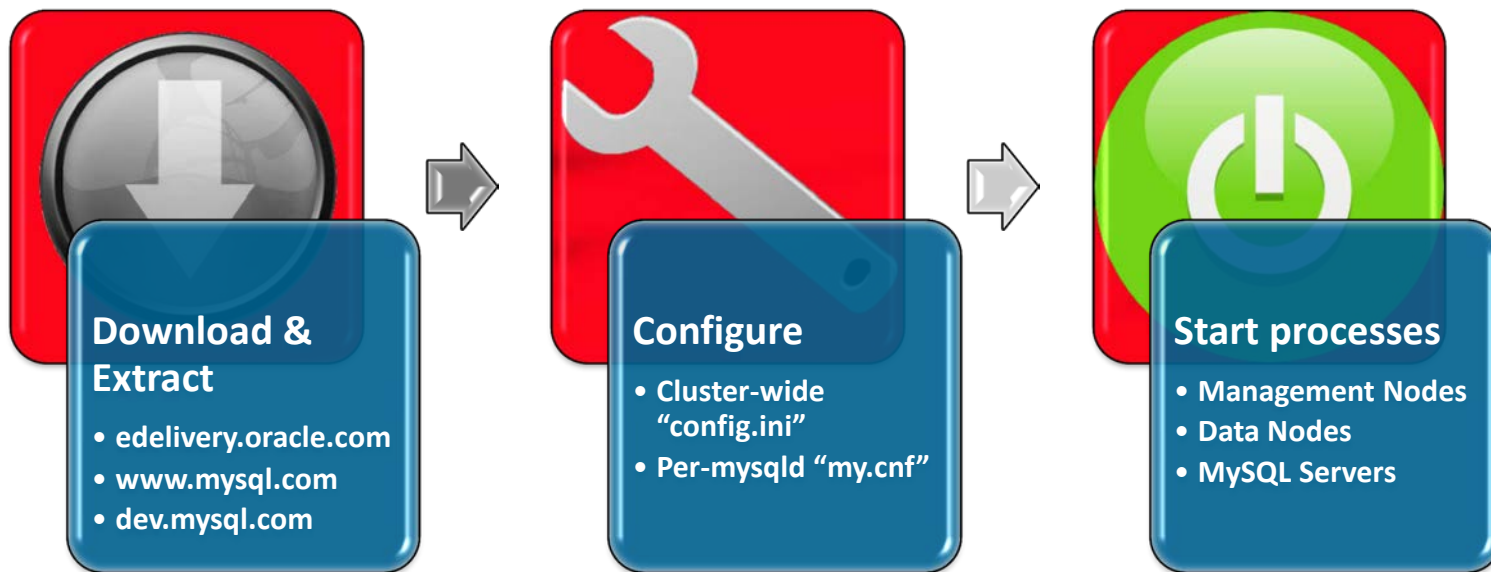
Replication Flexibility



- Synchronous replication within a Cluster node group for HA
- Bi-Direction asynchronous replication to remote Cluster for geographic redundancy
- Asynchronous replication to non-Cluster databases for specialised activities such as report generation
- Mix and match replication types

Creating & running your first Cluster

The traditional way (pre-MCM) – **Up and running in 15 mins**



- Up & running in 10-15 minutes using Quick Start guides from <http://dev.mysql.com/downloads/cluster/>
 - Versions for Linux, Windows & Solaris

Scalability	✓
Performance	✓
HA	✓
Ease of use	✓
SQL/Joins	✓
ACID Transactions	✓

MySQL Cluster Manager 1.1.2

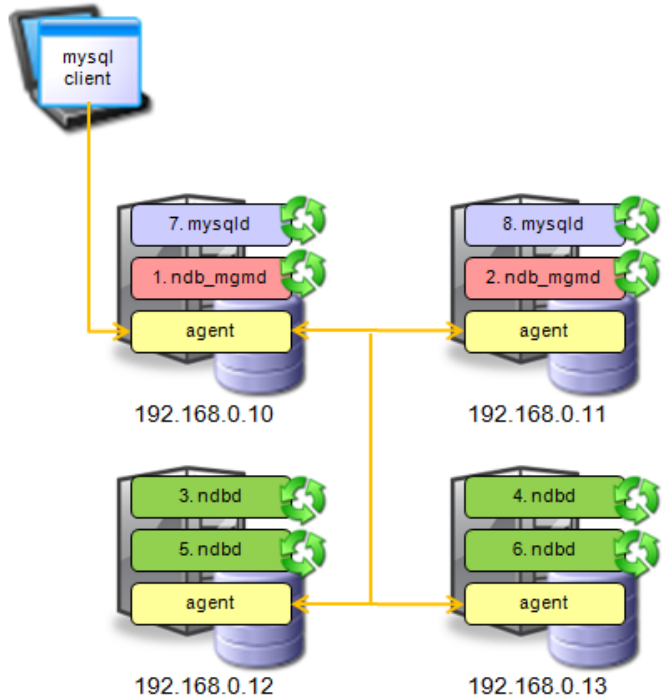
Bootstrap single host Cluster

1. Download MCM/Cluster package from edelivery.oracle.com:
2. Unzip
3. Run agent, define, create & start Cluster!

```
$> bin\mcmd -bootstrap
```

```
MySQL Cluster Manager 1.1.2 started
Connect to MySQL Cluster Manager by running "D:\Andrew\Documents\MySQL\mcm\bin\mcm" -a NOVA:1862
Configuring default cluster 'mycluster'...
Starting default cluster 'mycluster'...
Cluster 'mycluster' started successfully
  ndb_mgmd NOVA:1186
  ndbd NOVA
  ndbd NOVA
  mysqld NOVA:3306
  mysqld NOVA:3307
  ndbapi *
Connect to the database by running "D:\Andrew\Documents\MySQL\mcm\cluster\bin\mysql" -h NOVA -P 3306 -u root
```

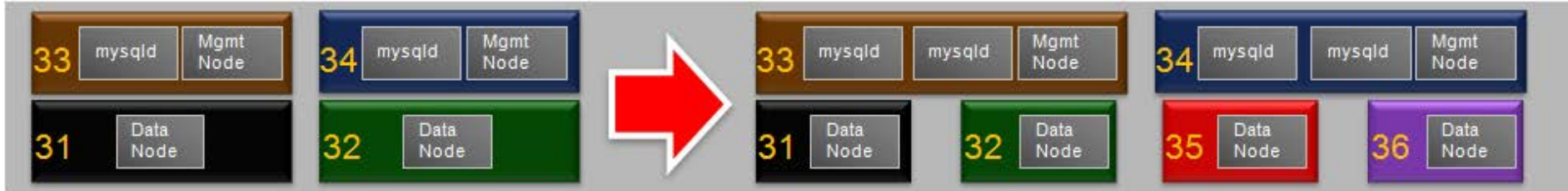
MCM: Upgrade Cluster



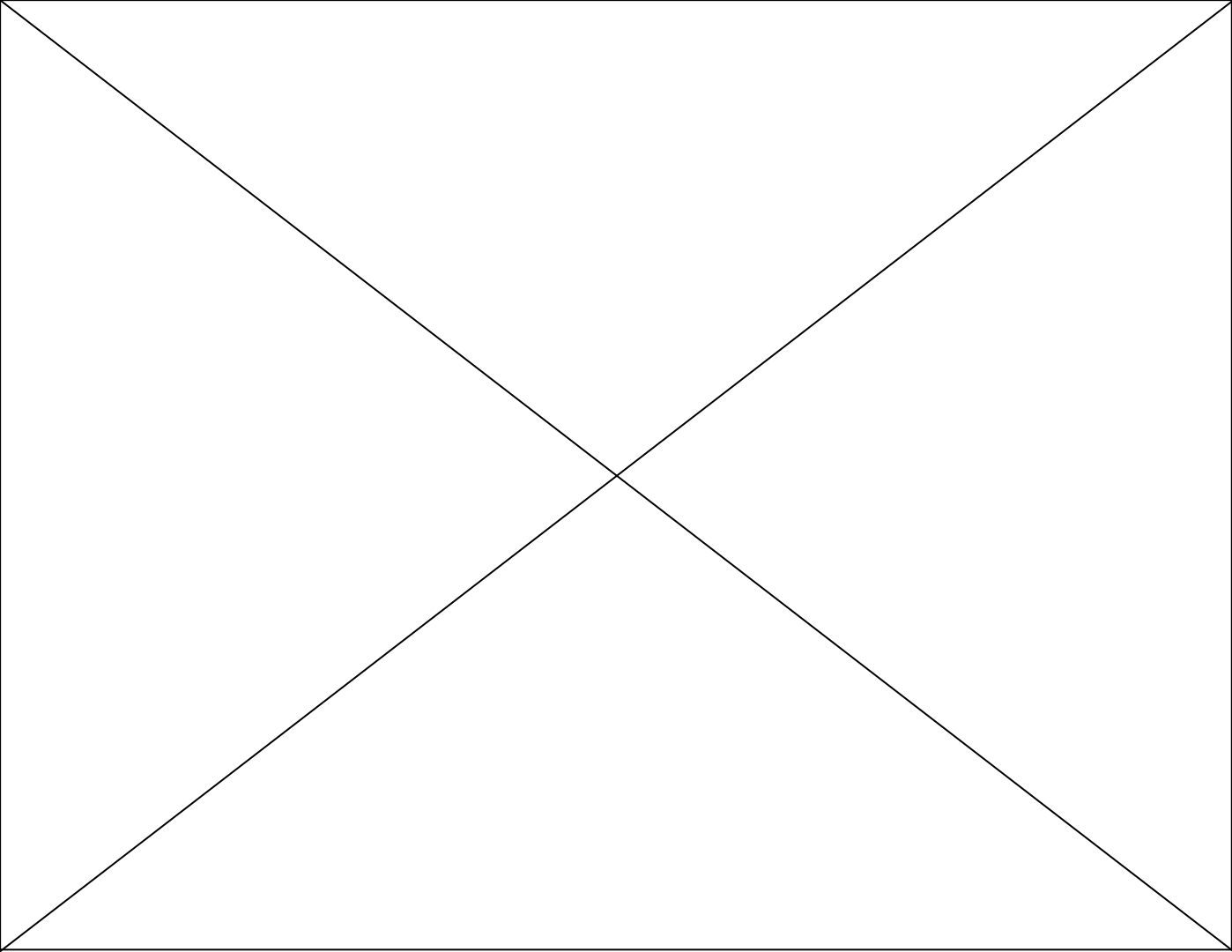
```
mysql> upgrade cluster  
--package=7.1 mycluster;
```

MCM: On-Line Add-Node

Scale out with no loss of service



```
mysql> add hosts --hosts=192.168.0.35,192.168.0.36 mysite;  
mysql> add package --basedir=/usr/local/mysql_7_1_9a -  
hosts=192.168.0.35,192.168.0.36 7.1;  
mysql> add process --  
processhosts=mysqlid@192.168.0.33,mysqlid@192.168.0.34,ndbd@192.  
168.0.35,ndbd@192.168.0.36 mycluster;  
mysql> start process --added mycluster;
```

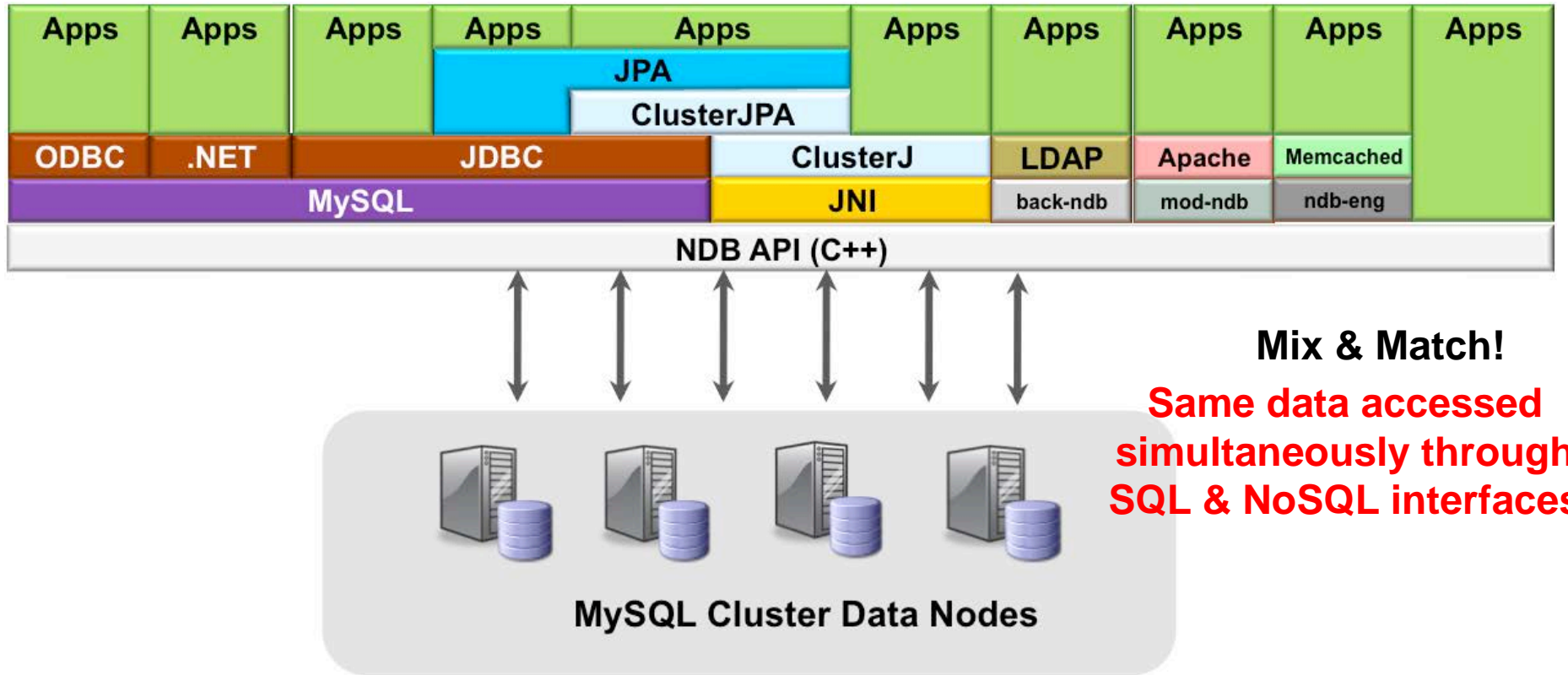


Make schema
changes without
interrupting read
or write traffic

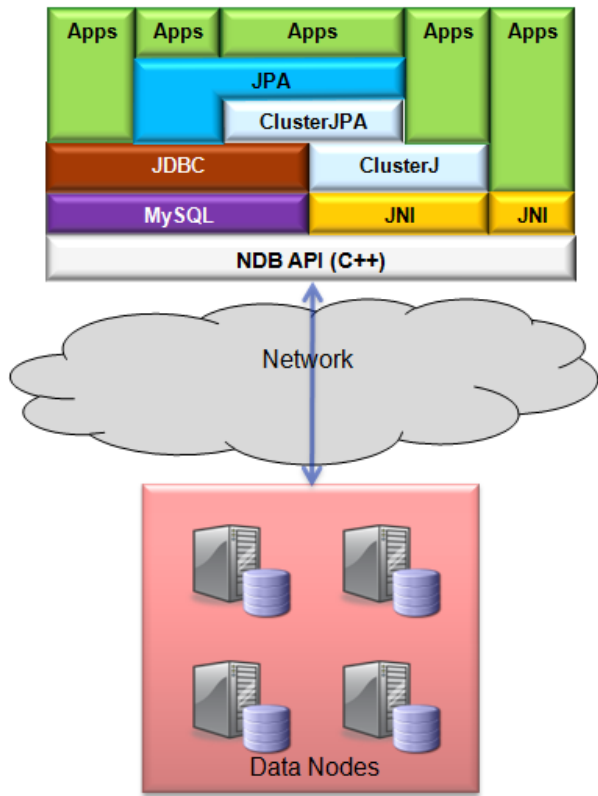
Add columns &
indexes

Scalability	✓
Performance	✓
HA	✓
Ease of use	✓
SQL/Joins	✓
ACID Transactions	✓

NoSQL Access to MySQL Cluster data



MySQL Cluster 7.1: ClusterJ/JPA

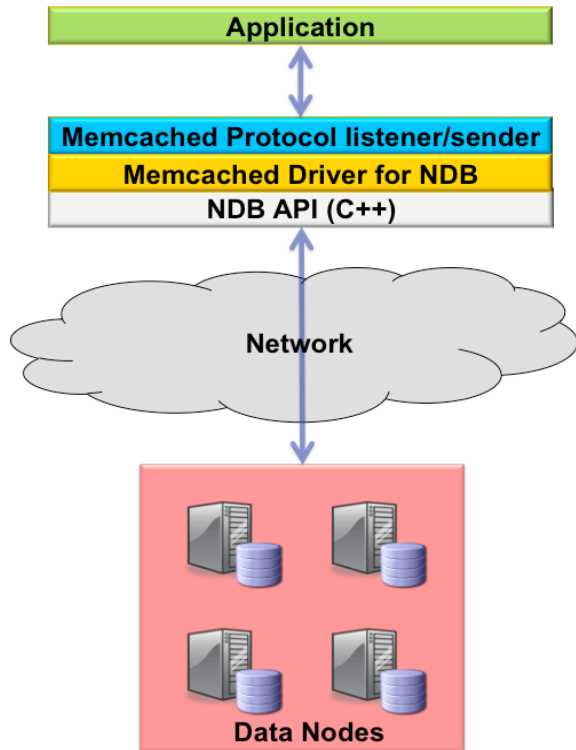


- New Domain Object Model Persistence API (ClusterJ) :
 - Java API
 - High performance, low latency
 - Feature rich
- JPA interface built upon this new Java layer:
 - Java Persistence API compliant
 - Implemented as an OpenJPA plugin
 - Uses ClusterJ where possible, reverts to JDBC for some operations
 - Higher performance than JDBC
 - More natural for most Java designers
 - Easier Cluster adoption for web applications

ClusterJ

- High Performance, Easy to Use
- In the style of Hibernate / JPA / JDO
- Domain Object Model DataMapper pattern
 - Data is represented as domain objects
 - Domain objects are separate from business logic
 - Domain objects are mapped to database tables
- Built on ndbjtie
 - JNI adapter
 - integral part of MySQL Cluster
 - Straight mapping of MySQL Cluster API (a.k.a NDB API) to Java
- Does not support relationships
 - Look at JDO / JPA for these modelling patterns

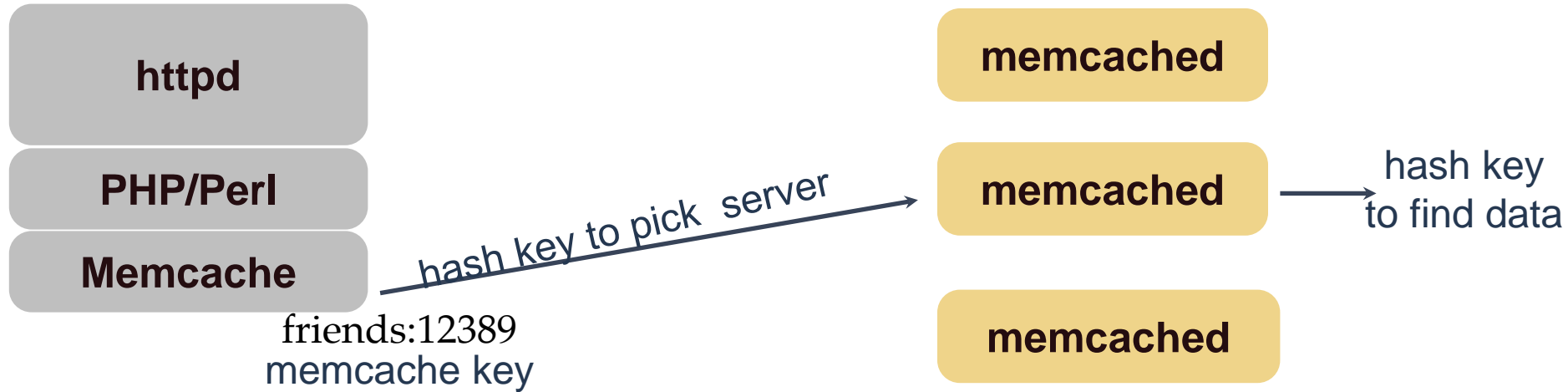
NoSQL with Memcached



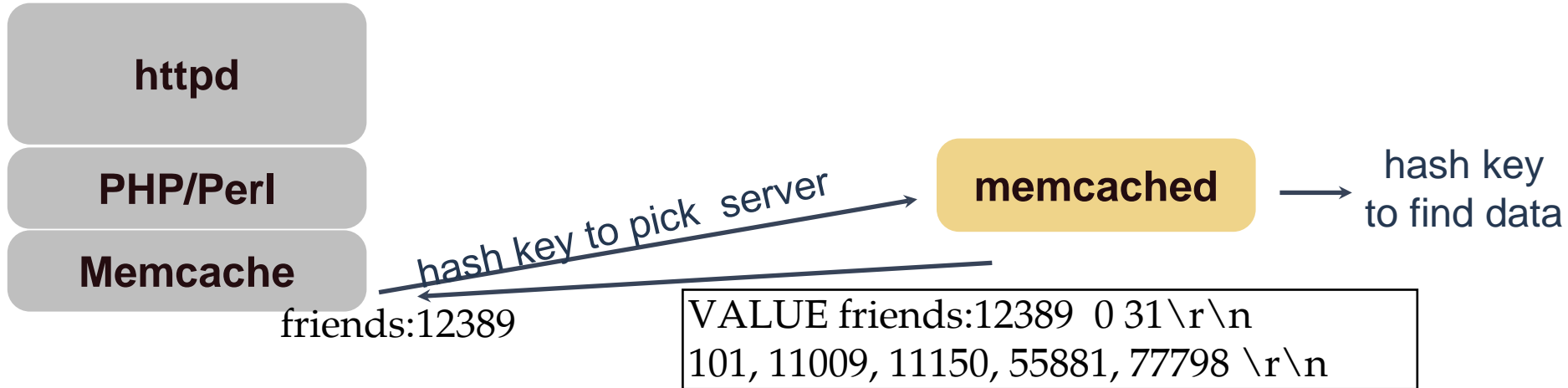
- Memcached is a distributed memory based hash-key/value store with no persistence to disk
- NoSQL, simple API, popular with developers
- MySQL Cluster already provides scalable, in-memory performance with NoSQL (hashed) access as well as persistence
 - Provide the Memcached API but map to NDB API calls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

Traditional Memcached Architecture

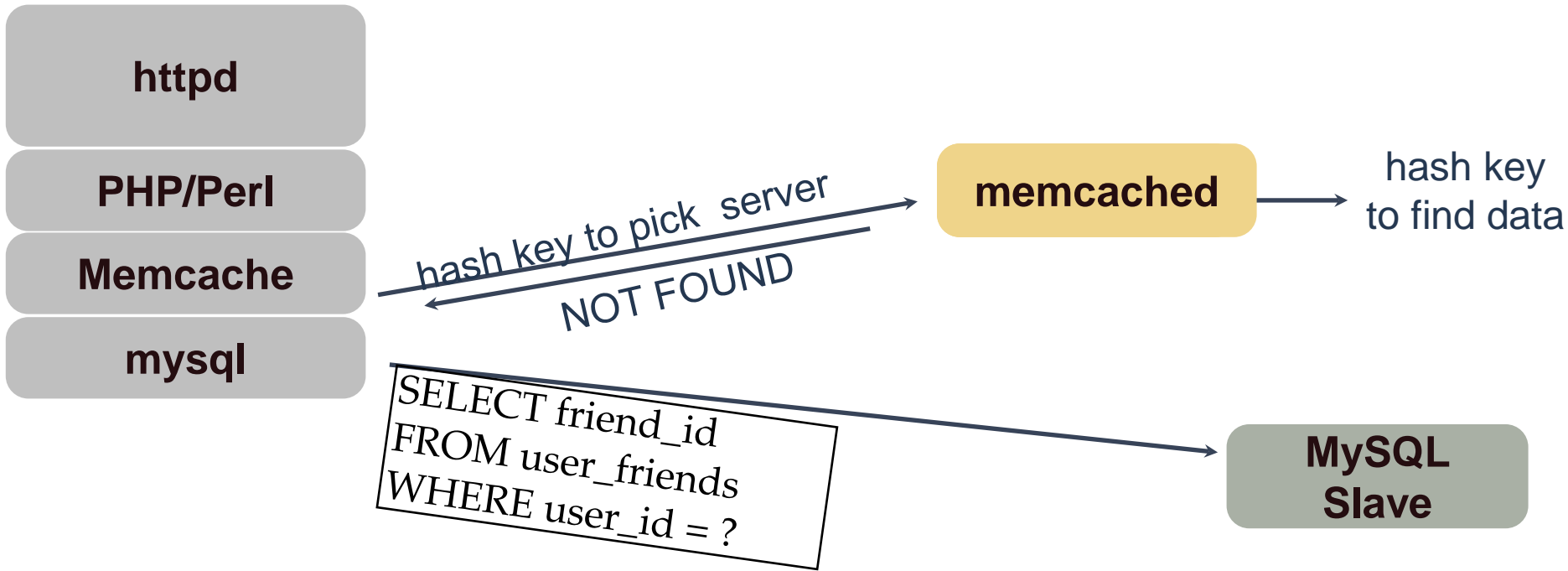
Two levels of hashing



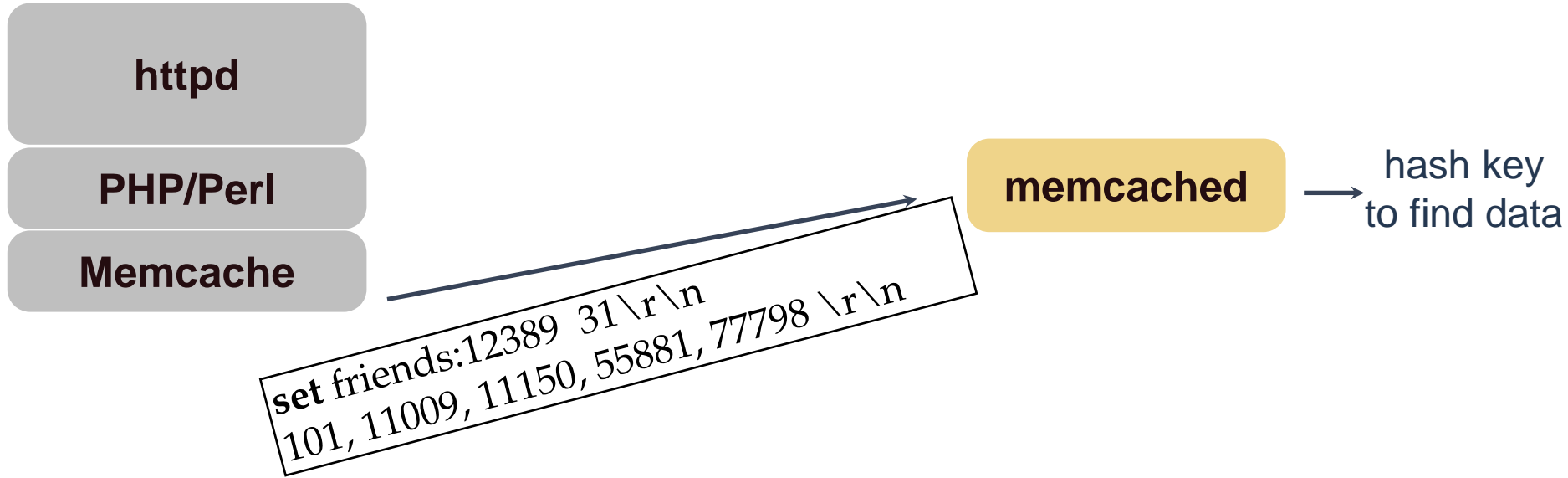
Cache hit



Cache miss (1): fetch from DB



Cache miss (2): manage cache



Data change (1): Write to DB

httpd

PHP/Perl

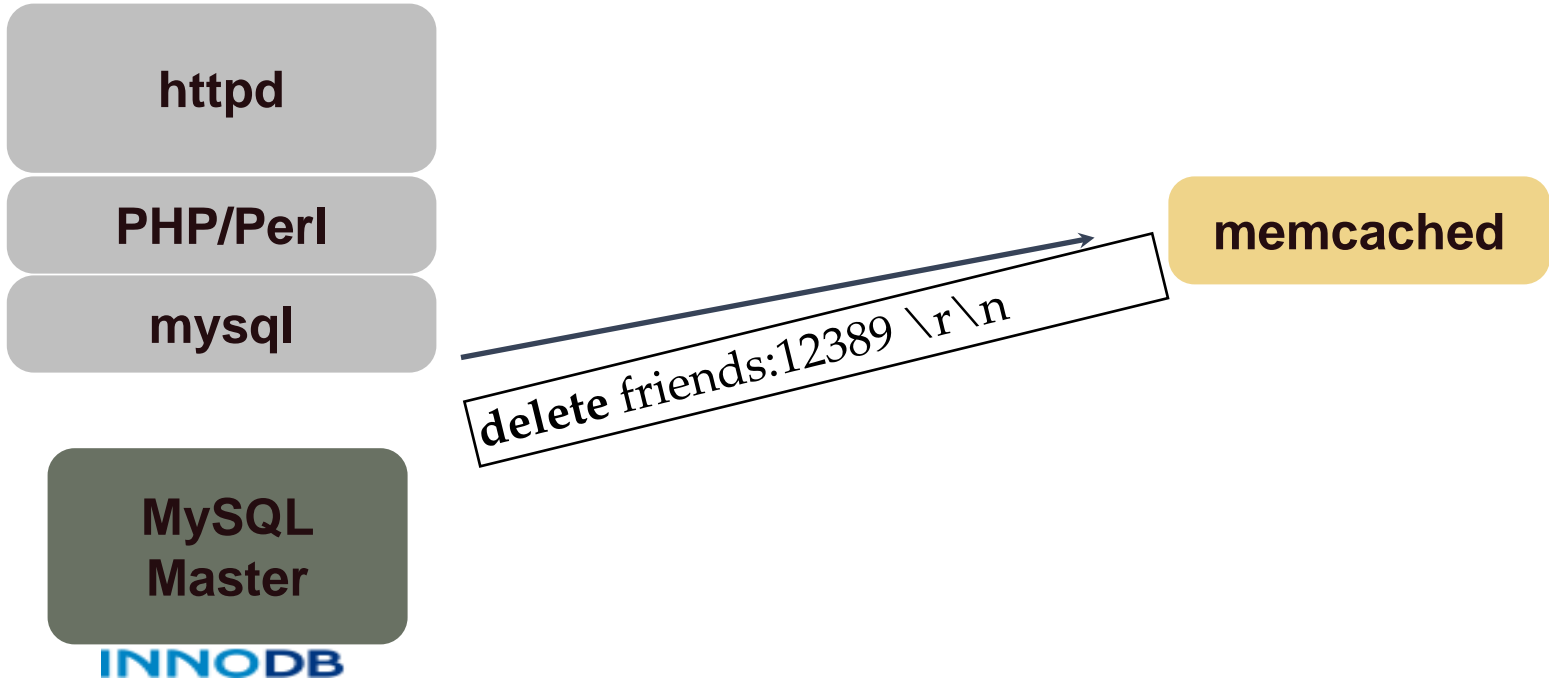
mysql

MySQL
Master

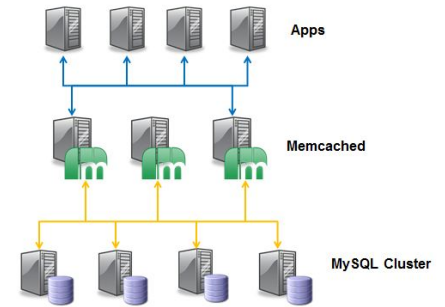
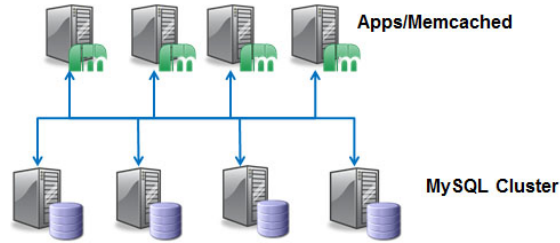
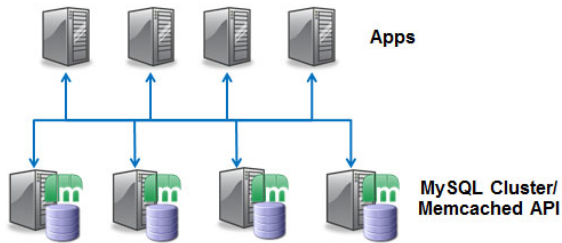
INNODB

```
INSERT INTO user_friends  
(user_id, friend_id)  
VALUES ( 12389, 999101);
```

Data change (2): manage cache



NoSQL with Memcached



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

```
set maidenhead 0 0 3
```

```
SL6
```

```
STORED
```

```
get maidenhead
```

```
VALUE maidenhead 0 3
```

```
SL6
```

```
END
```

Schema-Free apps

meal:lunch-cod random-96
age:fred-22 nick:james-jimmy home:blog-clusterdb.com
town:maidenhead-S16 edges:triangle-3
town:reading-RG1
edges:square-4
hair:fred-mohawk

- Rapid application evolution
 - New types of data constantly added
 - No time to get schema extended
 - Missing skills to extend schema
 - Initially roll out to just a few users
 - Constantly adding to live system

Cluster & Memcached – Schema-Free

meal:lunch-cod
random-96
home:blog-clusterdb.com
edges:triangle-3
town:reading-RG1
edges:square-4
hair:fred-mohawk
age:fred-22
nick:james-jimmy
town:maidenhead-SL6

Application view

key value
<town:maidenhead, SL6>

SQL view

key value
<town:maidenhead, SL6>

Key	Value
town:maidenhead	SL6

generic table

Cluster & Memcached - Configured Schema

meal:lunch-cod random-96
 home:blog-clusterdb.com
 edges:triangle-3
 town:reading-RG1
 edges:square-4
 hair:fred-mohawk
 age:fred-22
 nick:james-jimmy
 town:maidenhead-SL6

Application view

key value
 <town:maidenhead, SL6>

SQL view

prefix key value
 <town:maidenhead, SL6>

Prefix	Table	Key-col	Val-col	policy
town:	map.zip	town	code	cluster

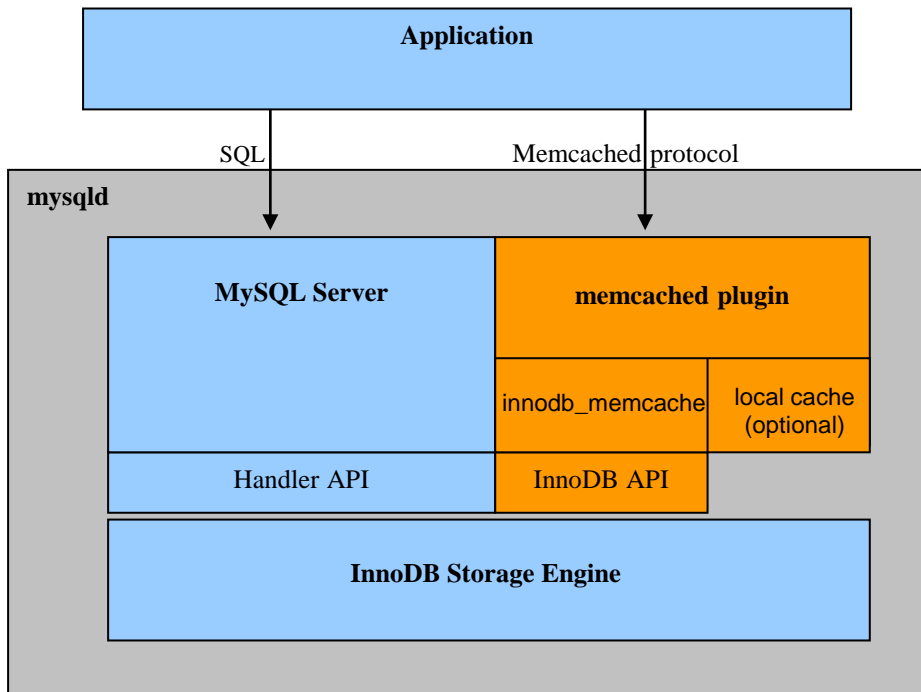
Config tables

town	...	code	...
maidenhead	...	SL6	...

map.zip

Memcached NoSQL Access with InnoDB

October labs release



- Memcached as a plugin of MySQL Server; same process space, with very low latency access to data
- Memcapable: supports both memcached ascii protocol and binary protocol
- Support multiple columns: users can map multiple columns into “value”
- Optional local caching: “innodb-only”, “cache-only”, and “caching”
- Batch operations for performance
- New in October labs:
 - Binlogging of DML operations
 - Memcached 1.6.0 (beta 1)
 - libevent to version 1.4.12

Which API to use?

SQL

- Industry standard
- Joins & complex queries
- Relational model

Memcached

- simple to use API
- key/value
- driver for many languages
- ideal as e.g. PHP proxy

mod_ndb

- REST/JSON
- HTML
- using Apache

ClusterJ

- simple to use Java API
- Web & telco
- Object Relational Mapping
- native & fast access to MySQL Cluster

C++

- experienced developer
- ultra low latency / real-time

Scalability	✓
Performance	✓
HA	✓
Ease of use	✓
SQL/Joins	✓
ACID Transactions	✓

Summary

Today's web workloads demand more from databases

Performance, scale-out, simpler access patterns & APIs

MySQL meets these needs while still delivering benefits of an ACID RDBMS



**ENGINEERED
FOR INNOVATION**

