MySQL

ORACLE

MySQL's NoSQL Interface

Dave Stokes
MySQL Community Manager

David.Stokes@Oracle.com
@stoker
Slideshare.net/davestokes

Safe Harbor Statement

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.

Quick Quiz – Who said

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed

Quick Quiz – Who said

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed – EF Codd, June 1970

MySQL

- Most popular database on the web
- Ubiquitous
- 16+ million instances
- Feeds 80% of Hadoop installs

Access SQL and NoSQL

At the same time!

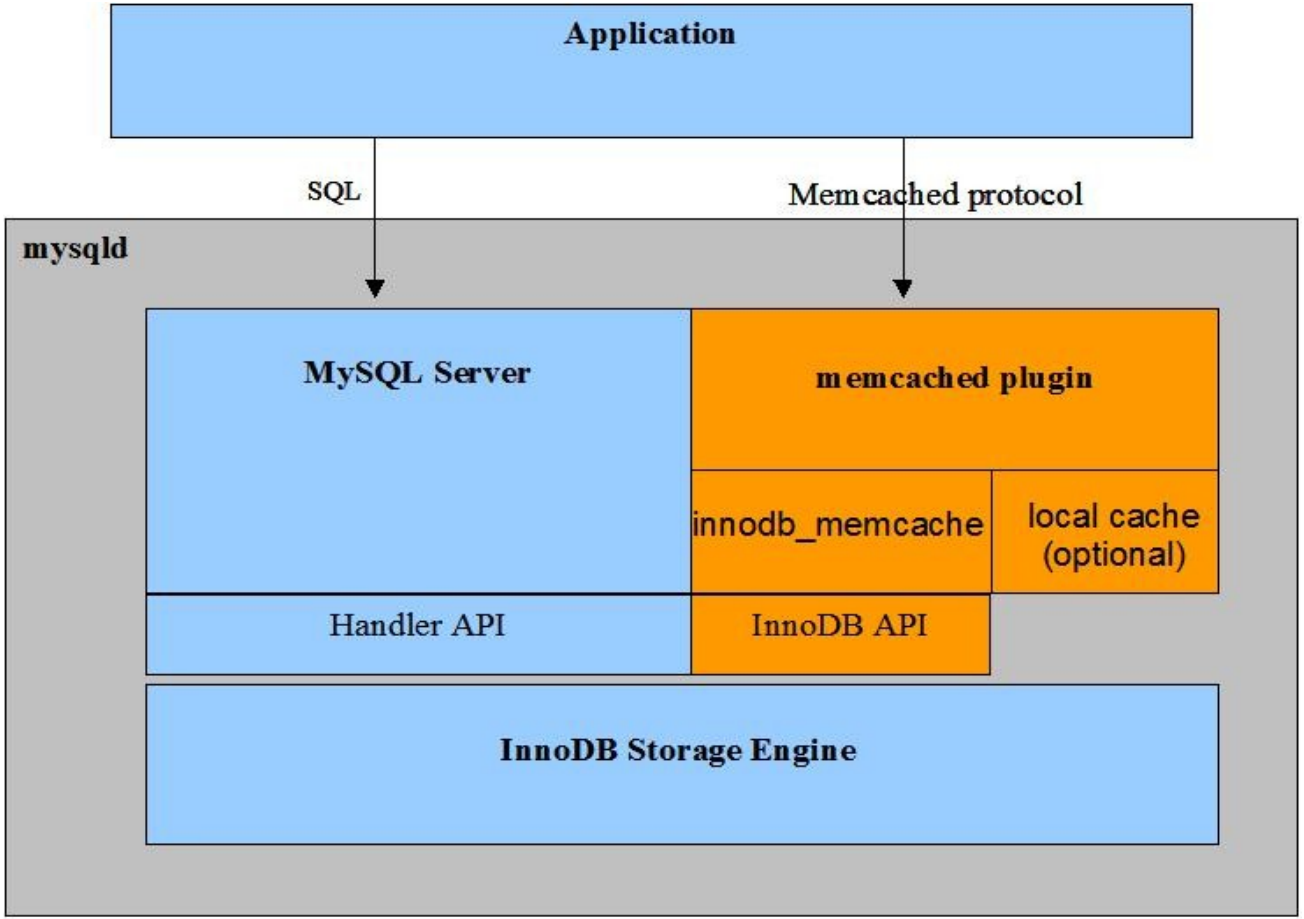
One set of disks

Simultaneous access

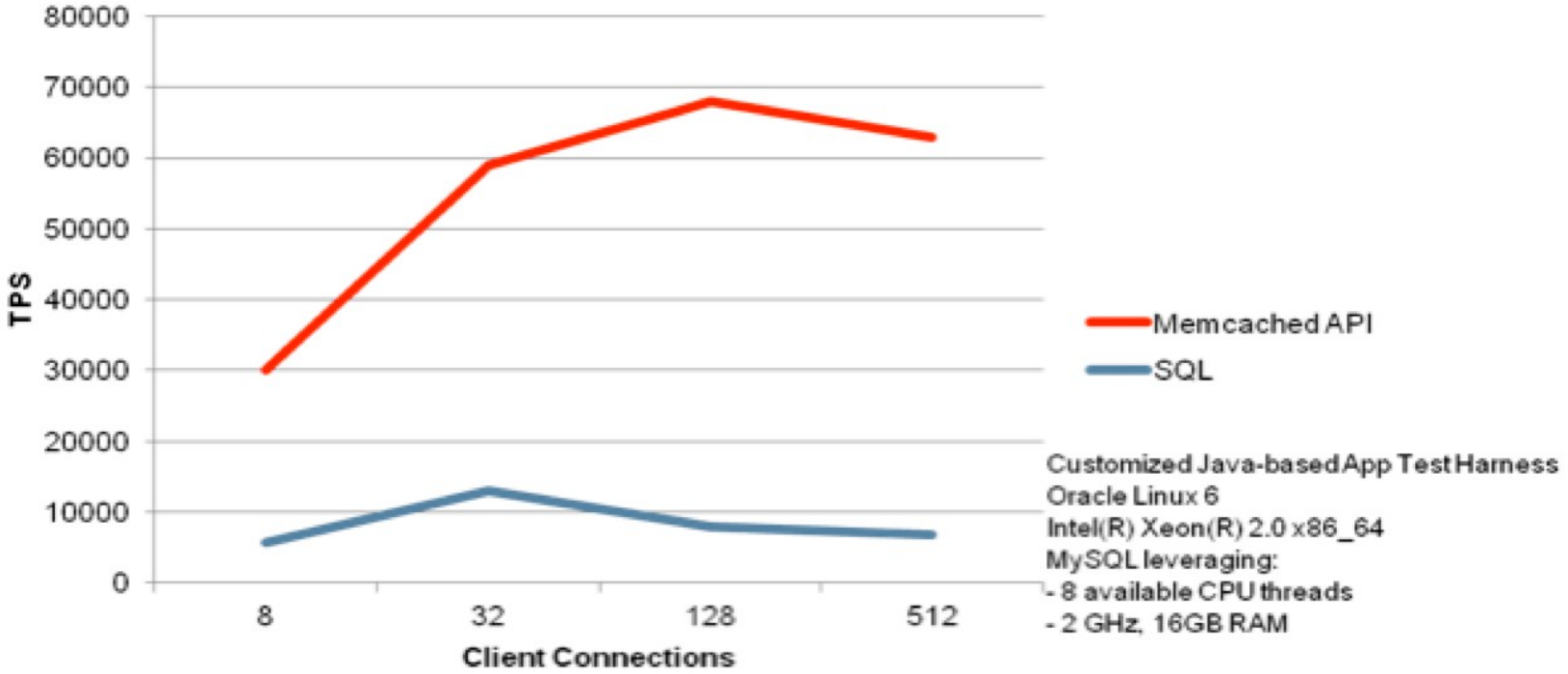
MySQL InnoDB tables and NDB tables

Use SQL and/or Key/Value pair

2,000,000,000 writes a minute with MySQL Cluster



MySQL 5.6: NoSQL Benchmarking



Benefits

Raw performance for simple lookups. Direct access to the InnoDB storage engine avoids the parsing and planning overhead of SQL. Running memcached in the same process space as the MySQL server avoids the network overhead of passing requests back and forth.

Data is stored in a MySQL database to protect against crashes, outages, and corruption.

The transfer between memory and disk is handled automatically, simplifying application logic.

Data can be unstructured or structured, depending on the type of application. You can make an all-new table for the data, or map the NoSQL-style

Benefits continued

You can still access the underlying table through SQL, for reporting, analysis, ad hoc queries, bulk loading, set operations such as union and intersection, and other operations well suited to the expressiveness and flexibility of SQL.

You can ensure high availability of the NoSQL data by using this feature on a master server in combination with MySQL replication.

The integration of memcached with MySQL provides a painless way to make the in-memory data persistent, so you can use it for more significant kinds of data. You can put more add, incr, and similar write operations into your application, without worrying that the data could disappear at any moment. You can stop and start the memcached server without losing

 |

More Benefits Continued

The serialization features of memcached, which can turn complex data structures, binary files, or even code blocks into storeable strings, offer a simple way to get such objects into a database.

Because you can access the underlying data through SQL, you can produce reports, search or update across multiple keys, and call functions such as AVG() and MAX() on the memcached data. All of these operations are expensive or complicated with the standalone memcached.

You do not need to manually load data into memcached at startup. As particular keys are requested by an application, the values are retrieved from the database automatically, and cached in memory using the InnoDB buffer pool.

Installation

```
mysql> install plugin daemon_memcached
soname "libmemcached.so";
```

Here is an example using telnet to send memcached commands and receive results through the ASCII protocol:

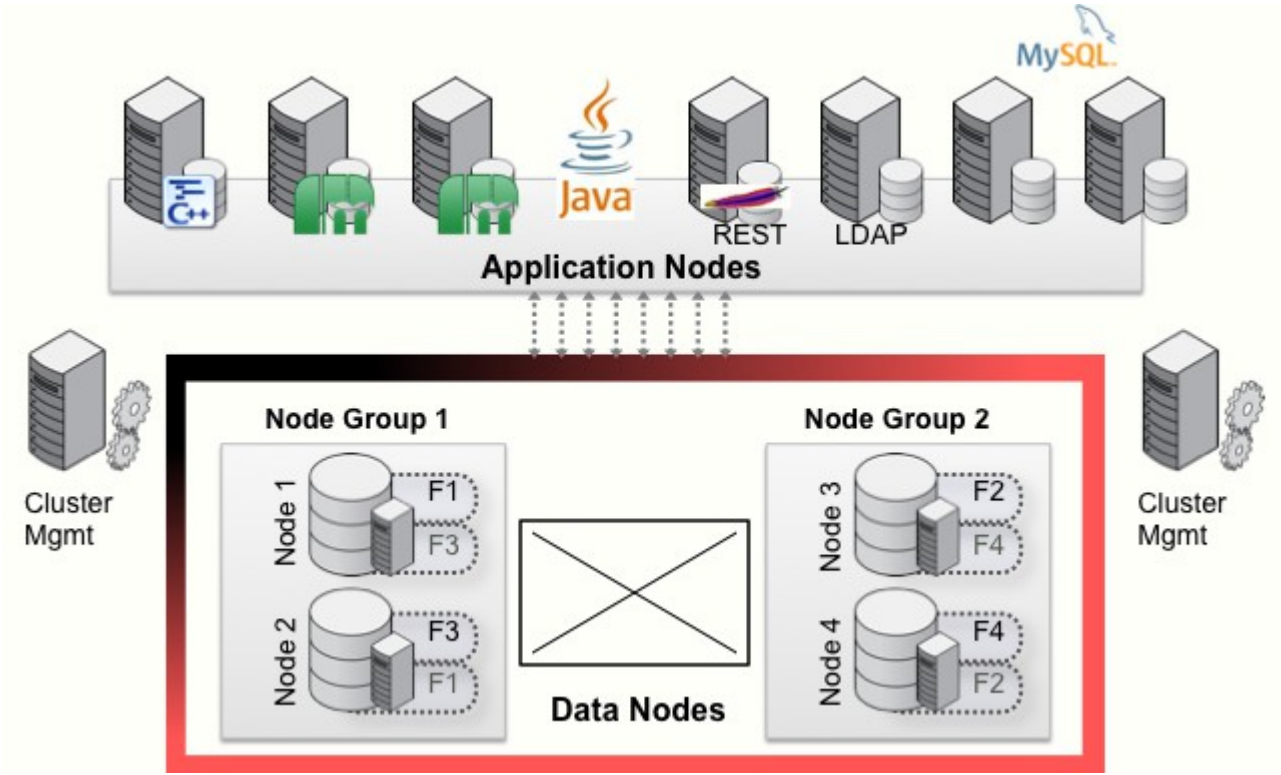
- telnet 127.0.0.1 11211
- set a11 10 0 9
- 123456789
- STORED
- get a11
- VALUE a11 0 9
- 123456789

Can it be used with MySQL Replication?

Because the InnoDB memcached daemon plugin supports the MySQL binary log, any updates made on a master server through the memcached interface can be replicated for backup, balancing intensive read workloads, and high availability. All memcached commands are supported for binlogging.

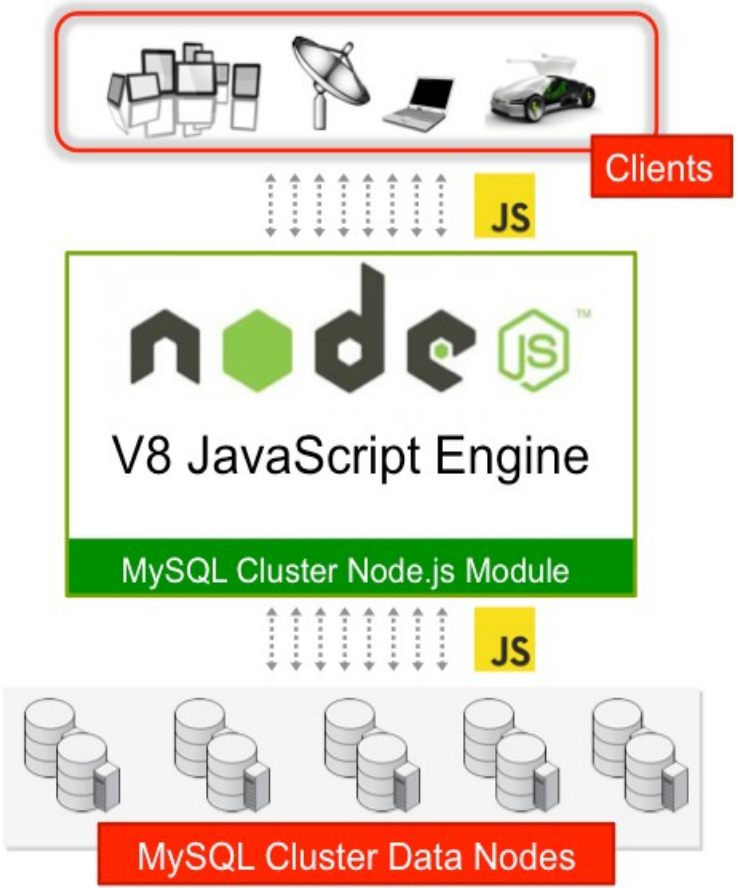
You do not need to set up the InnoDB memcached plugin on the slave servers. In this configuration, the primary advantage is increased write throughput on the master. The speed of the replication mechanism is not affected

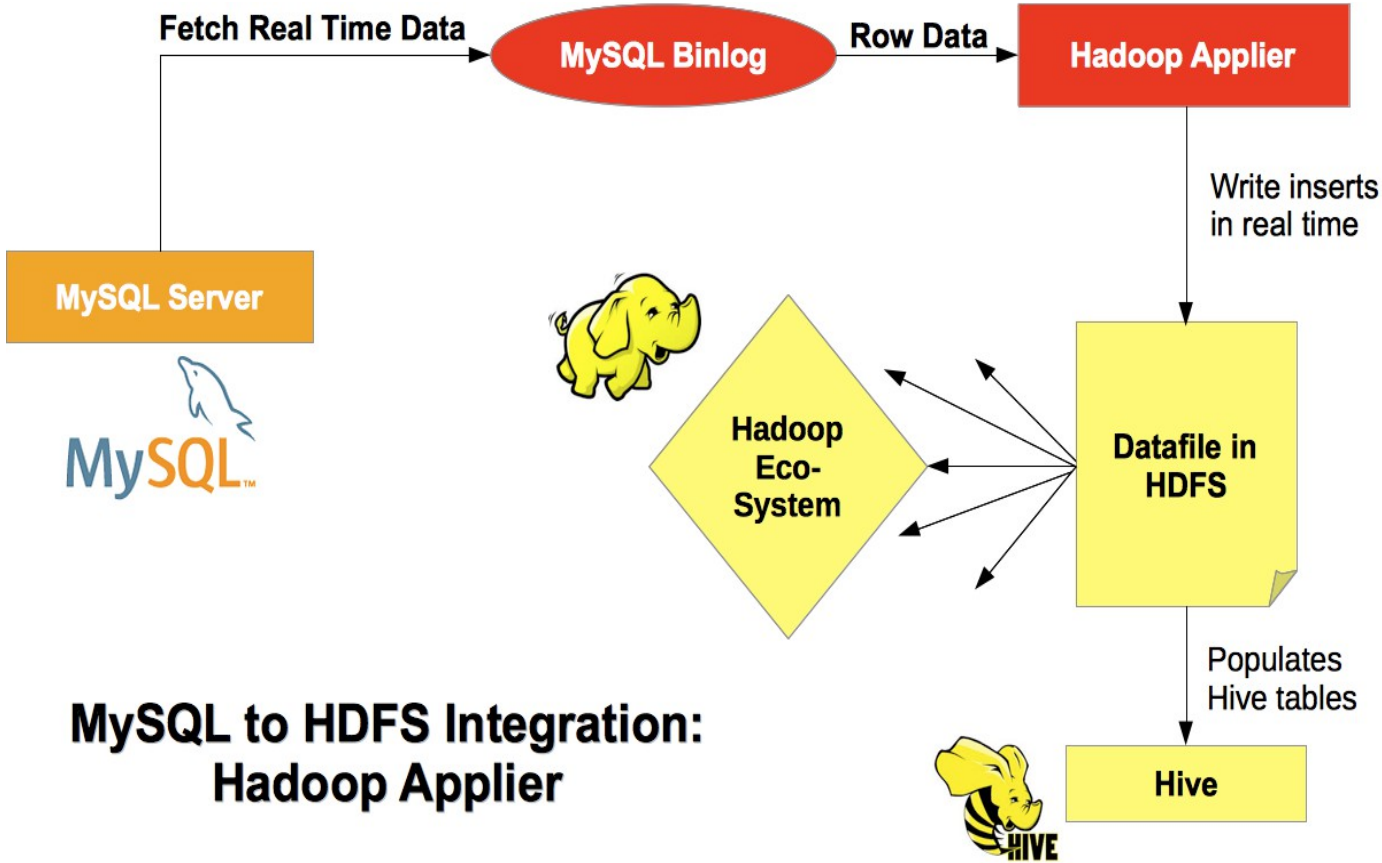
With MySQL Cluster



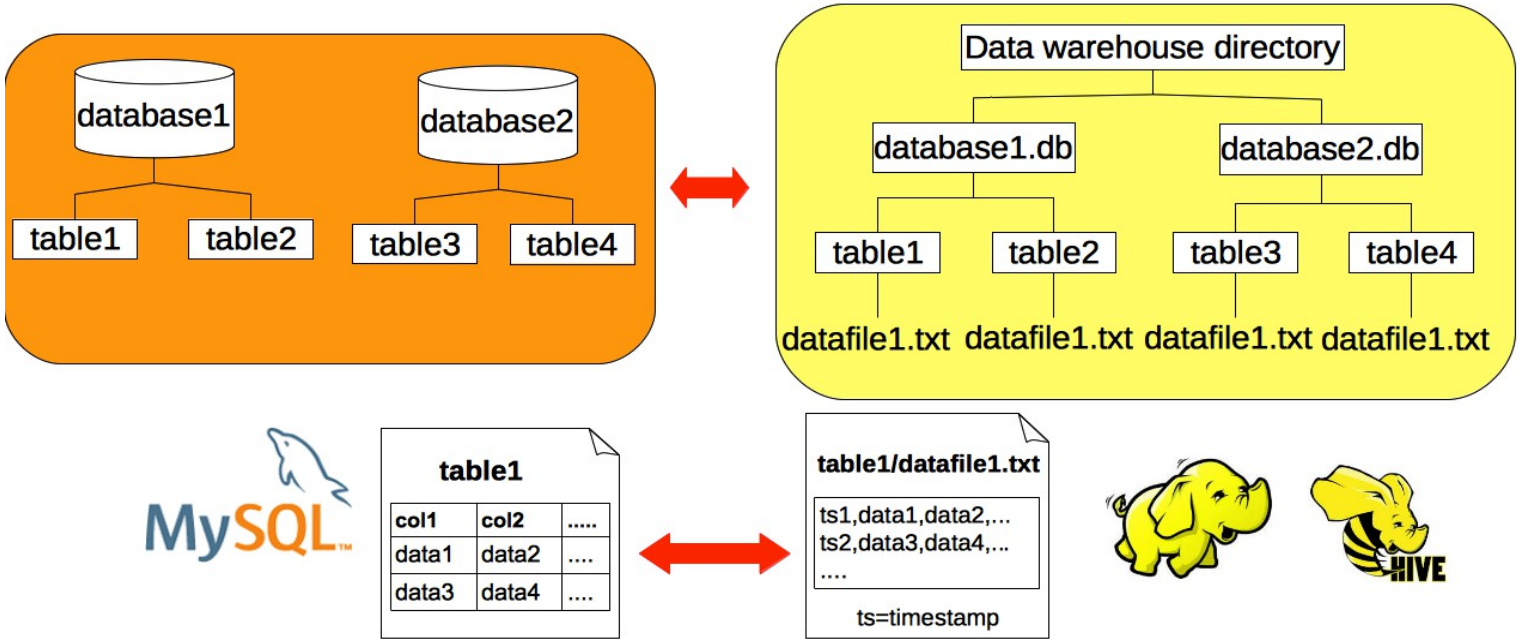
Bypass MySQL daemon!

- 4.2 billions reads/min
- 1.2 billion updates/min
- 2 billion writes a min
  - MySQL Cluster





MySQL to HDFS Integration: Hadoop Applier



For More Information

- Mysql.com
- Labs.mysql.com
- Planet.mysql.com

- Dave Stokes

  David.Stokes@Oracle.com

  @stoker

  slideshare.net/davestokes

Tonight – Improvements in MySQL 5.6 Query Tuning

- When Wed Oct 30, 2013 5:15pm – 7:30pm (CDT)
- Where ERANYC (214 west 29th street, New York, NY 10001)

ORACLE

MySQL™ Connect