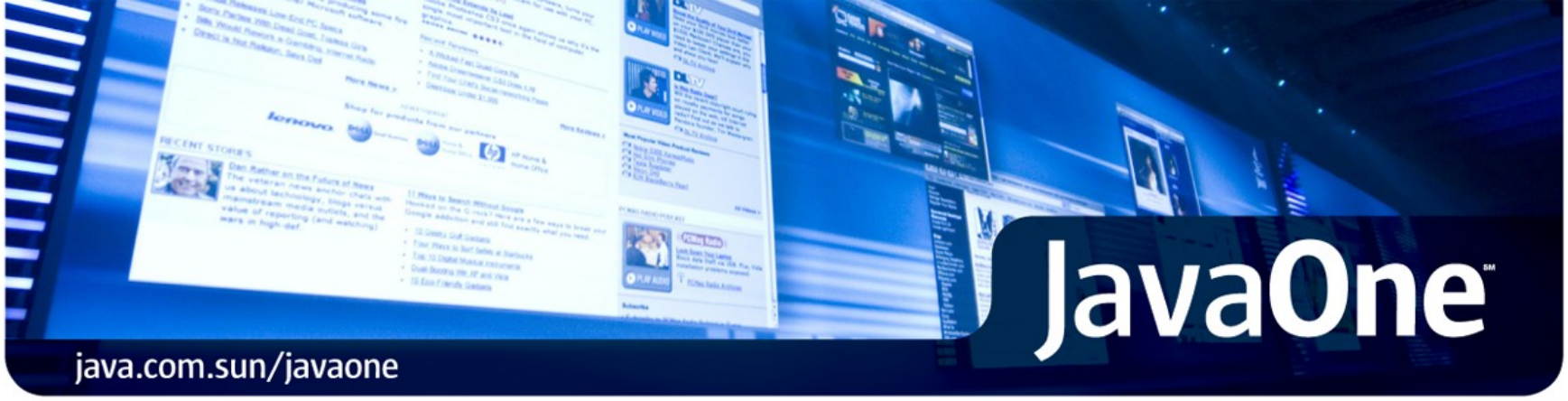


MySQL Cluster and Java Technology (and Python and Ruby and ...)

Monty Taylor, Senior Consultant, MySQL

ID#7814

mordred@sun.com

Java

Sun microsystems

Learn about MySQL™ Cluster, NDB API, NDB/J and a whole host of other exciting things

Excitement may or may not occur. The speaker takes no responsibility for overall excitement level, or any other spiritual or emotional responses.

GOAL

What are we going to talk about?

- MySQL Cluster
- NDB API
- Java™ (NDB/J)

What is MySQL Cluster?

- A High Availability

What is MySQL Cluster?

- A High Availability
- High Performance

What is MySQL Cluster?

- A High Availability
- High Performance
- In Memory

What is MySQL Cluster?

- A High Availability
- High Performance
- In Memory
- Shared Nothing

What is MySQL Cluster?

- A High Availability
- High Performance
- In Memory
- Shared Nothing
- Clustered

What is MySQL Cluster?

- A High Availability
- High Performance
- In Memory
- Shared Nothing
- Clustered
- Storage Engine

What is MySQL Cluster?

- A High Availability
- High Performance
- In Memory
- Shared Nothing
- Clustered
- Storage Engine

What is MySQL Cluster?

High Availability

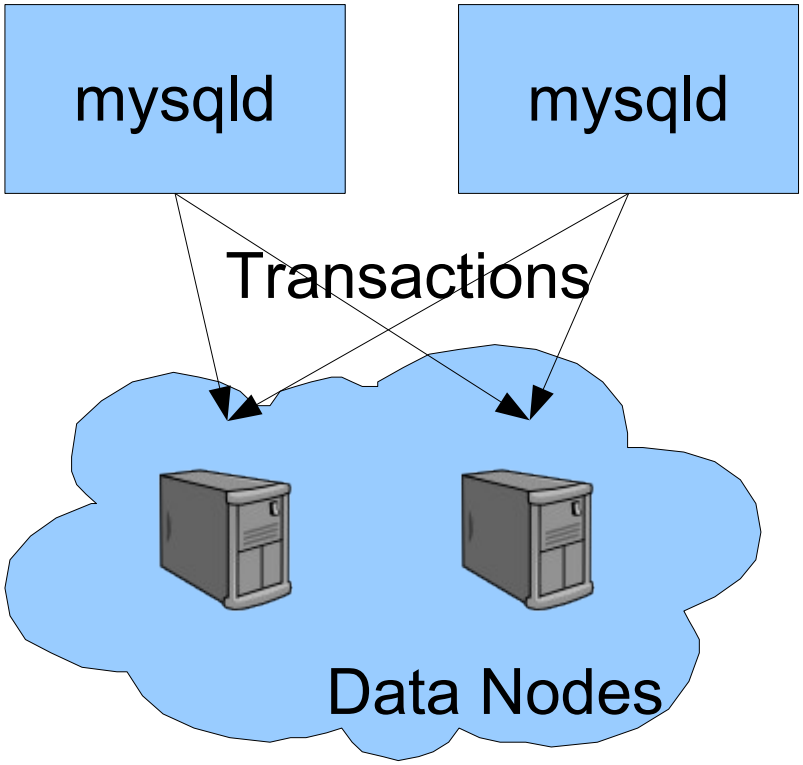
High Availability

Designed for Five Nines (99.999%) Uptime

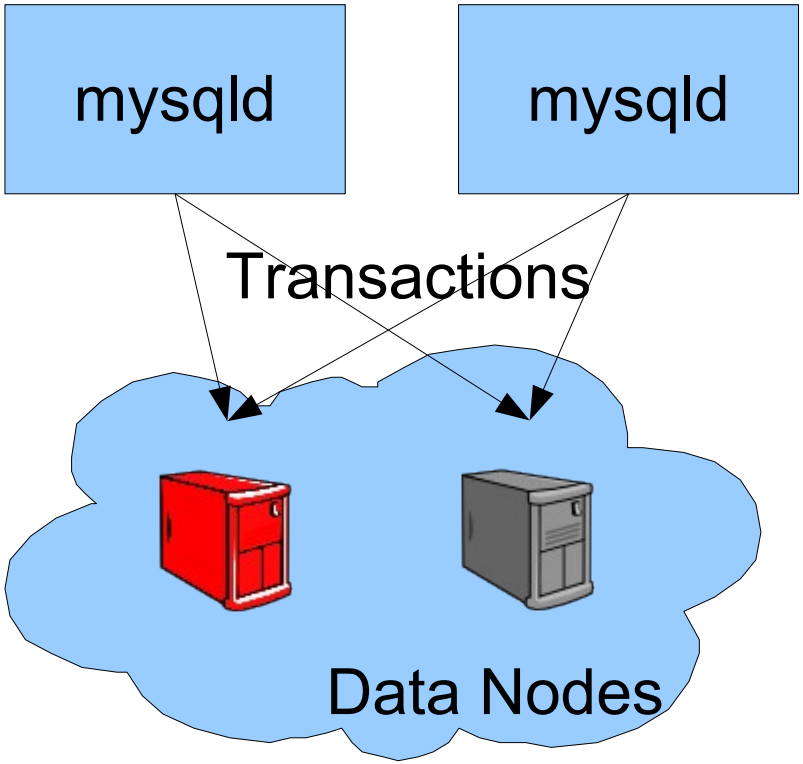
High Availability

Sub-Second Failover

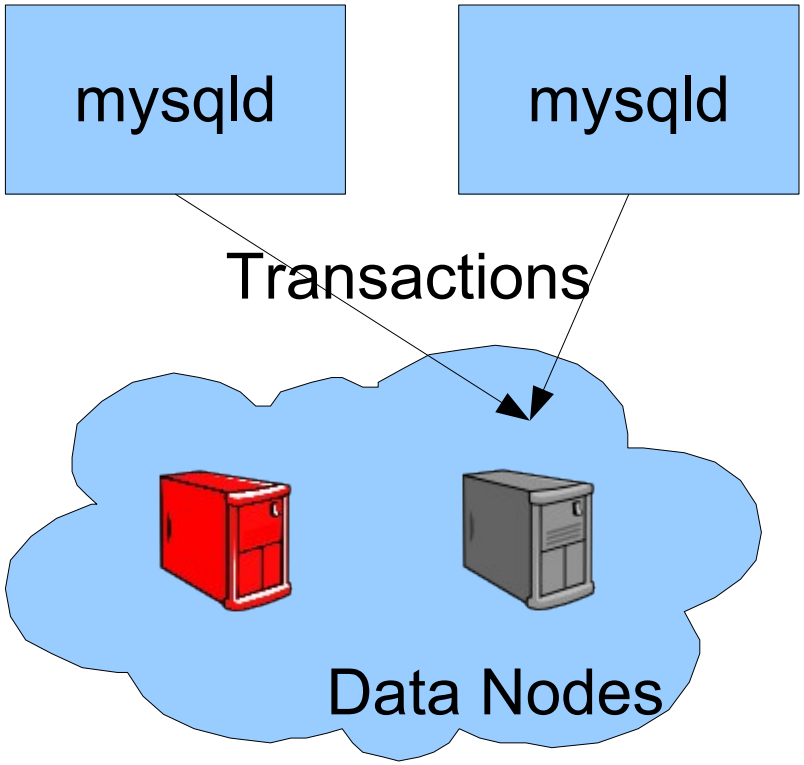
High Availability



High Availability



High Availability

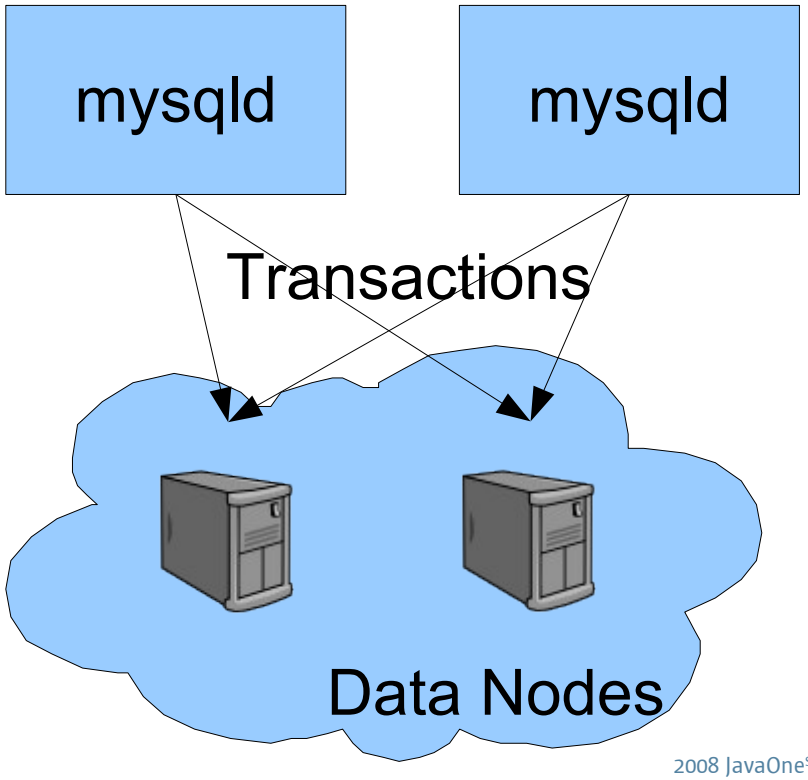


High Availability

Hot (Online) Backup

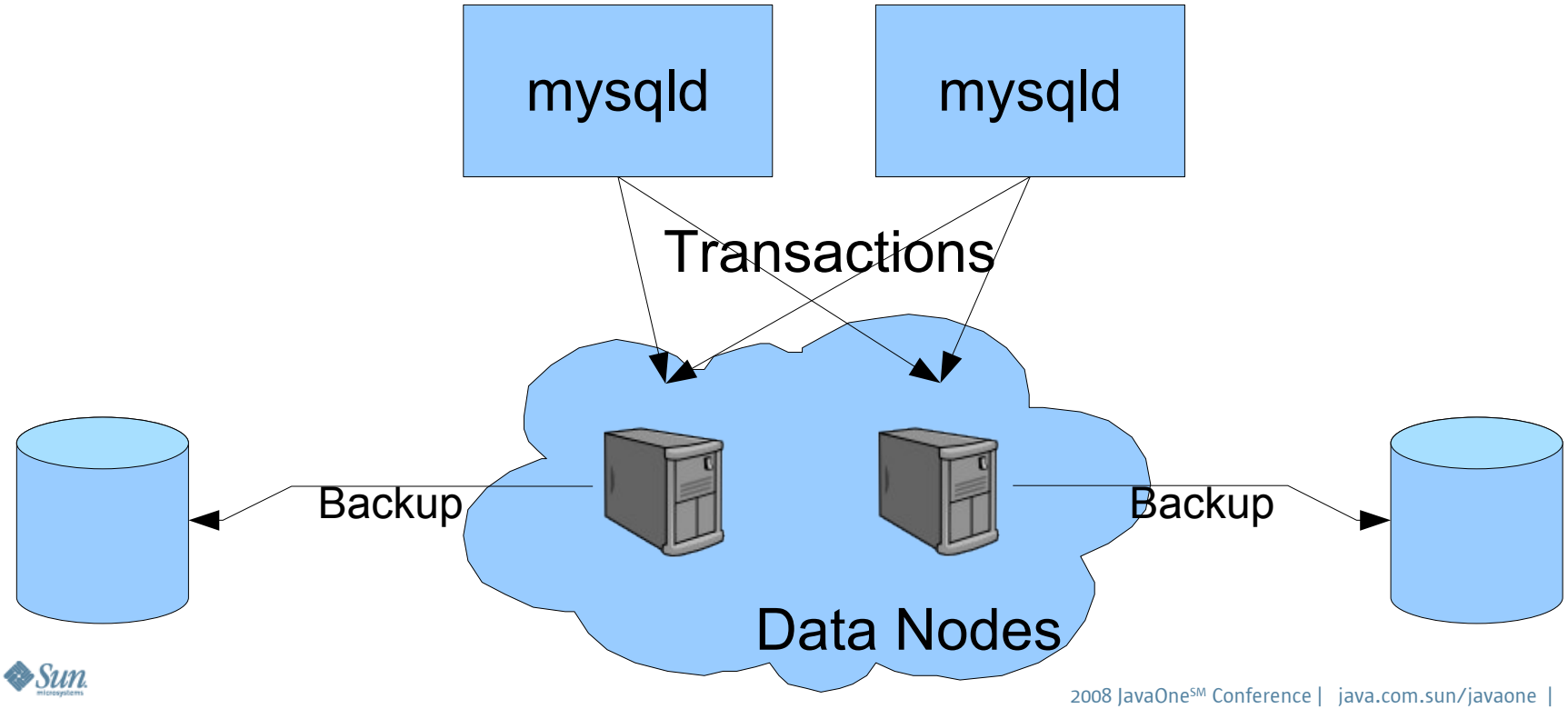
High Availability

Hot (Online) Backup



High Availability

Hot (Online) Backup



High Availability

Configurable Amount of Redundancy

High Availability

Configurable Amount of Redundancy

NoOfReplicas

High Availability

NoOfReplicas=1




D a t a

High Availability

NoOfReplicas=2




Da ta Da ta

High Availability

NoOfReplicas=3

Da Da ta Da ta ta

High Availability

NoOfReplicas=4




Data Data Data Data

What is MySQL Cluster?

- A High Availability
- High Performance
- In Memory
- Shared Nothing
- Clustered
- Storage Engine

High Performance

High Performance

High Performance

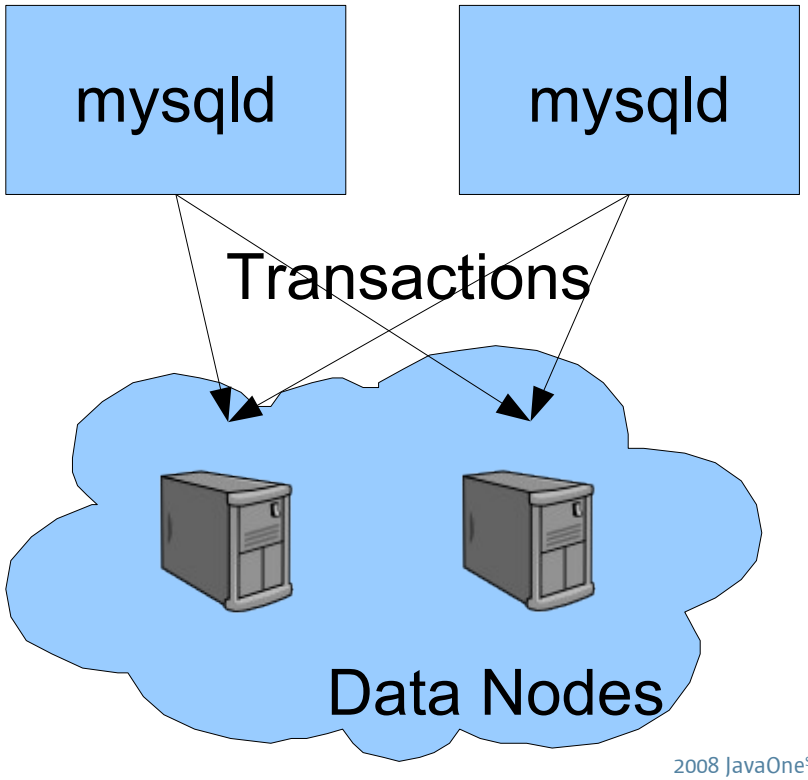
Not BEGIN to COMMIT

High Performance

Not BEGIN to COMMIT Through Parallelism

High Performance

Parallelism



What is MySQL Cluster?

- A High Availability
- High Performance
- In Memory
- Shared Nothing
- Clustered
- Storage Engine

What is MySQL Cluster?

In Memory

In Memory (4.1 and 5.0)

Data and Indexes kept in main memory

$$\text{Per Node} = \frac{\text{Size of Database} \times \text{No Of Replicas} \times 1.1}{\text{No Of Data Nodes}}$$

In Memory

Check point to disk

In Memory (4.1 and 5.0)

Check point to disk

Frequent, Configurable

In Memory (4.1 and 5.0)

Check point to disk

Not complete data loss after power outage

In Memory

Data on Disk in 5.1

In Memory

Data on Disk in 5.1

Indexed fields and Indexes in main memory

What is MySQL Cluster?

- A High Availability
- High Performance
- In Memory (4.1 and 5.0)
  Shared Nothing
- Clustered
- Storage Engine

What is MySQL Cluster?

- Shared Nothing

Shared Nothing

Commodity PCs

Shared Nothing

Commodity Interconnects

Shared Nothing

No Expensive Shared Disk

Shared Nothing

No Expensive Shared Disk

(so no single point of failure there)

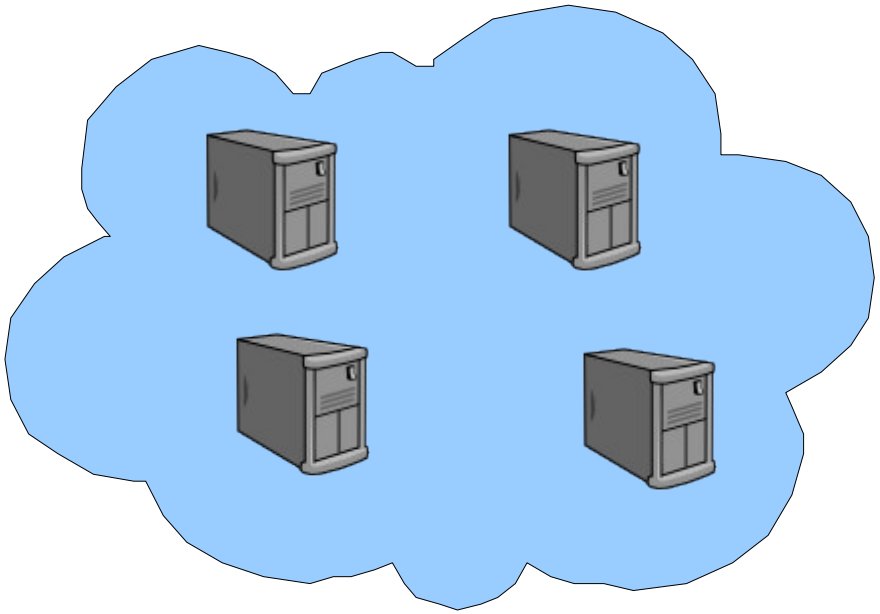
What is MySQL Cluster?

- A High Availability
- High Performance
- In Memory (4.1 and 5.0)
  Shared Nothing
- Clustered
- Storage Engine

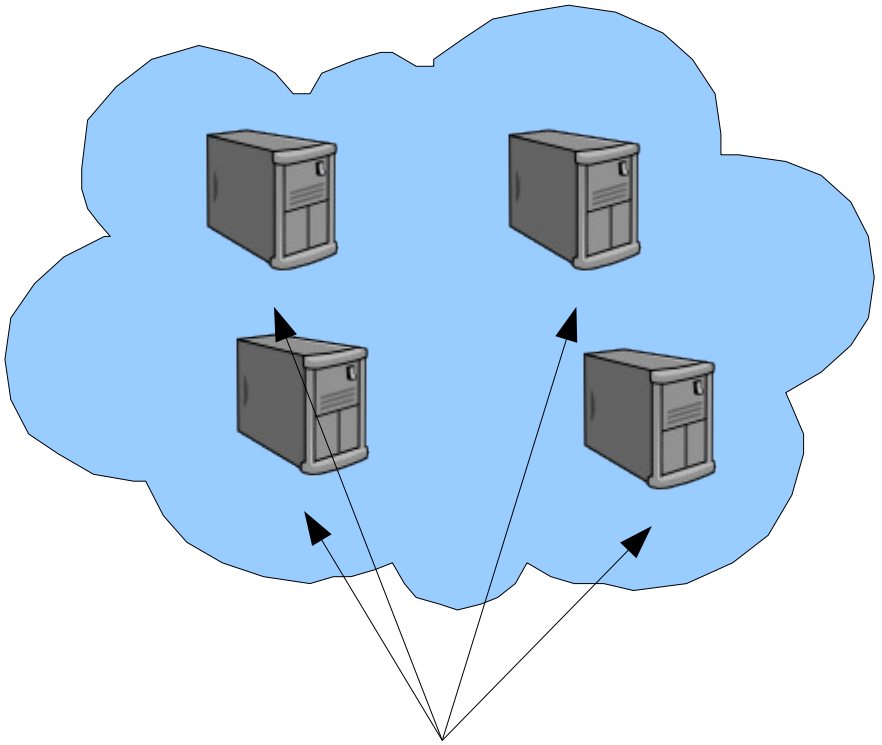
What is MySQL Cluster?

- Clustered

Data Nodes



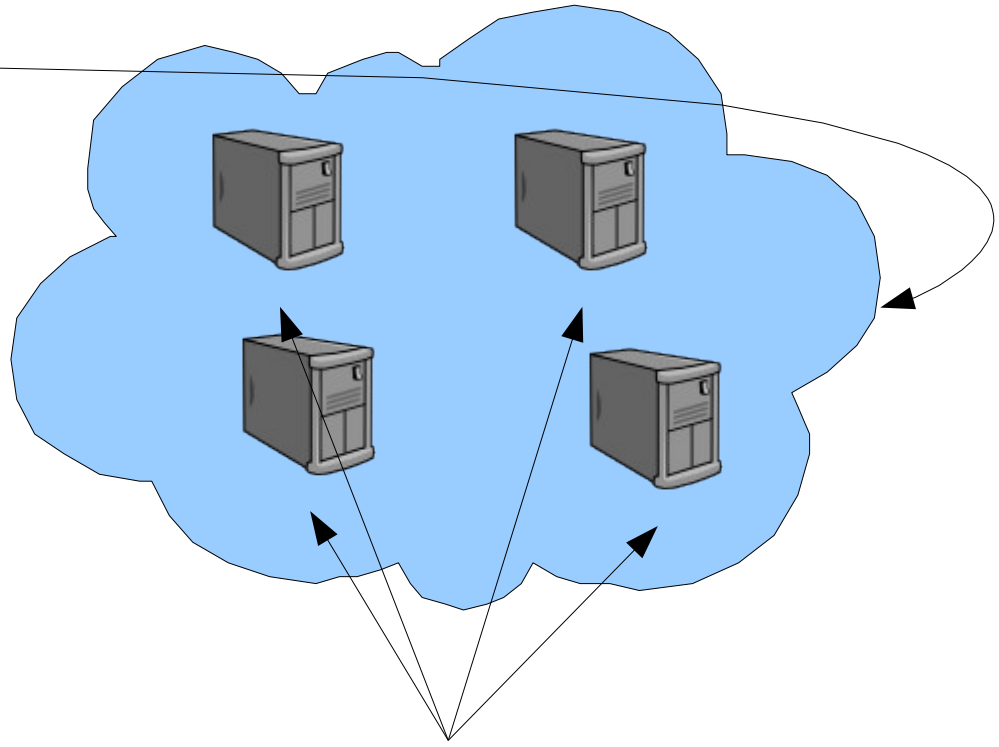
Data Nodes



Data nodes (running ndbd)

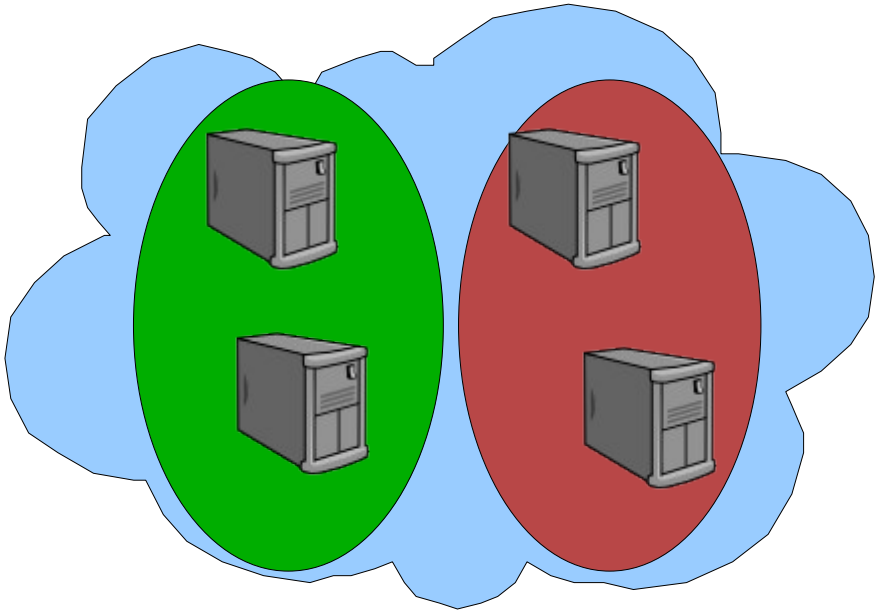
Data Nodes

This cloud means a cluster

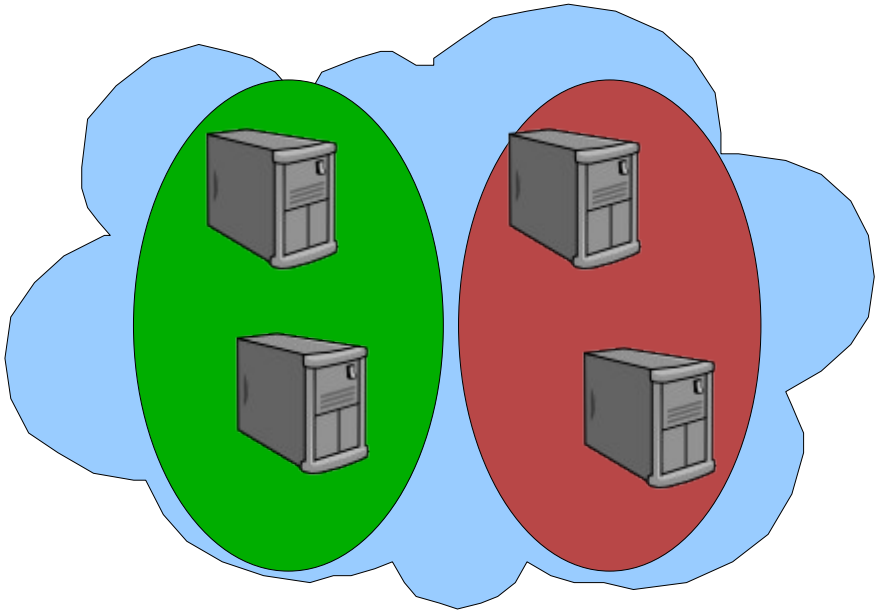


Data nodes (running ndbd)

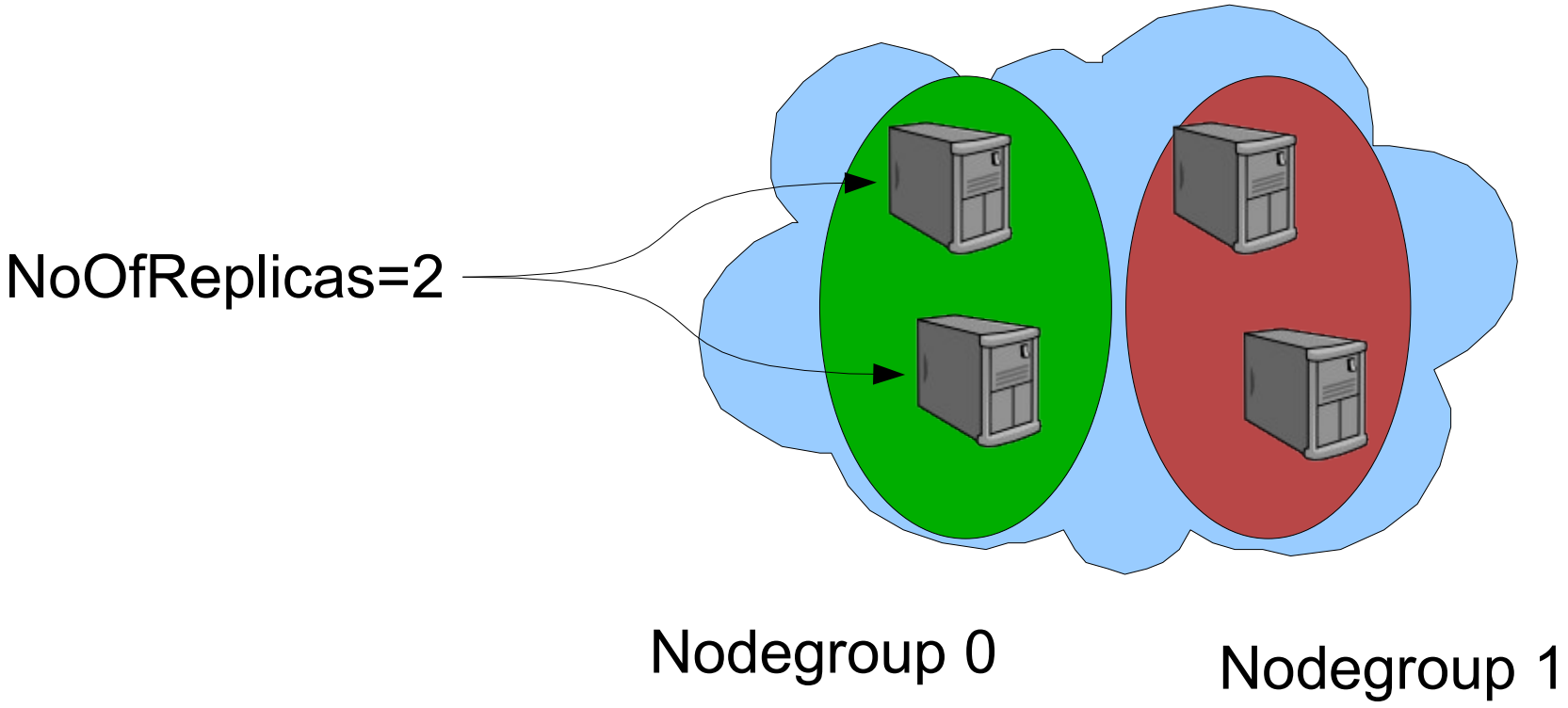
Data Nodes grouped



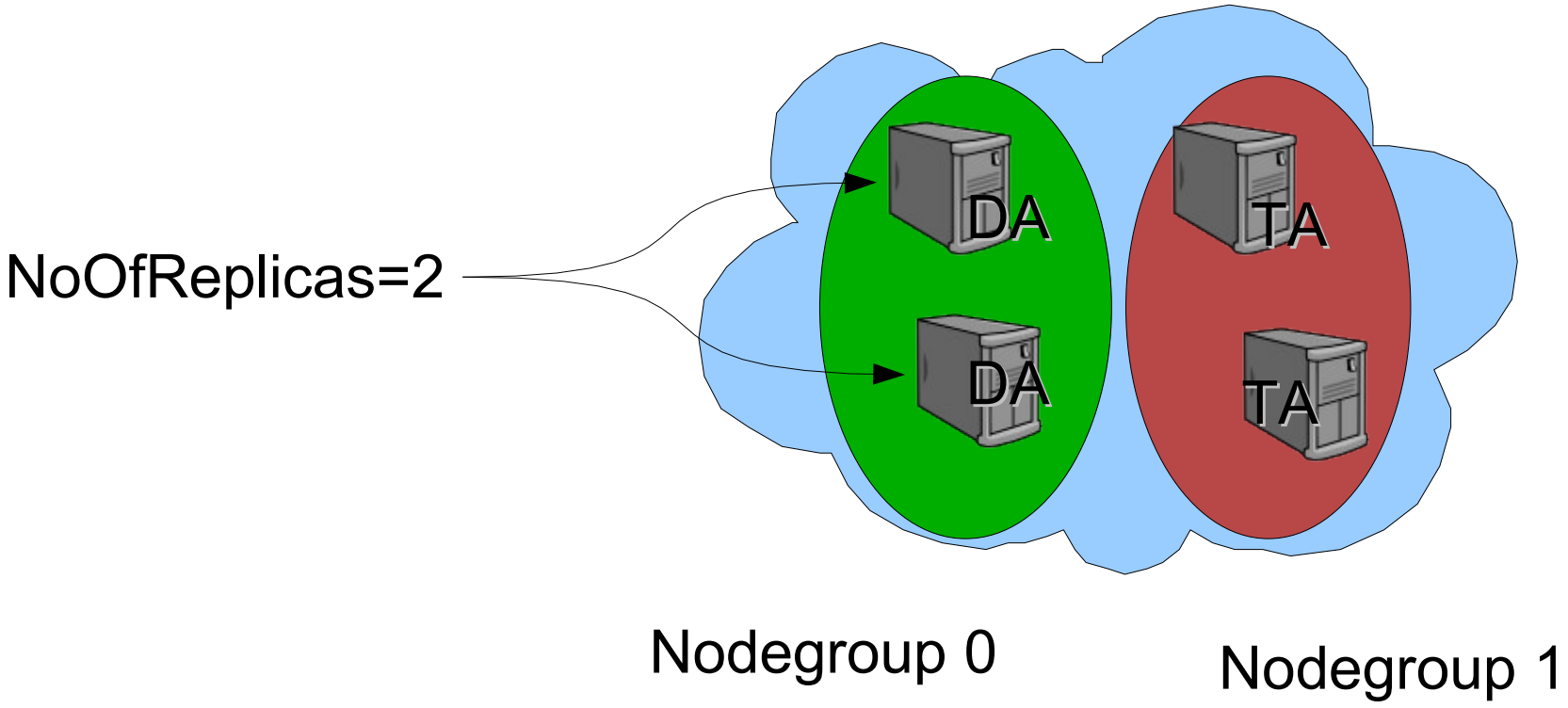
Data Nodes grouped into nodegroups



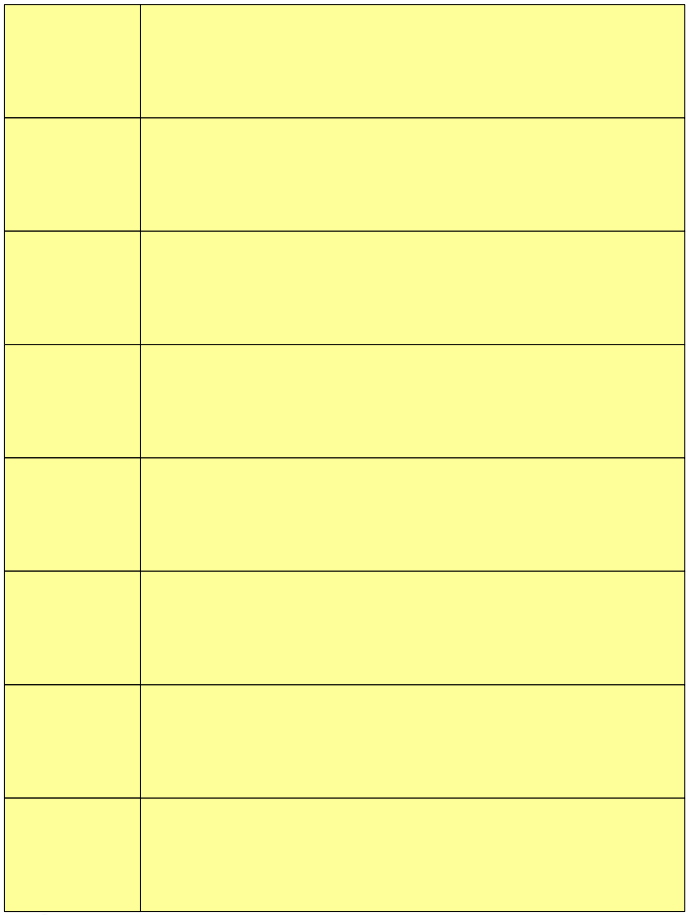
NoOfReplicas is number of nodes in node group

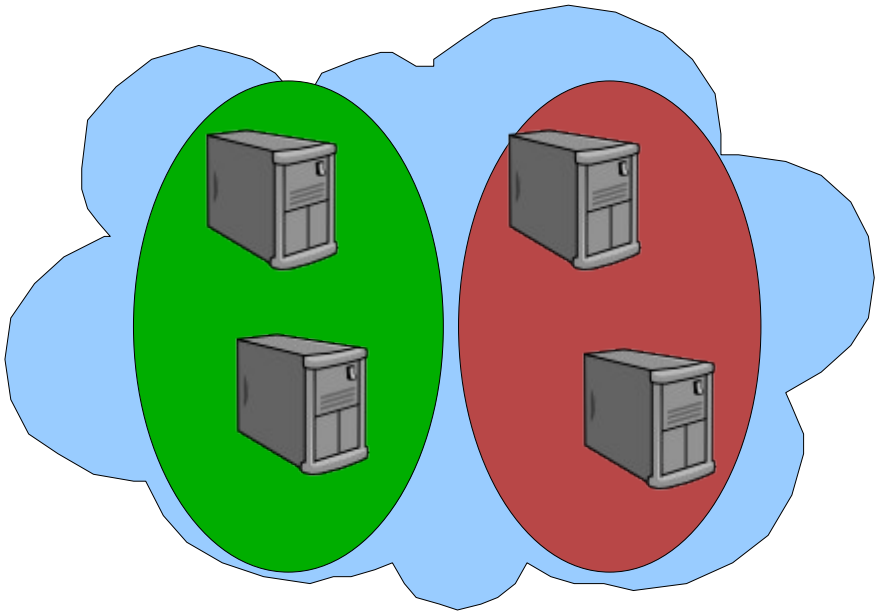


NoOfReplicas is number of nodes in node group



Data Nodes



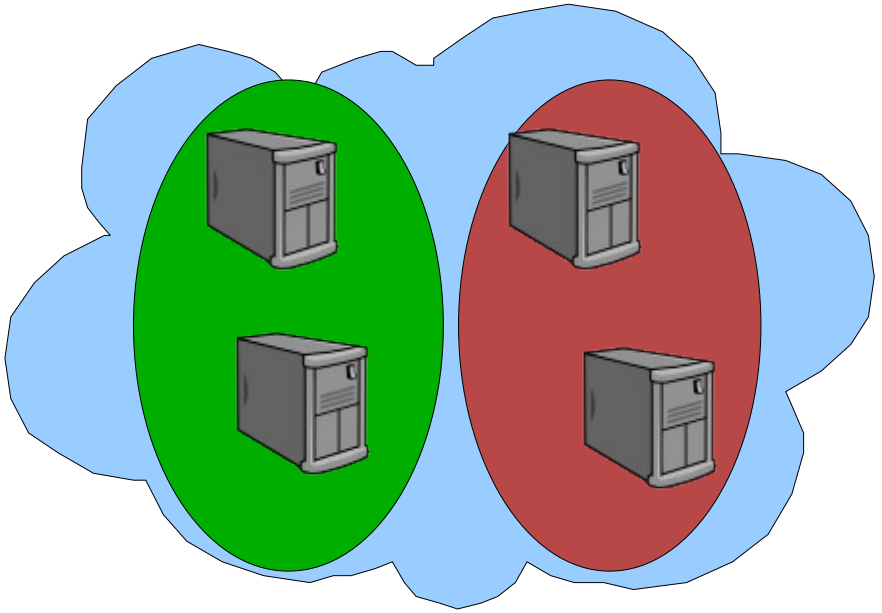


Nodegroup 0

Nodegroup 1

Data Nodes

pk

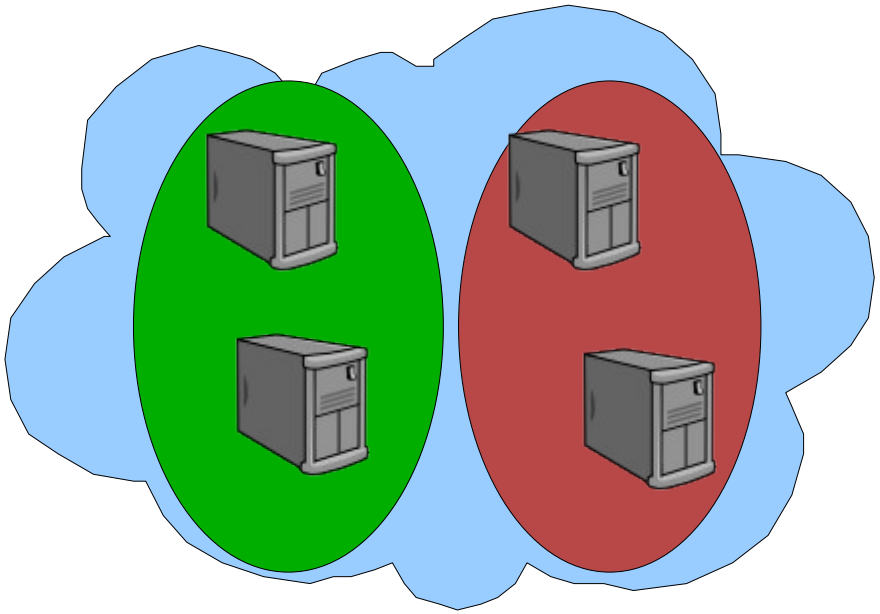


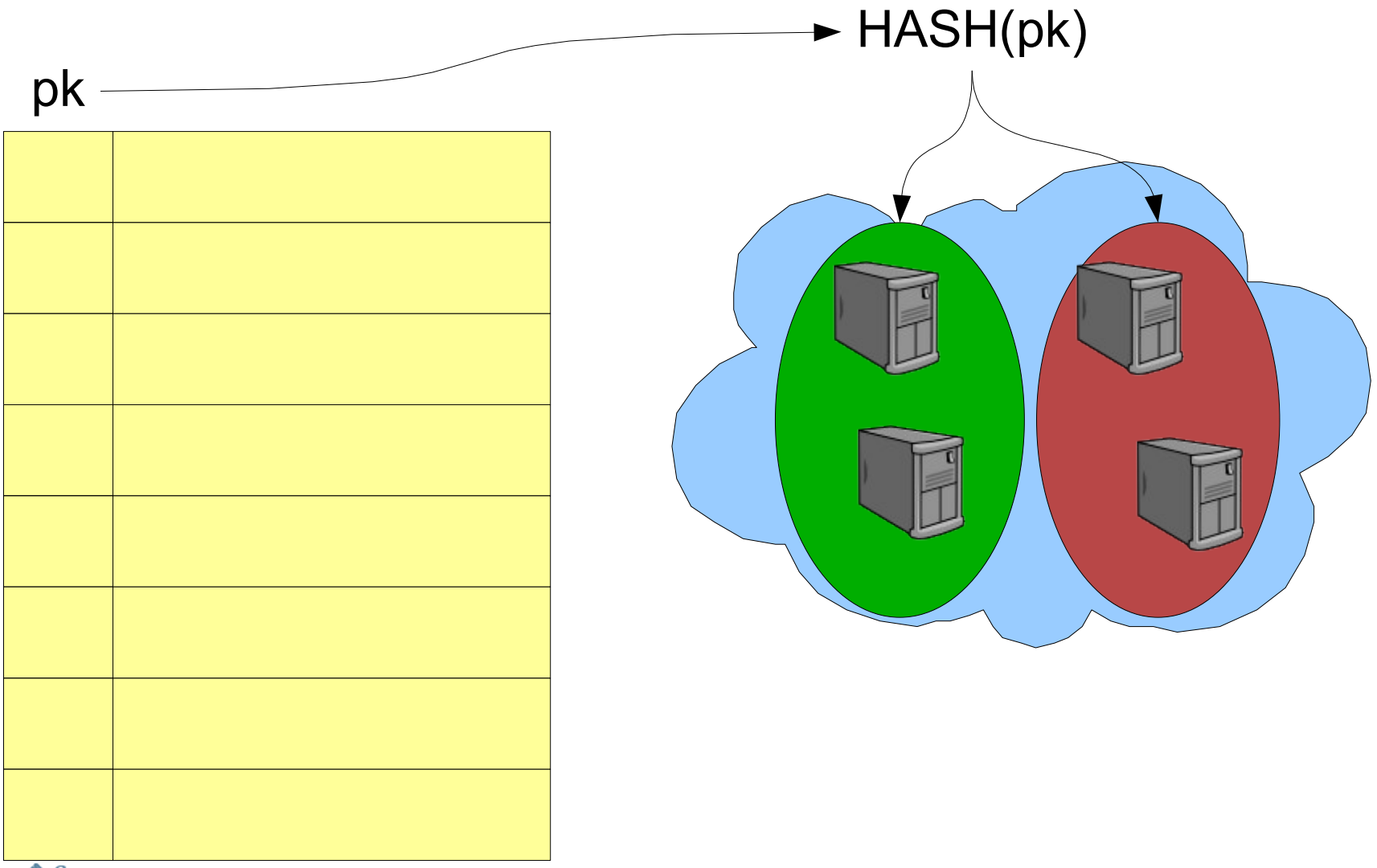
Nodegroup 0

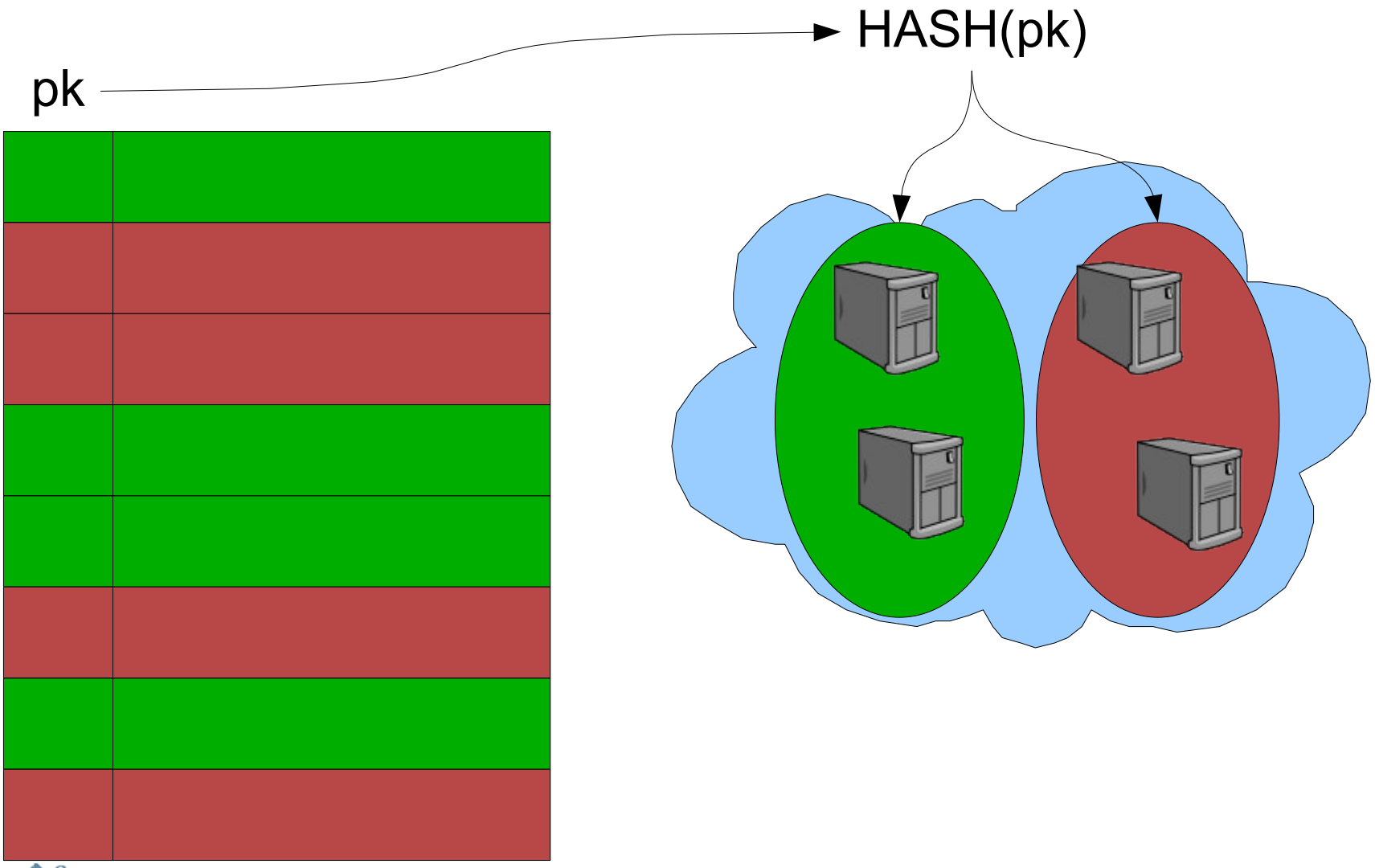
Nodegroup 1

HASH(pk)

pk

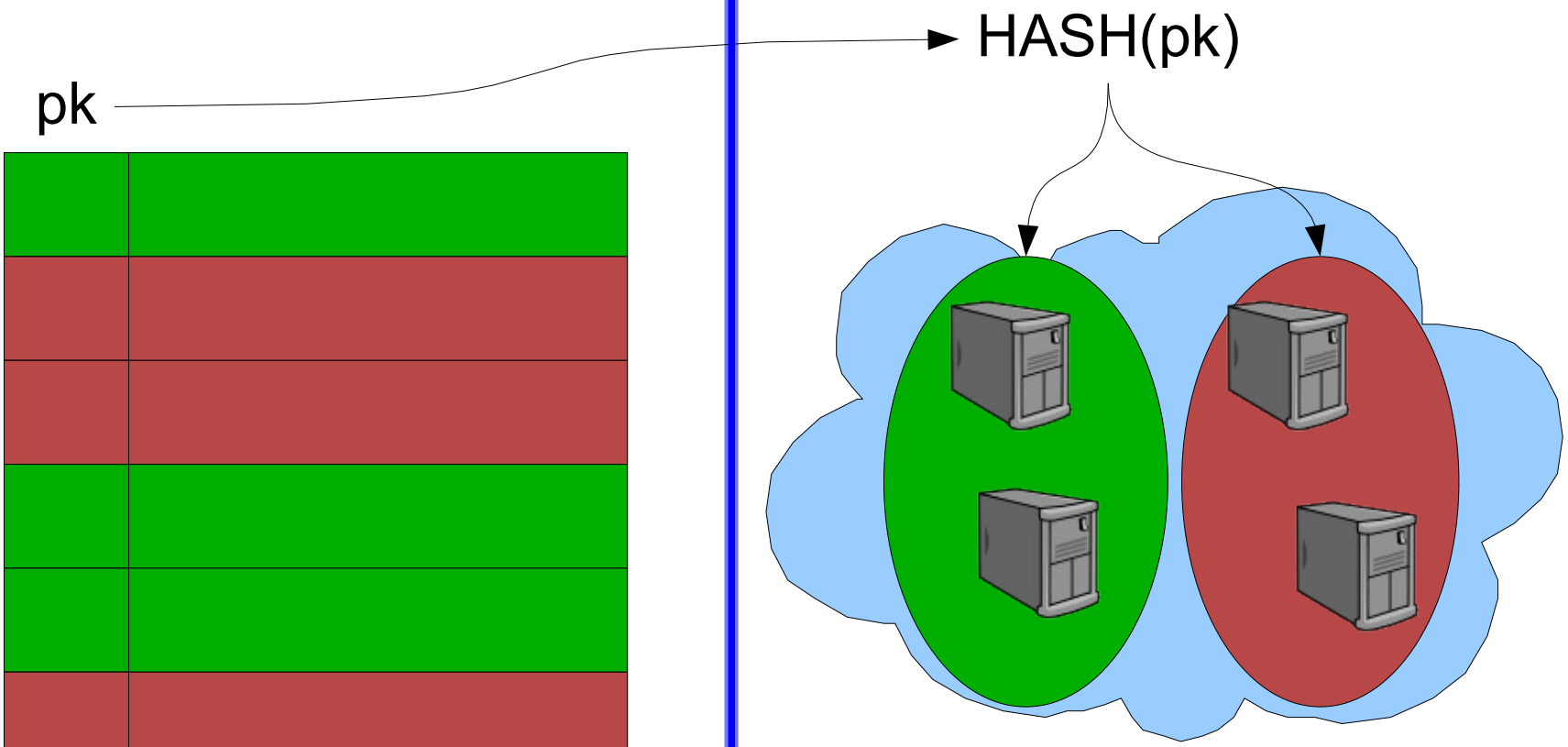


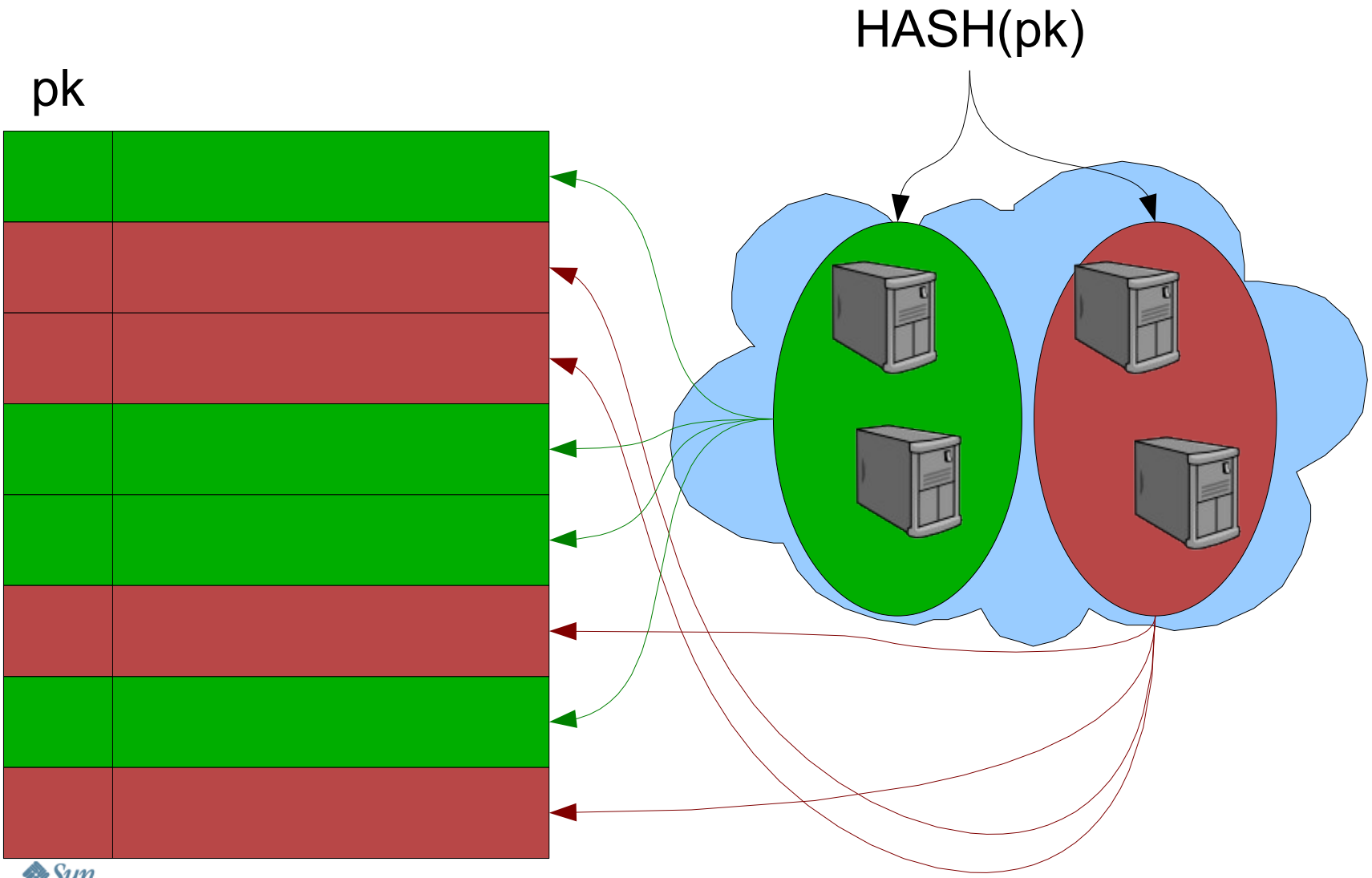

pk
HASH(pk)


pk
HASH(pk)

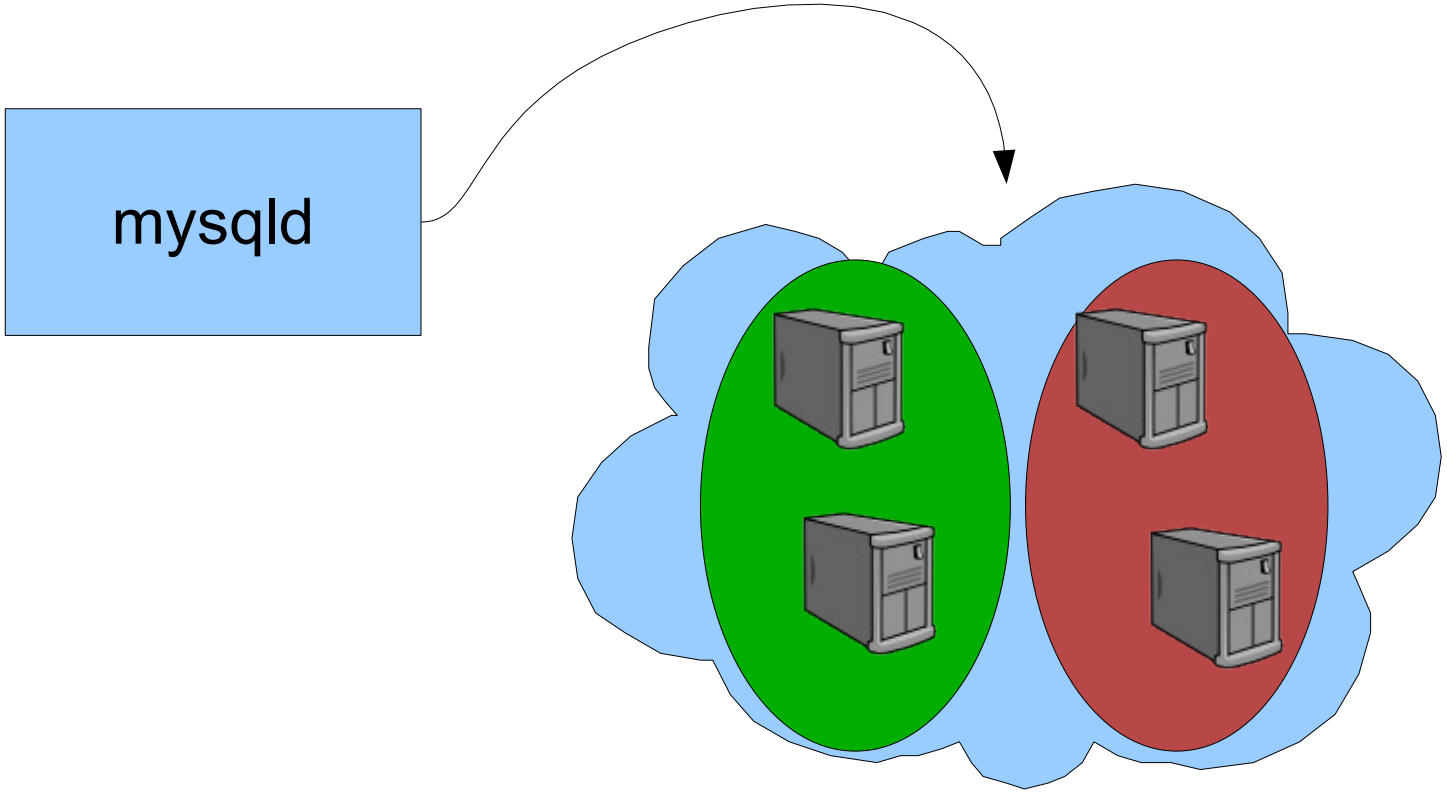
Perception

Reality




HASH(pk)
pk

MySQL Servers talk to the Data Nodes



What is MySQL Cluster?

- A High Availability
- High Performance
- In Memory (4.1 and 5.0)
  Shared Nothing
- Clustered
- Storage Engine

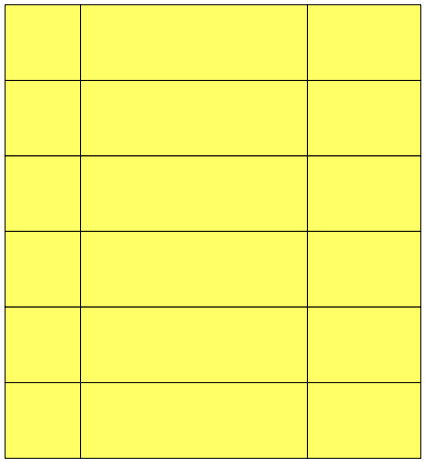
What is MySQL Cluster?

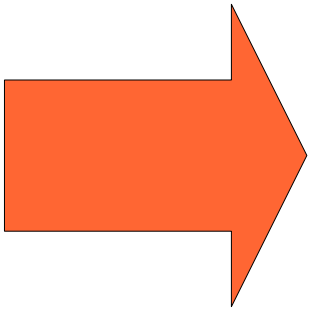
Storage Engine

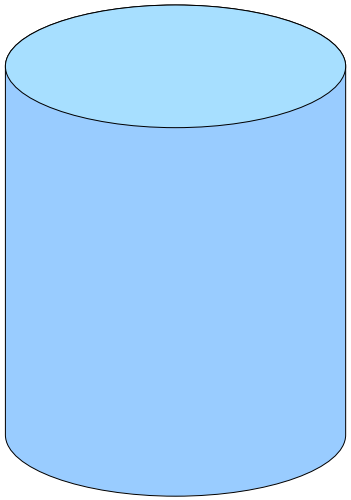
Storage Engines

- Unique feature of MySQL

No one best way to store tables

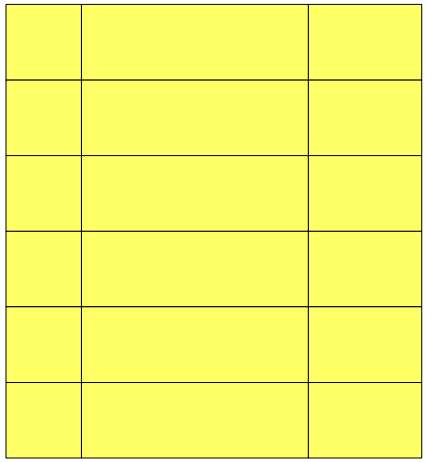


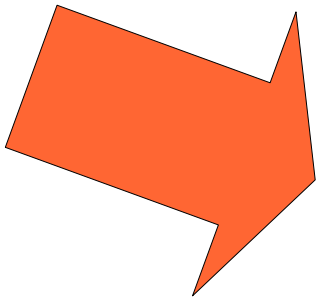


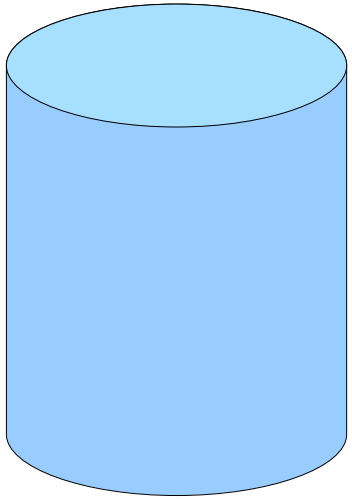


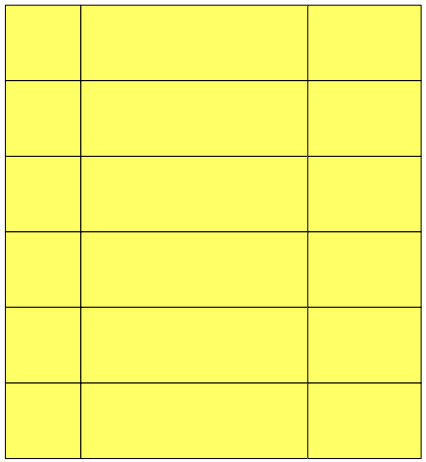
Choice of Storage Engines

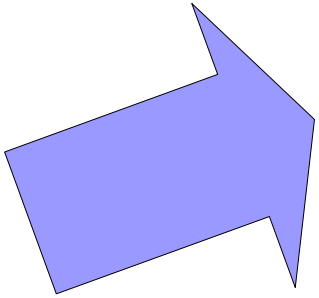
Different engine per table (if you want)



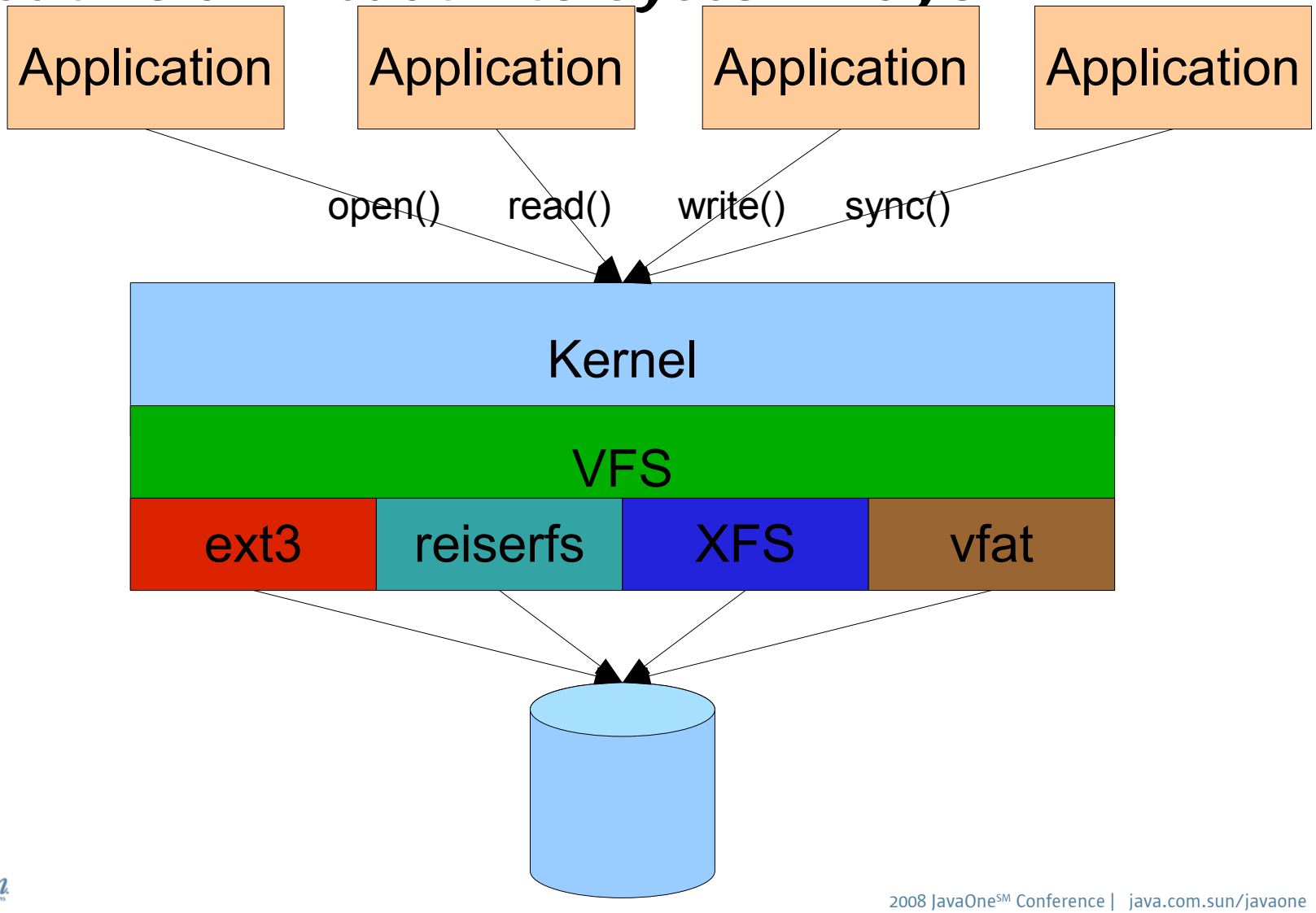




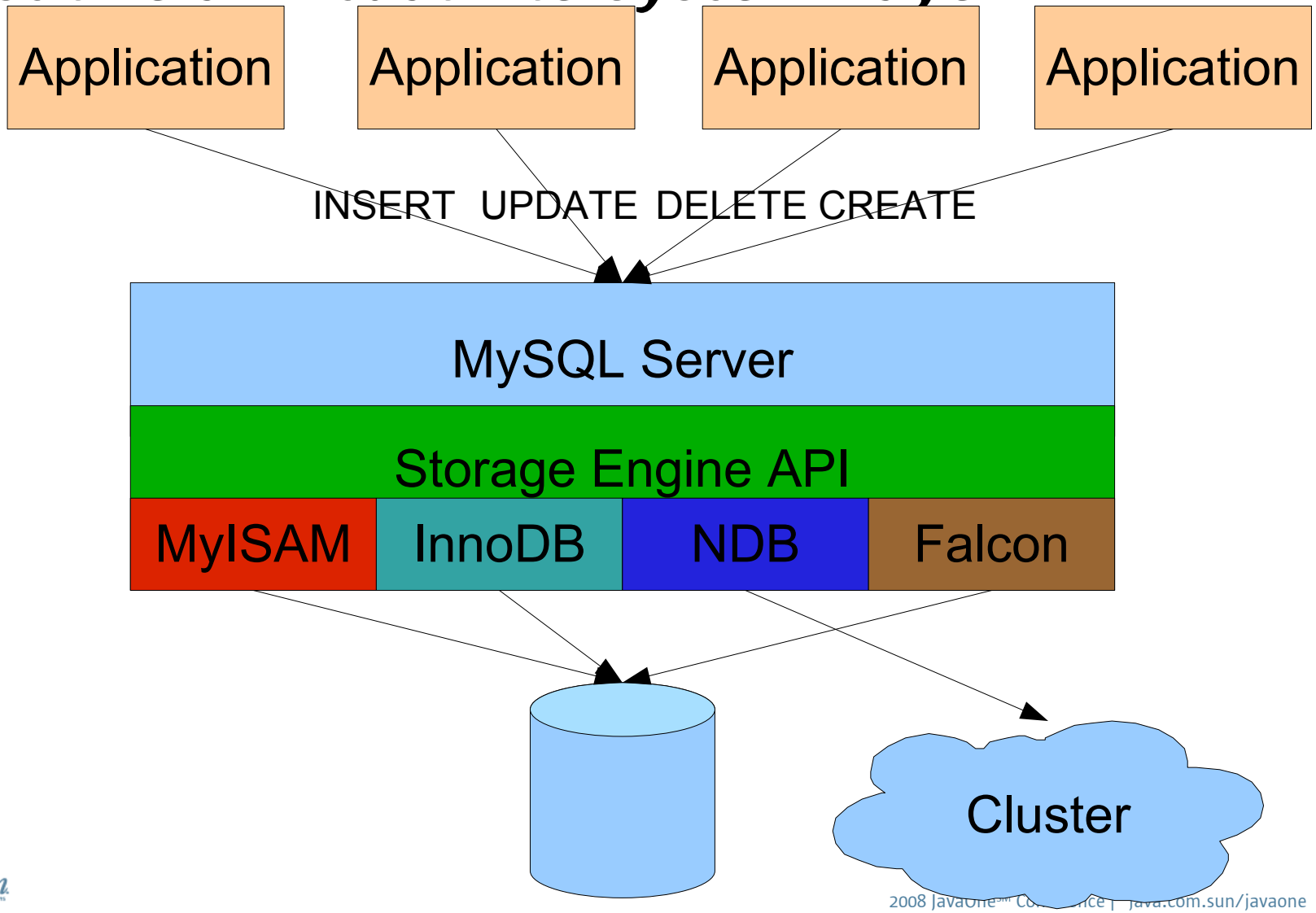




Just like a Virtual File System Layer



Just like a Virtual File System Layer



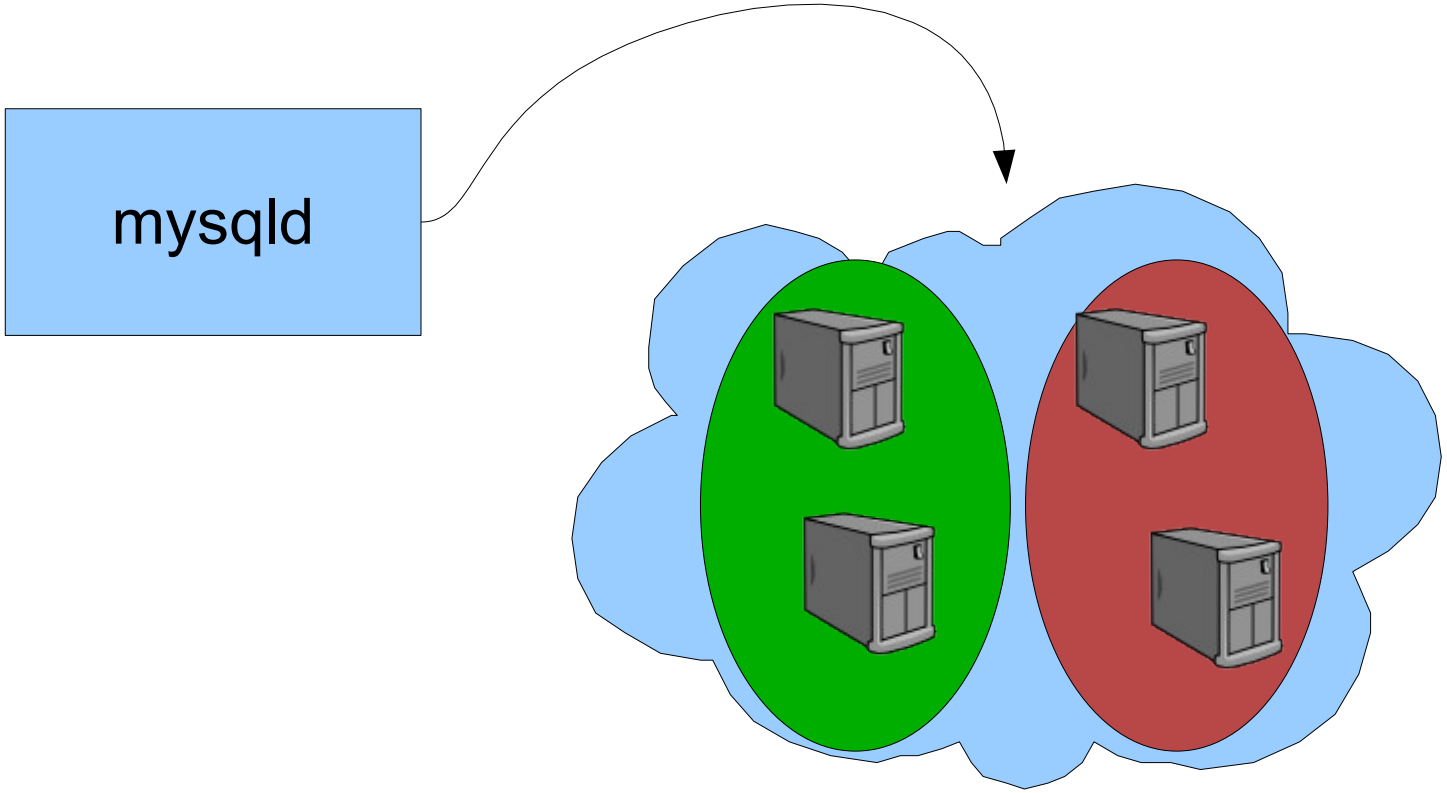
What is MySQL Cluster?

- A High Availability
- High Performance
- In Memory
- Shared Nothing
- Clustered
- Storage Engine

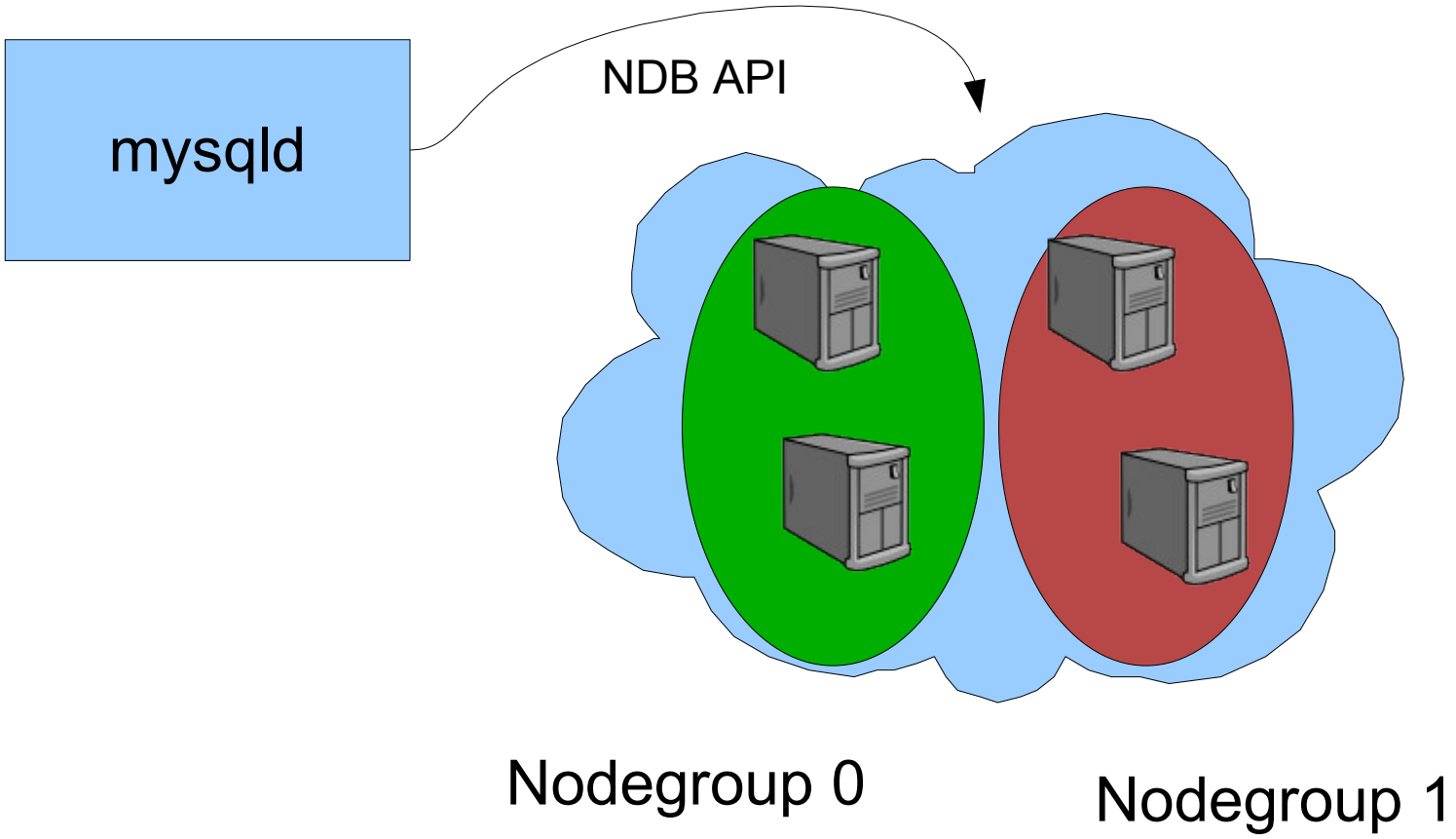
What are we going to talk about?

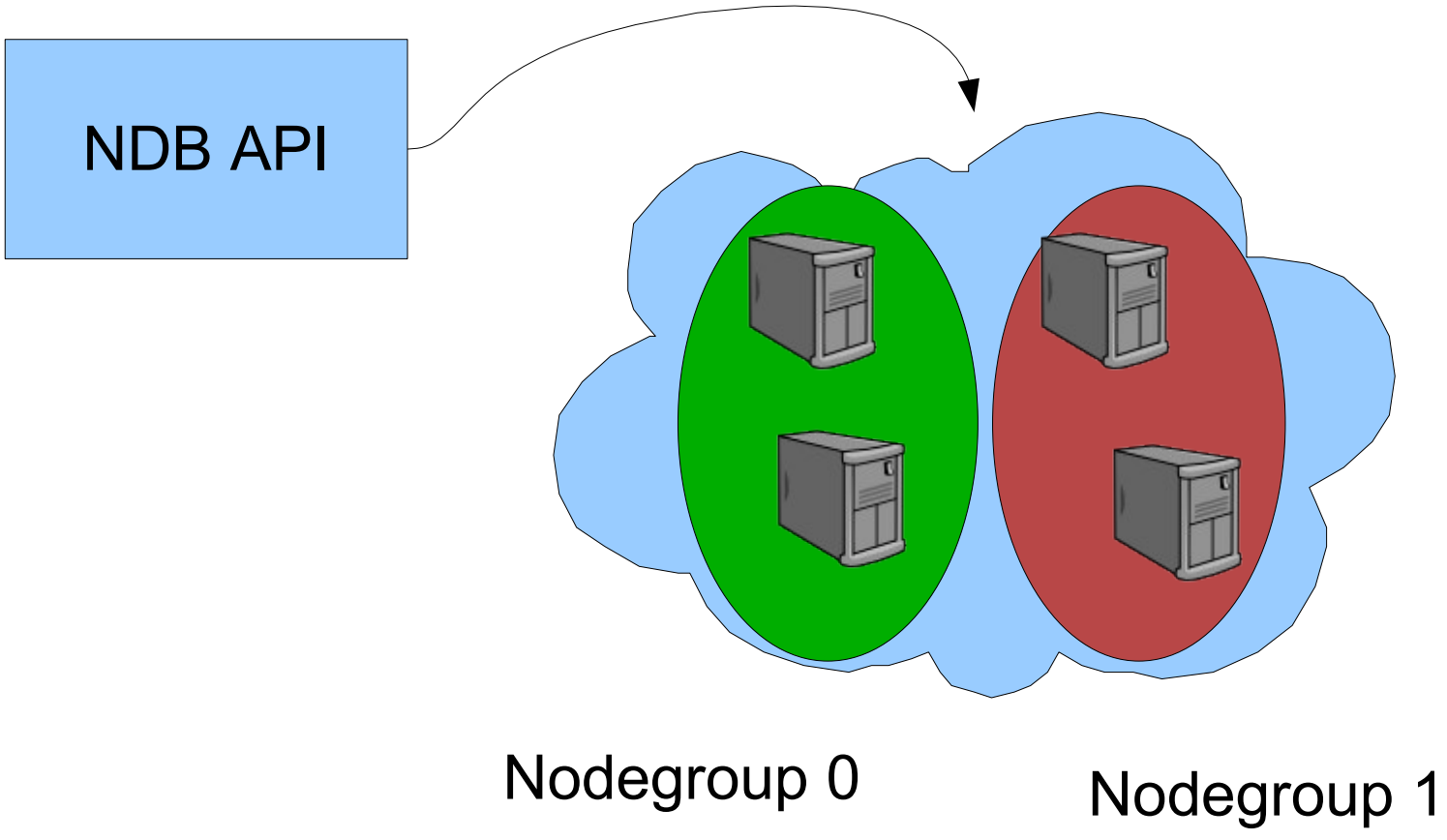
- NDB API

MySQL Servers talk to the Data Nodes



MySQL uses the NDB API to do this




NDB API
Nodegroup 0
Nodegroup 1

NDB API

- C++ direct API
- mysqld uses it to talk to Cluster
- MGM API in C for cluster management/monitoring

NDB API

So what does it actually look like?

select name from mytable where id=1

```
NdbTransaction *myTransaction= myNdb->startTransaction();
if (myTransaction == NULL) APIERROR(myNdb->getNdbError());

NdbOperation *myOperation=
        myTransaction->getNdbOperation("mytable");

if (myOperation == NULL)
   APIERROR(myTransaction->getNdbError());

myOperation->readTuple();

if (myOperation->equal("id",1)==-1)
   APIERROR(myTransaction->getNdbError());

NdbRecAttr * myAttr = myOperation->getValue("name", "monty");

if (myTransaction->execute( NdbTransaction::Commit ) == -1)
   APIERROR(myTransaction->getNdbError());

myNdb->closeTransaction(myTransaction);
```

Why?

Eeek!

So why would you want to bypass MySQL and do that instead of just using SQL?

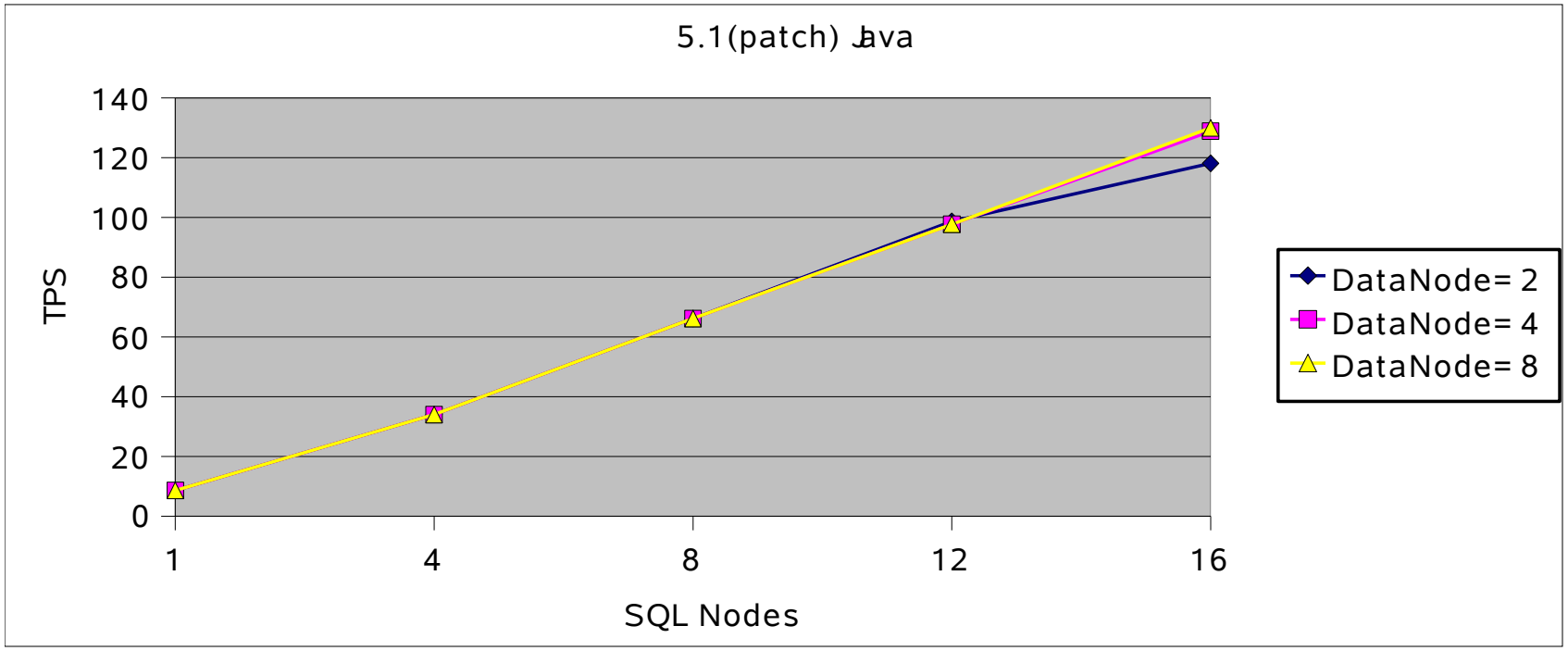
Why?

Speed

Much faster/more efficient
(about 5x over SQL)

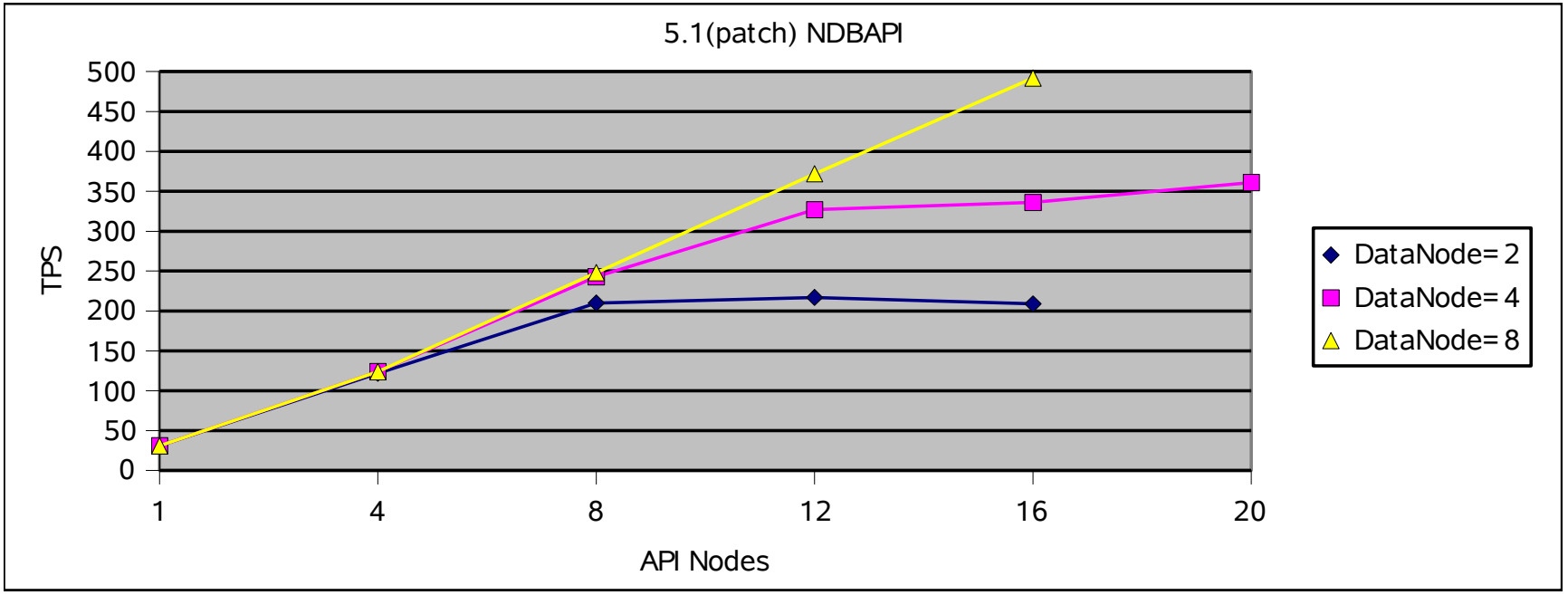
NDB API

- TPS here is actually Kilotransactions



NDB API

- TPS here is actually Kilotransactions



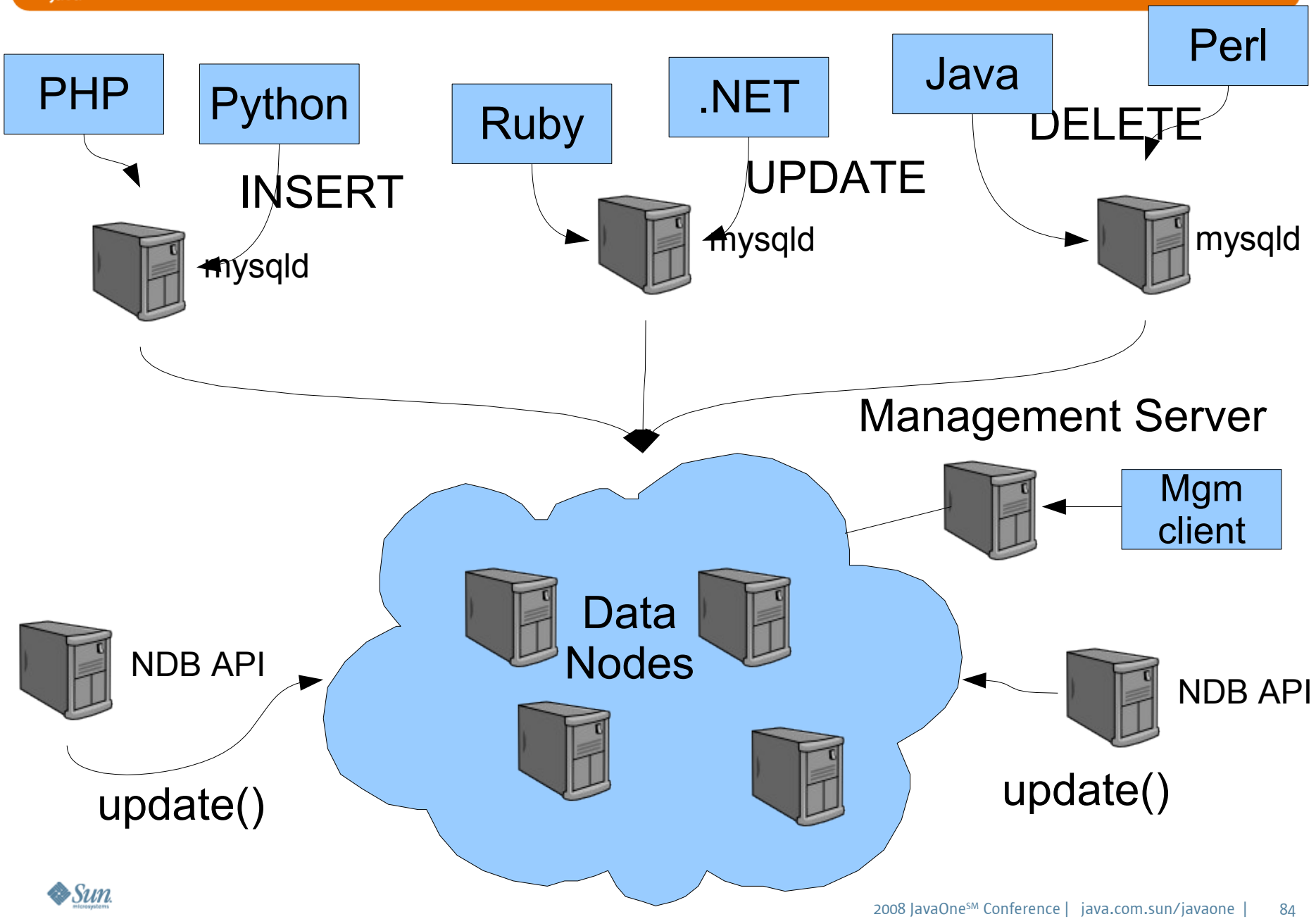
Don't like C++

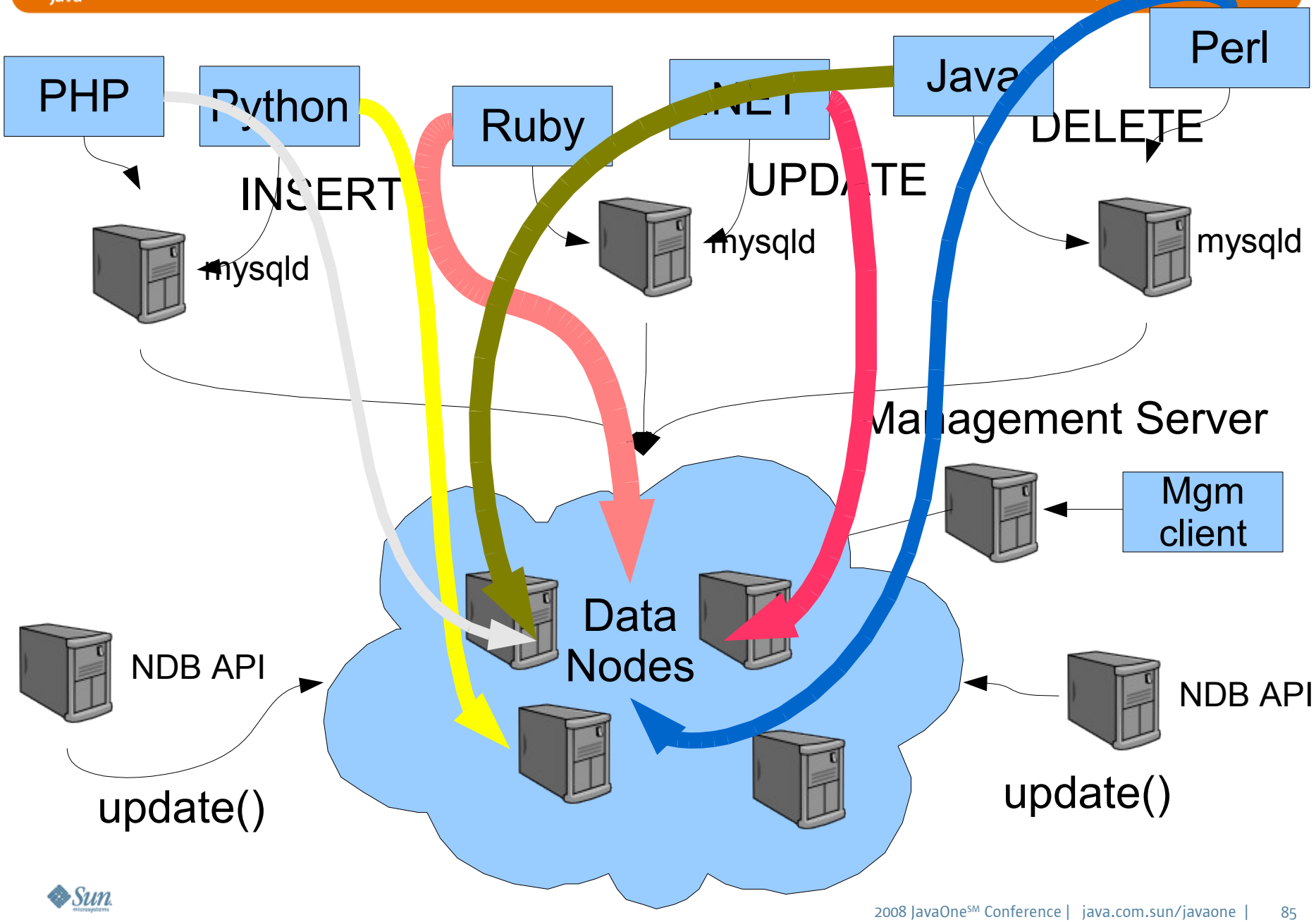
Ok. We're at the JavaOne℠ Conference. Nobody here likes C++

(Notice the shameless pandering)

NDB/Bindings

Many people don't want to write C++ code
Wrappers for NDB API and MGM API
Java, Python, Ruby, Perl, PHP, C#, Lua programming languages


PHP
Python
Ruby
.NET
Java
Perl
INSERT
UPDATE
DELETE
mysqld
mysqld
mysqld
Management Server
Mgm client
Data Nodes
NDB API
NDB API
update()
update()


PHP
Python
Ruby
.NET
Java
Perl
INSERT
UPDATE
DELETE
mysqld
mysqld
mysqld
Management Server
Mgm client
Data Nodes
NDB API
NDB API
update()
update()

Getting and using NDB/Bindings

bzr branch lp:ndb-bindings
Mailing list ndb-connectors@lists.mysql.com
Sign up at http://lists.mysql.com
IRC channel #mysql-ndb on freenode

Building NDB/Bindings

MySQL installed with libs and headers
MySQL 5.0 and 5.1 will NOT work
you need at least...
MySQL Cluster 6.2 or 6.3

You really want 6.2.16 or 6.3.14
(neither of these are released yet,
but I have patched versions of 6.2.14 and 6.3.13 available)

Implementation

C++ API (libndbclient) wrapped with SWIG

NDB/J and MGM/J

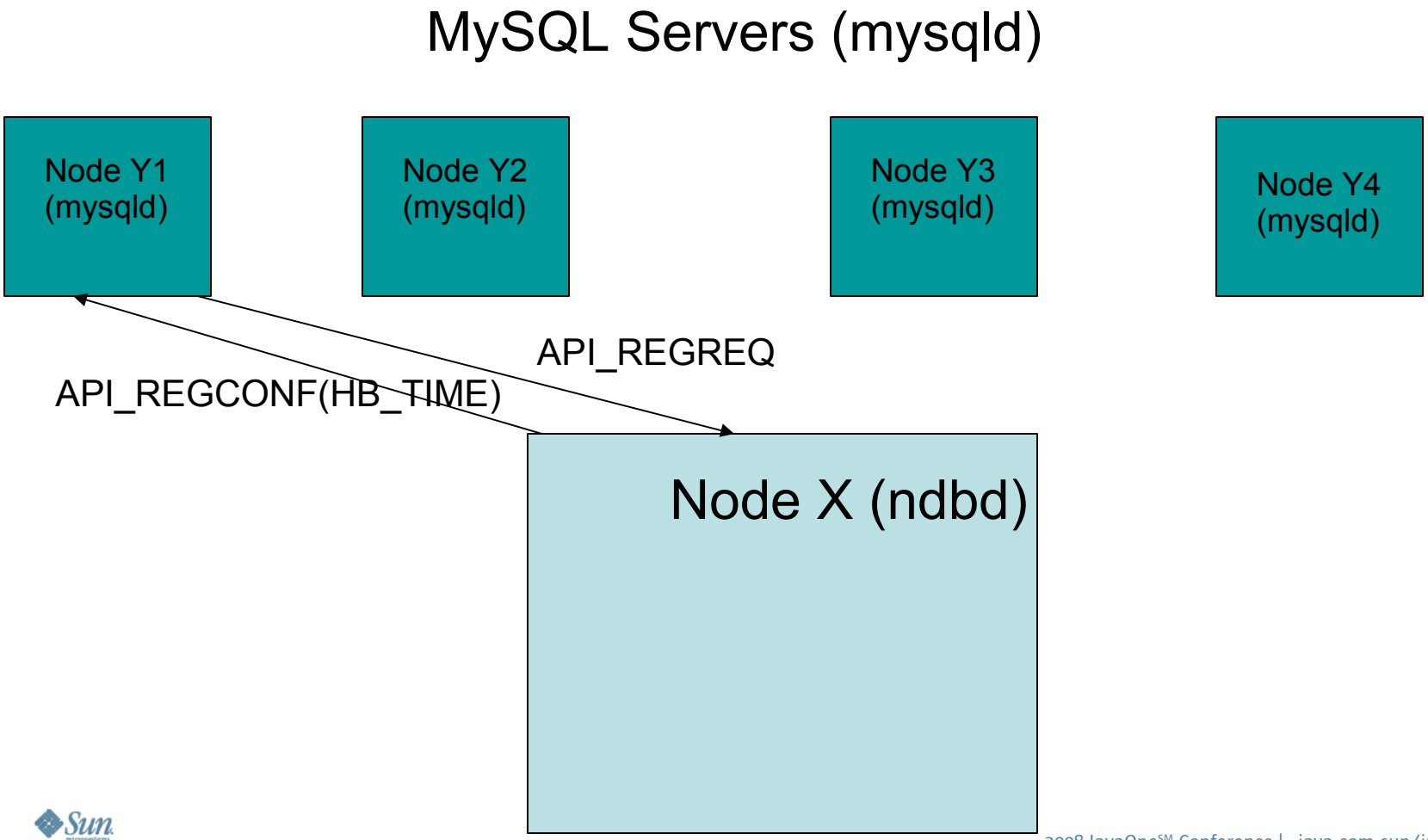
JNI™ implementation via SWIG

Must be PURE!!!!!!!

WHAT???

All things must be pure!
Why not do a pure approach?

API Nodes are Full Cluster Members



API Nodes are Full Cluster Members

Participates in Protocols, Node Failure and Recovery

Lock Management (remember sub-second failover?)

Pure Java Source Code

Honestly, we want to do it.

But it's a really big task, and the wrapped version is still stinking fast.

So it will happen – and it will use the same interfaces (yay interfaces) – but it's not going to be tomorrow.

NDB/J

Let's look at some code...

t1

id	int	PRIMARY KEY
x	int	
name	varchar(100)	

```
SELECT name FROM t1
  WHERE id=42;
```

Pseudo-NDB API

- connect to NDB
- start transaction
- set table t1
- get row where id=42
- get name

select name from t1 where id=42

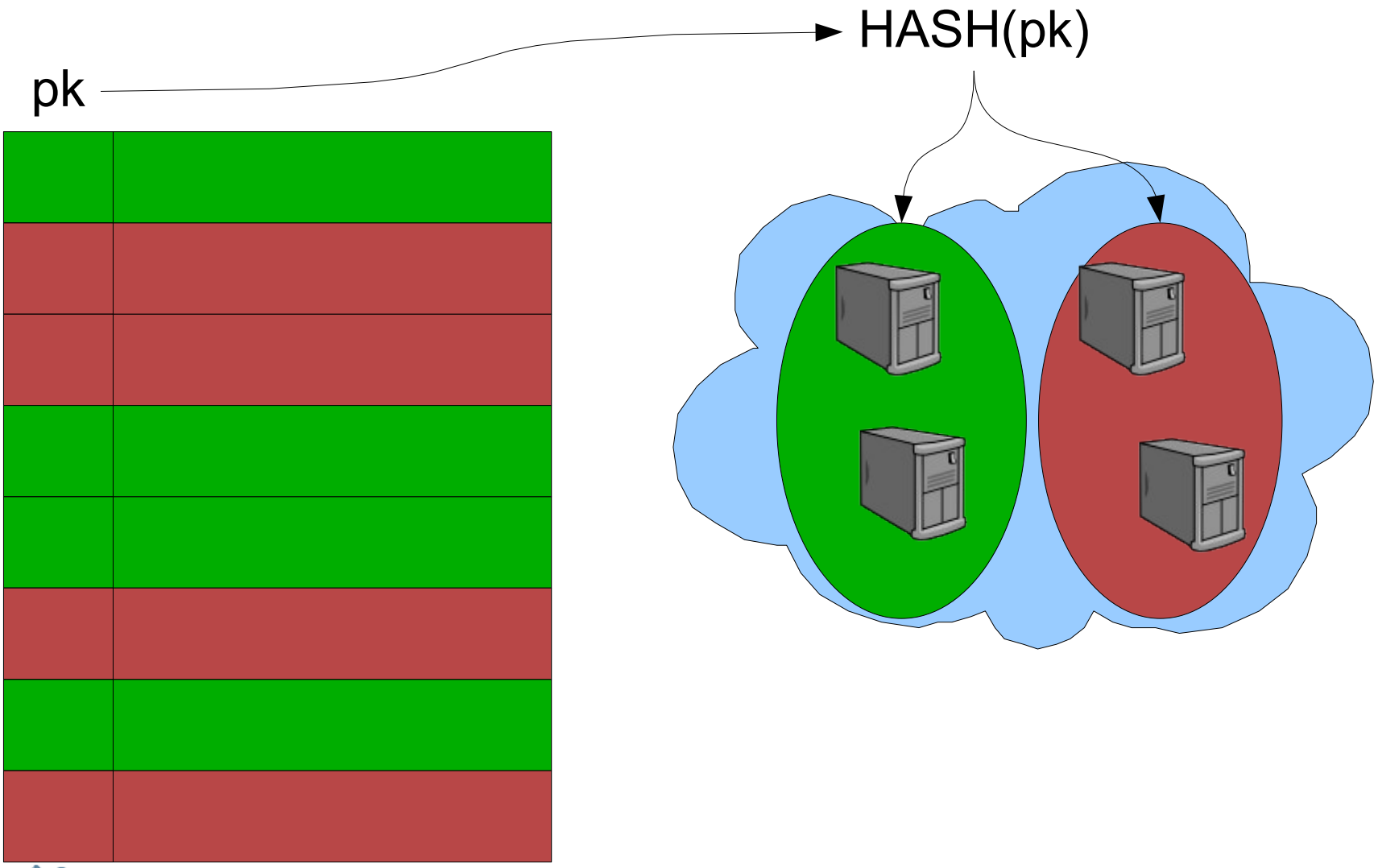
```
try {
  NdbClusterConnection connection =
    NdbFactory.createNdbClusterConnection();

  connection.connect(5, 3, true);
  connection.waitUntilReady(30, 30);

  Ndb myNdb = connection.createNdb("test", 4);

  NdbDictionary myDictionary = myNdb.getDictionary();
  NdbTable t1Table = myDictionary.getTable("t1");
```

select name from t1 where id=42

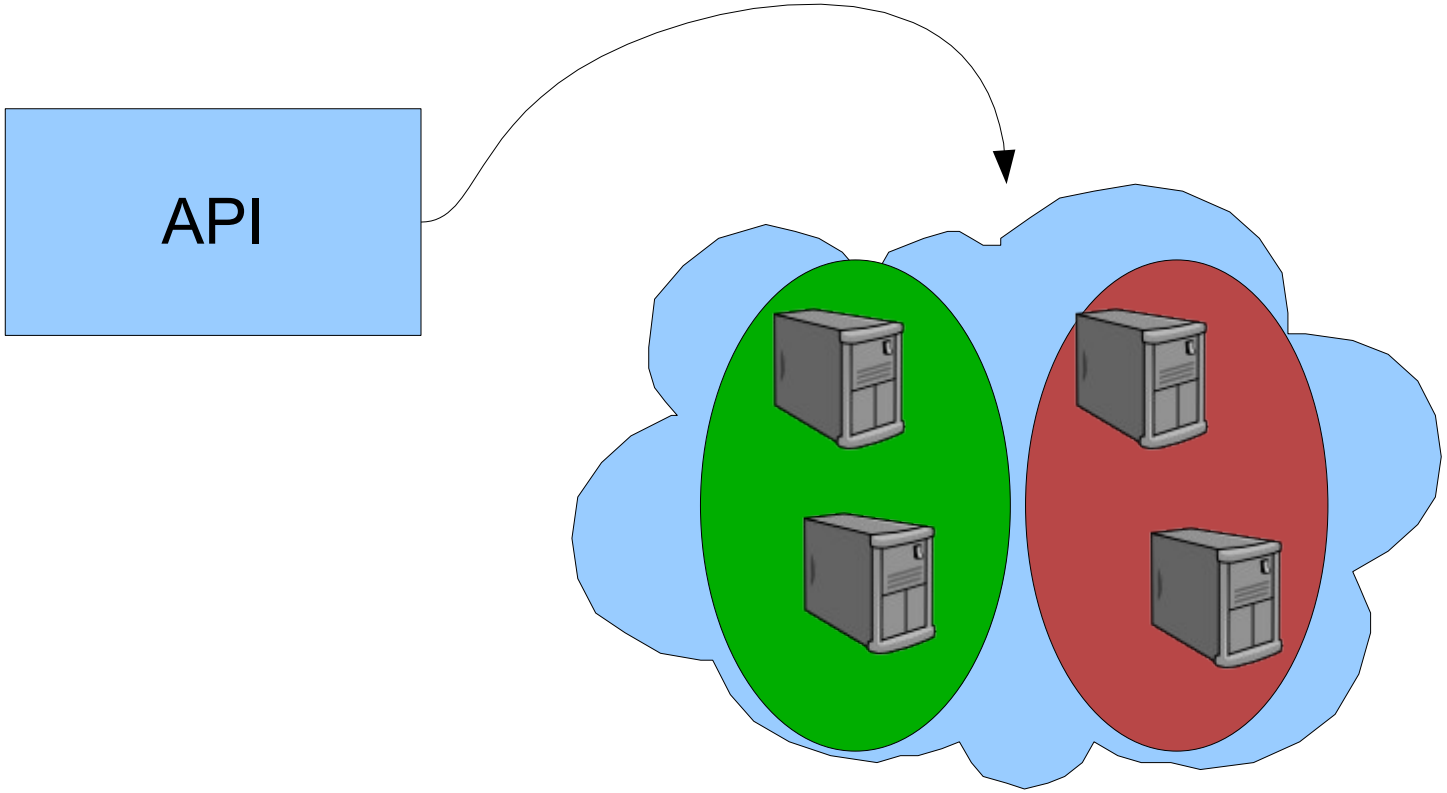
```
NdbTransaction myTransaction =
myNdb.startTransaction(t1Table, 42);

NdbOperation myOperation =
 myTransaction.getSelectOperation(t1Table,
        LockMode.LM_Exclusive);
myOperation.equalInt("id",42);
myOperation.getValue("name");
NdbResultSet myResults =
        myOperation.resultData();
```

select name from t1 where id=42

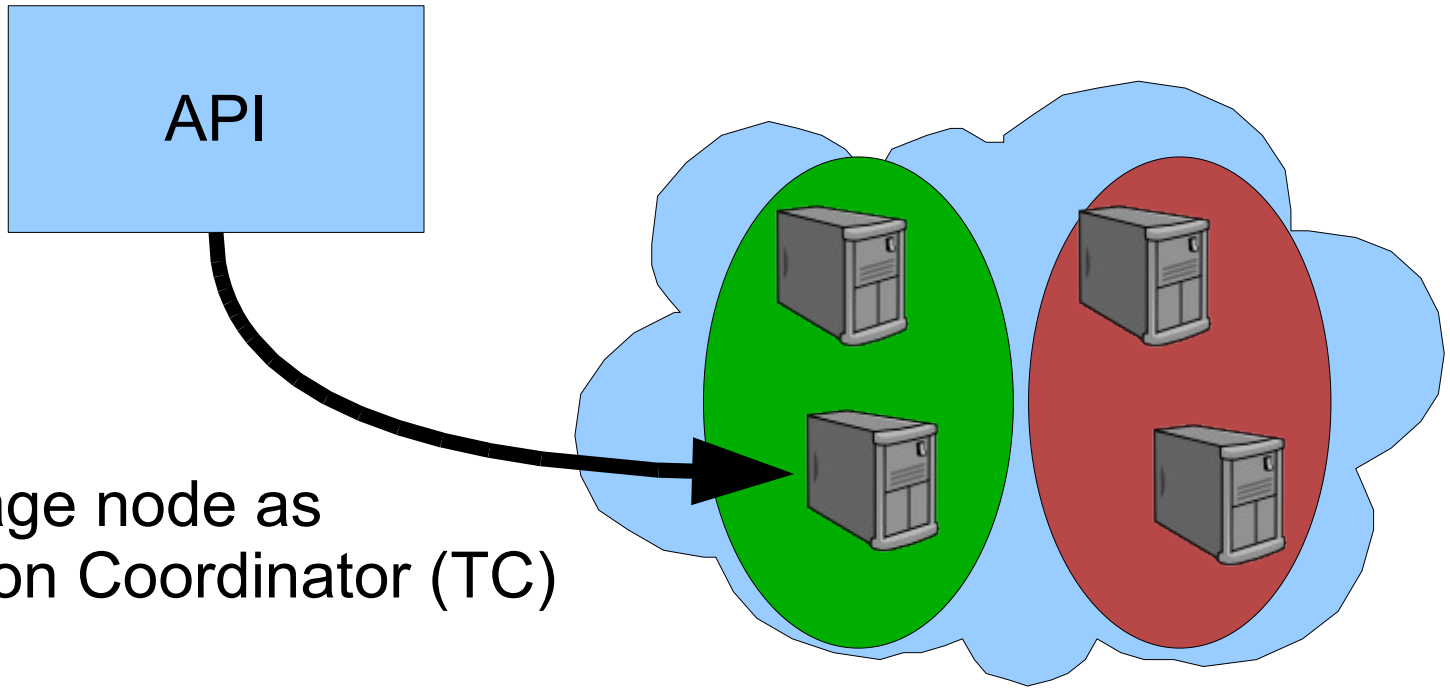
```
myTransaction.executeCommit(
          AbortOption.AbortOnError);

  if (myResults.next()) {
   System.out.println(myResults.getString("name"));
  }

} catch (NdbApiException e) {
    System.out.println(e.getMessage());
}
```


HASH(pk)
pk

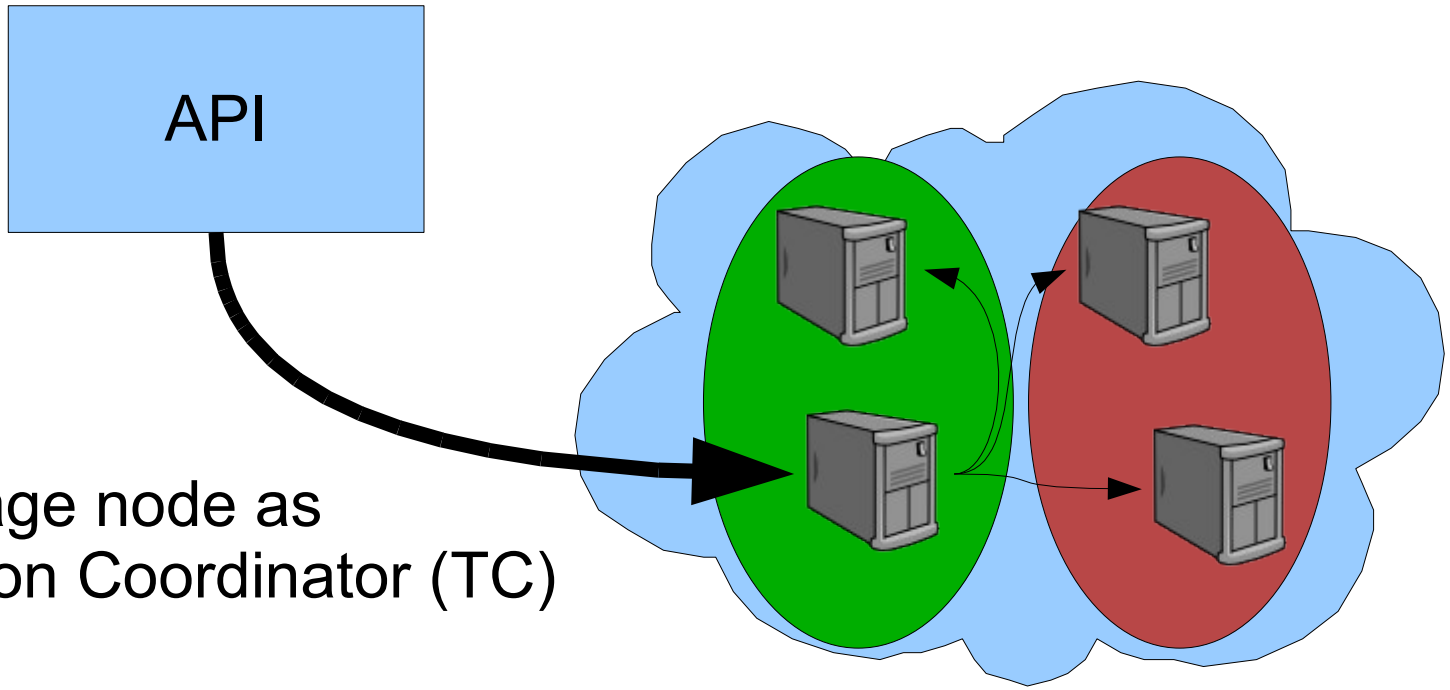
MySQL Servers talk to the Data Nodes



One used as a Transaction Coordinator

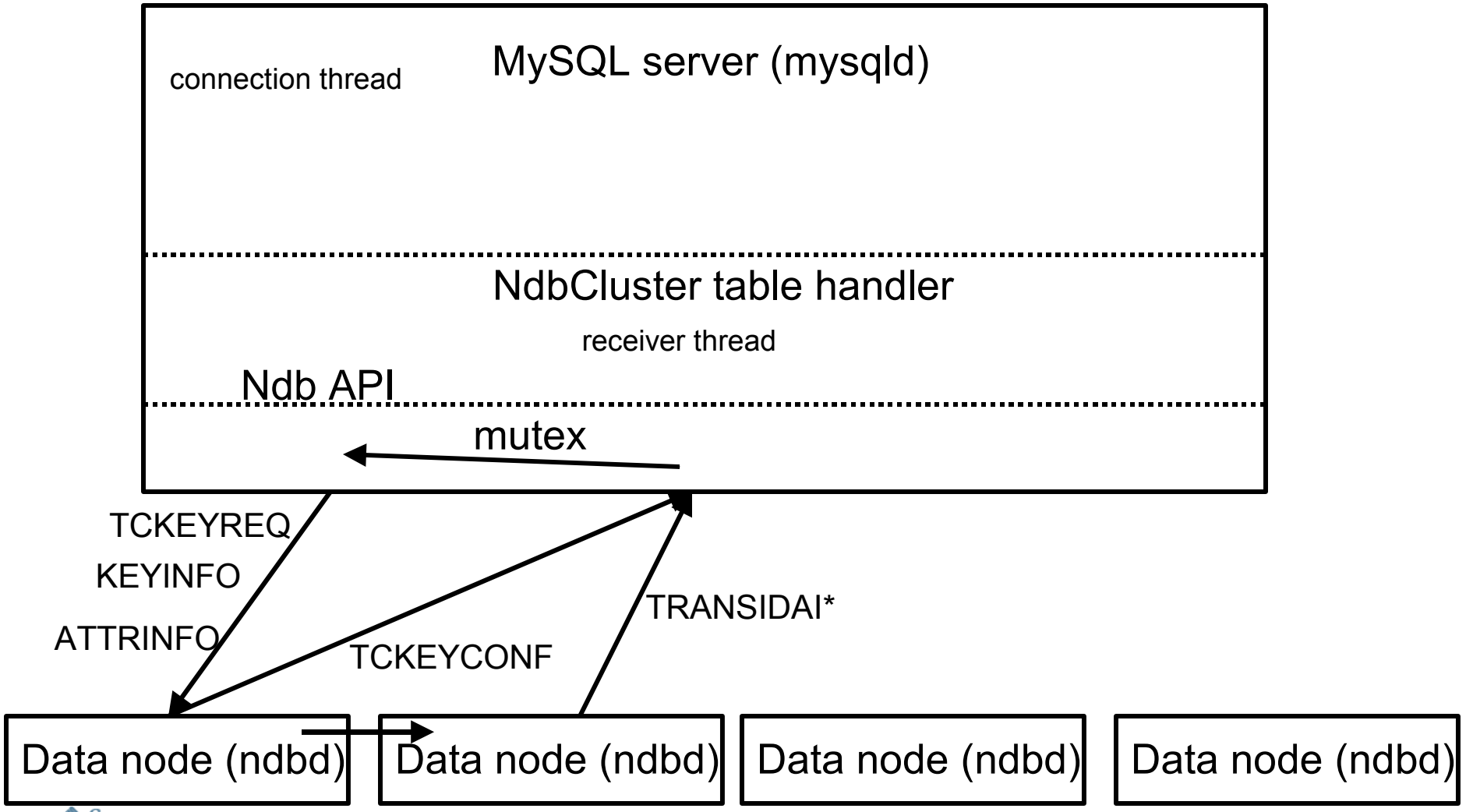


One used as a Transaction Coordinator

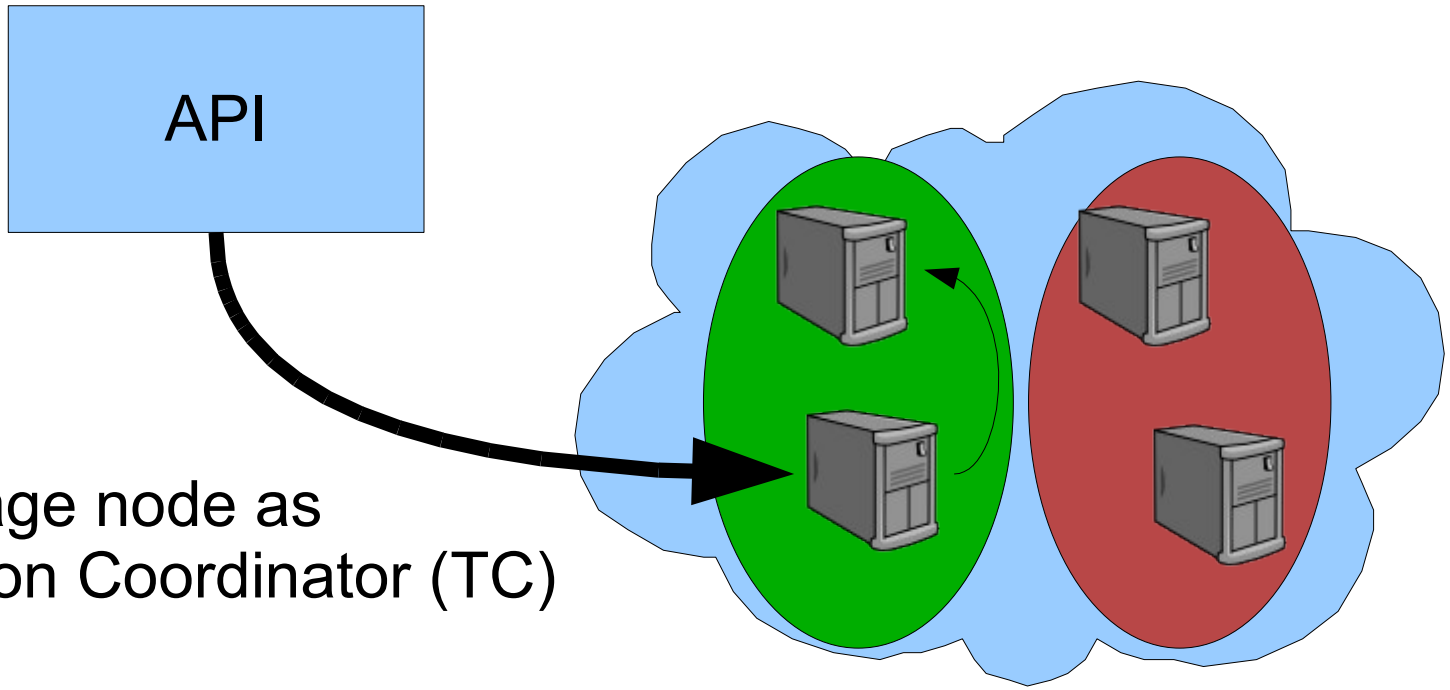


One storage node as
Transaction Coordinator (TC)

Primary Key Operation (read*,insert,update, delete), Signalling

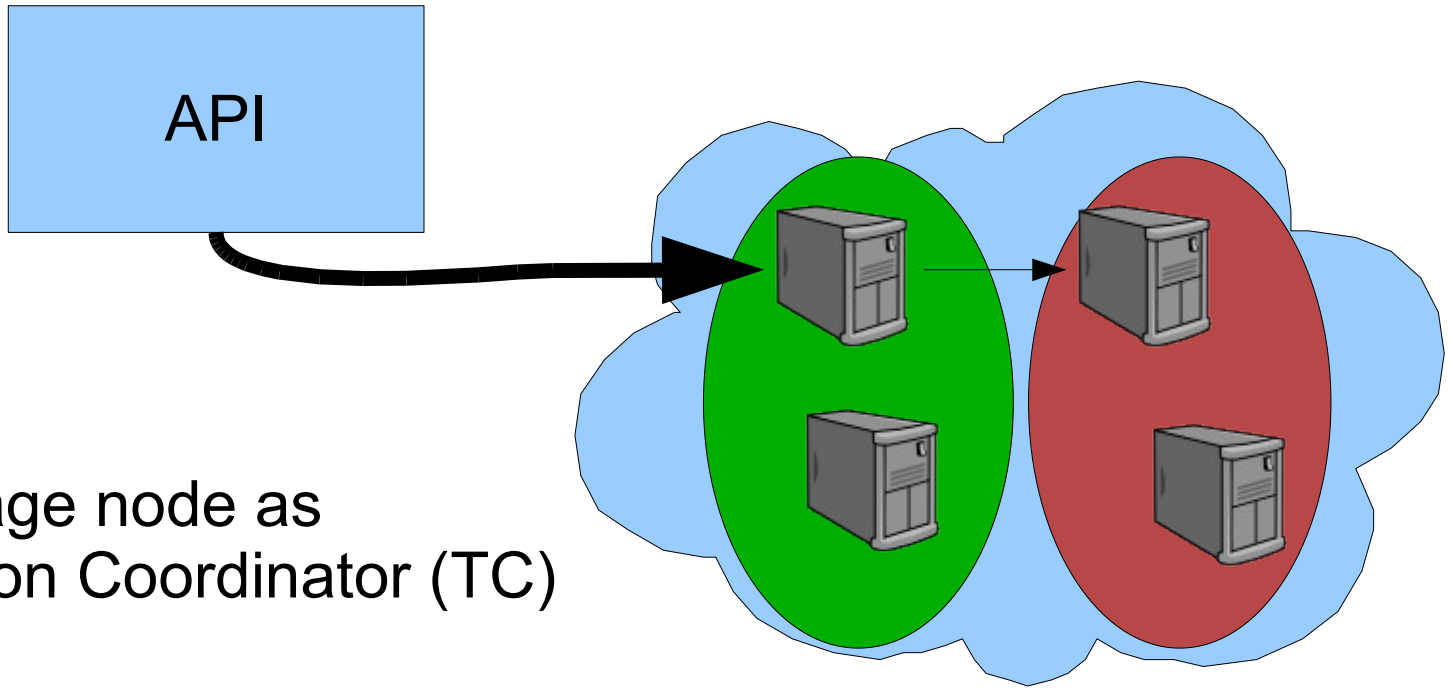


TC Selection



One storage node as
Transaction Coordinator (TC)

TC Selection

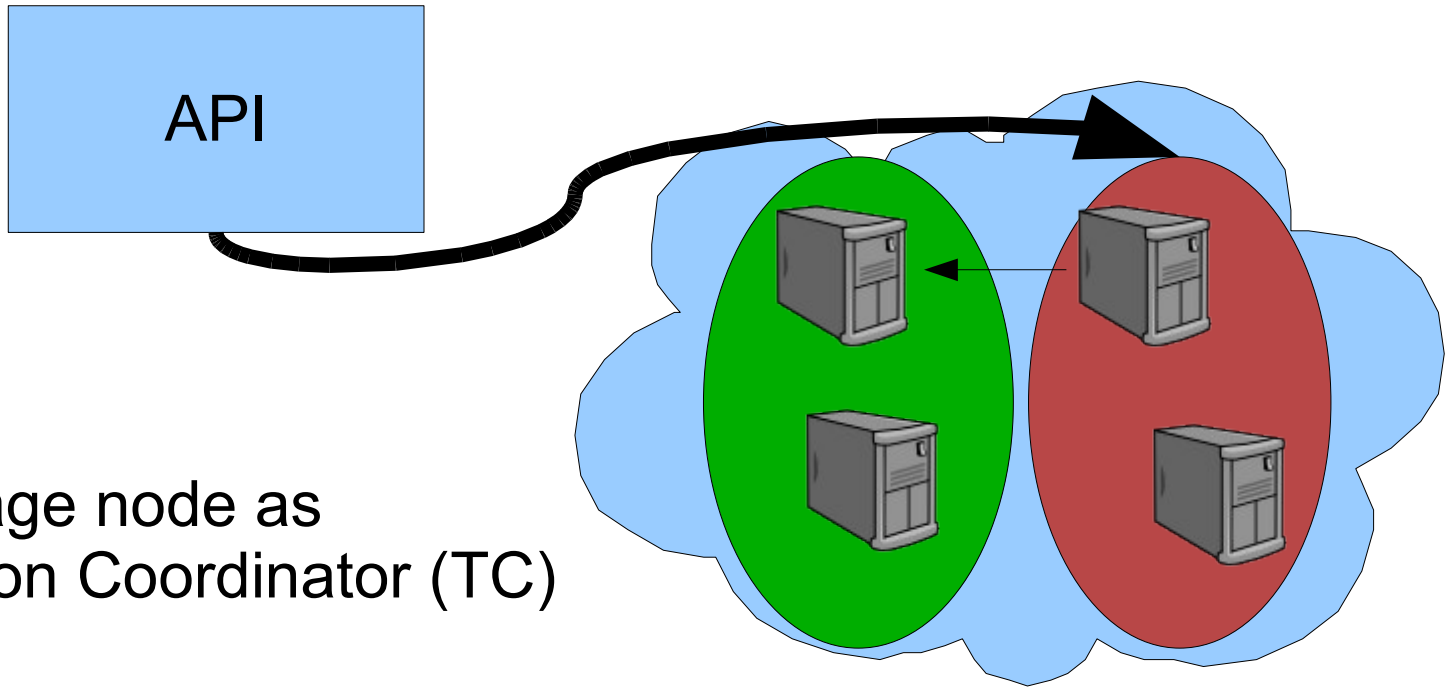


One storage node as
Transaction Coordinator (TC)

Nodegroup 0

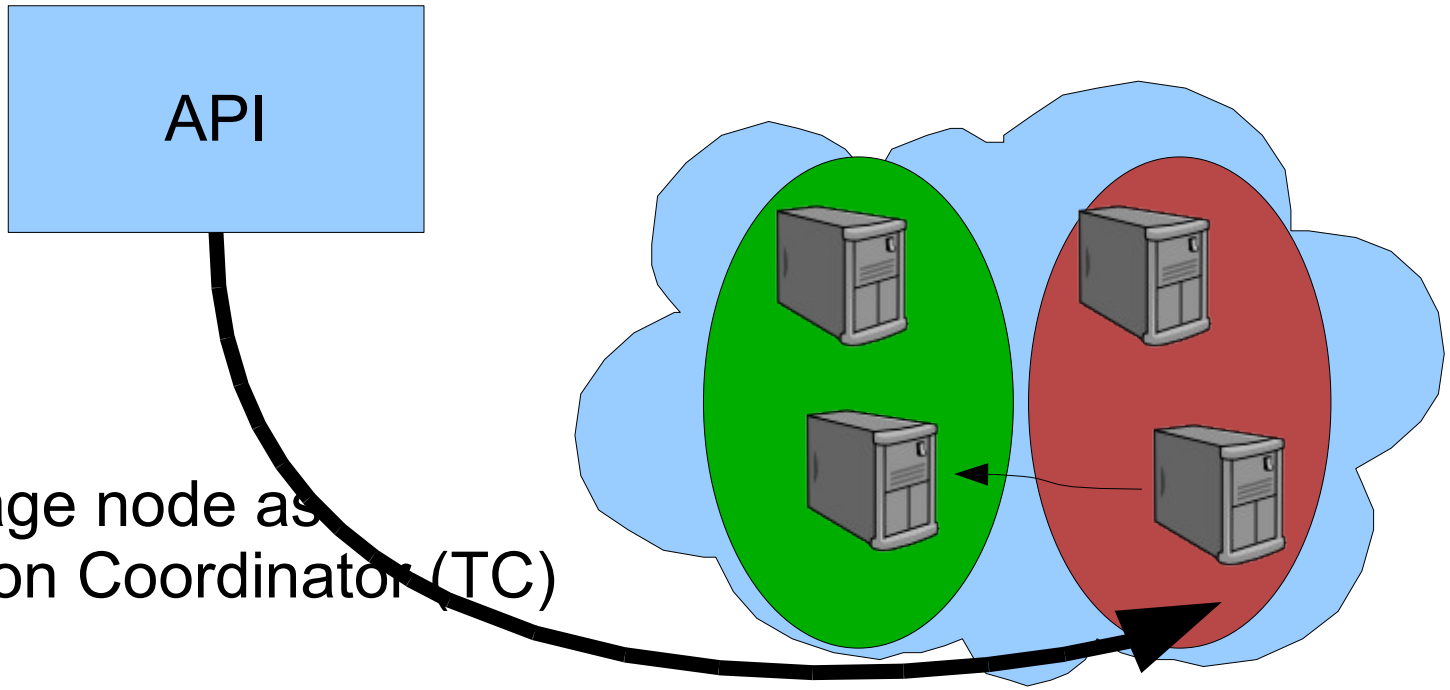
Nodegroup 1

TC Selection

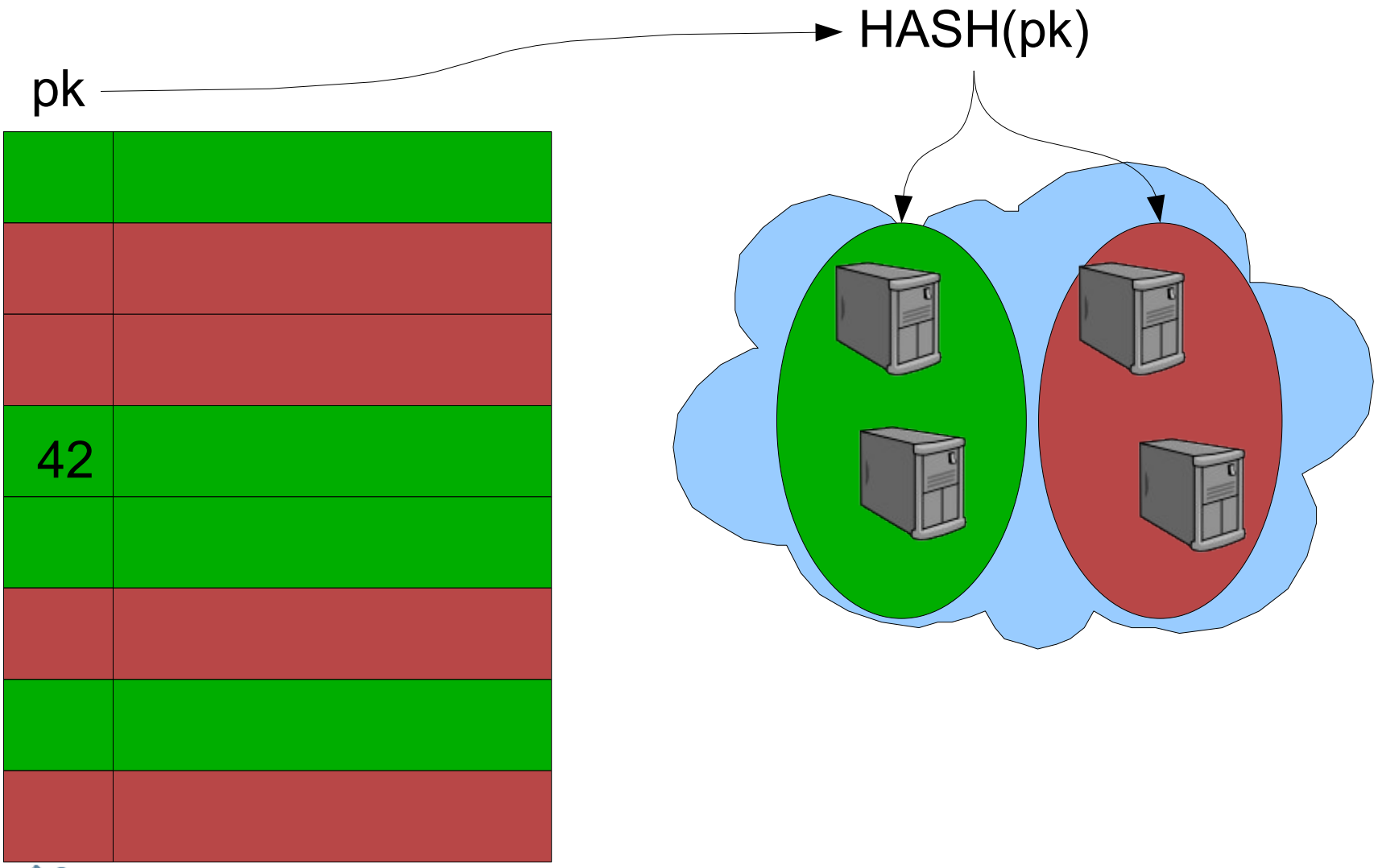


One storage node as
Transaction Coordinator (TC)

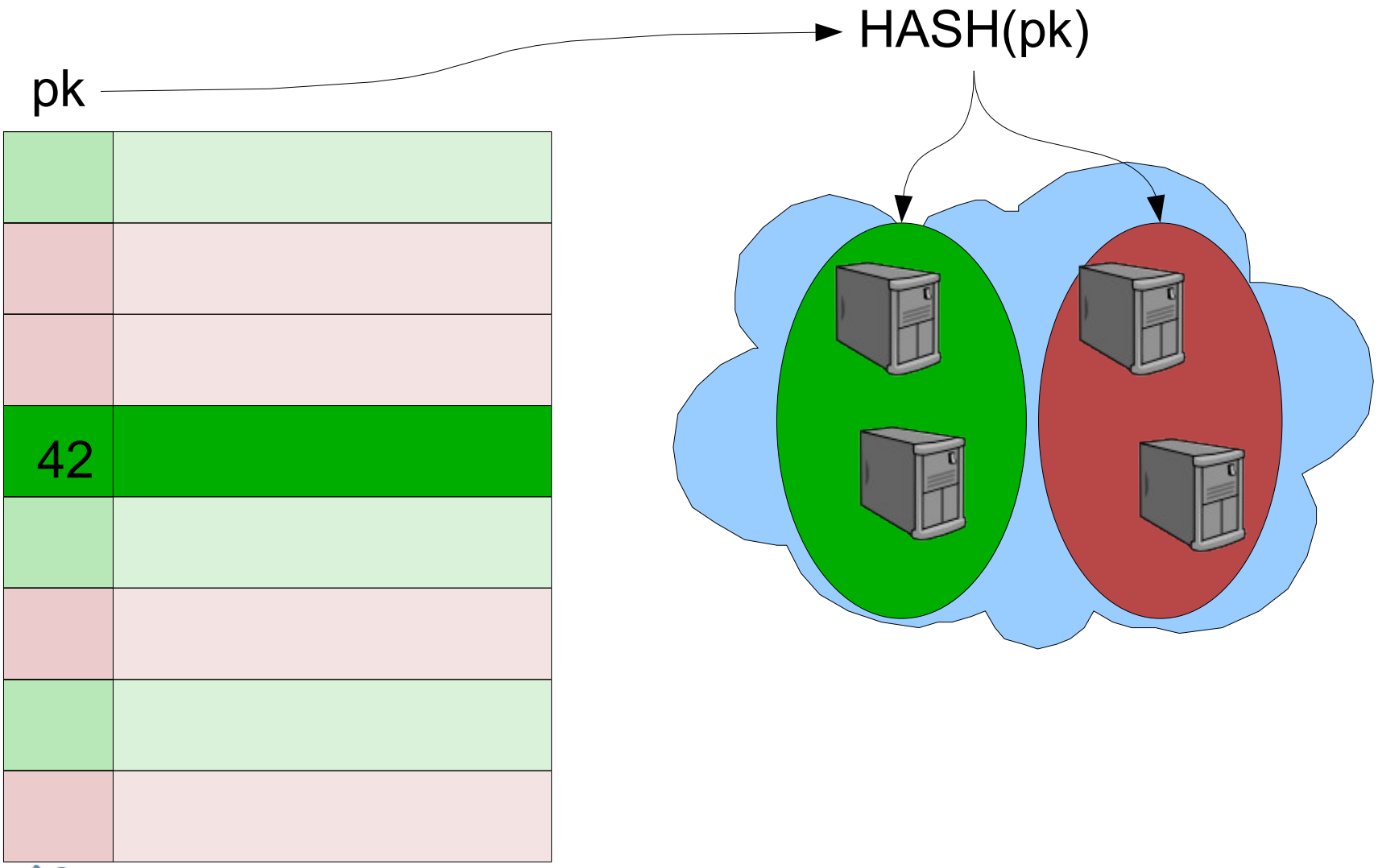
TC Selection



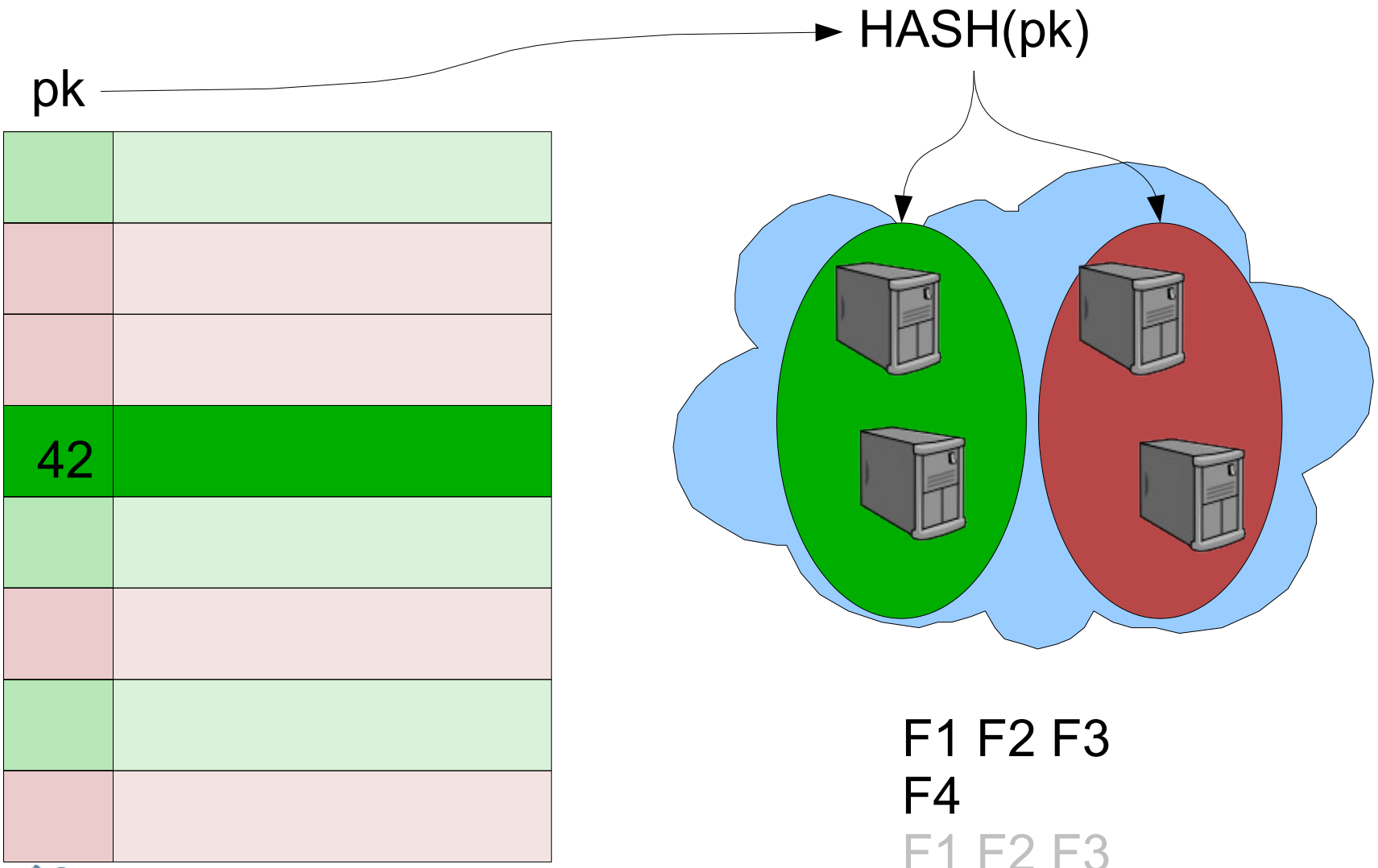
SELECT name FROM t1 WHERE id=42;



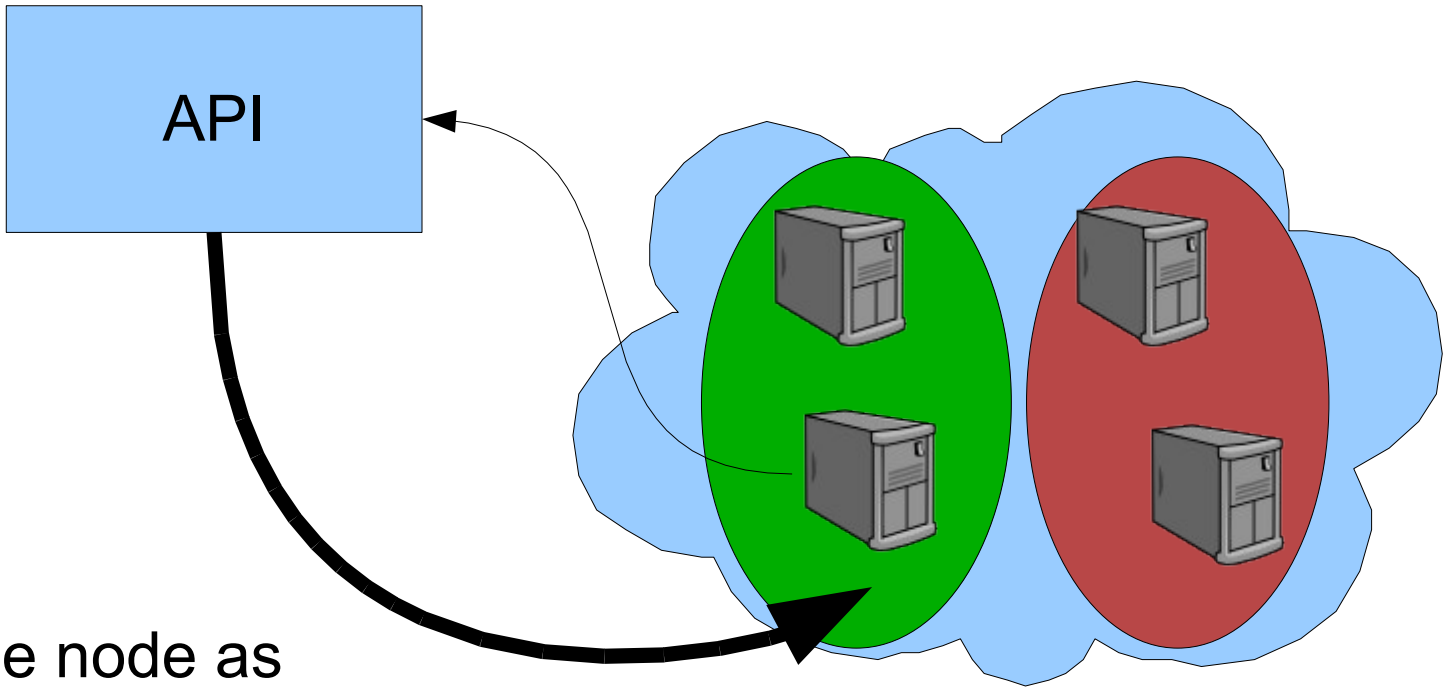
SELECT name FROM t1 WHERE id=42;



SELECT name FROM t1 WHERE id=42;



TC Selection



One storage node as Transaction Coordinator (TC)
Same node as primary replica for the row

Nodegroup 0

Nodegroup 1

TC close to row
=
faster

faster
=
good

select name from t1 where id=42

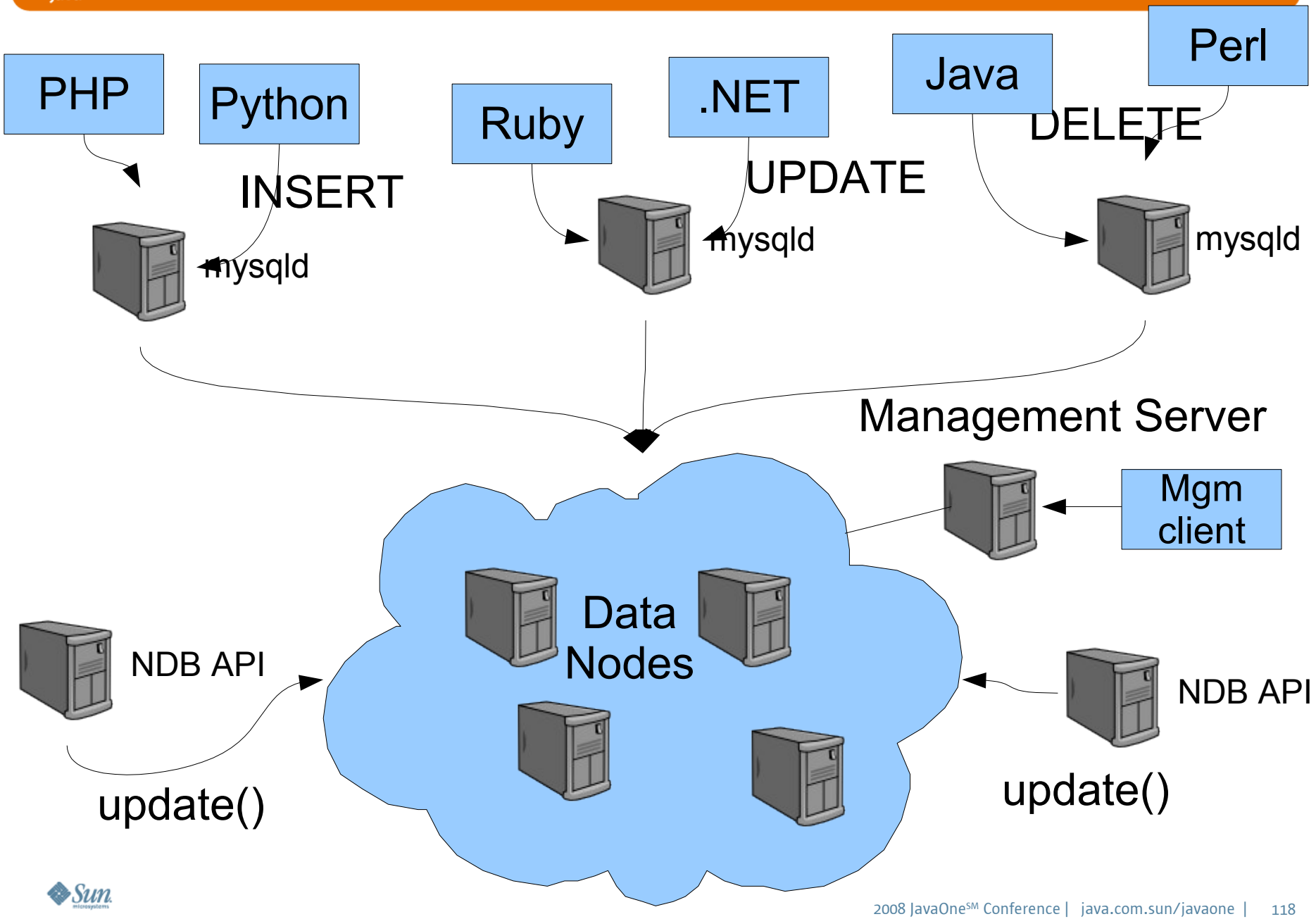
```
NdbTransaction myTransaction =
myNdb.startTransaction(t1Table, 42);

NdbOperation myOperation =
 myTransaction.getSelectOperation(t1Table,
        LockMode.LM_Exclusive);
myOperation.equalInt("id",42);
myOperation.getValue("name");
NdbResultSet myResults =
        myOperation.resultData();
```

select name from t1 where id=42

```
myTransaction.executeCommit(
          AbortOption.AbortOnError);

  if (myResults.next()) {
   System.out.println(myResults.getString("name"));
  }

} catch (NdbApiException e) {
    System.out.println(e.getMessage());
}
```

JDBC™ API

What about JDBC?


PHP
Python
Ruby
.NET
Java
Perl
INSERT
UPDATE
DELETE
mysqld
mysqld
mysqld
Management Server
Mgm client
Data Nodes
NDB API
NDB API
update()
update()

JDBC Interfaces

Some make sense (ResultSet, Blob)

Some do not make sense (Statement)

Truth in Advertising

I promised Python in the title too...

NDB API in Python

```
from mysql.cluster import ndbapi

connection = ndbapi.NdbClusterConnection()
connection.connect(5, 3, 1)
connection.waitUntilReady(30, 30)
myNdb = connection.createNdb("test", 4)

myTransaction = myNdb.startTransaction()
myOperation = myTransaction.getNdbOperation("t1")
myOperation.readTuple(ndbapi.NdbOperation.LM_Exclusive)
myOperation.equalInt("id", 42)
myRecAttr = myOperation.getValue("x")

myTransaction.executeNoCommit()
```

MGM/J

Some things you can't do through SQL at all!

```
NdbMgm mgm =
    NdbMgmFactory.createNdbMgm(connectString);

mgm.setConnectTimeout(4000);
mgm.connect(5, 3, true);
NdbFilterList theList = new NdbFilterList();
NdbFilterItem theItem1 = new NdbFilterItem(15,

NdbLogEventCategory.NDB_MGM_EVENT_CATEGORY_STATISTIC);


theList.add(theItem1);

NdbLogEventManager manager =
    mgm.createNdbLogEventManager(theList);
TransReportListener theListener1 =
    new TransReportListener();
manager.registerListener(theListener1);
```

```
while(true) {
   try {
     manager.pollEvents(5000);
   } catch(NdbMgmException e){
     break;
   }
}
```

```
class TransReportListener
  extends TransReportCountersTypeListener {

  @Override
  public void handleEvent(TransReportCounters event) {
    System.out.println("Node " +
      event.getSourceNodeId() +
      ": Trans count:"+ event.getTransCount());
    System.out.println("Node " +
      event.getSourceNodeId() +
      ": Read count:"+event.getReadCount());
    System.out.println("Node " +
      event.getSourceNodeId() +
      ": Scan count:"+event.getScanCount());
    System.out.println("Node " +
      event.getSourceNodeId() +
      ": Range Scan count:"
      +event.getRangeScanCount());
    }
}
```

MGM/J Implementation

Also currently done via SWIG (it's in the same library)

MGM protocol is actually a protocol – and actually a TEXT based protocol – so chances are that MGM/J will become pure Java source code rather soon.

Moving Forward

Working towards 1.0 release for August
(I'm not in marketing – don't hold me to that!)

Working on integrating NDB/J with various Sun Products

It's GPL – Feel free to help out! (I love patches)

THANK YOU

Monty Taylor, Senior Consultant, MySQL

ID#7814

mordred@sun.com