



MySQL Cluster 7.0 - New Features

A blue-toned network diagram showing a globe with glowing lines representing data connections between various nodes.

MySQL Cluster 7.0
Carrier-Grade Availability & Performance!

Johan Andersson
MySQL Cluster Consulting
johan.andersson@sun.com

Mat Keep
MySQL Cluster Product Management
matthew.keep@sun.com



Agenda



- MySQL Cluster Adoption
- Key Enhancements of MySQL Cluster 7.0
 - Dynamic Scalability
 - Internal optimizations



Customers and Applications

NEPs, ISVs and Communications Service Providers

- Network Subscriber databases
- Service Delivery Platforms
 - > Messaging / Presence
- Telco Application Servers
- VoIP Infrastructure
- Intelligent Networks
- Value-added services
- IPTV / VoD
- Mobile Content Delivery
- On-Line app stores and portals
- DNS/DHCP for Broadband
- Payment Gateways
- Data Store for LDAP Directories



Alcatel-Lucent



vodafone



<http://www.mysql.com/industry/telecom/>

MySQL Cluster 7.0

Key Enhancements

Enhanced Performance & Scalability

- Multi-Threaded Data Nodes
- Dynamic On-Line Scalability
- Large Record Handling
- Multi-Threaded Disk Data File Access

Expanded Platforms & Interoperability

- Windows Port
- Data Store for LDAP Directories

Simplified Maintenance

- Back-up Snapshot



Announce at MySQL UC, April 20th – April 23rd 2009

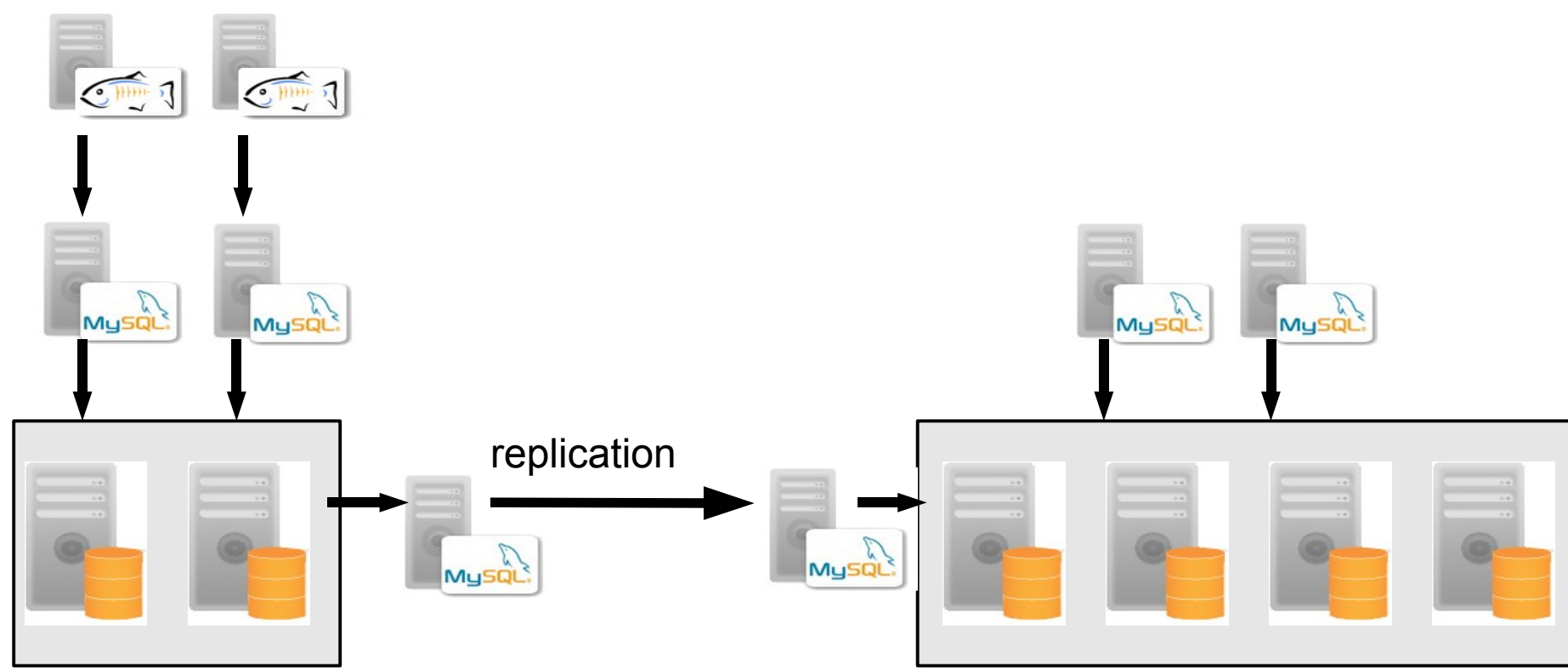
"We view MySQL Cluster Carrier Grade Edition as a **strategic technology** in our applications portfolio. With MySQL Cluster CGE 7.0 we are enabling our customers to **expose the rich capabilities of their networks** to drive a long-tailed innovation model. As a result, the operator can **leverage the creativity and agility of the web ecosystem** to deliver **new compelling, personalized and targeted services** to their subscribers without compromising reliability and scalability."

Jeff Cortley,
VP Subscriber Data Management,
Alcatel-Lucent



Alcatel-Lucent

Dynamic On-Line Scalability before - onlinish..



Dynamic On-Line Scalability

- before -

- Growing with MySQL Replication has a number of drawbacks:
 - Applications has to connect to the new cluster
 - Applications must wait for the binlog to be completely replicated to slave cluster before starting
 - Service interruption – applying the binary log can take time.
 - Possible to do it “faster”, but data will be lost
 - Switch applications before all binlog has been applied.
 - Extra hardware needed for the new Cluster
- Growing with Backup/Restore
 - Backup cluster
 - Stop cluster
 - Reconfigure and start new cluster
 - Restore cluster

Dynamic On-Line Scalability

ONLINE ADD NODE

Applications are not interrupted

No data loss

Easy procedure

Dynamic On-Line Scalability

- **Scale storage capacity - online**
 - Start with e.g two data nodes and extend the size of the Cluster over time.
 - Prior versions (≤ 6.3) and most other vendors requires downtime to accomplish this.
- **Scale transaction handling capacity – online**
 - More data nodes \rightarrow more transactions can be handled
 - Scale the application layer online by adding more MySQL Servers (has always been online)
- **Online means no service interruption!**
- **Other features**
 - No extra memory is needed on existing data nodes
 - Range scans / scans are not disturbed
 - Replication (mysql replication) is consistent and not affected.

Dynamic On-Line Scalability

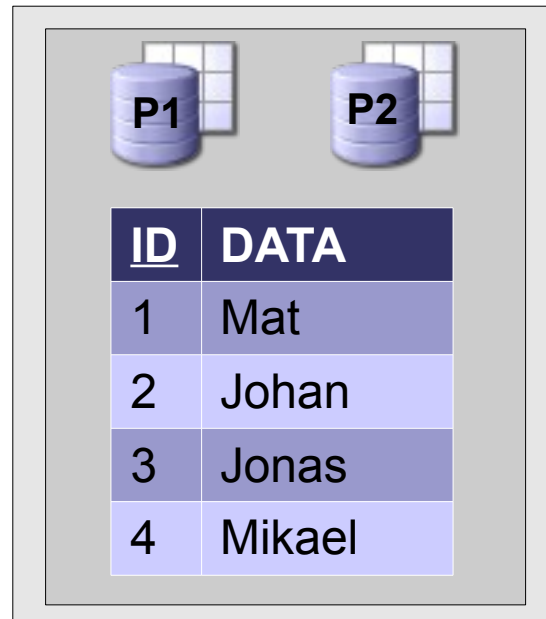
- Adding data nodes is done in three steps

Step 1: The new data nodes must join the cluster (two options to do this)

Step 2: Tell Cluster that new nodes has joined the cluster
`ndb_mgm> create nodegroup (5,6)`

Step 3: Data stored on the original set of data nodes must be repartitioned on the new data nodes
`mysql> ALTER TABLE <tablename> REORGANIZE PARTITIONS;`

Dynamic On-Line Scalability

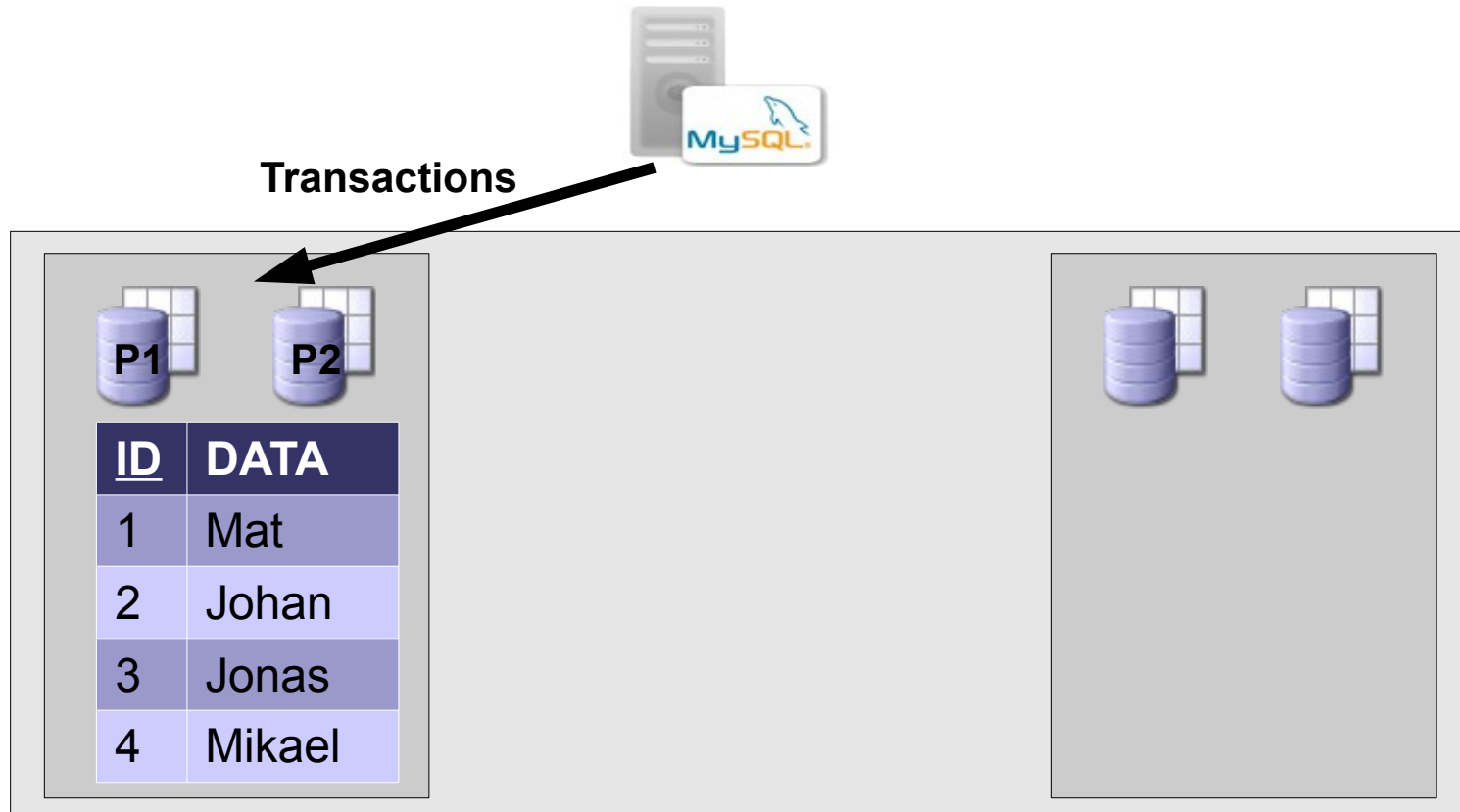


Starting point:

Two data nodes in one node group

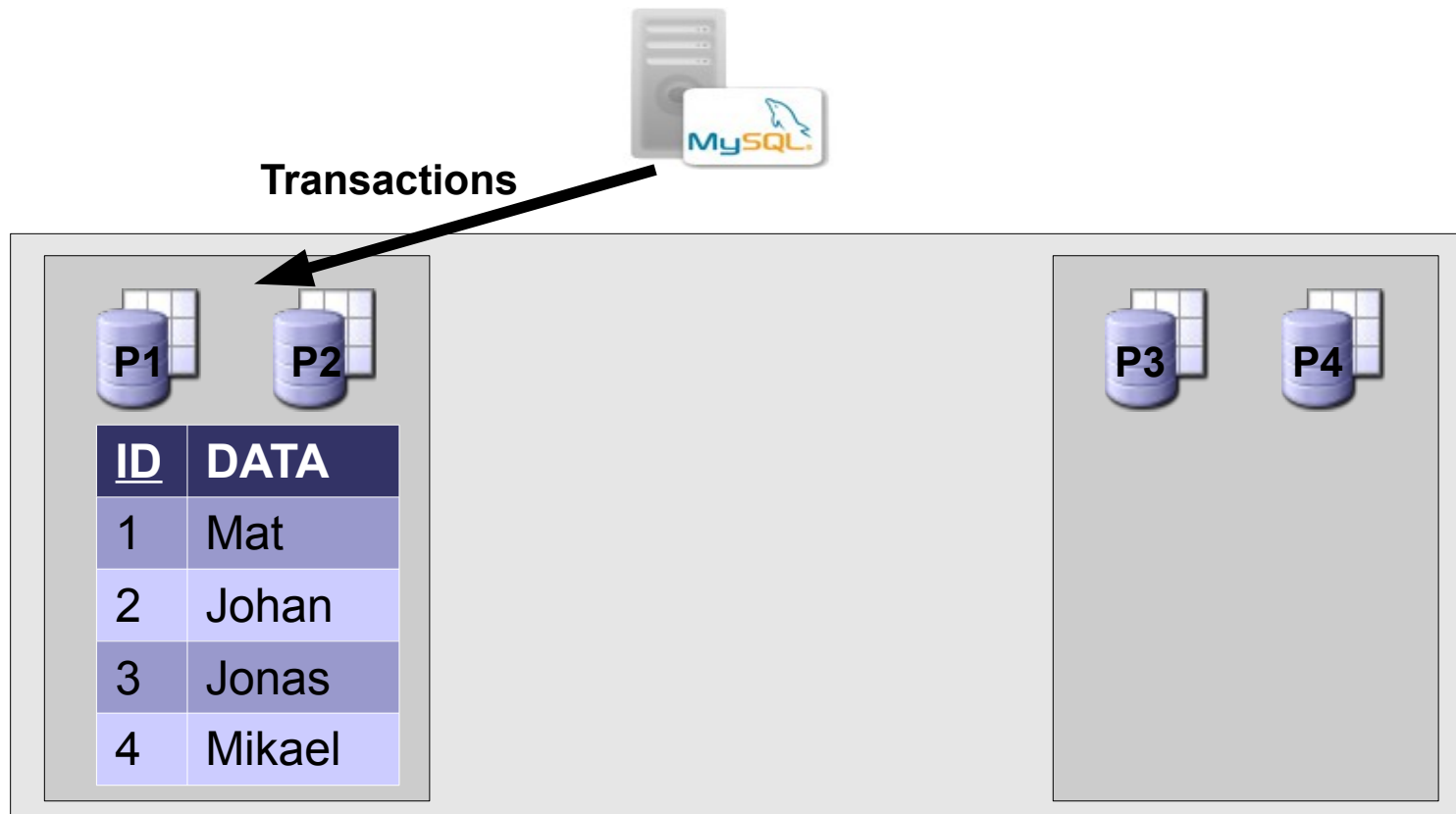
Two partitions, P1 and P2

Dynamic On-Line Scalability



1. Start new data nodes
2. Create a new empty node group (via `ndb_mgm`) with the new data nodes.
Data is not yet re-partitioned.

Dynamic On-Line Scalability



3. Data on P1 and P2 must be redistributed to P3 and P4:
mysql> ALTER TABLE .. REORGANIZE PARTITIONS
4. internal: New partitions are added (P3, and P4)
but not online

Dynamic On-Line Scalability

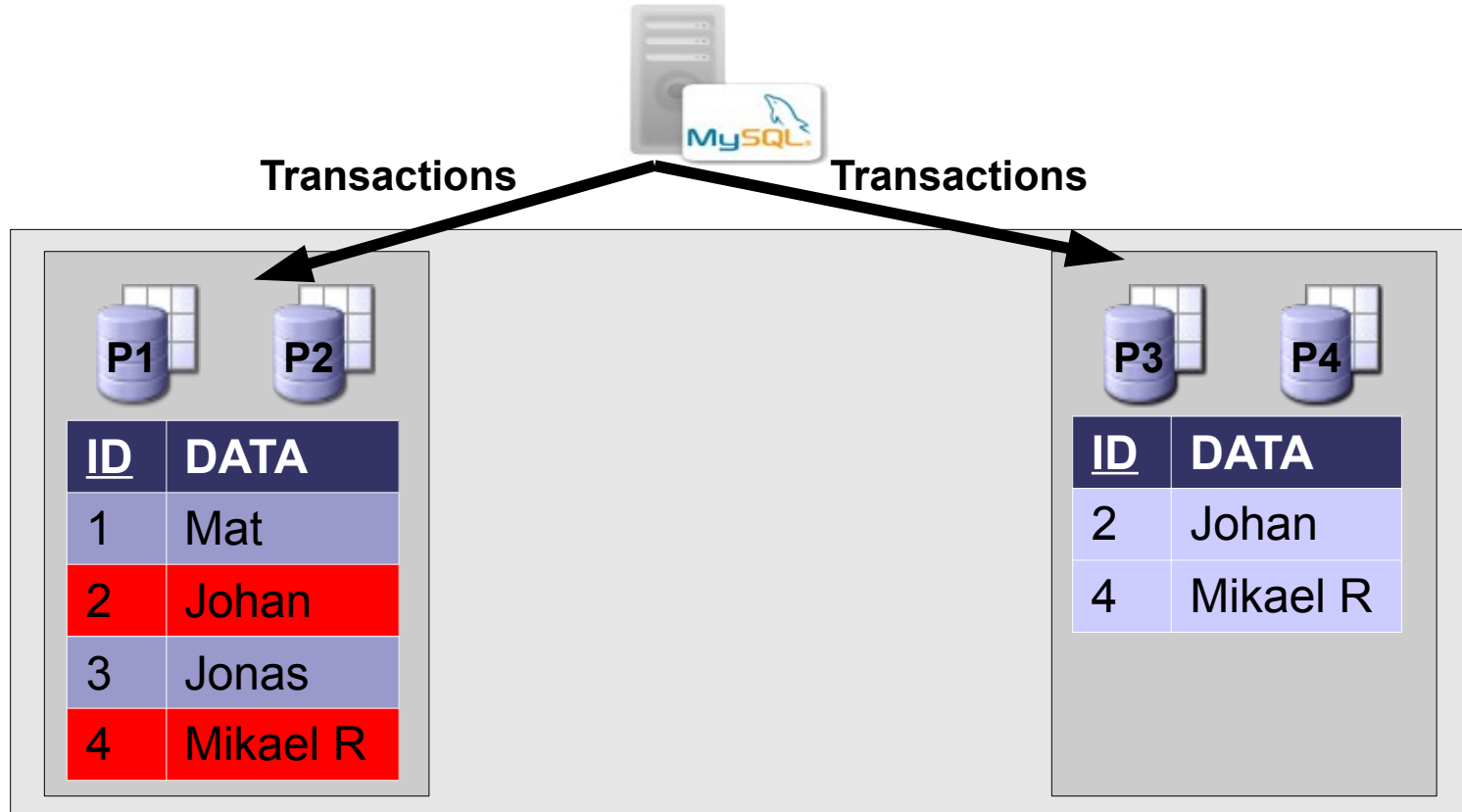
Transactions

Update: Mikael → Mikael R



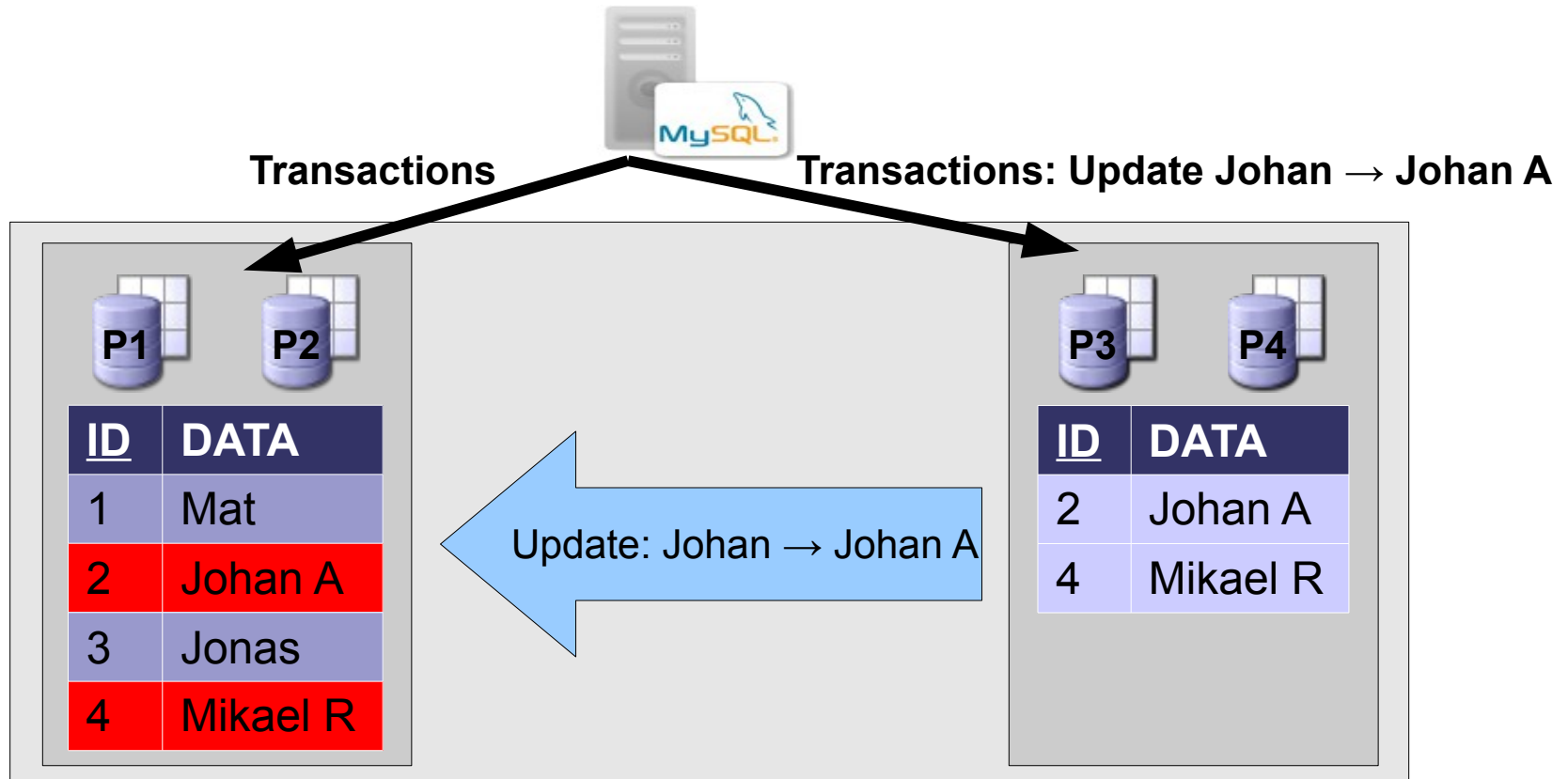
6. internal:
 - * Redistribute records (copy) and mark which records have been copied.
 - * Writes on already copied data are reflected (part of the transaction) on P3 and P4

Dynamic On-Line Scalability



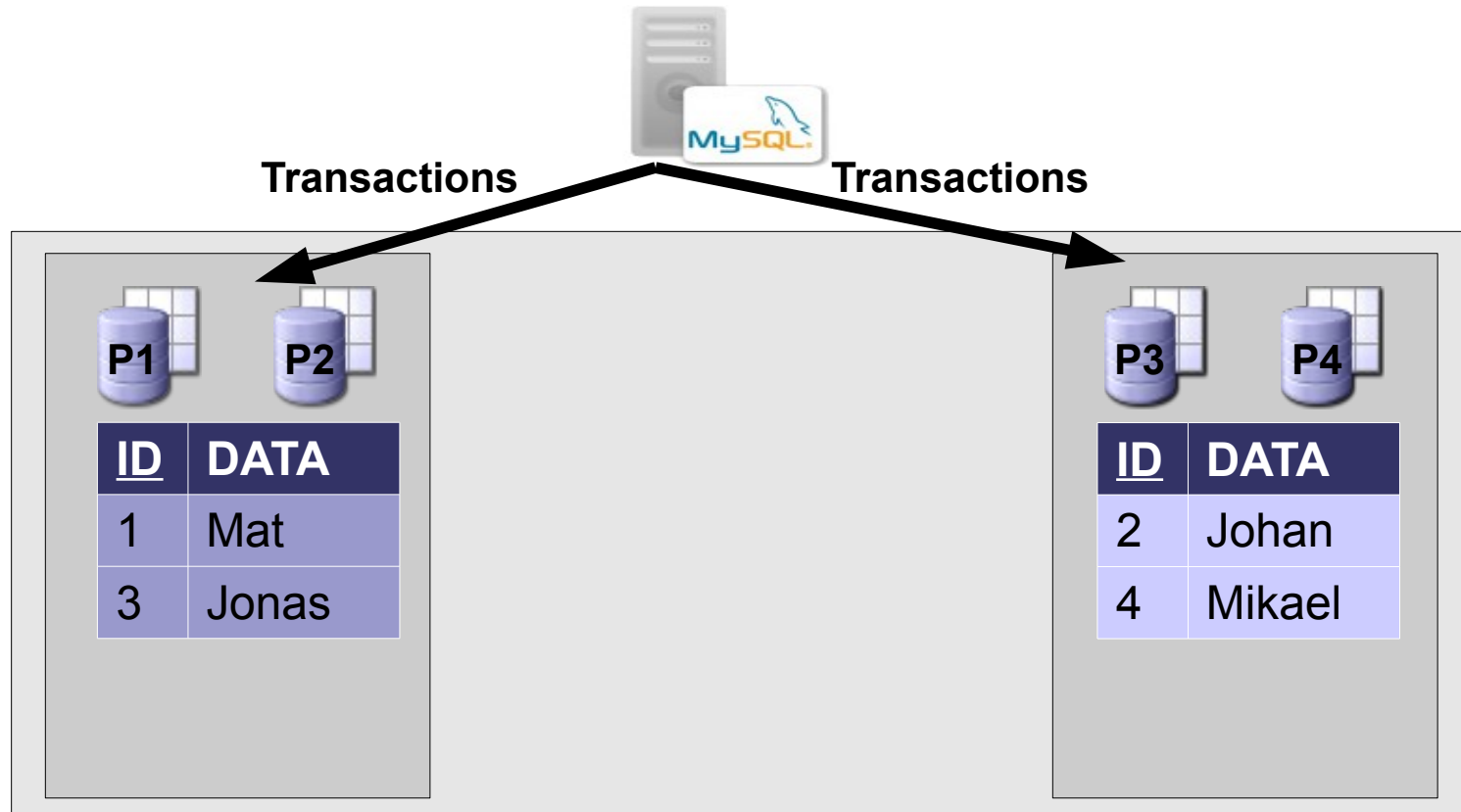
7. internal: * Copy completed.
 * Perform Switchover:
 Let everyone know that the distribution has changed.
 * Transactions now be started on P3 and P4.

Dynamic On-Line Scalability



8. internal: *
- * Wait for long running scans started on P1 or P2 *before* the switchover to finish.
 - * During this phase writes on P3 and P4 are reflected on the copied records on P1 and P2.

Dynamic On-Line Scalability



9. internal: *
- * Old scans on P1 and P2 are now completed
 - * Delete the marked records from P1 and P2.
 - * The ALTER TABLE... REORG is now completed!

Dynamic On-Line Scalability

- Add data nodes option 1) - Rolling Restart
 - Add new data nodes to config.ini
 - restart management servers,
 - restart existing data nodes
 - start added nodes
 - restart mysql servers (NDBAPI programs).
 - Pros: requires no planning at initial deployment.
 - Cons: rolling restart is rolling restart .. (but service is not interrupted)
- Add data nodes option 2) - Preallocate data nodes
 - Preallocate “empty” data nodes to config.ini at startup
 - These data nodes are inactive, and not started until you decide to add them.
 - Pros: Easier and faster than rolling restart
 - Cons: requires planning at initial deployment

Dynamic On-Line Scalability

- Preallocate data nodes for add node
 - Preallocate new data nodes in config.ini at beginning of time

```
[ndbd]
id=3
hostname=A
```

```
[ndbd]
id=5
hostname=C
nodegroup=65536
```

```
[ndbd]
id=4
hostname=B
```

```
[ndbd]
id=6
hostname=D
nodegroup=65536
```

- To add the new data nodes simply start one data node each on C and D, then do
- `ndb_mgm> CREATE NODEGROUP (5,6)`
- `mysql> ALTER TABLE .. REORGANIZE PARTITIONS`

Dynamic On-Line Scalability

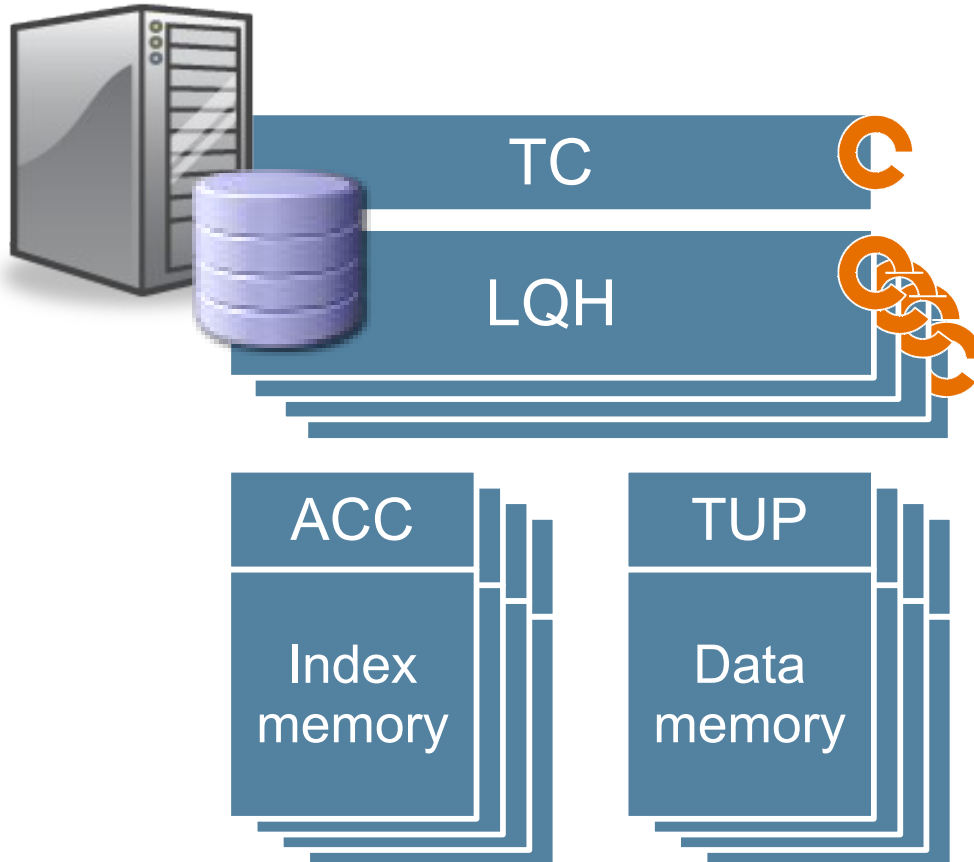
- 2M records and 2GB data in table – two data nodes
- Baseline:
 - 20 threads performing updates of 256B of data on random records.
 - Total throughput = **18315.06 qps**
 - Average response time=**1089us** for each update
- With reorg on + no update load – 2 → 4 data nodes
 - mysql> alter table t1 reorganize partition;
 - Query OK, 0 rows affected (**2 min 44.58 sec**)
 - Records: 0 Duplicates: 0 Warnings: 0

Dynamic On-Line Scalability

- With UPDATES and ongoing reorg (2 → 4 data nodes)
ongoing reorg means:
 ALTER TABLE t1 REORGANIZE PARTITIONS;
 20 threads each doing updates as in baseline
 Total throughput = **10907.70 qps**
 Average response time=**1827us** for each update

mysql> alter table t1 reorganize partition;
Query OK, 0 rows affected (**3 min 39.45 sec**)
Records: 0 Duplicates: 0 Warnings: 0
- Throughput decreases with about 40% during the reorg.
- Latency increases with about 68% during the reorg.

Scale Up: Multi-Threaded Data Nodes



Multi threaded ndbd

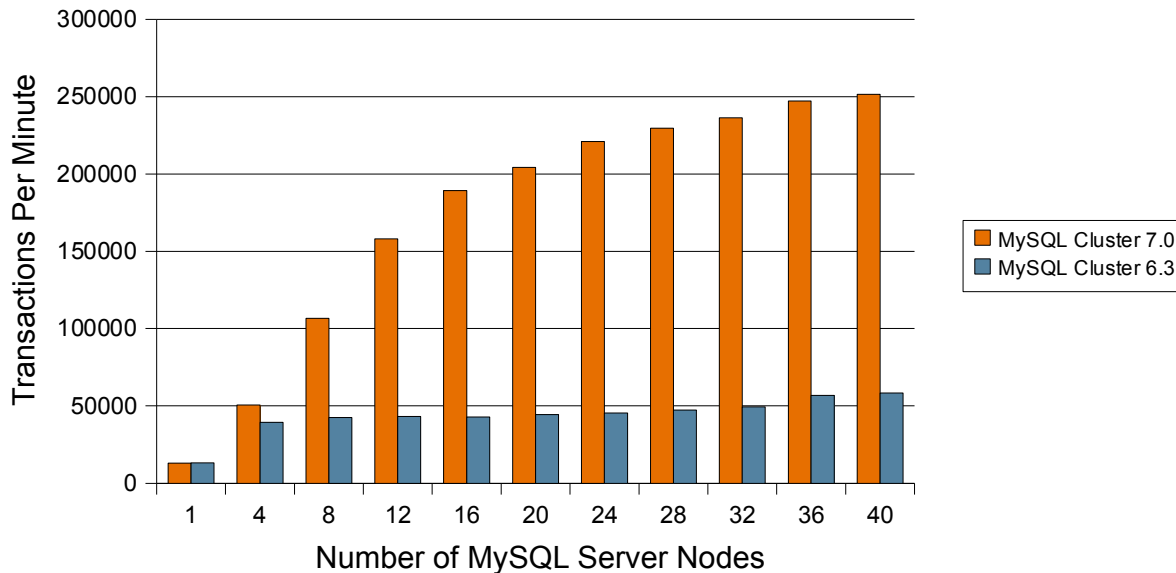
- Cluster 7.0
- Sub partition threads (up to 4)
- Separate TC and communication threads
- Utilizes 8 cores
- Scale out in 2D!

Improved Vertical Scalability on Multi-Core / Multi-Thread Hardware

- Increase Cluster throughput
- Reduce hardware requirements to achieve equivalent throughput
- Simpler maintenance

MySQL Cluster Benchmarks

DBT2 Benchmark, 4-MySQL Cluster Data Nodes



Data Nodes

Sun Fire x4450s

SQL Nodes

Sun Fire x4600s &
x4450s

OpenSolaris

Gigabit Ethernet

MySQL Cluster delivers

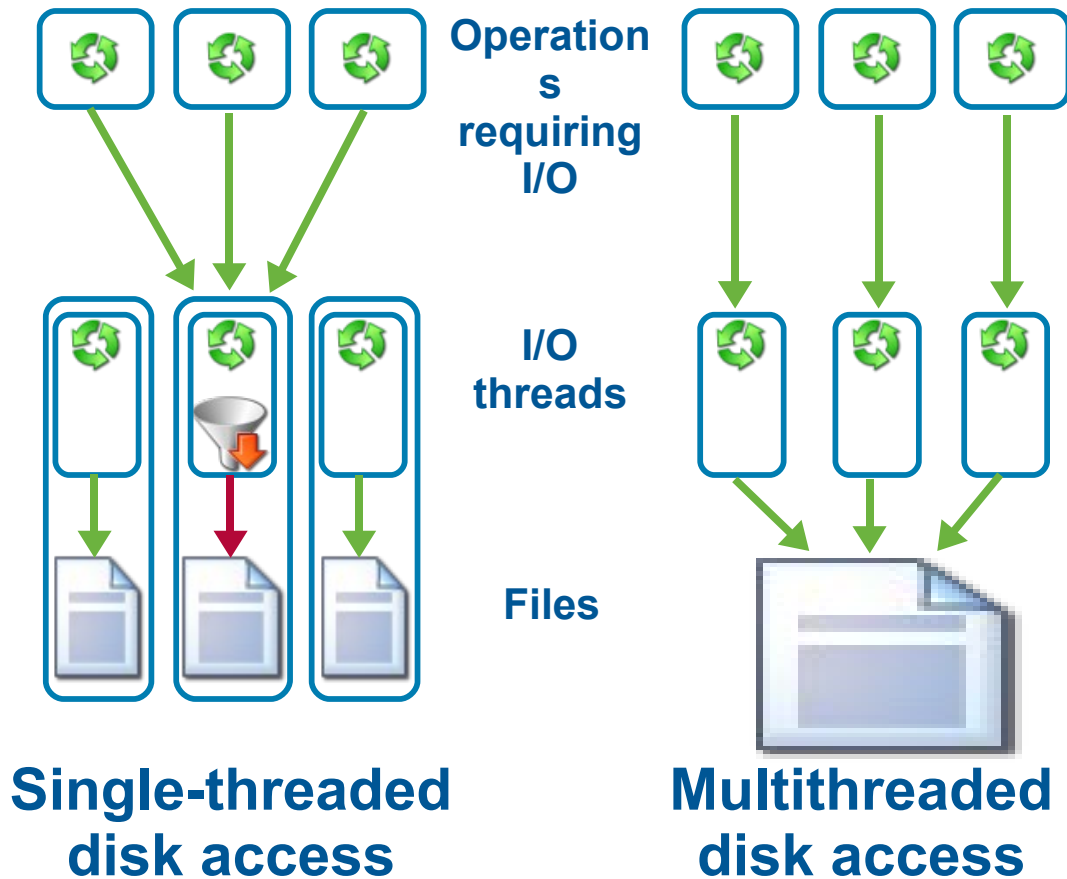
4.3x higher throughput at 4-nodes

MySQL Cluster 7 achieves 140k+ TPM with 2 data nodes vs 10 data nodes required for MySQL Cluster 6.3

4x less power and space consumption

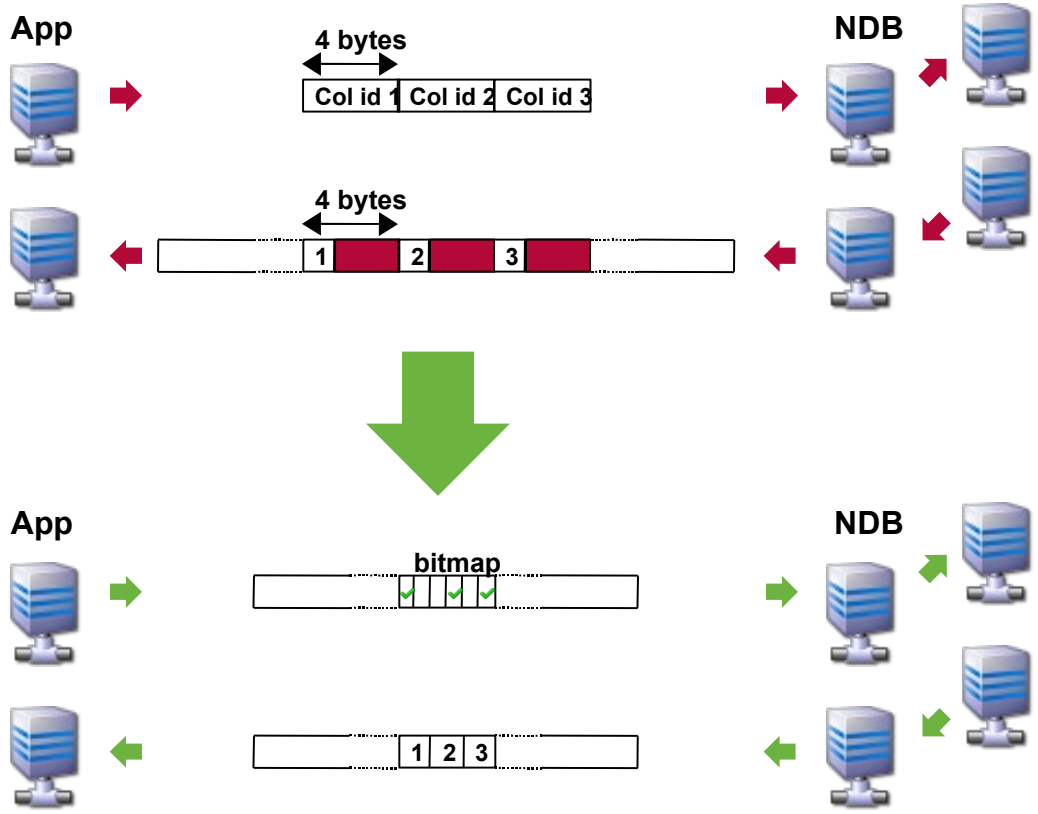
Higher cluster throughput with fewer nodes

Multithreaded Disk Data Access



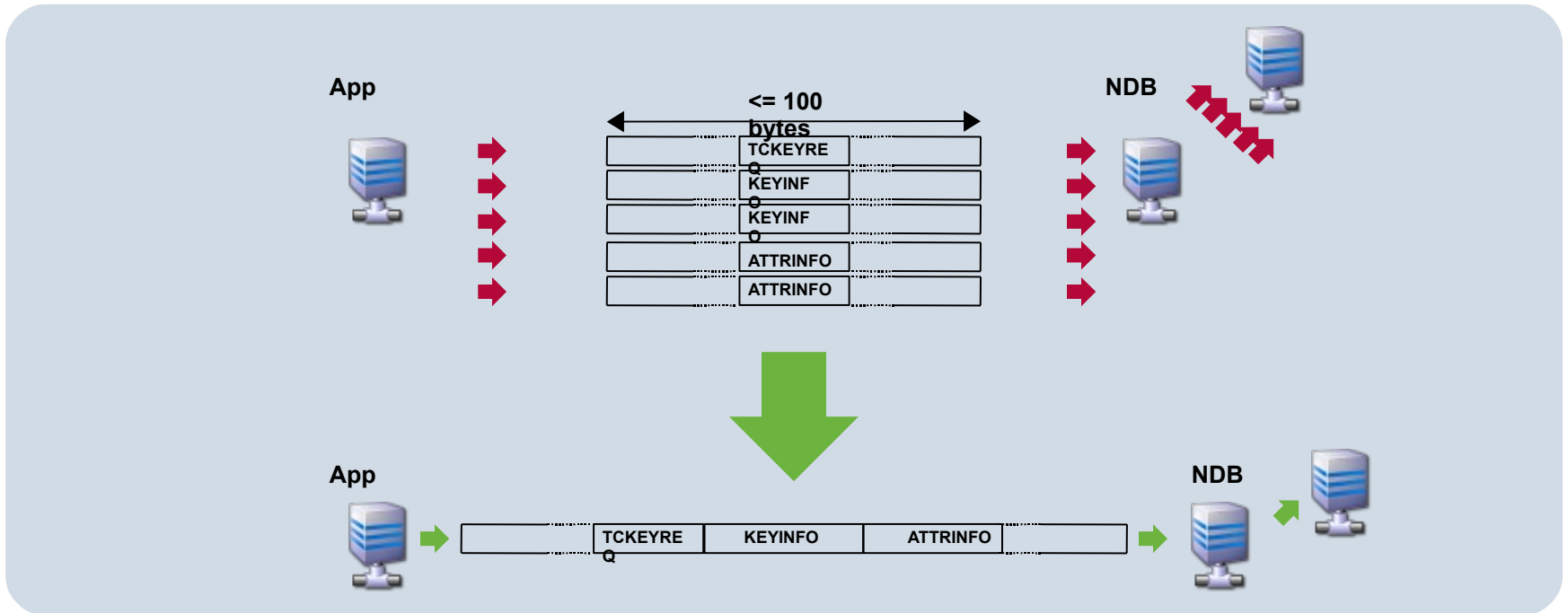
- All file access is through I/O threads
- Previously a one-to-one mapping between all open files and I/O threads
 - > Still used for some files where latency of single I/O critical — e.g., redo logs
- Can now use pooled I/O threads, breaking the one-to-one mapping
 - > Greater I/O throughput to the file
 - > Results in increased performance for disk-based table data
 - > Users no longer need to split large files into smaller ones as a workaround — saving admin work
 - > Extends to other files to reduce overall number of I/O threads in the system — saving memory

Large Record Handling (1)



- Optimization of the NDB messaging protocols between apps and NDB nodes and between NDB nodes
- Packed read reduces the size of read and response messages
 - > Reduces bandwidth used on the network — increasing throughput
- Read request: replaces a list of column identifiers with a bitmap
- Read response: column values no longer need to be packed to a 4 byte boundary
- Changes are transparent to the application, but designers might choose to design schemas that further exploit the mechanism so that fields pack tightly

Large Record Handling (2)



Optimization of the NDB messaging protocols between apps and NDB nodes and between NDB nodes

Long signal transactions:

Reduces bandwidth used on the network, increasing throughput (50–100% improvements observed)

Transparent to the application

Existing messages limited to 100 bytes so complex requests split across multiple messages:

Bandwidth wasted on OS and protocol overhead (header info) for each message

CPU consumed reassembling operation

LST enables large operations to be specified in a single message (up to 32 KB); fragmented signals can be used if larger messages are needed

Snapshot Backup

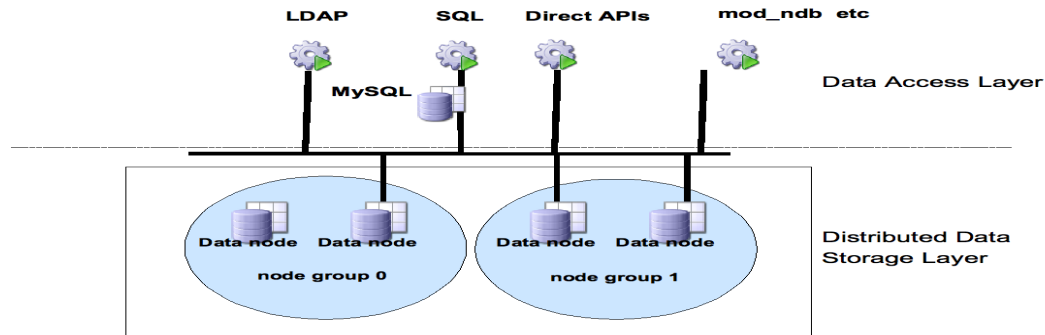
- Prior to 7.0, backups reflected the state of the Cluster at the end of the backup.
- With 7.0 the backup can reflect the state of the cluster as it looked at the start of the backup
- New options:
SNAPSHOTSTART or SNAPSHOTEND
Default is SNAPSHOTEND..
 - `ndb_mgm -e "start backup snapshotstart"`
- This is good in e.g sharding where you want all shards to have the backup from the same point in time.
 - Otherwise they may end at an arbitrary point in time.

Windows Support

- Alpha release of MySQL Cluster 7.0 for MS Windows.

Data Store For LDAP Directories

- OpenLDAP and OpenDS support MySQL Cluster
 - Native implementation on MySQL Cluster
 - Fast and High Availability
 - A paradigm shift for scaling out LDAP



Learn more at

Session: LDAP for MySQL Cluster - back-ndb

Ballroom B at 14:00 on Thursday (23 Apr 2009).

BOF: OpenDS

Tuesday 21st, 19.30, Ballroom Foyer



That's all

Questions?

We are here all week to
take your questions!

Webinar April 30th – Learn
more about Cluster 7.0

<http://www.mysql.com/news-and-events/web->

