# Introducing MySQL Cluster

全ノードアクティブ型分散データベース
＆トランザクショナルKVS

日本オラクル株式会社
MySQL Global Business Unit

# SAFE HARBOR STATEMENT

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメントするものではない為、購買決定を行う際の判断材料になさらないで下さい。

オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Server

# Cluster

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Agenda

1. MySQL Cluster概要
2. システムデザイン(構成とサイジング)
3. 構成パラメータと構成変更 (--reload, --init)
4. バックアップとリカバリー (backup, mysqldump)
5. スケールアウト(データノード追加とデータ再配置)
6. レプリケーション(NDB -> NDB,  NDB -> InnoDB )
7. その他、参考情報

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Agenda

1. MySQL Cluster概要
2. システムデザイン（構成とサイジング）
3. 構成パラメータと構成変更 (--reload, --init)
4. バックアップとリカバリー
5. スケールアウト(データノード追加とデータ再配置)
6. レプリケーション(NDB -> NDB,  NDB -> InnoDB )
7. その他、参考情報

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# History of MySQL Cluster ”NDB”

MySQL Clusterの基礎となっている技術は、通信機器ベンダのエリクソンで携帯通信網の加入者データベース向けに開発されたEricsson Network Data Base(NDB)と呼ばれていた技術が起源

**Original design paper:**

Design and Modelling of a Parallel Data Server for Telecom Application

**90年代後期**

Ericsson (Alzato) にて設計/開発

通信業者向けDBとして誕生

**2003年**

MySQL ABが NDB Cluster Projectを
Ericssonから引き継ぎ開発 (MySQL 4.1)

**2008年**

SunがNDB Cluster Projectを引き継ぎ開発
(MySQL 5.1/Cluster 6.x～ )

**2010年**

OracleがNDB Cluster Projectを引き継ぎ開発
(MySQL 5.1 / Cluster 7.x～)

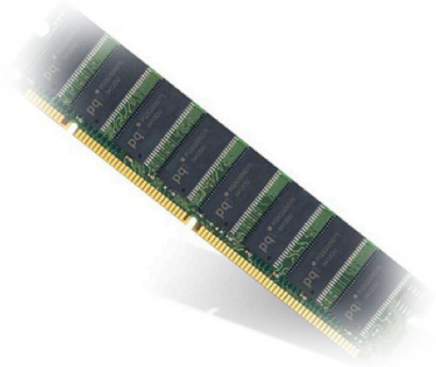
Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# MySQL Clusterとは？

- MySQLとは開発ツリーの異なる別製品
- 共有ディスクを使わずに、アクティブ-アクティブのクラスタ構成が組める
- 元々はSQLを使わないデータベース(NOSQL)だったが、MySQLと統合され、SQLも追加で使えるようになった(NoSQLとSQL共に使えるデータベース)
- 各テーブルのストレージエンジンを選択する事が出来る(InnoDB or NDB)
- SQL, NoSQL共にACID処理可能なインメモリーデータベース

## 適したシステム

- 高可用性が求められるシステム
- 同時多発的に大量の短いトランザクションが発生するシステム
- 読込み処理だけでなく、書込み処理に対しても拡張性が求められるシステム



参照：MySQL Cluster Evaluation Guide

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# MySQL Cluster Overview

| | |
|---|---|
| リアルタイム | • 永続性を持つインメモリテーブル ＋ ディスクテーブル<br>• 低レイテンシ |
| 参照更新性能の高い拡張性 | • 自動シャーディング、マルチマスタ<br>• ACIDトランザクション、OLTPとリアルタイム分析 |
| 99.999% の可用性 | • シェアードナッシング、単一障害点無し<br>• 自動復旧、オンラインメンテナンス |
| SQL + NoSQL | • キー・バリュー型＋複雑なリレーショナルな処理<br>• SQL + Memcached + JavaScript + Java + HTTP/REST & C++ |
| 低コスト | • オープンソース＋商用版運用支援ツール<br>• 特殊なハードウェア不要、管理監視ツール群、サポート |

ORACLE

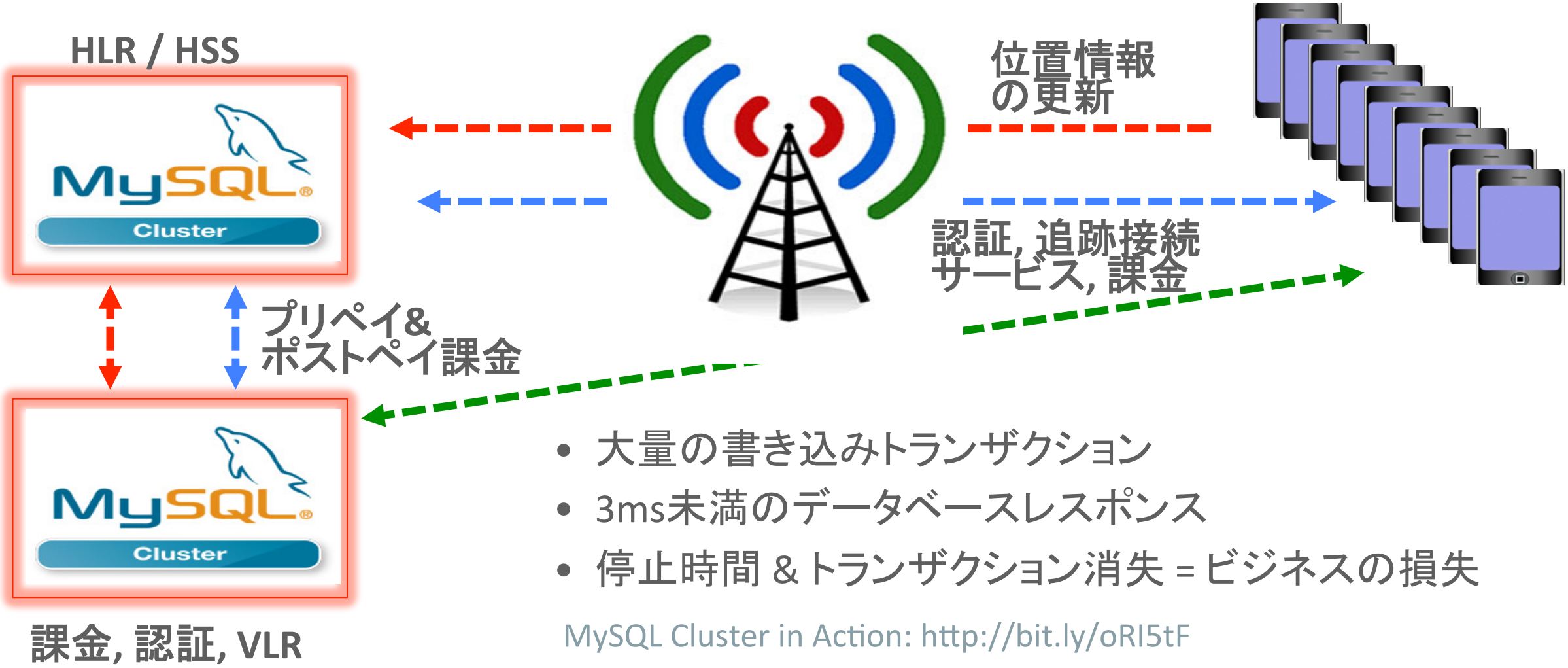
Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# When to Consider MySQL Cluster

- ダウンタイムやパフォーマンス不足がビジネスへ及ぼす影響は?
- 高可用性実現の為に、どれだけの労力やコストを費やしましたか?
- 書き込み処理を拡張する為に、データベースのシャーディングを検討していますか? それは、アプリケーションや開発者にどれだけ影響がありますか?
- 対象サービスにリアルタイム処理は必要でしょうか?
- 対象サービスは急なニーズ増加に対して、書き込みを含む拡張性は必要でしょうか?
- データ処理にSQLだけでなく、NoSQLなども柔軟に活用出来れば良いと考えてますか ?

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 導入事例: 携帯電話ネットワーク



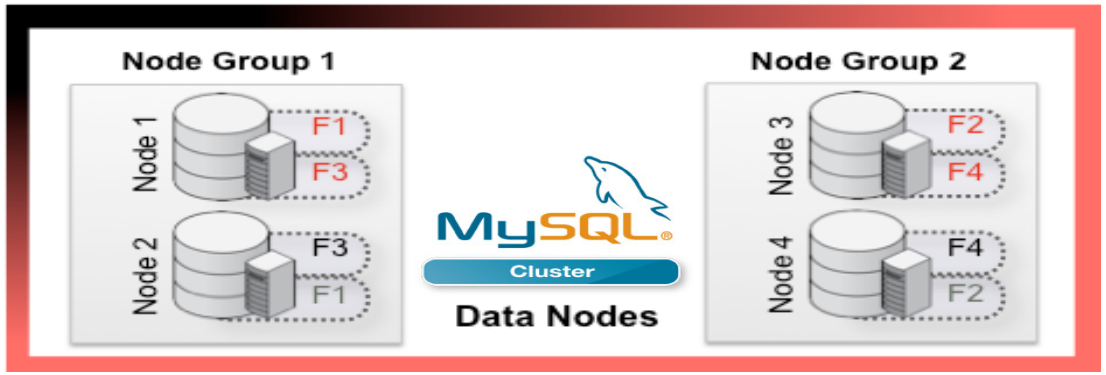

- 大量の書き込みトランザクション
- 3ms未満のデータベースレスポンス
- 停止時間 & トランザクション消失 = ビジネスの損失

MySQL Cluster in Action: http://bit.ly/oRI5tF

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 導入事例: 航空機管制システム




- 米国海軍航空母艦
- 包括的航空機運用管制システム
  - メンテナンス記録
  - 燃料搭載量管理
  - 気象状況
  - 飛行甲板管理
- システム要件
  - 単一障害点無し
  - 完全な冗長性
  - 小さなフットプリント & 過酷な利用環境での利用
- 4台のMySQL Clusterノード LinuxおよびWindows

MySQL User Conference Session: http://bit.ly/ogeid3

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 導入事例: PayPal

## PAYPAL OVERVIEW

- Processed $145bn in transactions (CY2012)
- 前年比成長率22%、1億23百万アカウント、190マーケット

## 課題/機会

- グローバル単位での不正検知システム構築
- ユーザセッション、トランザクション追跡のリアルタイム処理

## DATABASEに求められた要件

- 100TB のデータ & 1億ユーザ に対応可能
- ACID 準拠のトランザクション
- 書込まれたデータを1秒以内に全世界のどこからでも確認可能
- ユーザのトランザクション履歴をリアルタイムで分析可能
- クラウド環境における、高い性能拡張性と99.999%の高可用性

http://www.mysql.com/customers/view/?id=1223

## CUSTOMER PERSPECTIVE

***"You can achieve high performance and availability without giving up relational models and read consistency."***

Daniel Austin, Chief Architect, PayPal

## 課題解決策

- MySQL Cluster 7.2 with Geo-Replication
- AWS

## 結果

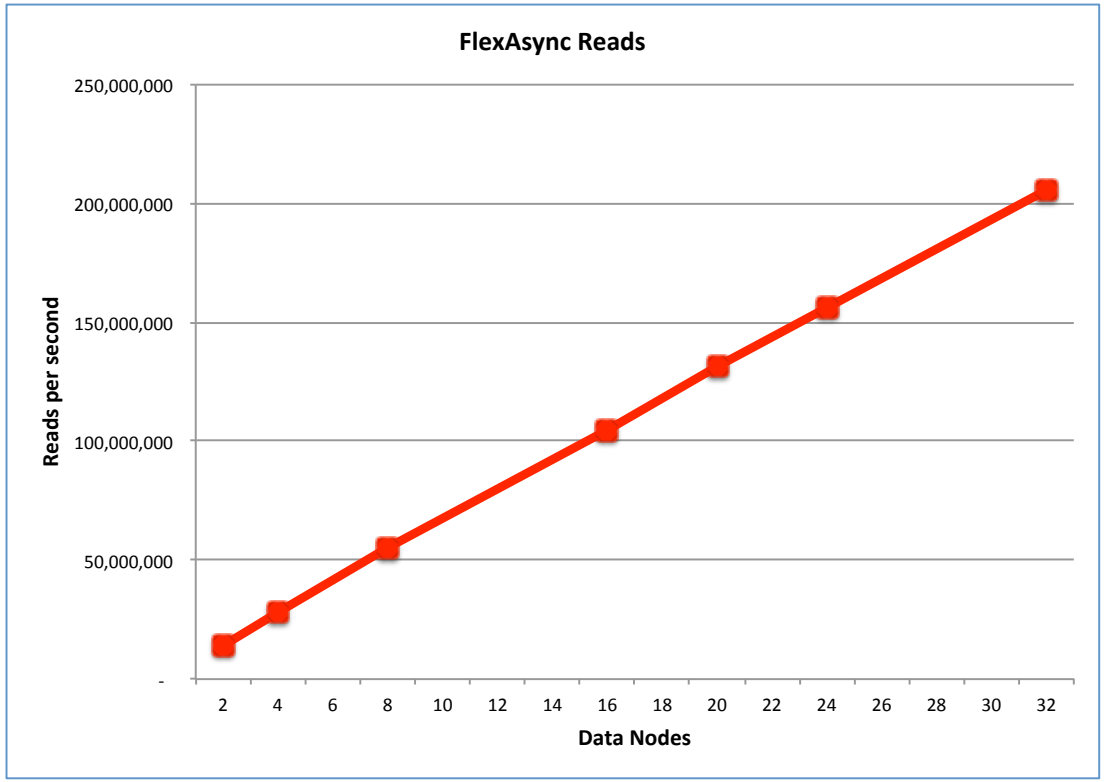
- 設計目標より3倍速いパフォーマンス
- 40TB のデータをCluster間で同期
- 5つのAWS リージョンにMySQL Clusterを構築
- Self-healing

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# MySQL NDB Cluster 7.4 NoSQL Performance
## 200 Million NoSQL Reads/Second

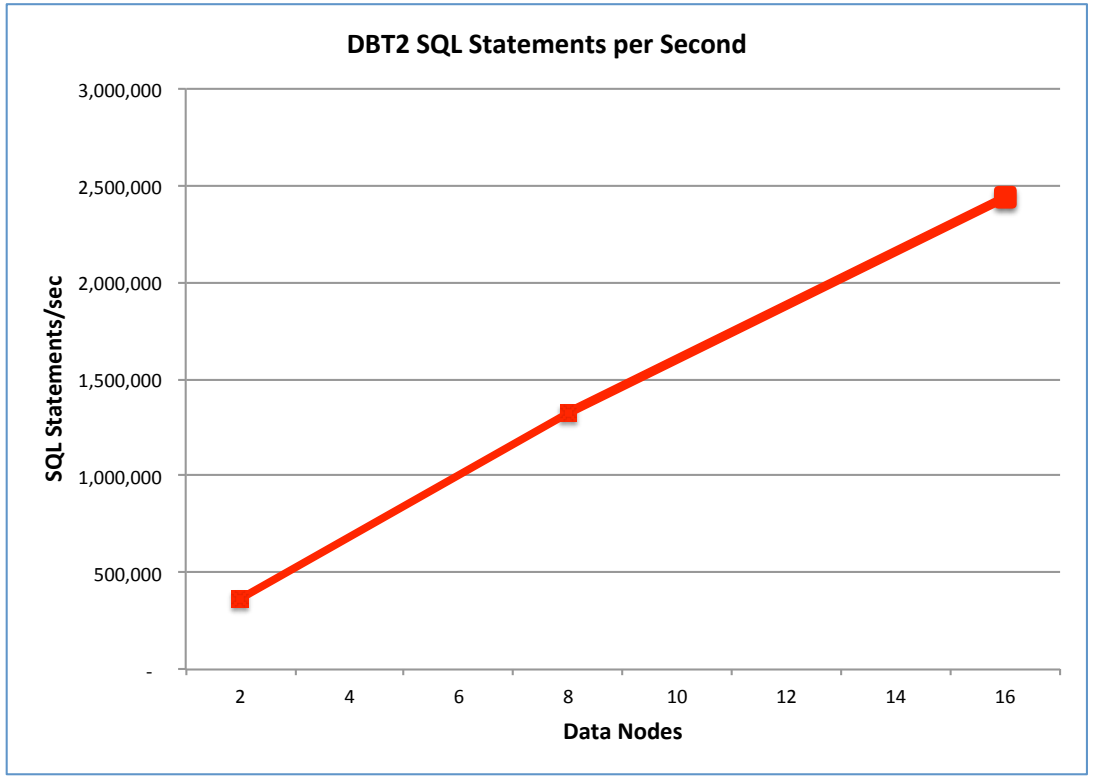
- Memory optimized tables
  - Durable
  - Mix with disk-based tables
- Massively concurrent OLTP
- Distributed Joins for analytics
- Parallel table scans for non-indexed searches
- MySQL NDB Cluster 7.4 FlexAsych
  - 200M NoSQL Reads/Second




Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# MySQL NDB Cluster 7.4 SQL Performance
# 2.5M SQL Statements/Second

- Memory optimized tables
  - Durable
  - Mix with disk-based tables
- Massively concurrent OLTP
- Distributed Joins for analytics
- Parallel table scans for non-indexed searches
- MySQL NDB Cluster 7.4 DBT2 BM
  - 2.5M SQL Statements/Second




Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Where would I **not** Use MySQL Cluster?

- データセット > 3TB
  - 64Gバイトの RAM を備えた48 個のデータノードを基にした場合の上限
- 実行時間の長いトランザクション
- BLOB以外で大きな行データが必要な場合
- 多くのフルテーブルスキャンが発生する処理
- Geo-Spatialインデックスを利用する場合

***"InnoDB storage engine would be the right choice"***

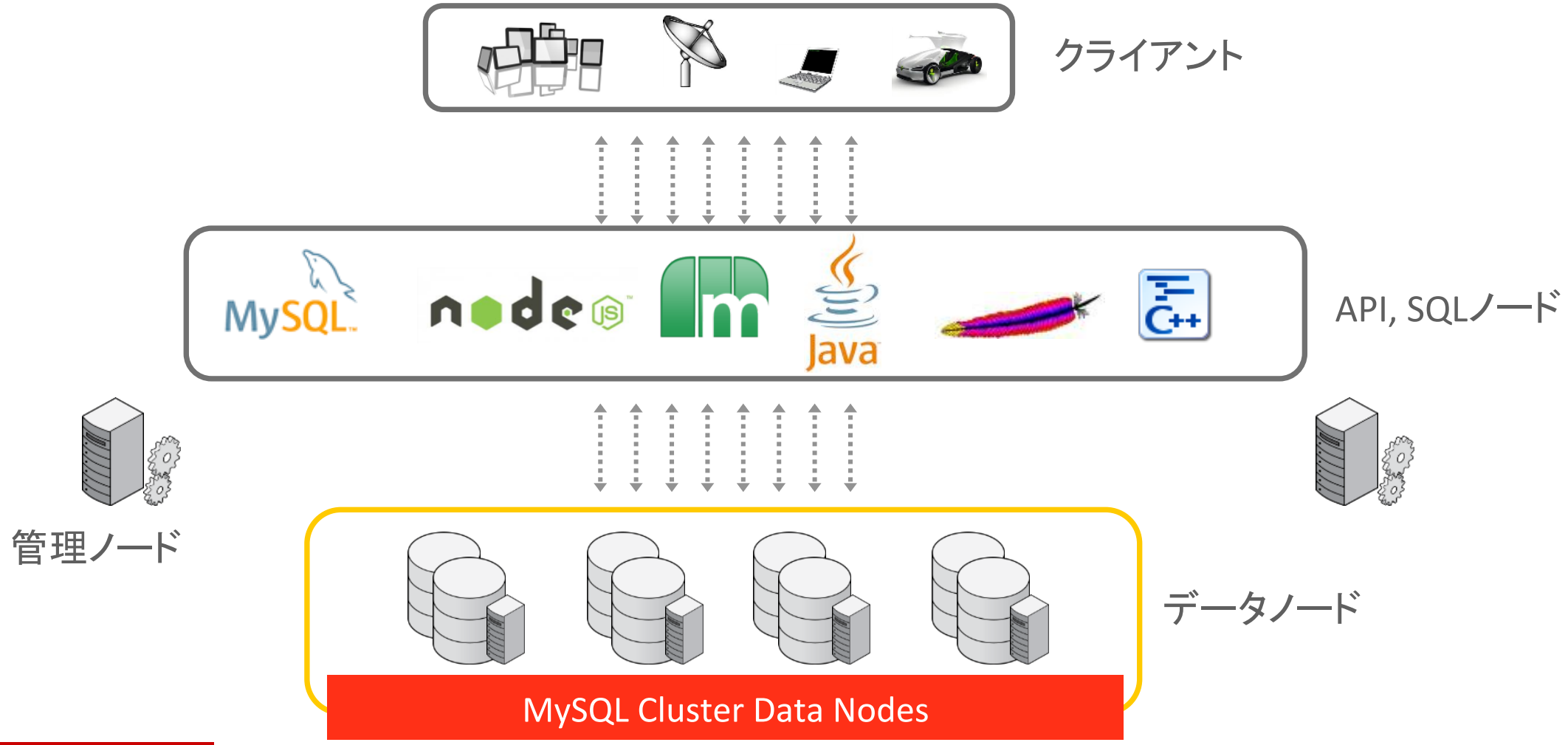
INNODB®

詳細：https://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-limitations.html

ORACLE

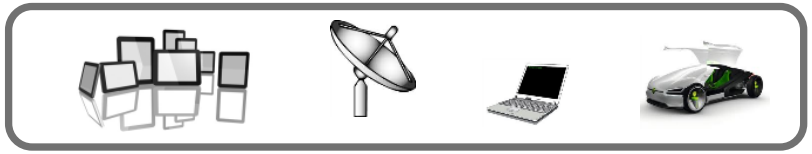
Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# MySQL Clusterのアーキテクチャ

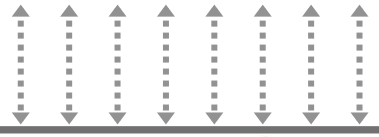



Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# MySQL Clusterの拡張性

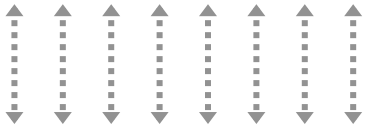


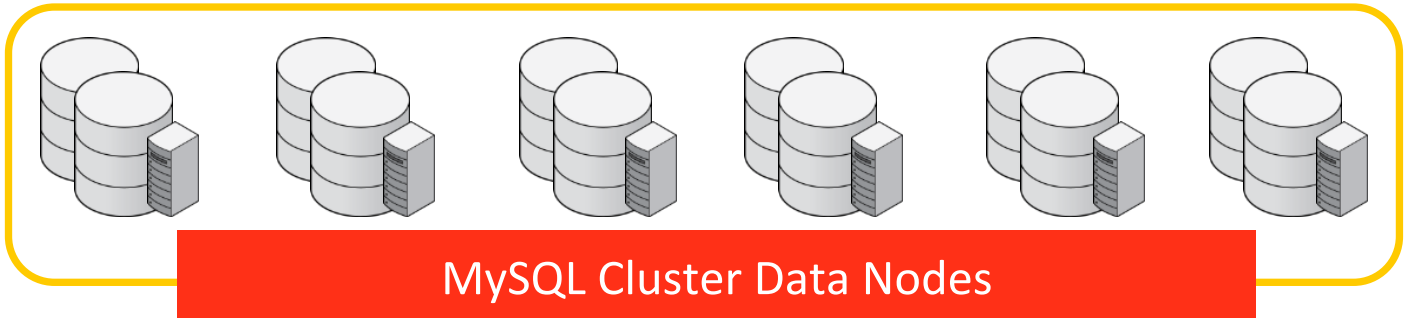
クライアント



API, SQLノード

管理ノード






データノード

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 管理ノード (management node)

ndb_mgmd -f config.ini [--reload/--initial] --configdir

- 全てのノードに構成情報を提供
- クラスターのモニタリングとロギング

  FILE:filename=ndb_node_id_cluster.log,maxsize=1000000,maxfiles=6

  例) LogDestination=CONSOLE;SYSLOG:facility=local0;FILE:filename=/var/log/mgmd
- Arbitratorとしての役割 (ArbitrationRank - 0:not used,1-high priority, 2-low priority)

  データノード間の通信がダウンした場合にsplit-brainを回避
- データノードと管理ノードを開始・停止・リスタート
- クラスターにてバックアップを開始
- 簡易レポート

```
ndb_mgm> HELP REPORT
---------------------------------------------------------------------------
 NDB Cluster -- Management Client -- Help for REPORT command
---------------------------------------------------------------------------
REPORT  Displays a report of type <report-type> for the specified data
        node, or for all data nodes using ALL
  <report-type> =
    BackupStatus        - Report backup status of respective node
    MemoryUsage - Report memory usage of respective node
    EventLog    - Report events in datanodes circular event log buffer
```

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# データノード (Data nodes)

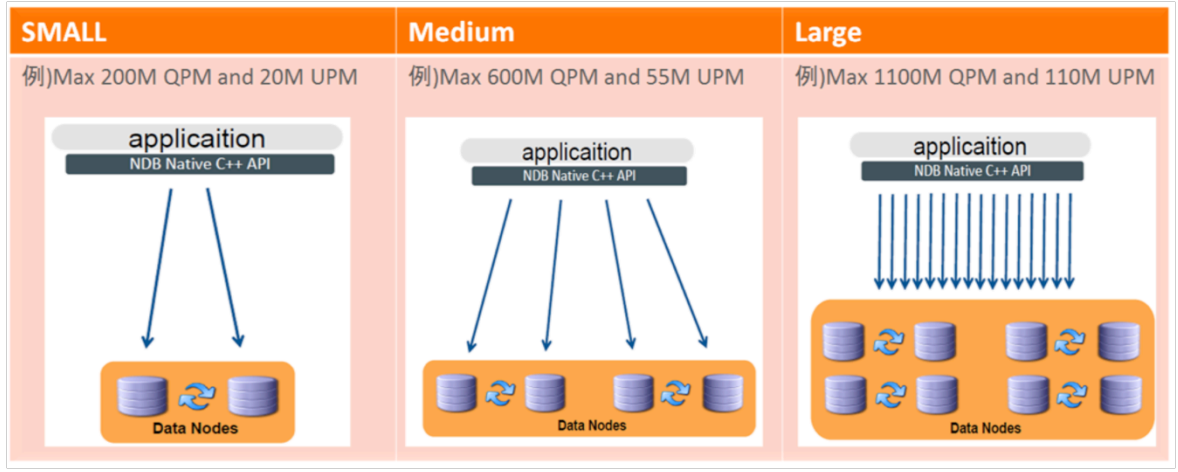
ndbmtd -c <mgt-node-host1>:<port1>[,<mgt-node-host2><port2>]

- データ、インデックス、メタ情報を保存
  - In memory,  Non-indexed data possible on disk
- メモリー内のデータをディスクへ書き込み(Checkpoint)
- トランザクションコーディネーター
- Fail Over処理
  - heartbeatsで他のデータノードのHealth Check
- オンラインバックアップ処理
- 最大48台までのスケールアウトサポート

NOTE:
1.データノードと管理ノードは、別々のノードに構築
2.ndbmtdはマルチスレッドですが、Defaultはシングルスレッドです。MaxNoOfExecutionThreadsかThreadConfigでスレッド数を定義する必要があります。

**MaxNoOfExecutionThreads**
- Single value for number of threads
- System will allocate these threads to blocks in a reasonable way

**ThreadConfig**
- Specify explicitly how many threads for each block type
- Lock threads to CPUs for further performance gains
- **Threadconfig=main={cpubind=0},ldm={count=4,cpubind=1,2,5,6},io={count=2,cpubind=3,4}**




Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 例) データノードパラメータ

**MaxNoOfConcurrentScan**
このパラメータは、クラスタ内で実行できる並列スキャンの数を制御するために使用されます。各トランザクションコーディネータは、このパラメータで定義された数の並列スキャンを処理できます。各スキャンクエリーは、すべてのパーティションを並列でスキャンして実行されます。各パーティションスキャンでは、パーティションが配置されているノード内のスキャンレコードが使用されます。レコードの数は、このパラメータの値にノードの数を掛けた値です。

**MaxNoOfLocalScans**
多くのスキャンが完全に並列化されていない場合の、ローカルスキャンレコードの数を指定します。ローカルスキャンレコードの数が指定されていない場合は、ここに示すように計算されます。
MaxNoOfLocalScans= 4 * MaxNoOfConcurrentScans * [# data nodes] + 2

**DiskSyncSize** (ODirect が有効になっている場合は、DiskSyncSize を設定する必要はありません)
ローカルチェックポイントファイルにデータをフラッシュするまでに格納される最大バイト数です。これは、パフォーマンスを大幅に低下させる可能性がある書き込みバッファリングを禁止するために行われます。これは、TimeBetweenLocalCheckpointsの代わりに使用するためのものではありません。

**DiskCheckpointSpeed**
ローカルチェックポイント中にディスクに転送されるデータの量 (バイト/秒)。
MinDiskWriteSpeed, MaxDiskWriteSpeed、MaxDiskWriteSpeedOtherNodeRestart、MaxDiskWriteSpeedOwnRestartでLCPとBACKUPの書き込み速度を制御。

**NoOfFragmentLogParts**
ndbmtd に属する Redo ログのログファイルグループの数を設定します。MySQL Cluster NDB 7.3.3 より前では、この値は 4 から 16 までの (これらを含む) 4 の偶数倍である必要があります。MySQL Cluster NDB 7.3.3 以降では、最大は 32 です。

**TimeBetweenGlobalCheckPoints**
トランザクションがコミットされると、データがミラー化されているノードのメインメモリーでコミットされます。デフォルトは2000 ミリ秒

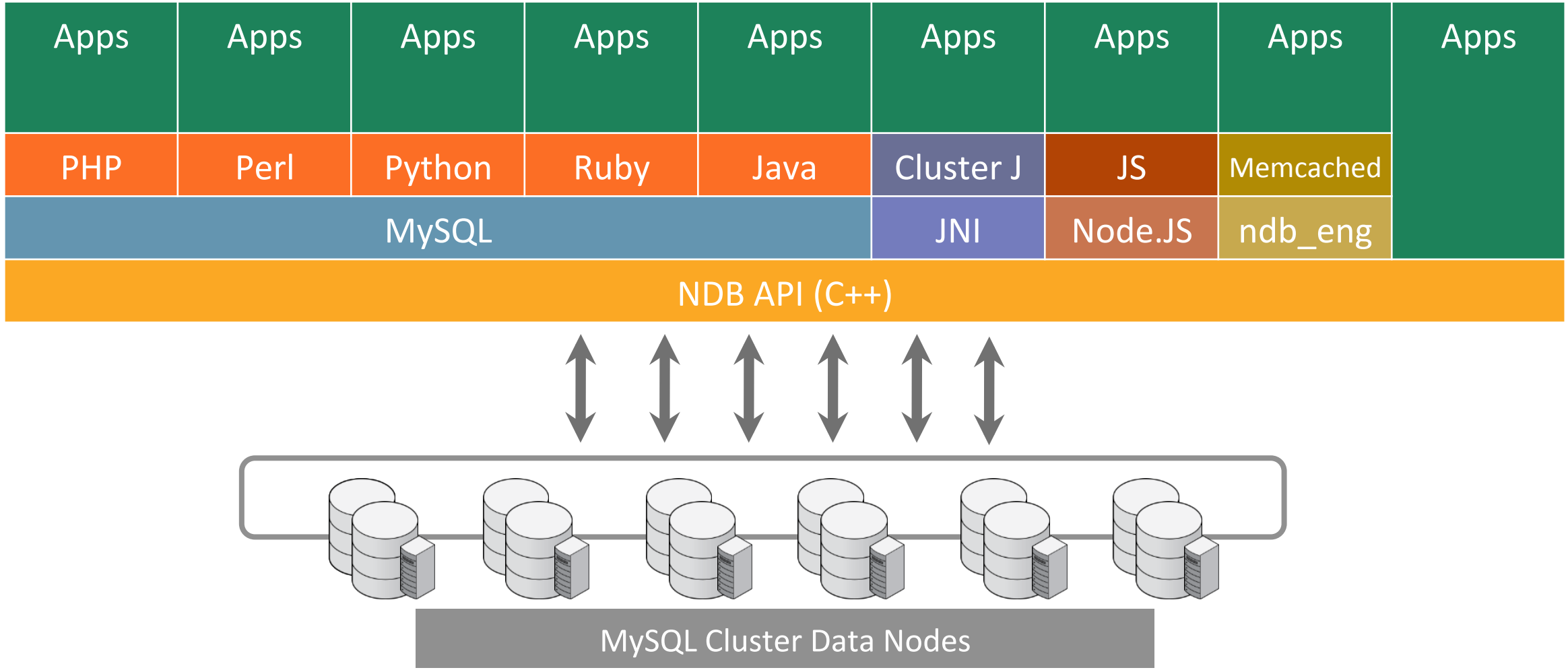
**BuildIndexTherads**
システムまたはノードの起動中に順序付けされたインデックスの再構築時、および**ndb_restore** --rebuild-indexes の実行時に作成するスレッドの数を指定

参考: http://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-ndbd-definition.html

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# NoSQL Access to MySQL Cluster data



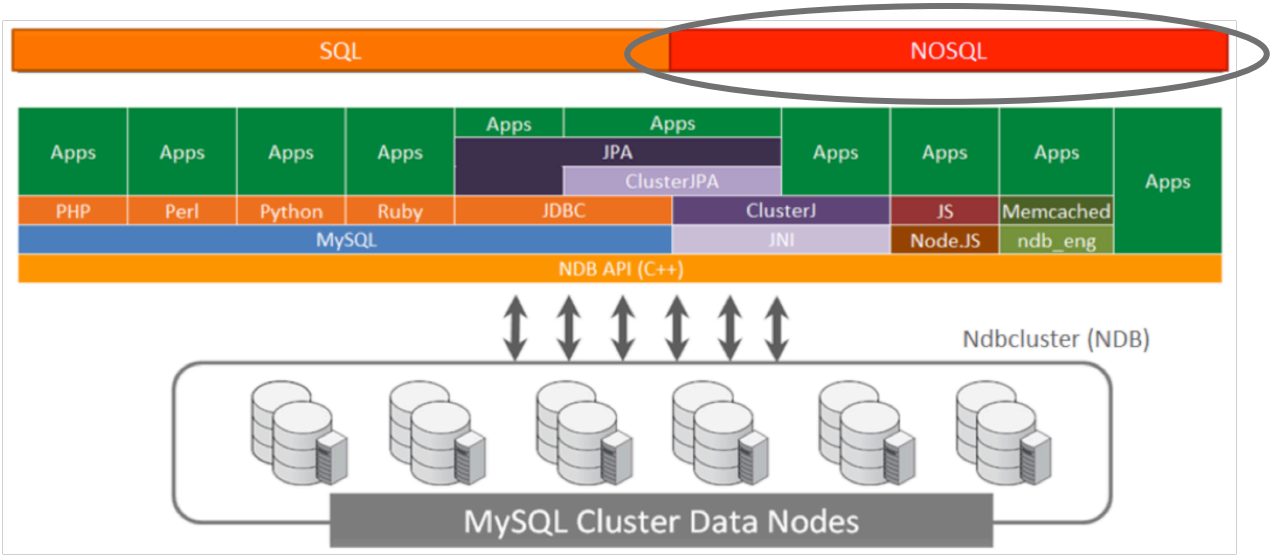

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# API Nodes

Communicate with external processes, supports many interfaces/protocols

- NDB APIを利用してアプリケーションを開発
  - C/C++/Java/Memcached/Node.js/JavaScript
- SQLをパースしない高速な処理

```
// Load the properties from the clusterj.properties file
File propsFile = new File("clusterj.properties");
InputStream inStream = new FileInputStream(propsFile);
Properties props = new Properties();
<SNIP>
session.currentTransaction().begin();
for (int i=700;i<800;i++) {
 newEmployee.setLast("NDB NoSQL Transaction"+i);
 newEmployee.setId(i+1000);
 newEmployee.setDepartment(i);
 session.persist(newEmployee);
 }
session.currentTransaction().commit();
```



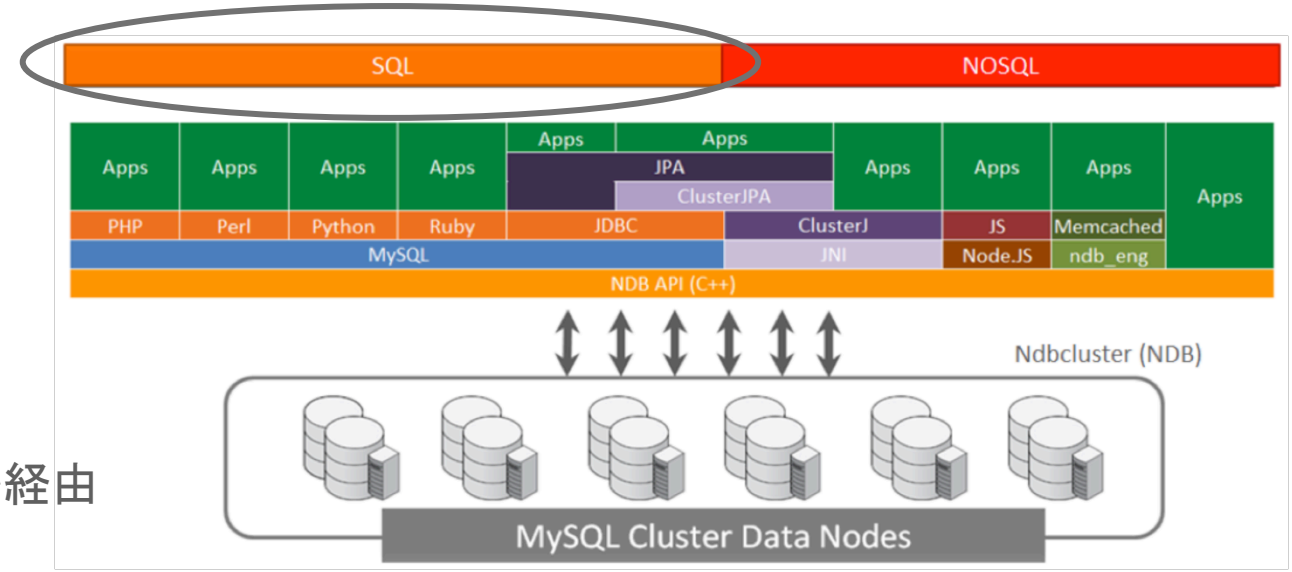

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# SQL Nodes

mysqld –ndb-connectstring=<mgt-node-host1>:<port1>[,<mgt-node-host2><port2> --ndbcluster]
--defaults-fileオプションを指定して、my.cnfから構成を読み込む。

```
[mysqld]
ndbcluster
ndb-connectstring="192.168.56.1:1186"
```

- MySQL はNDBClusterストレージエンジンを利用
  - API ノードの一つ
- 殆どのアプリケーションに対し透過的
- テーブル作成に利用
- Replicationに利用
  - 全ての変更をbinaryログに記録
- Arbitratorとしても設定可能
- 全てのデータノードに接続




NOTE:SQLを利用する場合も、全ての処理はNDB APIを経由
ndb-cluster-connection-poolオプションを調整

参照: http://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-program-options-mysqld.html

ORACLE
Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# MySQLから見たNDB(Storage Engine)

NDB API (C++)を経由してアクセスする事で,MySQLから独立して利用する事が可能です。
MySQLサーバにとって,NDB Clusterは行のテーブルを格納するためのストレージエンジンの一つです。
NDB Clusterの観点からは、MySQLサーバインスタンスはクラスタに接続されているAPIのプロセスの一つ。

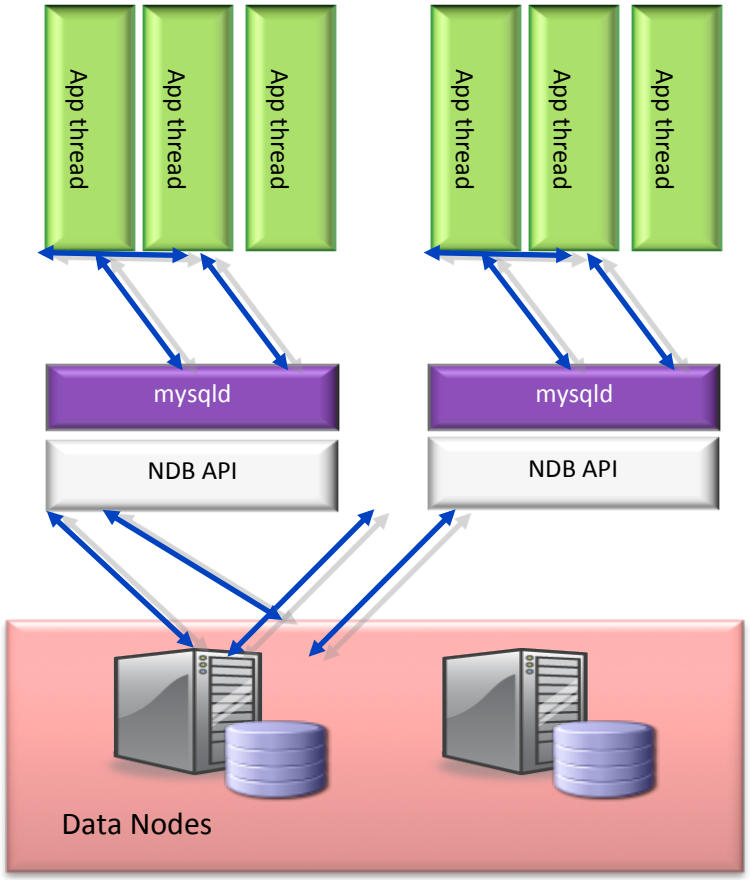
```
mysql> show engines;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| ndbcluster         | YES     | Clustered, fault-tolerant tables                               | YES          | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| ndbinfo            | YES     | MySQL Cluster system information storage engine                | NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
```

**NDB (ndbcluster)**
in-memory clustered storage engine called NDB (which stands for “Network DataBase”).

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# スループット最適化の為の並列処理




- ネットワークにおけるHOP数の増加はLatencyを増加
  - 並列に複数の処理を行う事でスループットを向上させる
- 各MySQL Serverにて複数クライアントコネクションを処理
- 複数のMySQLインスタンス
- MySQLサーバとデータ・ノード間の接続プーリング
  - My.cnfにてndb-cluster-connection-poolの設定を1以上に設定
  - config.iniに十分な[SQL], [api]セクションを確保しておく

NOTE:
ndb-cluster-connection-poolを 1 より大きい値に設定すると、mysqld プロセスはクラスタに対して複数の接続を使用し、複数の SQL ノードを模倣します。

このオプションは、複数の CPU、コア、またはその両方を搭載したホストマシンで mysqld を実行する場合にのみ有効です。

参照：https://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-program-options-mysqld.html

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# InnoDBとの違い

- デフォルトでは、インメモリデータベースである
  ⇒キャッシュ上のデータが有る/無しでの性能差異が無い
- スキャン系の処理は、複数台のデータ・ノードにアクセスする必要がある
- データがSQLノード(MySQL Server)ではなくデータ・ノードに存在するため、データアクセス時にネットワーク経由でのアクセスになる
  ⇒ネットワークの通信時間がレスポンスタイムに影響を与える
- 参照処理だけでなく、更新処理に対しても拡張性が出せる
- サポートされているトランザクションの分離レベルは「READ-COMMITTED」のみ(InnoDBは全ての分離レベルをサポートし、デフォルトは「REPEATABLE-READ」)
- InnoDBはMVCCに対応だが、NDBは対応では無い

詳細：http://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-ndb-innodb-engines.html

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

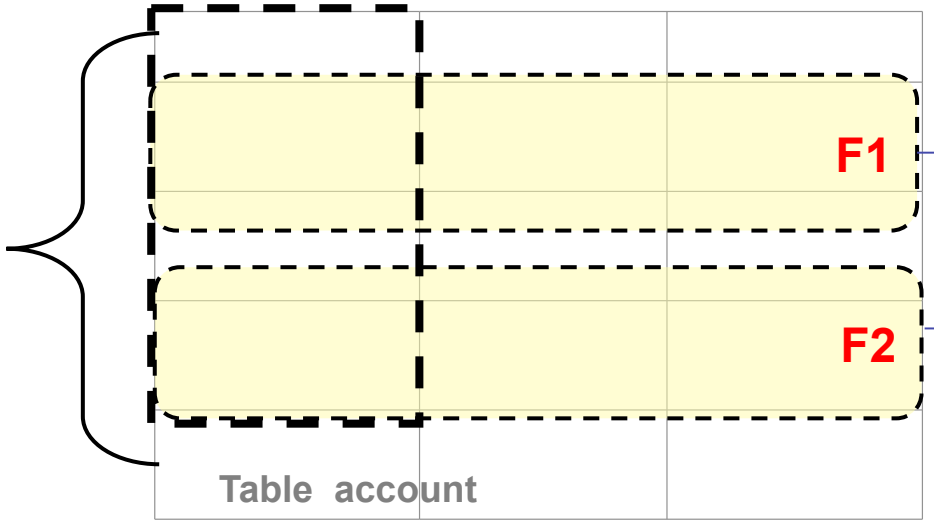
# データ・ノードでのデータ保持方法

・主キーのハッシュ値で水平分割して、各データノードにデータを保持
・レプリカを別のデータ・ノードと持ち合う

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

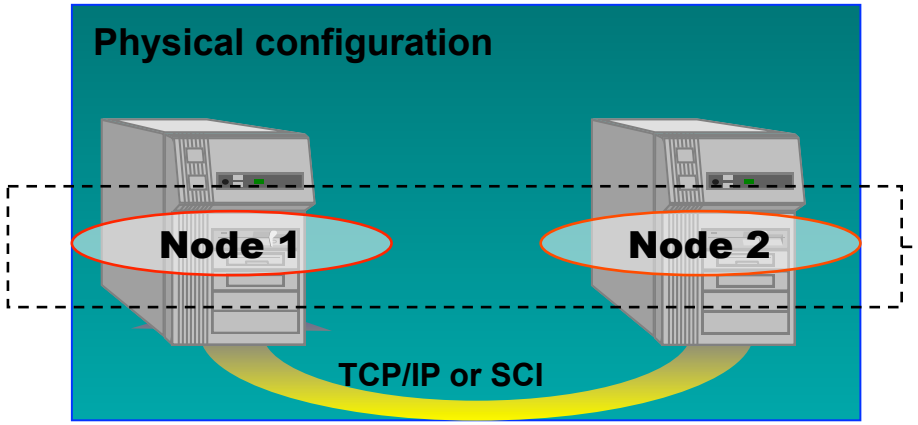
# データノード2台でのデータ保持方法

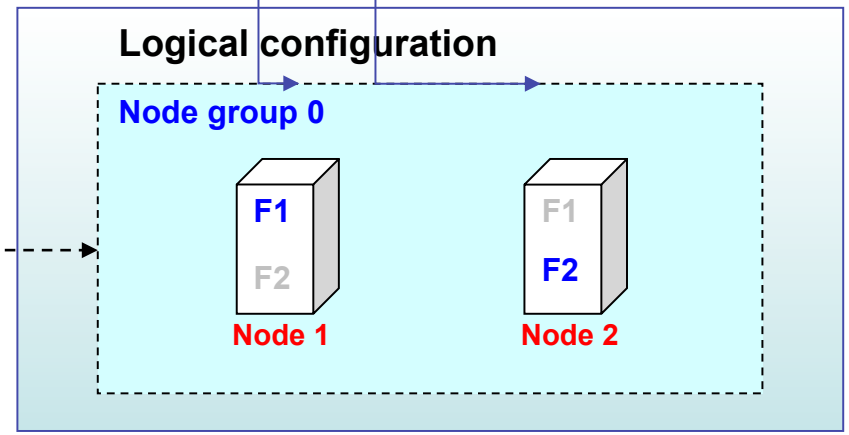
- Horizontal fragmentation of table t1 (2 fragments)
- Fragments distributed on nodes




2 copies of data

Fx – primary replica
Fx – secondary replica



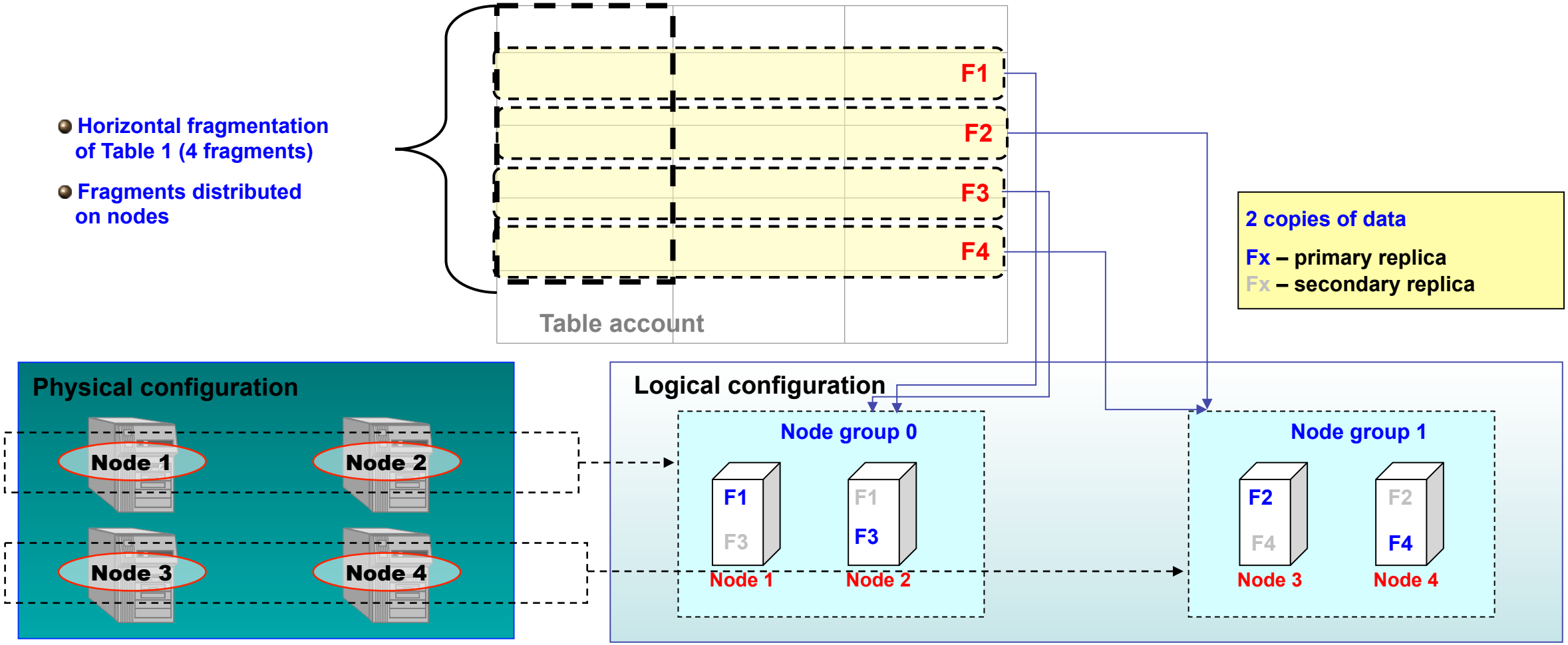




ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

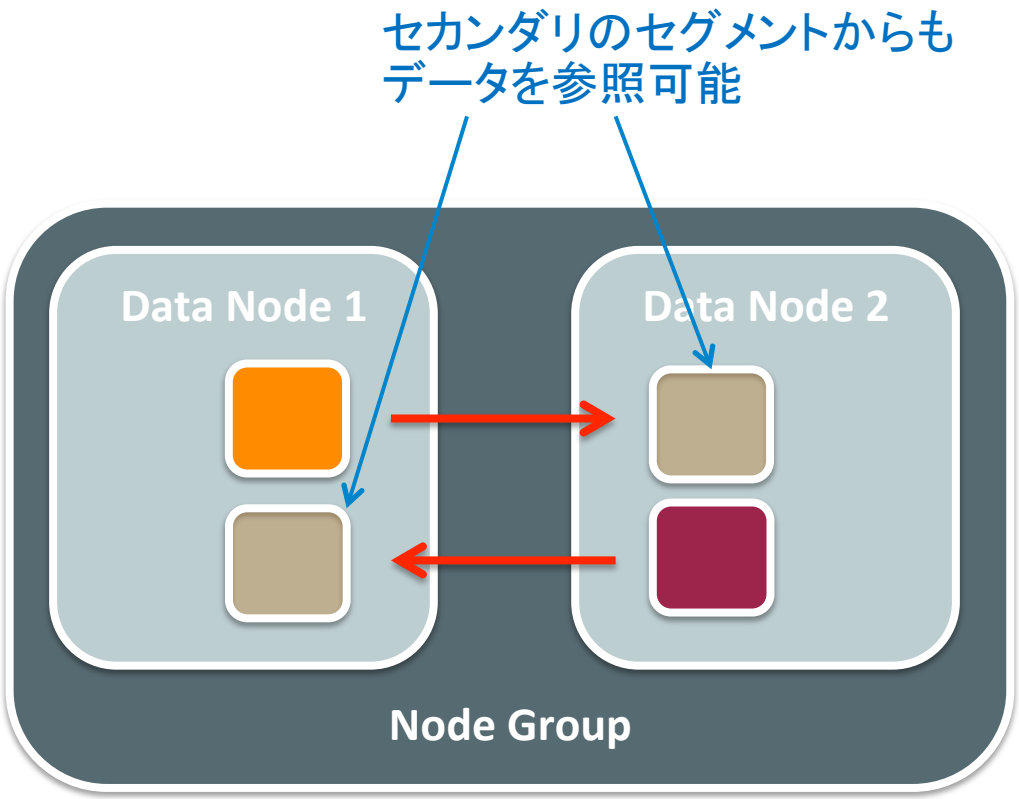
# データノード4台でのデータ保持方法

- Horizontal fragmentation of Table 1 (4 fragments)
- Fragments distributed on nodes




Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

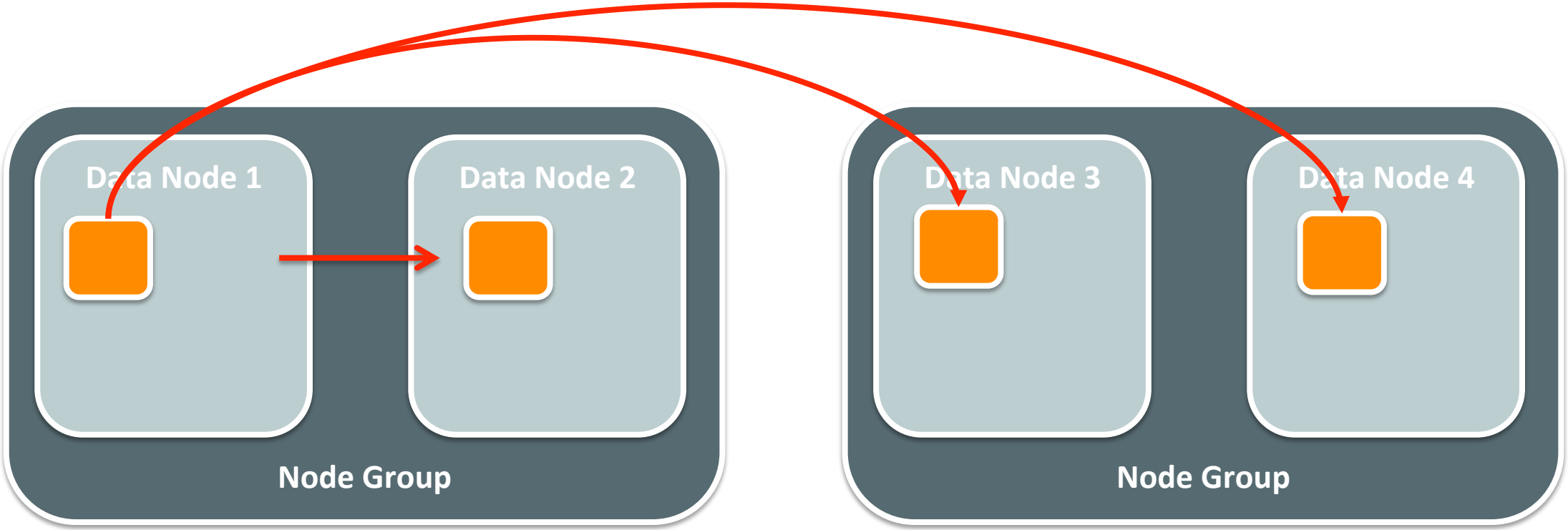
# Reading from backup (ndb_read_backup, READ_BACKUP)




- バックアップからの読み取りは、どのコピーからもデータの読み取りを可能にします。
- 7.4までは、全ての参照処理は、プライマリーフラグメントのみ参照。

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Fully replicated (FULLY_REPLICATED)




- Fully replicatedは、対象テーブルに対して、どのデータノードにおいても、ローカルでデータの読み書きが出来るようになります。
- 静的データに最適で、より高速なJOIN処理を実現可能。

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# MySQL Clusterにおけるインデックス

- Primary Key:
  - データはPKのハッシュ値に基づいて、データノードに分散されます
  - Ordered Indexは、各データノードに作成されます。
- Unique Index:
  - データノード上のHidden translation table
  - 各データノード上のOrdered Index
- Ordered Index:
  - Ordered indexは、全てのデータノードに分散されます
  - 全てのデータノードでリクエストを処理します

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# インデックスとハッシュ値

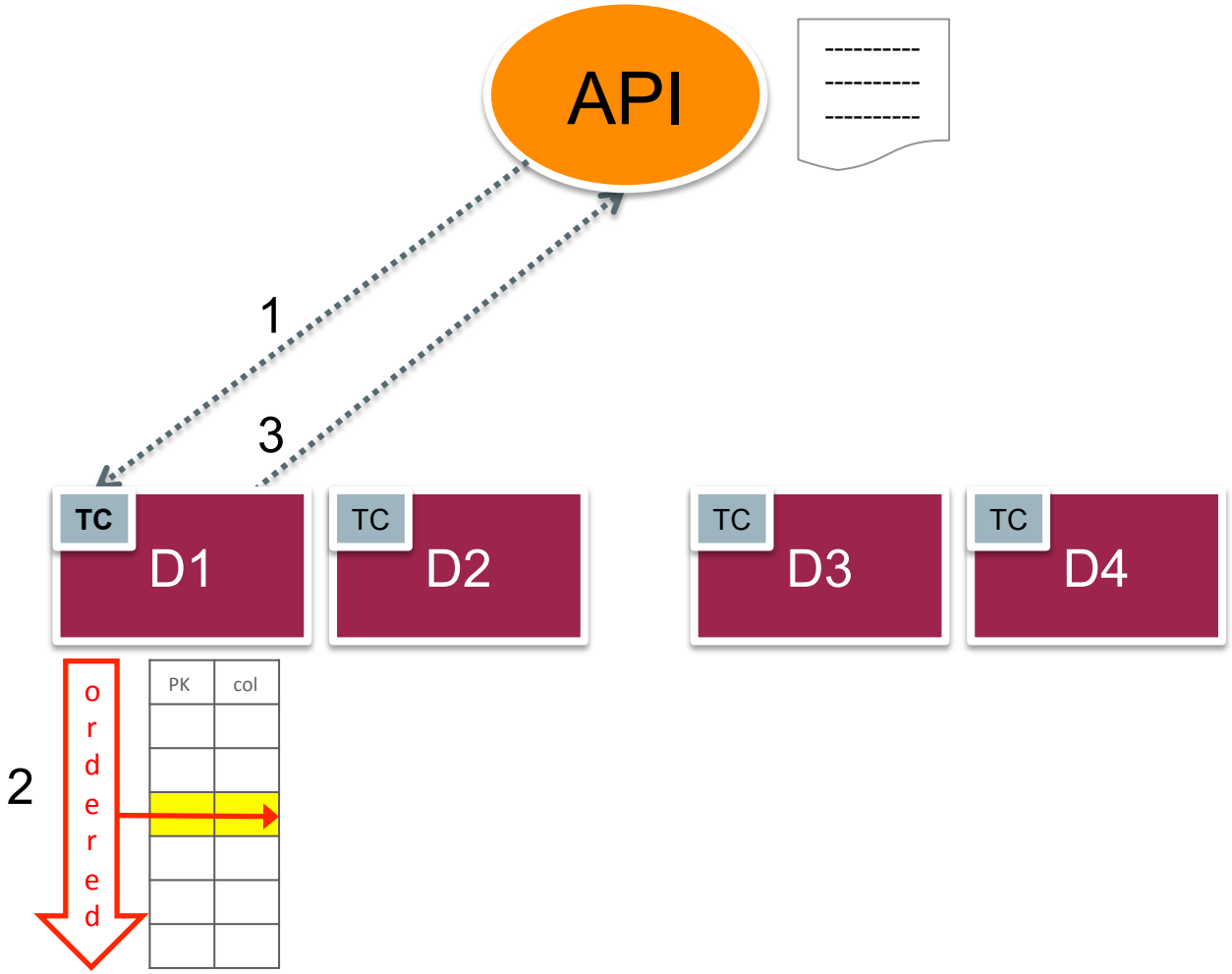
```
mysql> show create table City\G
*************************** 1. row ***************************
       Table: City
Create Table: CREATE TABLE `City` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` char(35) NOT NULL DEFAULT '',
  `CountryCode` char(3) NOT NULL DEFAULT '',
  `District` char(20) NOT NULL DEFAULT '',
  `Population` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  KEY `CountryCode` (`CountryCode`)
) ENGINE=ndbcluster AUTO_INCREMENT=4080 DEFAULT CHARSET=latin1
1 row in set (0.01 sec)

mysql>
```

Primary Keyは、Hash IndexとOrdered Indexとして利用されます

CountryCodeは、Ordered Indexとして利用されます。

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# データアクセス: Primary Key lookup




1.APIは、(ex: MySQL) 正しいデータノードを見付ける為に、透過的に行のPKをハッシュします。

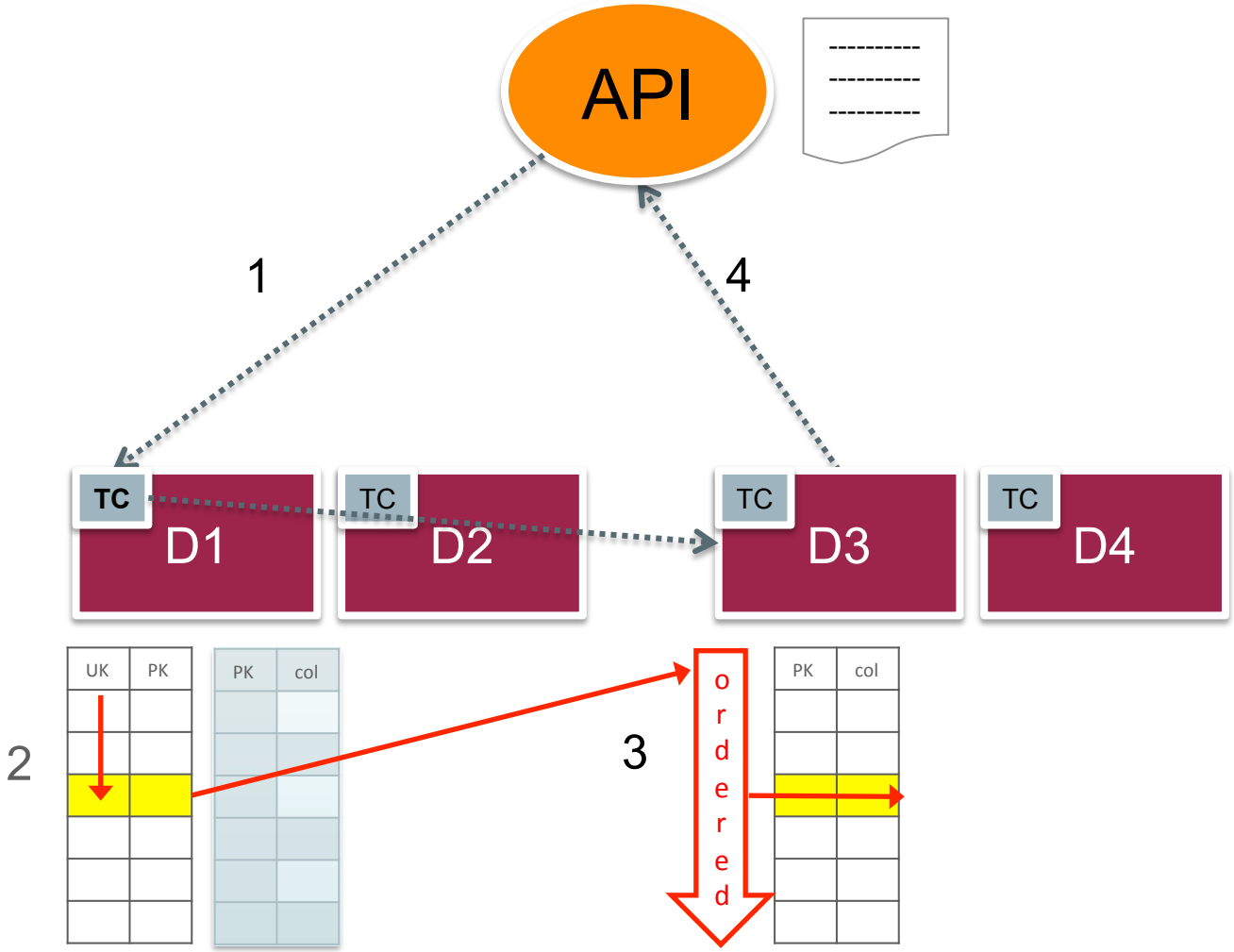
2.行を参照する為に、データノード上のOrdered Indexを参照します。

3. 対象行データをAPIに返します

補足:
※ 高速且つ効率的でスケーラブル
※ PKには、ハッシュインデックスとオーダードインデックスの両方が作成されます。これにより、高速なルックアップと範囲検索の両方が可能になっています。オーダードインデックスは必要がなければテーブル定義においてUSING HASHオプションを使います。
但し、等価比較のみ利用出来ます。

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# データアクセス: Unique key lookup




1. Uniqueなハッシュ値をベースにどのデータノードがPKハッシュ値を保持しているか判断

2. PKハッシュ値を参照してどのデータノードが対象行を保持しているか判断します。

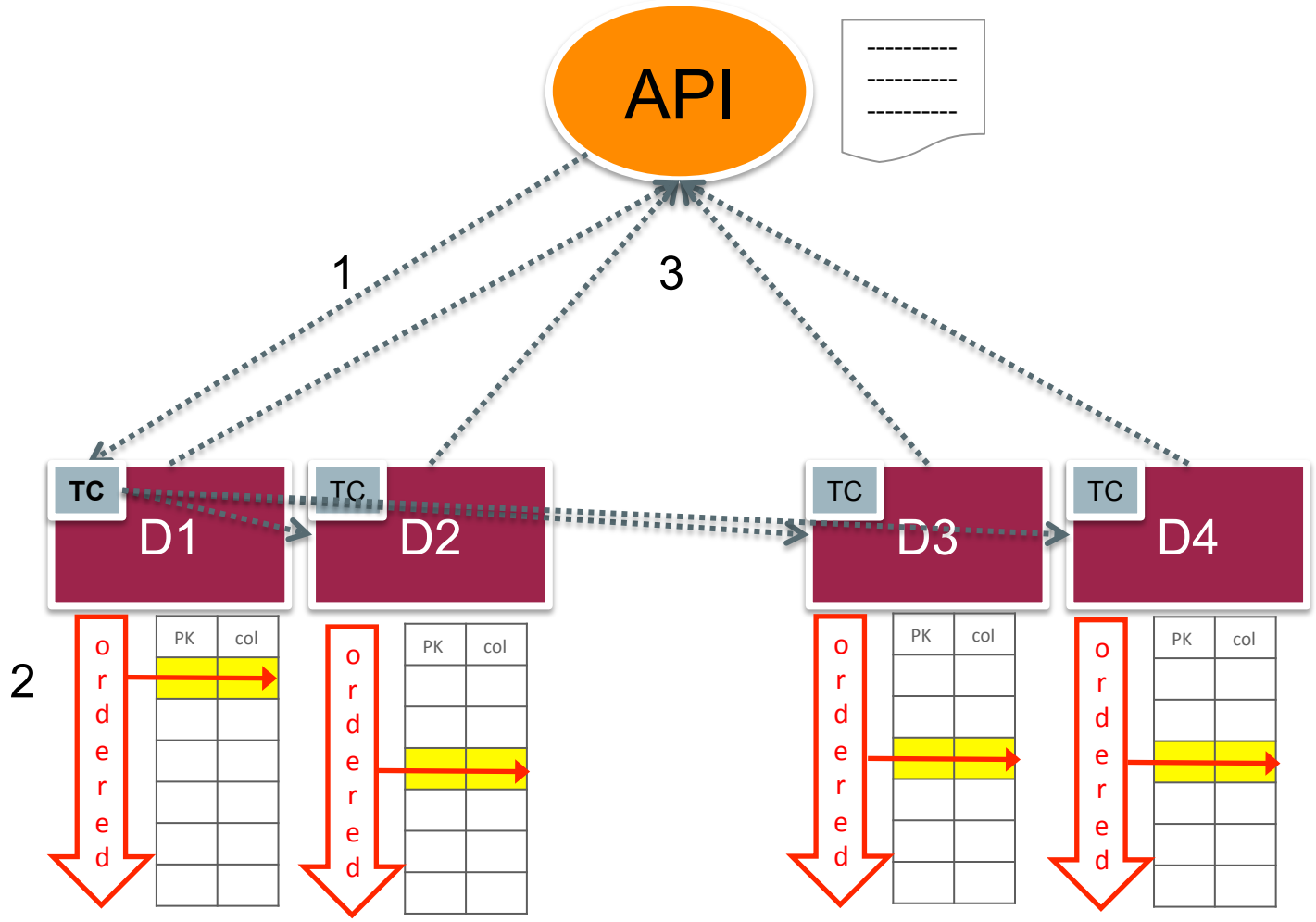
3.行を参照する為に、データノード上のOrdered Indexを参照します。

4.対象行データをAPIに返します

補足:
※Secondary keysは,hidden/systemテーブルとして実装
※ Hiddenテーブルは、新たなセカンダリーキーをPKと して保持し、
　ベーステーブルのPKを値として保持。
※データは、同一ノード若しくは他のノード
※ネットワークをまたぐ事による、若干Latencyが増加

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# データアクセス: Ordered Index




1. 参照リクエストをランダムに選定されたノードに送信

2. 全てのデータノードは行を参照する為に、データノード上のOrdered Indexを参照します

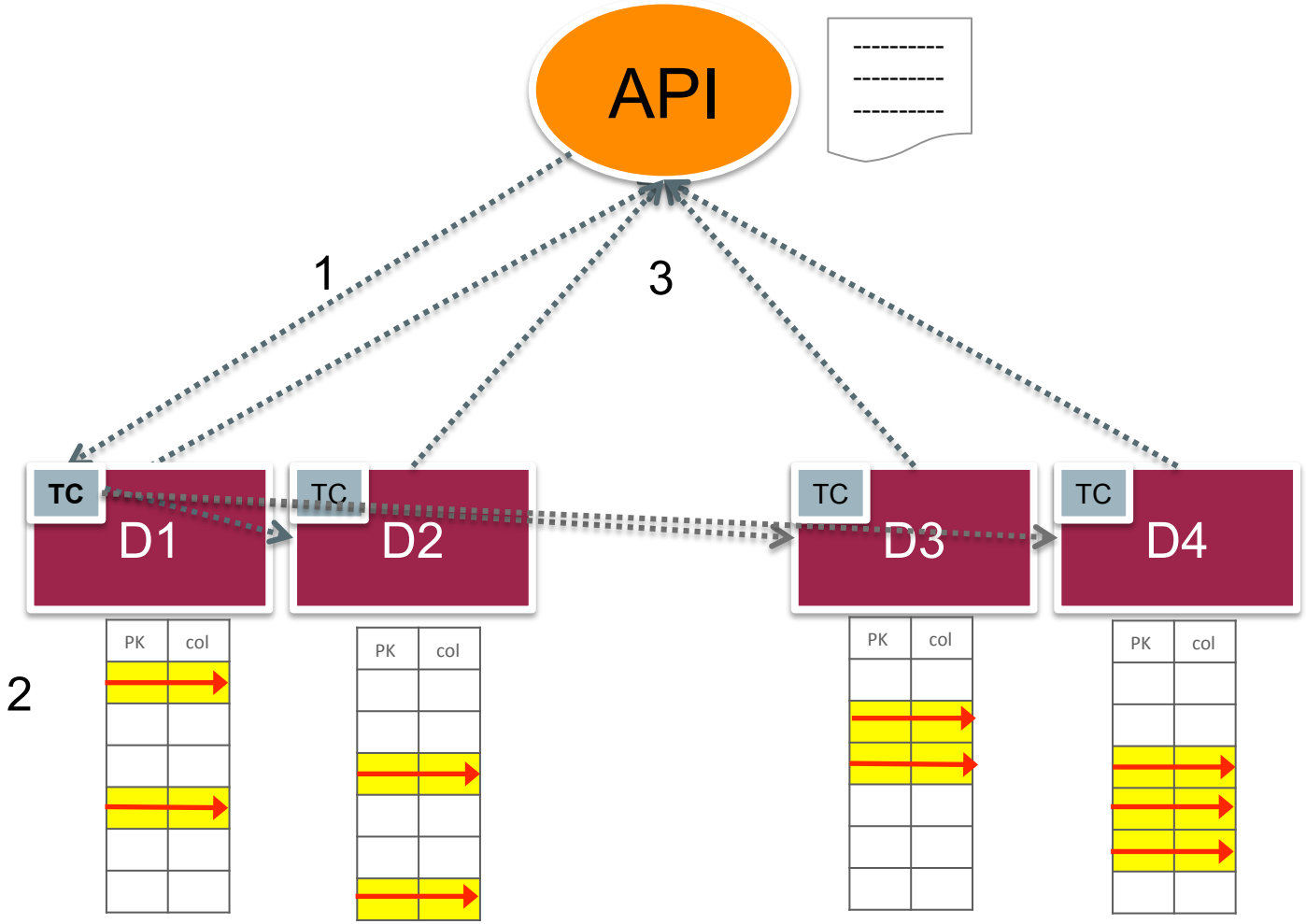
3.対象行データをAPIに返します

- データはアプリケーションに返される前にマージされます

補足:
※分散型スキャンを実行する為にランダムにノードを選定
※データノードは、直接APIにデータを返す
※データはアプリケーションに返される前にマージされる。

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# データアクセス: Condition Pushdown (No Indexes Used)




1. WHERE句を含む参照リクエストをランダムに選定されたノードに送信

2. WHERE句の要素を各データノードでフィルターします。

3.対象行データをAPIに返します

- データはアプリケーションに返される前にマージされます

補足:
engine_condition_pushdown

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Engine Condition Pushdown

インデックスが設定されていないカラムと定数との直接比較の効率性を向上。クラスタのデータノードとクエリーを発行した MySQL Server 間で、ネットワーク経由で一致しない行を送る必要性をなくすことが出来、5 - 10 倍高速化可能です。

mysql> SET optimizer_switch='engine_condition_pushdown=OFF';

```
mysql> EXPLAIN SELECT id,comment FROM
    -> T_NDB01 WHERE id < 3\G
*************** 1. row *************
           id: 1
  select_type: SIMPLE
        table: T_NDB01
         type: range
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: NULL
         rows: 3
        Extra: Using where; Using MRR
1 row in set (0.00 sec)
```

mysql> SET optimizer_switch='engine_condition_pushdown=ON';

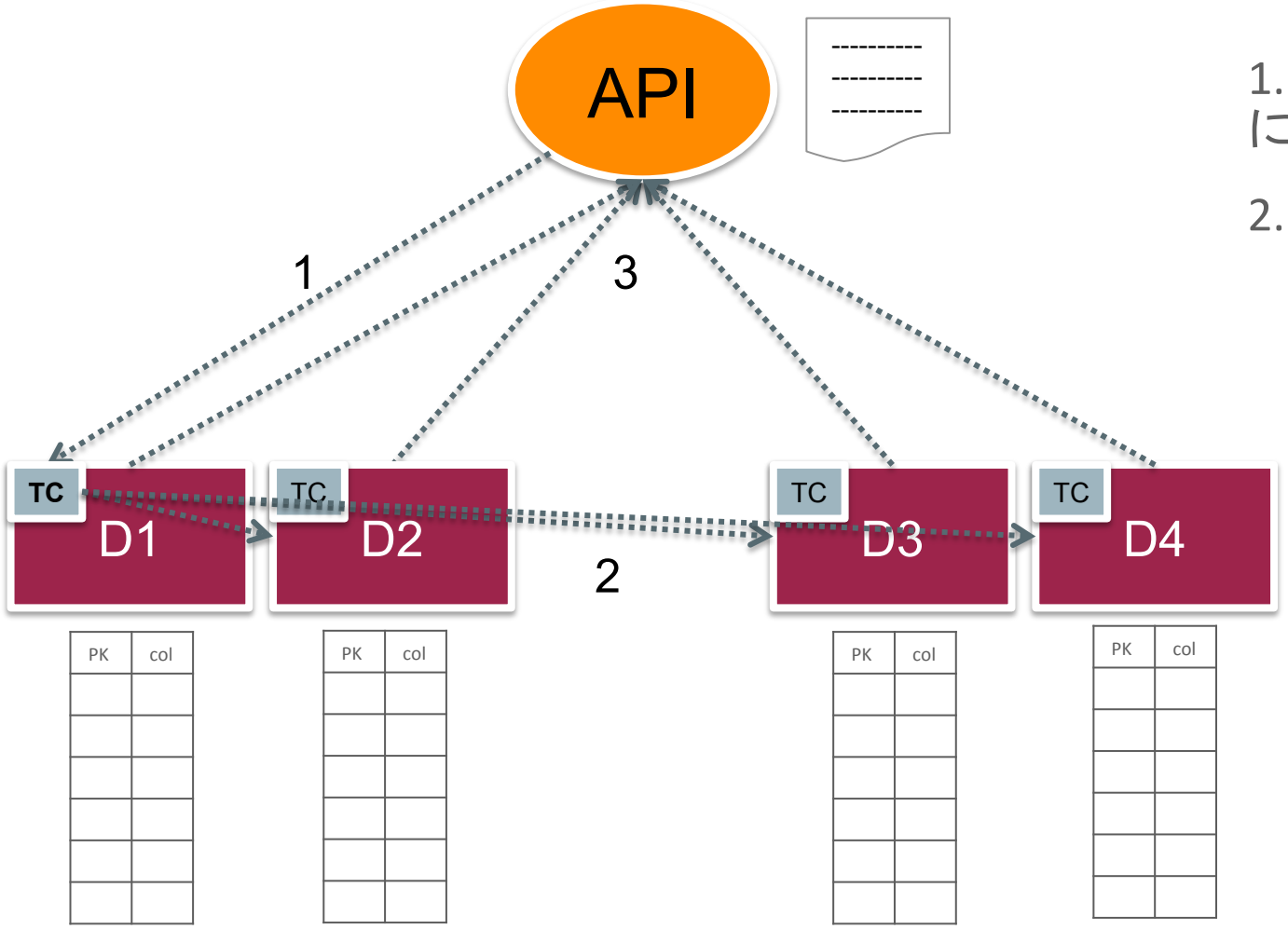
```
mysql> EXPLAIN SELECT id,comment FROM
    -> T_NDB01 WHERE id < 3\G
*************** 1. row *************
           id: 1
  select_type: SIMPLE
        table: T_NDB01
         type: range
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: NULL
         rows: 3
        Extra: Using where with pushed condition; Using MRR
1 row in set (0.00 sec)
```

参照： https://dev.mysql.com/doc/refman/5.6/ja/condition-pushdown-optimization.html

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# データアクセス: Table scan




1. 分散型スキャンを実行する為にランダムにノードを選定

2.データノードは、APIにデータを返す

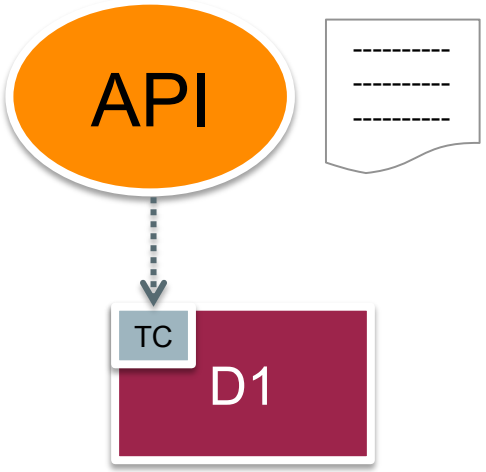
- データはアプリケーションに返される前にマージされます
- Whereで絞り込まない場合は、全てのデータノードからデータを返す為、シングルサーバーで参照するよりも高速にデータ処理出来る事も有る

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Transaction Coordinatorの選定

## SQL ノードがトランザクションコーディネータとして使用するためにクラスタデータノードを選択する方法を決定

- SQL ノードは、近接性を使用してトランザクションコーディネータを指定します。
- つまり、その SQL ノードから「もっとも近い」データノードがトランザクションコーディネータとして選択されます。
- この指定では、SQL ノードとの共有メモリー接続を持つデータノードがその SQL ノードから「最も近い」とみなされます。
- 次に近いのは、(近接性が低くなる順に) localhost との TCP 接続、SCI 接続、localhost以外のホストからのTCP接続です。
- SQL スレッドは、分布の認識を使用してデータノードを選択します。
- つまり、特定のトランザクションの最初のステートメントによってアクセスされるクラスタパーティションを収容するデータノードが、トランザクション全体のトランザクションコーディネータとして使用されます。




参照: https://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-system-variables.html#sysvar_ndb_optimized_node_selection

ORACLE
Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Transaction Coordinatorの選定

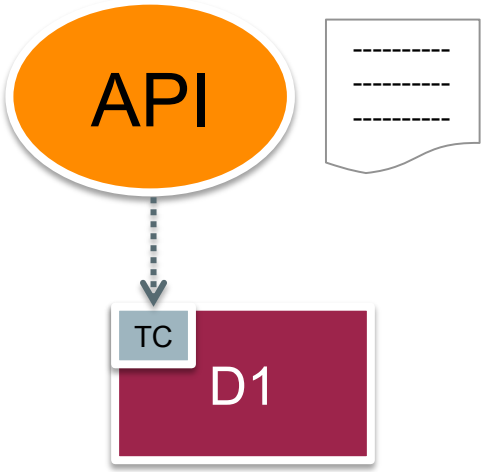
## パラメータ: ndb_optimized_node_selection

0: ノード選択は最適化されません。SQL スレッドが次のデータノードに進む前に、
各データノードがトランザクションコーディネータとして8回ずつ採用されます

1: SQLノードへの近接性を使用してトランザクションコーディネータが指定されます

2: 分布の認識を使用してトランザクションコーディネータが選択されます。
ただし、トランザクションの最初のステートメントが
複数のクラスタパーティションにアクセスする場合は、
SQLノードはこのオプションを 0 と同様 (ラウンドロビン)

3: トランザクションコーディネータを指定するために
分布の認識を使用できる場合は、それが使用されます。
そうでない場合は、近接性を使用してトランザクションコーディネータが
選択されます。(Default)




参照: https://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-system-variables.html#sysvar_ndb_optimized_node_selection

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 補足：Distribution Aware Apps

```
SELECT SUM(population) FROM towns WHERE country="UK";
```

Partition Key

Primary Key

| town | country | population |
|---|---|---|
| Maidenhead | UK | 78000 |
| Paris | France | 2193031 |
| Boston | UK | 58124 |
| Boston | USA | 617594 |

```
SELECT SUM(population) FROM towns WHERE town="Boston";
```

Partition Key

Primary Key

| town | country | population |
|---|---|---|
| Maidenhead | UK | 78000 |
| Paris | France | 2193031 |
| Boston | UK | 58124 |
| Boston | USA | 617594 |

- デフォルトでは、データは各ノードに主キーのハッシュ値に基づいて分散されるため、ランダムに分散される
- パーティショニングテーブルを使ってシャーディングキーを変更でき、特定のキーのデータを同じデータノードに格納可能
  - トランザクションで必要となるデータが同じパーティション（同じデータノード）にある場合、ノード間の通信量が少なくなる
    ⇒　高速になる
- 頻繁に実行されるクエリーでアクセスする行を同じパーティションに格納することで高速化できる可能性がある

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 補足：Distribution Aware Apps – 複数テーブル

Partition Key

Primary Key

| sub_id | age | gender |
|---|---|---|
| 19724 | 25 | male |
| 84539 | 43 | female |
| 19724 | 16 | female |
| 74574 | 21 | female |

Partition Key

Primary Key

| service | sub_id | svc_id |
|---|---|---|
| twitter | 19724 | 76325732 |
| twitter | 84539 | 67324782 |
| facebook | 19724 | 83753984 |
| facebook | 73642 | 87324793 |

- 複数テーブルを意識したパーティション設計
- 同じ情報を持つ列でパーティショニングすることで、複数テーブルであっても同じ属性のデータを同じパーティション（同じデータノード）に格納できる
- 例）

```
ALTER TABLE service_ids PARTITION BY
KEY(sub_id);
```

- 特定のデータノードに負荷が集中してボトルネックが発生する可能性に注意

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Distribution Aware – 確認 (ユーザー定義Partition設定前)

```
mysql> select * from CountryLanguage where CountryCode = 'JPN';
+-------------+----------------------+------------+------------+
| CountryCode | Language             | IsOfficial | Percentage |
+-------------+----------------------+------------+------------+
| JPN         | Chinese              | F          |        0.2 |
| JPN         | Japanese             | T          |       99.1 |
| JPN         | Korean               | F          |        0.5 |
| JPN         | Philippene Languages | F          |        0.1 |
| JPN         | Ainu                 | F          |        0.0 |
| JPN         | English              | F          |        0.1 |
+-------------+----------------------+------------+------------+

mysql> SHOW GLOBAL STATUS LIKE 'ndb_pruned_scan_count';
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| Ndb_pruned_scan_count | 0     |
+----------------------+-------+
```

データがPrunedされていない為、カウントは0のまま

2つのパーティションが読み取られているのが分かる

```
mysql> explain partitions select * from CountryLanguage where CountryCode = 'JPN';
+----+-------------+-----------------+------------+------+---------------------+-------------+---------+-------+------+----------------------------------+
| id | select_type | table           | partitions | type | possible_keys       | key         | key_len | ref   | rows | Extra                            |
+----+-------------+-----------------+------------+------+---------------------+-------------+---------+-------+------+----------------------------------+
|  1 | SIMPLE      | CountryLanguage | p0,p1      | ref  | PRIMARY,CountryCode | CountryCode | 3       | const |    3 | Using where with pushed condition |
+----+-------------+-----------------+------------+------+---------------------+-------------+---------+-------+------+----------------------------------+
```

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Distribution Aware – 確認 (ユーザー定義Partition設定後)

```
mysql> alter table CountryLanguage partition by key(CountryCode);
Query OK, 984 rows affected (0.41 sec)
Records: 984  Duplicates: 0  Warnings: 0

mysql> explain partitions select * from CountryLanguage where CountryCode = 'JPN';
+----+-------------+----------------+------------+------+--------------------+-------------+---------+-------+------+----------------------------------+
| id | select_type | table          | partitions | type | possible_keys      | key         | key_len | ref   | rows | Extra                            |
+----+-------------+----------------+------------+------+--------------------+-------------+---------+-------+------+----------------------------------+
|  1 | SIMPLE      | CountryLanguage | p0         | ref  | PRIMARY,CountryCode | CountryCode | 3       | const |    3 | Using where with pushed condition |
+----+-------------+----------------+------------+------+--------------------+-------------+---------+-------+------+----------------------------------+
```

```
mysql> select * from CountryLanguage where CountryCode = 'JPN';
+-------------+----------------------+------------+------------+
| CountryCode | Language             | IsOfficial | Percentage |
+-------------+----------------------+------------+------------+
| JPN         | Japanese             | T          |       99.1 |
| JPN         | Philippene Languages | F          |        0.1 |
| JPN         | Ainu                 | F          |        0.0 |
| JPN         | Korean               | F          |        0.5 |
| JPN         | Chinese              | F          |        0.2 |
| JPN         | English              | F          |        0.1 |
+-------------+----------------------+------------+------------+

mysql> SHOW GLOBAL STATUS LIKE 'ndb_pruned_scan_count';
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| Ndb_pruned_scan_count | 1     |
+----------------------+-------+
1 row in set (0.00 sec)
```

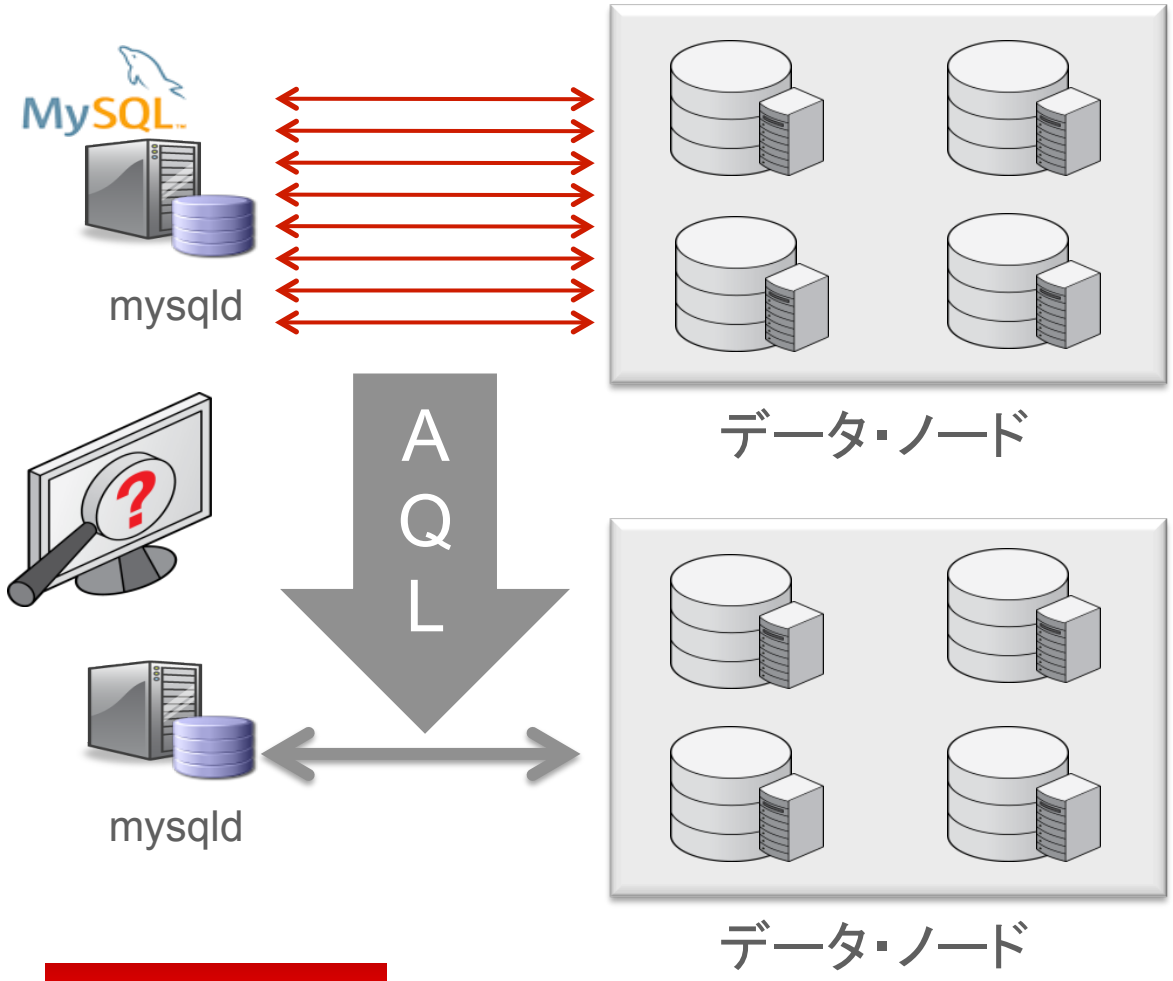
1つのパーティションに絞り込まれ、データが読み取られている

Prunedが増加していて、データのフィルタリング状況が確認出来る

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 補足：アダプティブ・クエリー・ローカライゼーション(AQL)

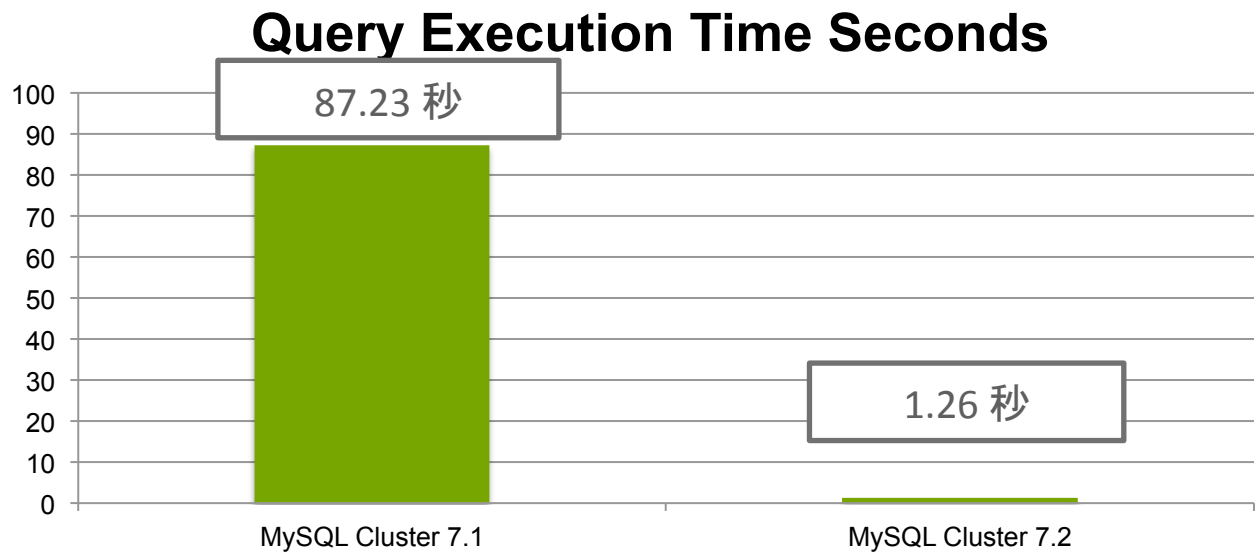



- シャード間で複雑なクエリを実行
  - JOIN処理をデータ・ノードに移行
  - 並列実行
  - 一つの結果セットをMySQLに戻す
- これまでは性能的に難しかった処理も実行可能に
  - リアルタイム分析
  - レコメンデーション・エンジン
  - クリックストリームを分析

※ SQLとデータノード間の通信は減少しますが、データノード間の通信量は増加します。

ORACLE
Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 補足：MySQL Cluster 7.2～： JOIN性能の向上




例) 70倍以上の
パフォーマンス向上

- Webベース・コンテンツ管理システム
  - 11テーブル、33,500行を結合(JOIN)
  - 結果セット：2,060行、1行あたり19列を返す

```
mysql> explain select a.id,b.memo from T_NDB01 a join T_TABLE01 b on a.id = b.id;
+----+-------------+-------+--------+---------------+---------+---------+--------------+------+-----------------------------+
| id | select_type | table | type   | possible_keys | key     | key_len | ref          | rows | Extra                       |
+----+-------------+-------+--------+---------------+---------+---------+--------------+------+-----------------------------+
|  1 | SIMPLE      | b     | ALL    | PRIMARY       | NULL    | NULL    | NULL         |   10 | Parent of 2 pushed join@1   |
|  1 | SIMPLE      | a     | eq_ref | PRIMARY       | PRIMARY | 4       | TEST_DB.b.id |    1 | Child of 'b' in pushed join@1 |
+----+-------------+-------+--------+---------------+---------+---------+--------------+------+-----------------------------+
```

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 補足：AQL – 使用方法

- **ndb_join_pushdown** がONで有効（デフォルト）
- 結合（JOIN)を移行するためのルール:
  1. 結合する列のタイプは全て完全に同じ
  2. BLOBまたはTEXT列への参照なし
  3. 明示的ロックなし
  4. 結合内の子テーブルは**ref**、**eq_ref**、または**const**を使用してアクセス
  5. テーブルは**[LINEAR] HASH**、**LIST**、または **RANGE**を使用して明示的にパーティショニングされていない
  6. クエリー・プランが ‘**Using join buffer**’ を選択していない
  7. 結合のルートが **eq_ref** または **const** の場合、子テーブルは **eq_ref**で結合
- 各テーブルに対し **ANALYZE TABLE <tab-name>** を実行
- **EXPLAIN** を使用してどのコンポーネントが移行されるか確認:
  - **Extra: Child of 'd' in pushed join@1**

参照: https://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-system-variables.html#sysvar_ndb_join_pushdown

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# チューニングのポイント

- 出来る限り、主キーベース、ユニークキーベースの処理にする
- トランザクションを分割できる場合は、細かく分割する
- 必要に応じて、ユーザー定義パーティショニング(Distribution Aware)も活用
- JOINの処理は、AQLが使える場合はAQLで処理する
- スキャン系の処理が多発する場合は、InnoDBへのレプリケーションも検討
- NoSQL APIの使用を検討する
- パラメータチューニングを実施する
  - 参考資料： MySQL Cluster でも フラッシュドライブを活用してみる
    http://h50146.www5.hp.com/services/ci/opensource/pdfs/HP_OpenServices.pdf

NOTE: AQLは、SPJとも呼ばれています。

ORACLE

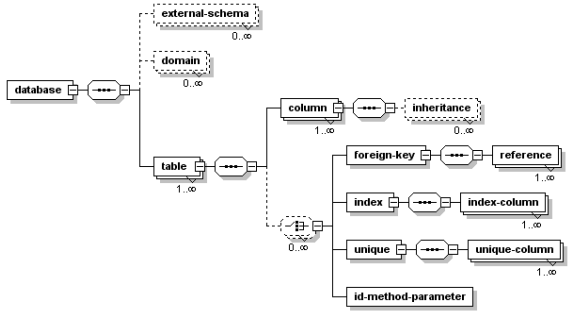
Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# オンライン・メンテナンサビリティ

## メンテナンスによるサービス停止と機会損失を可能な限り削減

**Scale**

ノード追加 & 削除




**Evolve**

テーブル再分割& スキーマ変更

**Backup**

バックアップ

**Upgrade**

MySQL Cluster & OS

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Agenda

1. MySQL Cluster概要
2. システムデザイン（構成とサイジング）
3. 構成パラメータと構成変更 (--reload, --init)
4. バックアップとリカバリー
5. スケールアウト(データノード追加とデータ再配置)
6. レプリケーション(NDB -> NDB,  NDB -> InnoDB )
7. その他、参考情報

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 推奨される初期構成

MySQL Cluster で最良のパフォーマンスを実現できるかどうかは、
次を含むいくつかの要因に依存します。

1. MySQL Cluster ソフトウェアのバージョン (機能改善・バグFIX等)
2. データノードおよび SQL ノードの数 (必要に応じてPOCを実施)
3. ハードウェア (CPU, MEMORY, DISK, NETWORK)
4. オペレーティングシステム (OSバージョン、設定等)
5. 格納されるデータの量 (グループ分散等を考慮)
6. クラスタ稼働時の負荷のサイズとタイプ (PK, UKベースの参照)

参照: http://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-config-starting.html

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# ハードの見積もり

| | |
|---|---|
| OS: | サポートされているプラットフォームの最新リスト 参照 |
| CPU: | Intel/AMD x86, UltraSPARC |
| Memory: | Minimum 1GB RAM |
| HDD: | 3GB |
| Network: | 1+ nodes (Gigabit Ethernet - TCP/IP) |

SQLノード　物理・仮想環境 (既存システムを確認し調整)
データノード　仮想環境も可能だが物理が望ましい
管理ノード　スペックは低くて良い。SQLノードと共存可能

MySQL Node (+Management Node) x 2 Servers
Data Node (1 Group) x 2 Servers (xxx GB Memory)

※ メモリーが256GBの場合は、600GBのデータ処理する為に(3Group)6サーバーが必要。
※ ディスクサイズはメモリーの7～10倍程度必要。512GBだとすると各ノードで概算5TBとなる。
各データノードを256GBの設定にした場合は、2.5TB程度のディスク容量が必要となります。
※ SQLサーバーに関しては、ディスクよりもCPUやメモリー処理が重要になります。
※ Networkに関しては、GB以上を推奨しております。全ての処理がネットワークを経由しますのでLatencyを考慮
※クラスタ管理ノードのデフォルトポートは 1186 です。データノードのデフォルトポートは 2202 です。

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# データノードにおけるメモリー
## データメモリー、インデックスメモリー、および文字列メモリー

DataMemory と IndexMemory は、実際のレコードとインデックスの格納に使用されるメモリーセグメントのサイズを指定する [ndbd] パラメータ

**DataMemory**

DataMemory によって割り当てられたメモリーは、実際のレコードとインデックスの両方を格納するために使用されます。最大レコードサイズは 14000 バイトです。DataMemory によって定義されたメモリースペースは、順序付けされたインデックスを格納するためにも使用されます。

**IndexMemory**

このパラメータは、MySQL Cluster 内のハッシュインデックスに使用されるストレージの量を制御します。ハッシュインデックスは、常に主キーのインデックス、一意のインデックス、および一意 の制約に使用されます。主キーまたは一意のインデックスを定義すると、2つのインデックスが作成され、そのうちの1つのハッシュインデックスが、すべてのタプルへのアクセスとロックの処理に使用されます。
このインデックスは、一意の制約を適用するためにも使用されます。

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 考慮事項:データノードにおけるメモリー

IndexMemory と DataMemory の両方で重要な点は、各ノードグループのすべてのデータメモリーと
インデックスメモリーの合計がデータベースの合計サイズになることです。各ノードグルー プはレプリケートされた情
報の格納に使用されるため、2 つのレプリカを含むノードが 4 つある場合は、2 つのノードグループが作成されます。

DataMemory設定値 X ノードグループ数 = クラスター全体で利用可能なDataMemory量
IndexMemory設定値X ノードグループ数 = クラスター全体で利用可能なIndexMemory量

すべてのノードのDataMemory を同じ値に設定することを強くお勧めします。(IndexMemory も同様)
データの分布はクラスタ内のすべてのノードで均等であるため、各ノードで使用できるスペースの最大量は
クラスタ内でもっとも小さいノードの最大量と同じになります。

DataMemory と IndexMemory は変更できますが、これらのいずれかを減らすのは危険です。
その場合、ノードまたは MySQL Cluster 全体がメモリースペースの不足によって再起動できなくなる可能性が高くなり
ます。これらの値を増やすことは問題ありませんが、このようなアップグレードは ソフトウェアのアップグレードと同じ
方法で行うことをお勧めします。つまり、最初に構成ファイルを更新し、次に管理サーバーを起動して、
各データノードを 順に再起動します。

NOTE: MySQL Clusterはデータとインデックスをメモリーにロードします。MySQLでディスク容量を考慮するのと
同じようにメモリー容量をサイジングして下さい。共に不足すれば、処理がエラーになります。

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# インデックスによるメモリーの利用

- Index Memory:
  - Primary Key hash for each row.
  - Unique Index hash for each row.
  - Unique Constraints.
- Data Memory:
  - Every table columns.
  - Hidden Unique Index table: PK value and UK value.
  - Ordered Indexes.

Enterprise Monitorを利用してメモリー不足にならないように監視をお勧めします

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 参考) ndb_size.pl

18.4.25 ndb_size.pl — NDBCLUSTER Size Requirement Estimator
http://dev.mysql.com/doc/refman/5.6/en/mysql-cluster-programs-ndb-size-pl.html

サイジングツールの1つですが、5.1以降アップデートしておりません。参考指標の一つとしてのみご利用下さい。

```
[admin@GA01 ~]$ ./ndb_size.pl --user=root --password=password --database=world --socket=/tmp/
mysql.sock > ndb_size_world.txt
<SNIP>
Parameter Minimum Requirements
-----------------------------
* indicates greater than default

            Parameter        Default          4.1          5.0          5.1
        DataMemory (KB)        81920        19328        19328        20384
     NoOfOrderedIndexes          128           12           12           12
            NoOfTables          128           12           12           12
       IndexMemory (KB)        18432         7136         3904         3904
   NoOfUniqueHashIndexes           64            3            3            3
        NoOfAttributes         1000           69           69           69
          NoOfTriggers          768           69           69           69
```

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# データの永続化

**LCP（Local CheckPoint）** 一時点でのデータ全体
LCPが実行されると、各データノード毎にメモリ上にあるすべてのデータがディスクに書き込まれます。
デフォルト設定では4Mバイトのデータ更新毎

**GCP（Global CheckPoint）** LCPとLCPの間の差分ログ
GCPはLCPよりも細かい間隔で実行され、連続的に更新された内容をログに書き続けます。
GCPでは一定周期毎の更新内容をEpochと呼ばれる単位にまとめ、REDOログとして記録します。

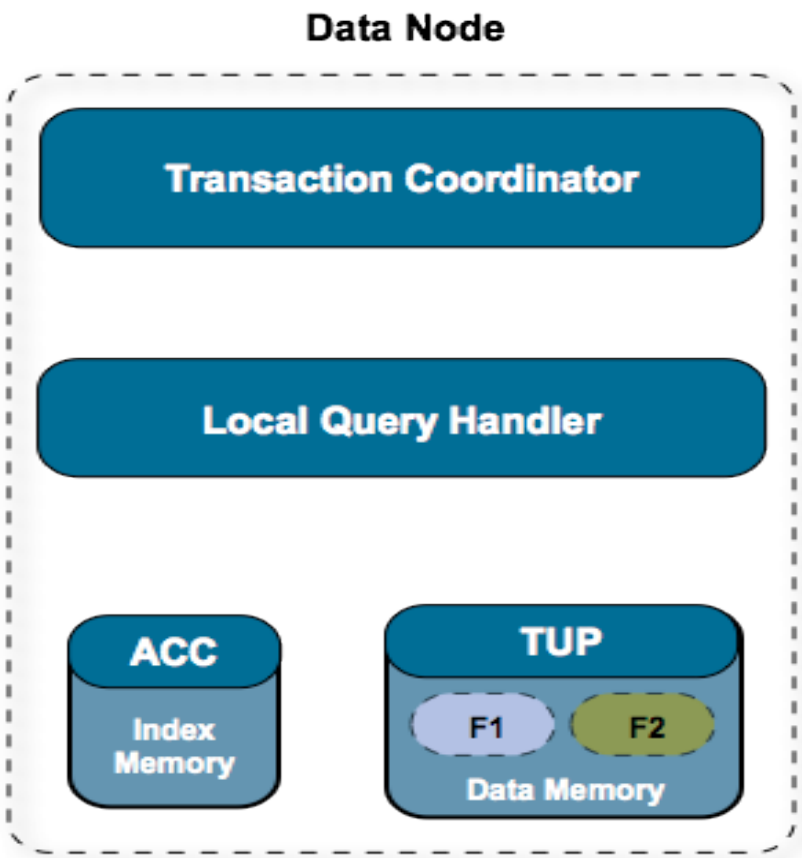
NOTE: GCPやLCPの書き込み状況を確認し、パラメータを調整可能。

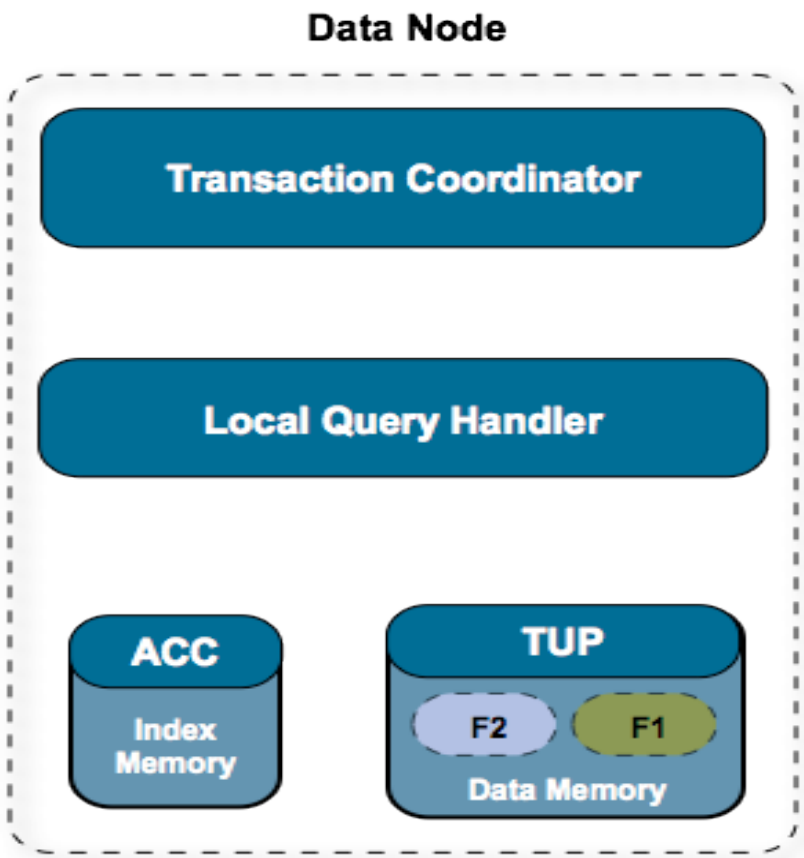
```
mysql> select * from ndbinfo.disk_write_speed_base;
+---------+--------+------------+---------------+-------------------------+--------------------+-------------------------+
| node_id | thr_no | millis_ago | millis_passed | backup_lcp_bytes_written | redo_bytes_written | target_disk_write_speed |
+---------+--------+------------+---------------+-------------------------+--------------------+-------------------------+
|       1 |      0 |          0 |          1003 |                       0 |                  0 |                20971520 |
|       1 |      0 |       1003 |          1012 |                       0 |             131072 |                20971520 |
|       1 |      0 |       2015 |          1004 |                       0 |                  0 |                20971520 |
|       1 |      0 |       3019 |          1002 |                       0 |             131072 |                20971520 |
|       1 |      0 |       4021 |          1002 |                       0 |                  0 |                20971520 |
```

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Internal Replication

Simplistic view of two Data Nodes

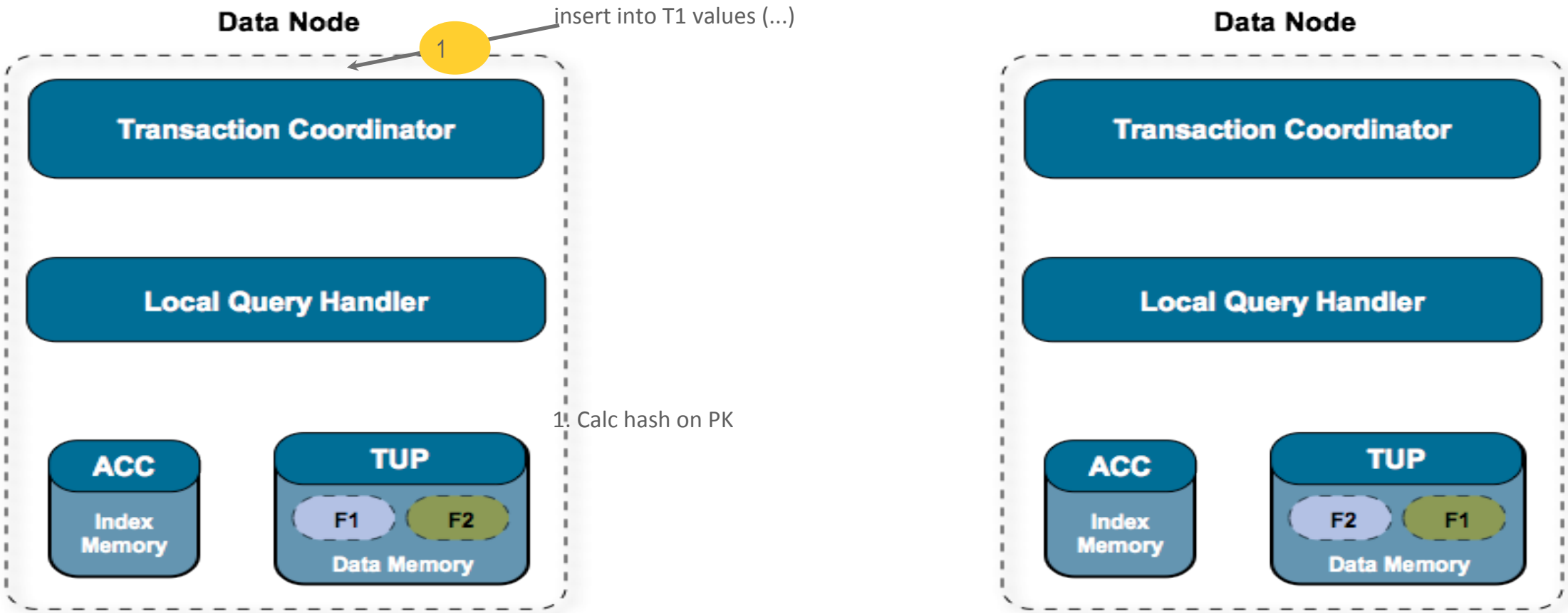



Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

# Internal Replication

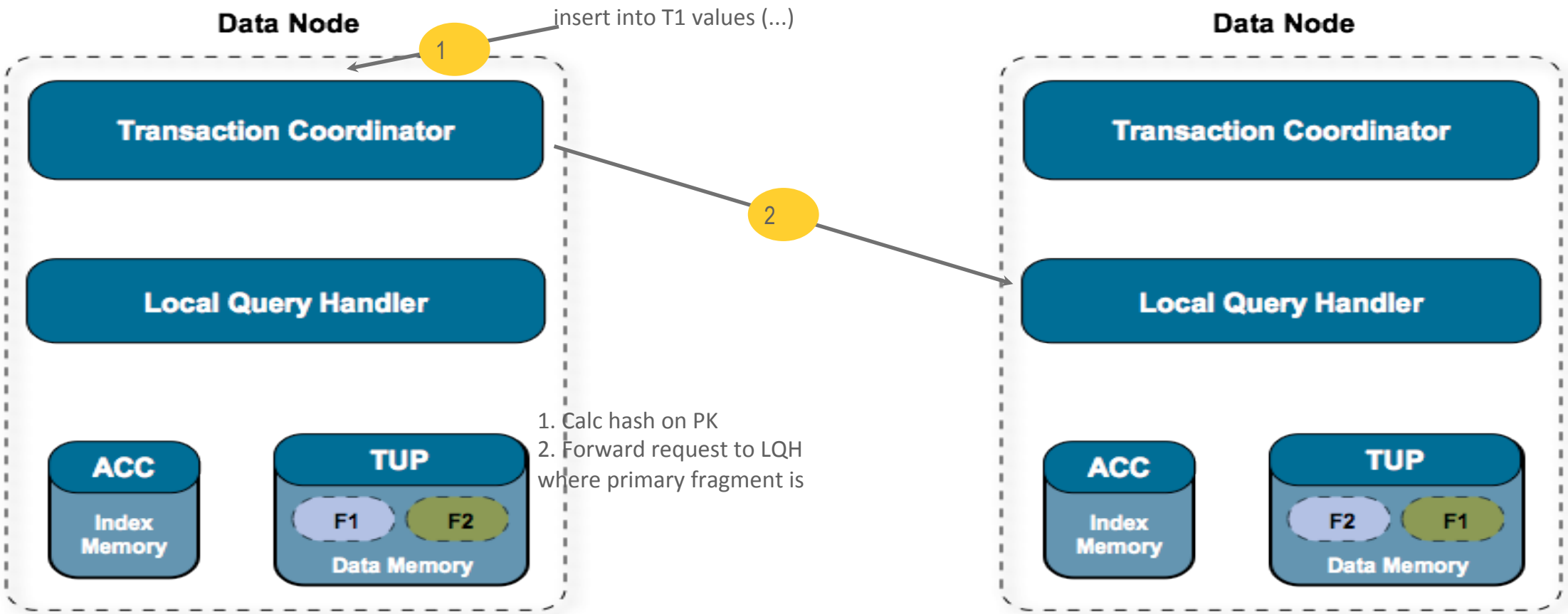
## Prepare Phase




Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

# Internal Replication

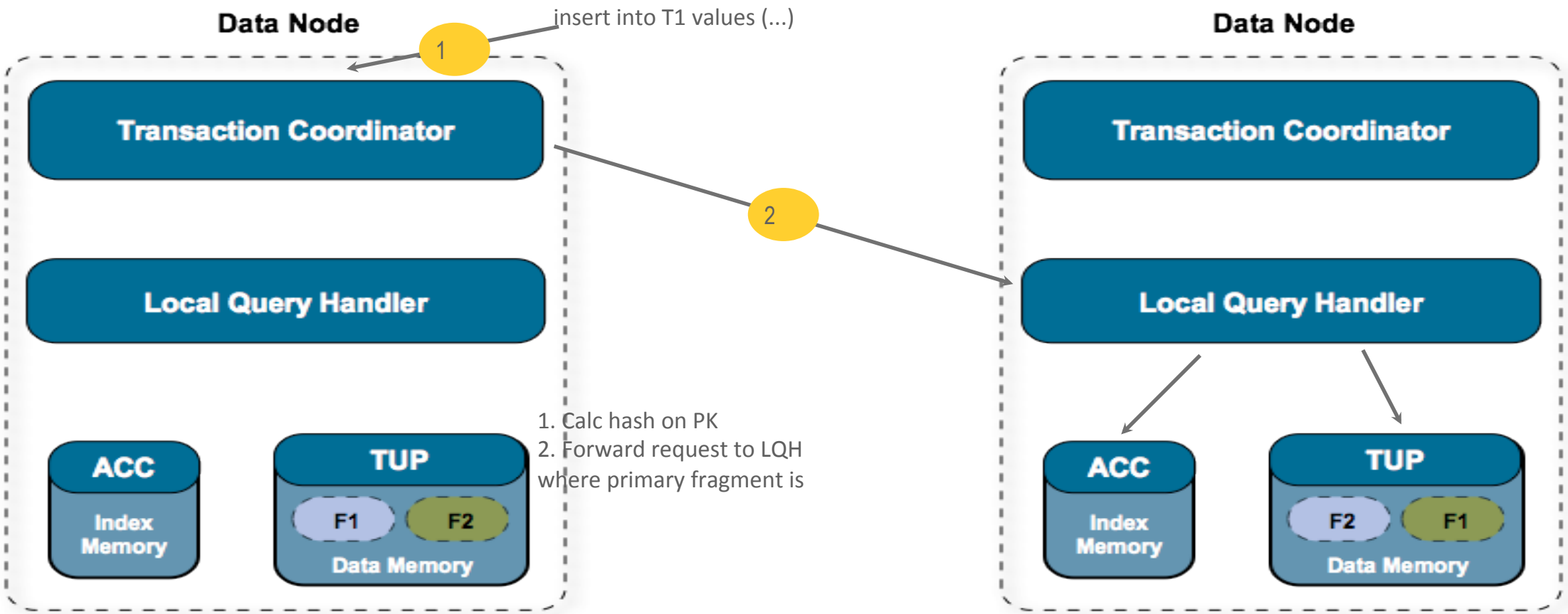
## Prepare Phase




Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

# Internal Replication

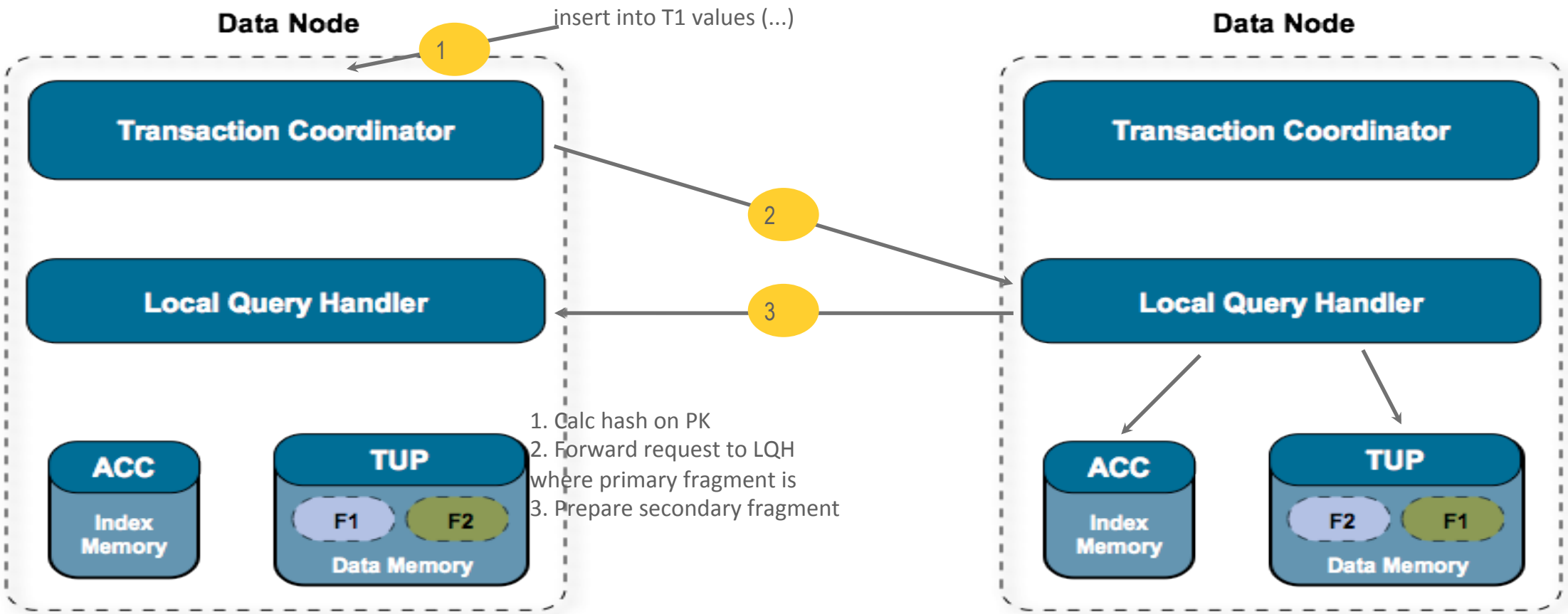
## Prepare Phase




Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

# Internal Replication

## Prepare Phase

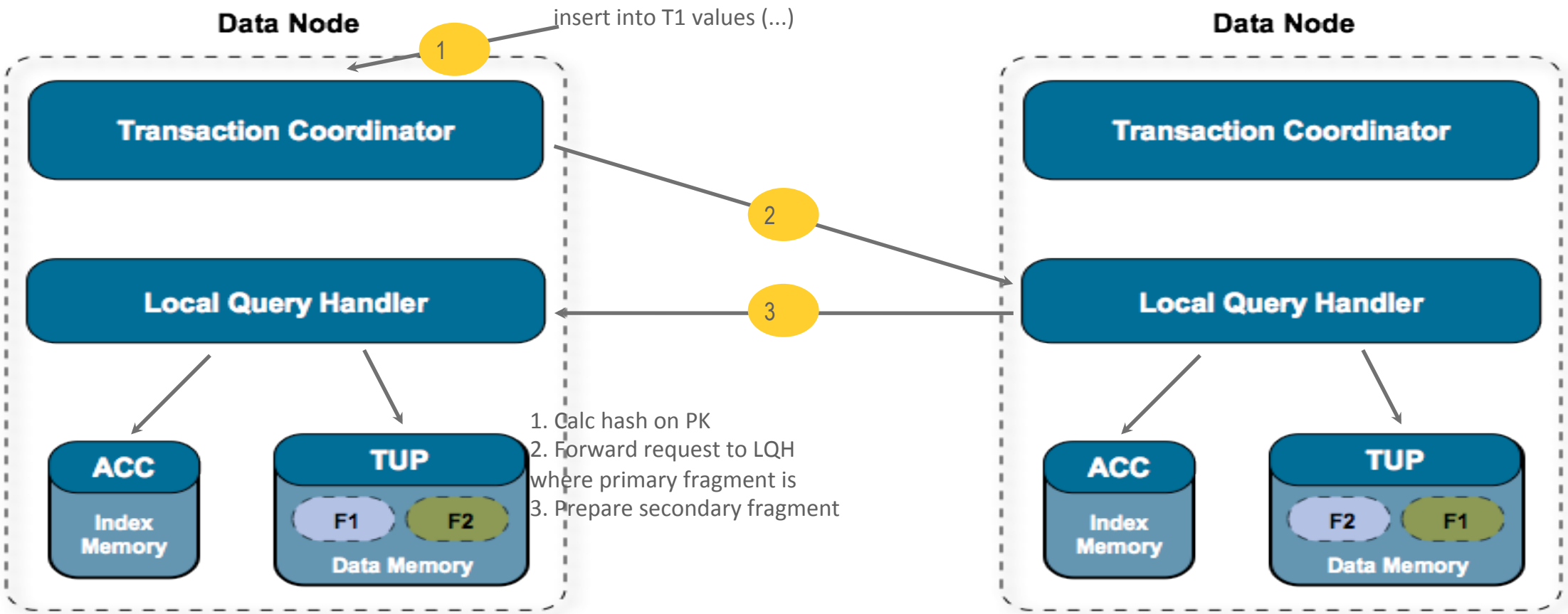



ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

# Internal Replication

## Prepare Phase

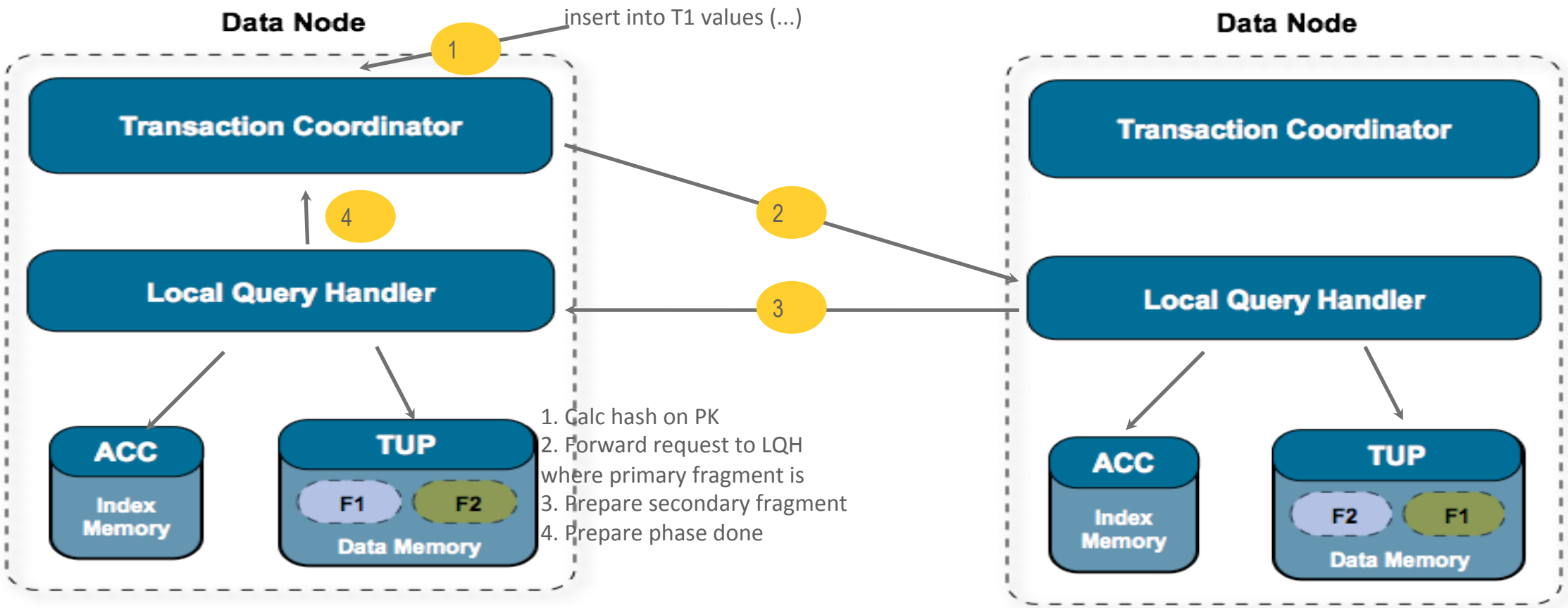



ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

# Internal Replication

## Prepare Phase




1. Calc hash on PK
2. Forward request to LQH where primary fragment is
3. Prepare secondary fragment
4. Prepare phase done

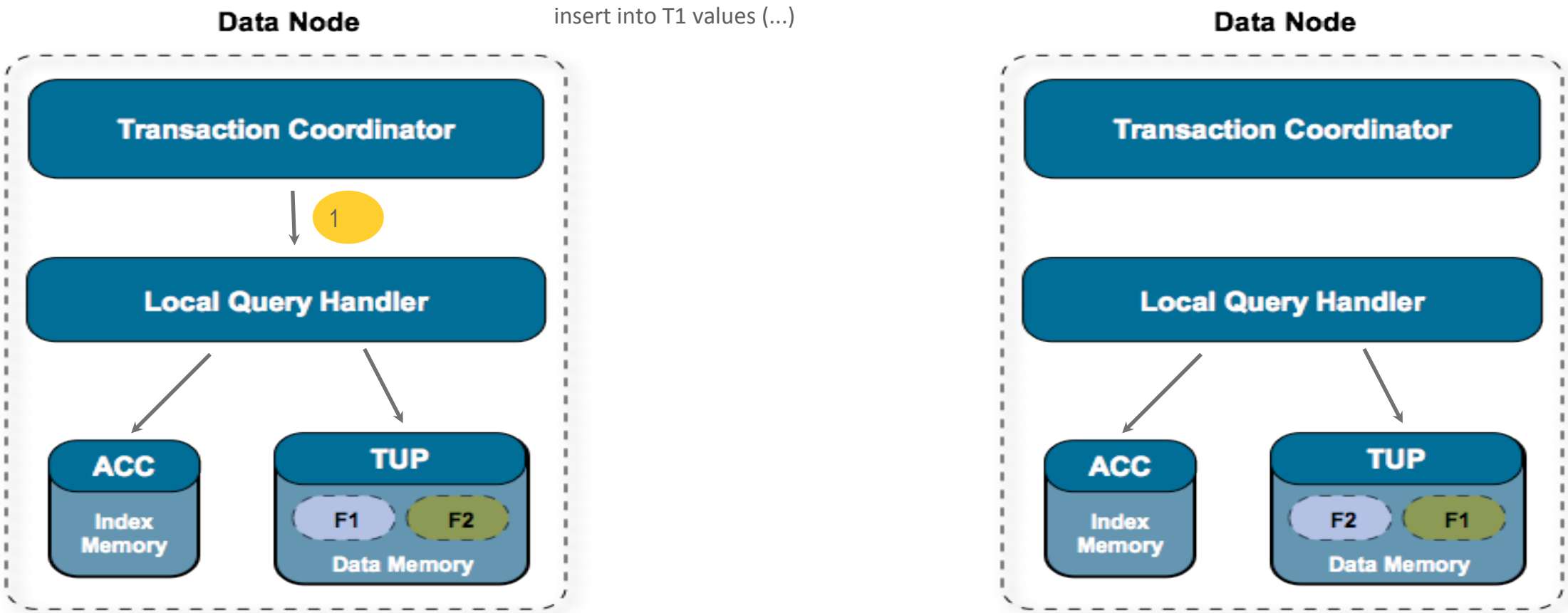
ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

# Internal Replication

Commit Phase

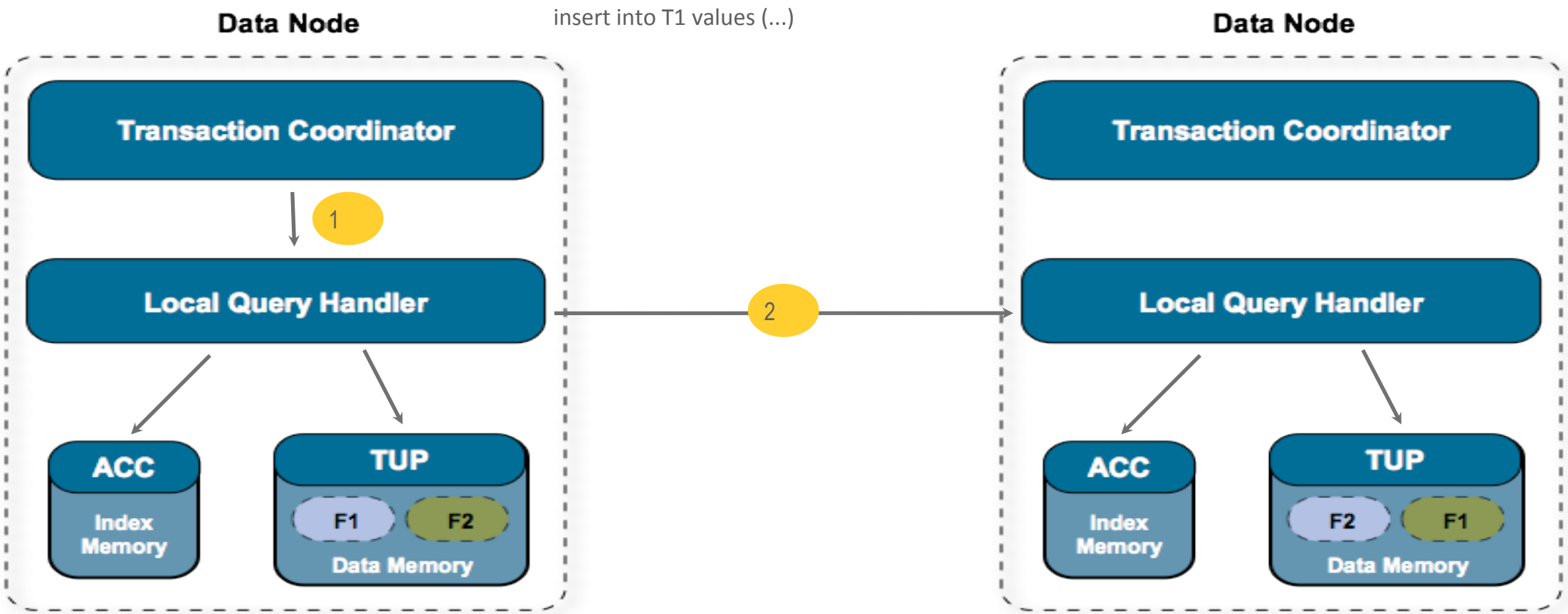
insert into T1 values (...)




Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

# Internal Replication

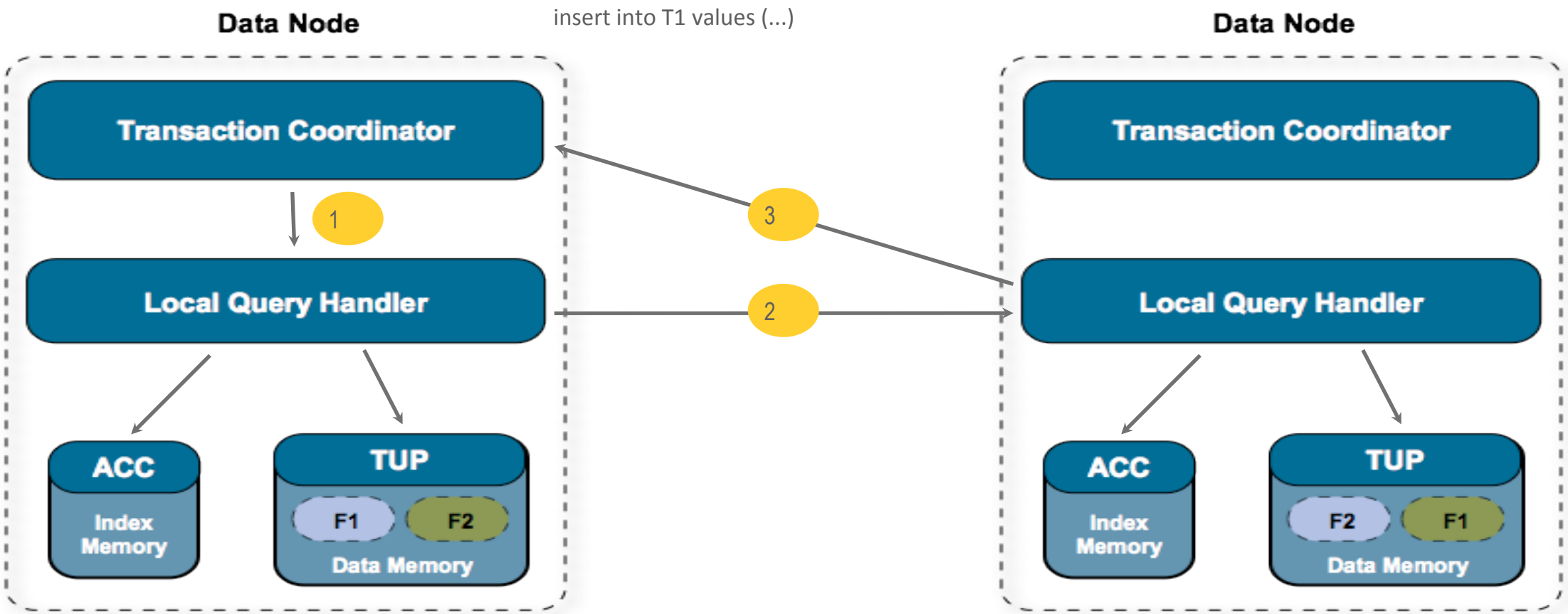
Commit Phase




Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

# Internal Replication

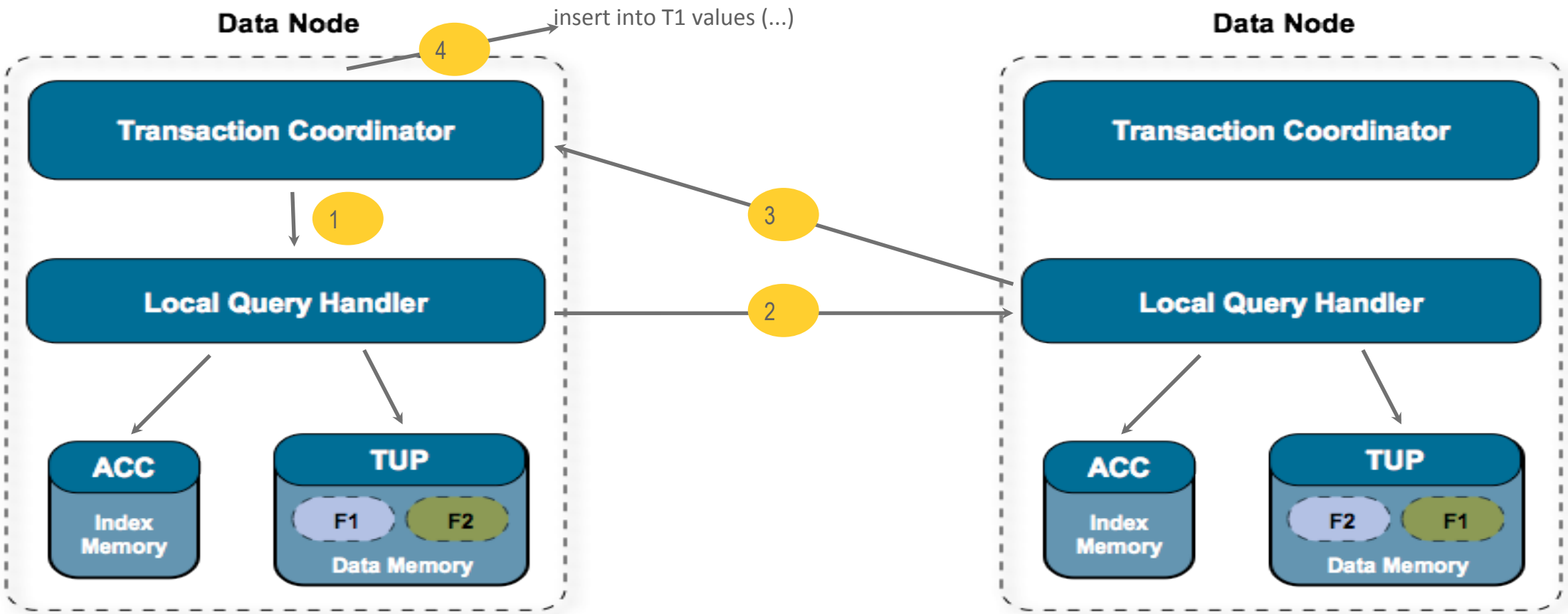
Commit Phase




Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

# Internal Replication

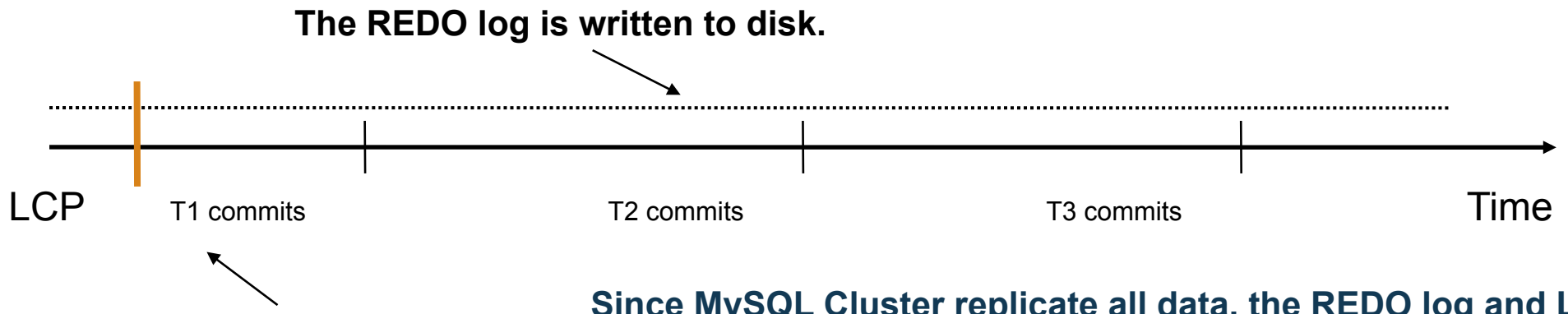
Commit Phase




Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

# Local Checkpoint (LCP)

Flushes the Data Nodes' data to disk. After 2 LCP the REDO log is cut
Frequency controlled by **TimebetweenLocalCheckpoints** setting
Specifies the amount of data that can change before flushing to disk
Not a time! Base-2 logarithm of the number of 4-byte words
Ex: Default value of 20 means 4*2^20 = 4MB of data changes, value of 21 = 8MB




**A *local checkpoint* LCP saves an image of the database on disk.**
**This enables cutting the tail of the REDO log.**

**Since MySQL Cluster replicate all data, the REDO log and LCPs are only used to recover from whole system failures.**

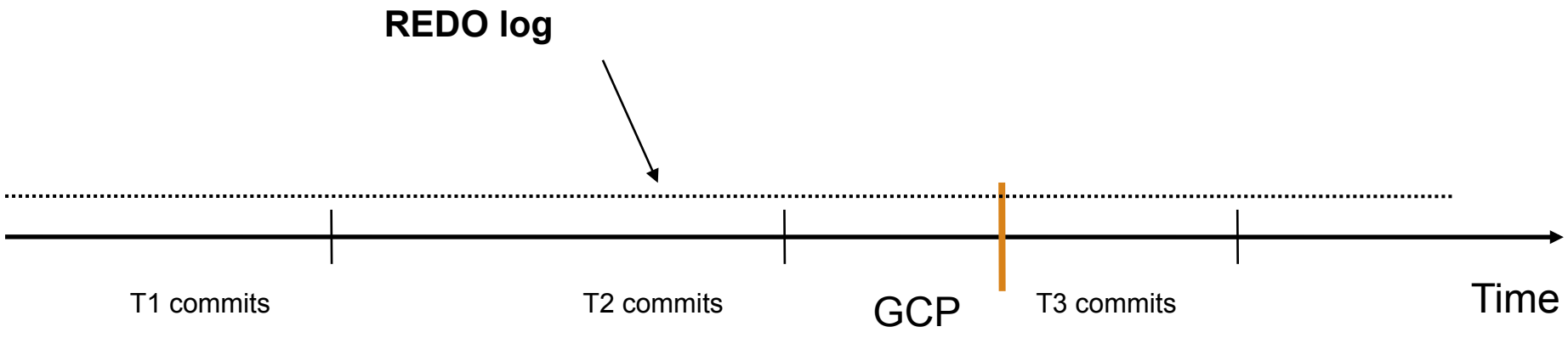
LCP and REDO Log are used to bring back the cluster online
- System failure or planned shutdown
  1st) Data Nodes are restored using the latest LCP
  2nd) The REDO logs are applied until the latest GCP

**Transactions T1, T2 and T3 are *main memory persistent*.**

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Global Checkpoint (GCP)

REDO log, synchronized between the Data Nodes.
Writes transactions that have been recorded in the REDO log buffer to disk/REDO log
Frequency controlled by **TimebetweenGlobalCheckpoints** setting (Default is 2000ms)
Size of the REDO log set by **NumOfFragmentLogFiles** and **FragmentLogFileSize**.

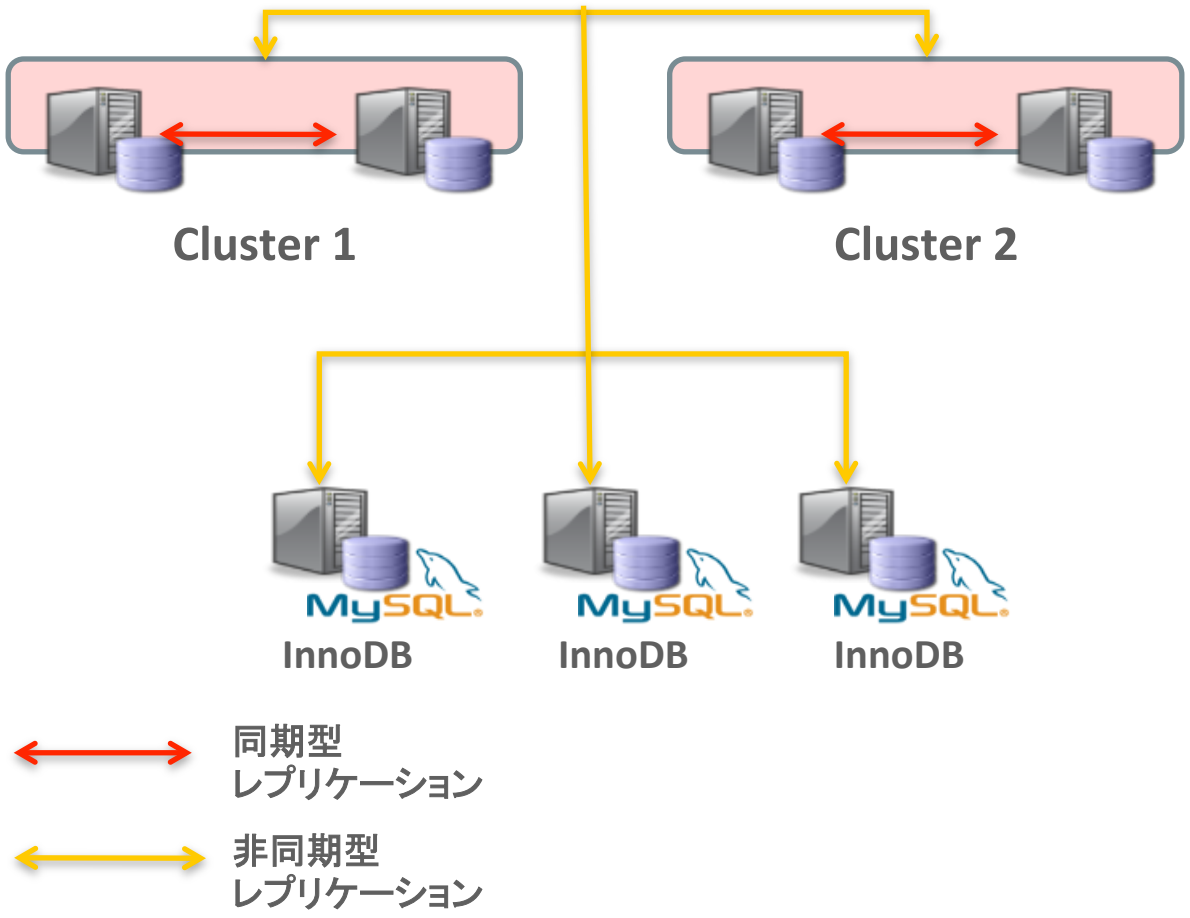



**A *global checkpoint* GCP flushes REDO log to disk.**
**Transactions T1 and T2 become *disk persistent*.**
**(Also called *group commit*.)**

NOTE: GCPが行われるまで（デフォルトでは2000ミリ秒ごと）に更新のあったデータはすべてメモリ上に格納しておかなければなりません。その為のバッファ サイズは、RedoBufferオプションで指定します。一度に大量の変更を加えたいような場合にはRedoBufferを大きくする必要があります。(Default:8MB)

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

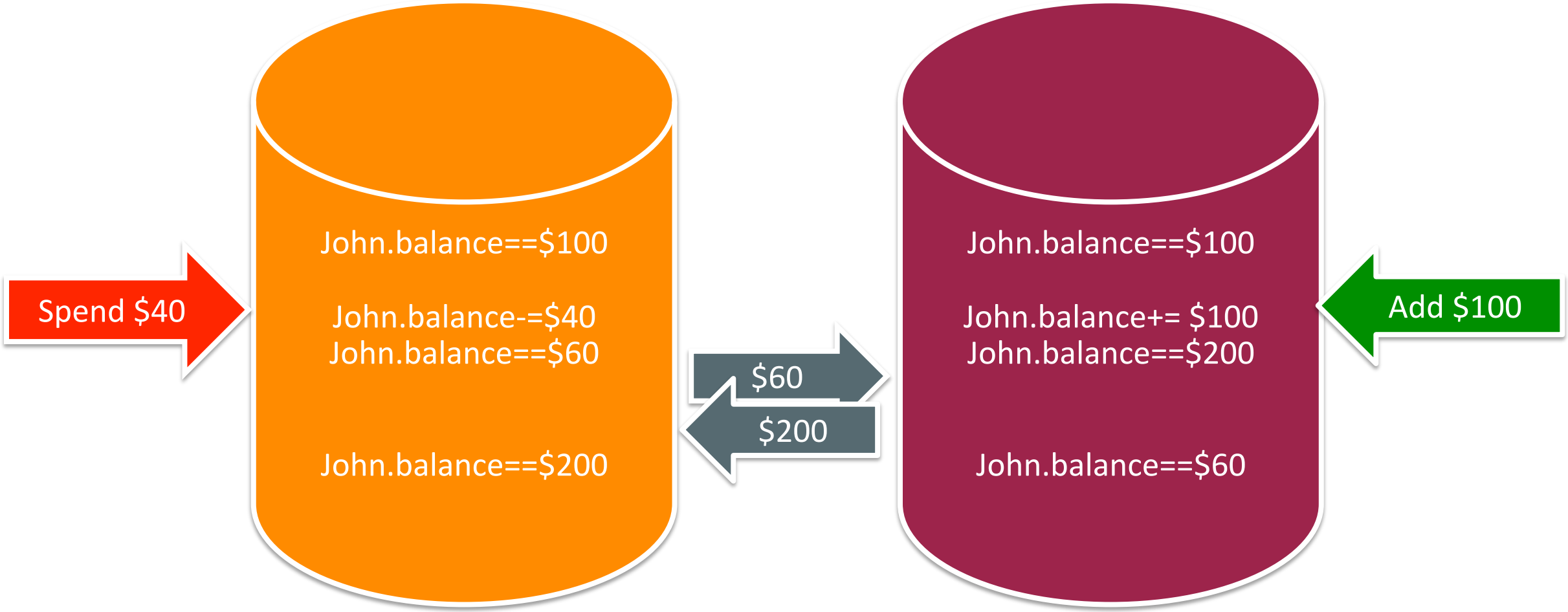
# MySQL Serverも組み合わせた複合型の構成




- クラスタのノードグループ間では<u>同期型</u>データ複製で冗長性を確保
- 地理的に離れたクラスタ間で、双方向の<u>非同期型</u>レプリケーションを行い、地理的冗長性を確保
- (MySQL Clusterではない)通常のMySQLサーバへ非同期型のレプリケーションを行い、レポート生成や課金処理などのアプリケーションを実行
- NoSQL経由の処理もレプリケーション可能

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# What is a conflict?




Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 補足：データの永続化(ディスクサイズ)

**Datamemory x (2+4+4+2)　バックアップを含むと Data memoryの 10倍の容量を安定稼働指標とする。**

※通常6倍～7倍を目安としていますが、安定稼働という意味では10倍程度を見積もっておくと安全です。
※バックアップは、CompressedBackupパラメーターで圧縮する事が可能です。

**LCP用のディスク領域には、DataMemoryの2倍のサイズを見積もります**

LCP用のディスク領域のサイズを設定するパラメータはありませんが、LCPはDataMemory上のデータをディスクに書き出す処理であるため、1回のLCPによるディスク書き込みは最大でDataMemoryと同じサイズを消費します。2倍のサイズを見積もるのは、LCPにより書き込まれたデータはディスク上に2世代保管される仕組みになっているためです。

**GCP用のディスク領域には、DataMemoryの4倍程度のサイズを見積ります**

GCPはLCPよりも細かい間隔で実行されるチェックポイント処理です。データノードの再起動時など最新のデータを復元する時には、LCPによりディスクに書き込まれたデータを読み込んだ後で、GCPによりディスクに書き込まれたREDOログを適用します。

**REDOログファイルの総容量**

安定稼働には、この計算式の結果が**DataMemoryの4倍程度**になるように、FragmentLogFileSizeパラメータを増加させます（NoOfFragmentLogFilesのデフォルト値（16）は一般的にREDOログファイルの数として十分であるため、通常はFragmentLogFileSizeによって調整する）。
REDOログファイルの総容量(指標: Data Memory x 4)＝NoOfFragmentLogFiles * FragmentLogFileSize * 4

GCPによるREDOログ出力と同様に1回のLCPが完了する間にバックアップの取得が完了すると仮定して必要領域を考えると、安定稼働のためのマージンを 含めて**1世代あたりDataMemoryの約2倍**の領域が必要となります。
そのため、「データノードが保持しているデータ（=DataMemory）」と 合わせて1世代あたりDataMemoryの3倍の領域を確保。

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Disk-Based Data

## ディスク上に列とテーブルのデータを格納するよう定義することが可能

To store the nonindexed columns of NDB tables on disk, rather than in RAM.
To handle of very large amounts (terabytes) of data during node recovery and restart.

1) ログファイルグループを作成し1つ以上のUNDO LOGファイルを割り当てる。

```
CREATE LOGFILE GROUP logfile1 ADD UNDOFILE 'undo_file' engine=ndbcluster;
```

2)表領域を作成し表領域にログファイルグループ、ならびに1つ以上のデータファイルを割り当てる。

```
CREATE TABLESPACE ts_1 ADD DATAFILE 'data_1.dat' USE LOGFILE GROUP logfile1 engine=ndbcluster;
```

3) データ保存のためのこの表領域を使用してディスクデータテーブルを作成します。

Disk data related parameters:
DiskPageBufferMemory
SharedGlobalMemory
DiskIOThreadPool

NOTE: ディスクテーブルはディスクベースとなり処理は遅いので,
I/O分散を考慮して下さい。

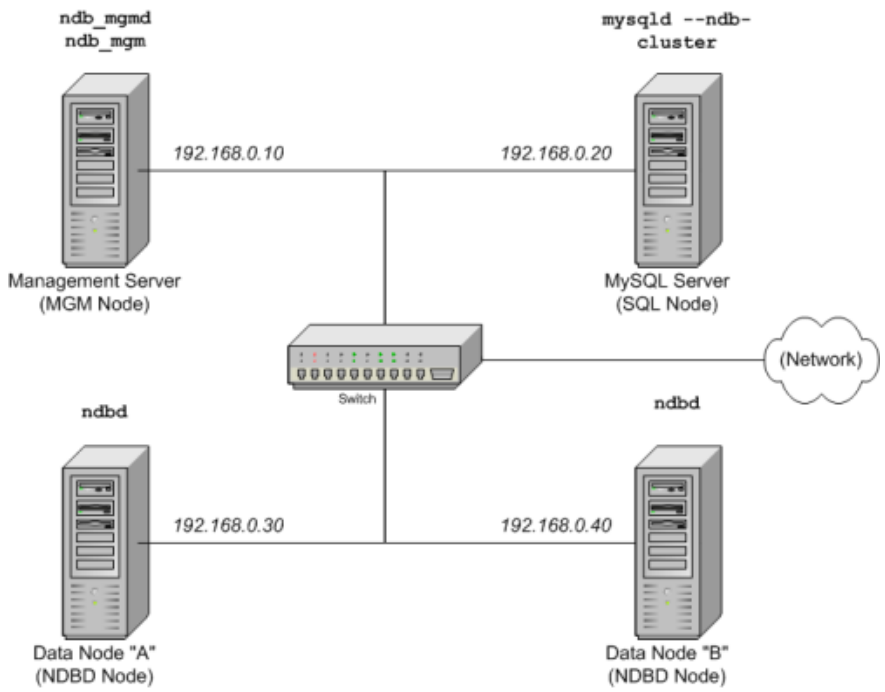
```
mysql> show create table T_STORAGE_DISK\G
*************************** 1. row ***************************
       Table: T_STORAGE_DISK
Create Table: CREATE TABLE `T_STORAGE_DISK` (
  `member_id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `last_name` varchar(50) NOT NULL,
  `first_name` varchar(50) NOT NULL,
  `dob` date NOT NULL,
  `joined` date NOT NULL,
  PRIMARY KEY (`member_id`),
  KEY `last_name` (`last_name`,`first_name`)
) /*!50100 TABLESPACE ts_1 STORAGE DISK */ ENGINE=ndbcluster
DEFAULT CHARSET=utf8
```

参照:
https://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-disk-data-objects.html

ORACLE
Copyright © 2019 Oracle and/or its affiliates. All rights reserved.

# ネットワーク

TCP/IP (local), TCP/IP (remote), SCI, and SHM.




ほとんどのユーザーは現在、広く普及していることから、TCP/IP over Ethernet (GB Ethernet)を採用しています。TCP/IP は、MySQL Cluster での使用に関して最も検証及びテストされているトランスポータでもあります。

利用者が希望する場合は、クラスタのインターコネクトを使用してパフォーマンスをさらに向上させることも出来ます。これには 2 つの実現方法があります。このケースを処理できるようにカスタムトランスポータを設計するか、TCP/IP スタックをある程度バイパスするソケット実装を使用できます。弊社は、Dolphin Interconnect Solutions が開発した SCI (スケーラブルコヒーラントインタフェース) テクノロジを使用して、両方の手法を試しました。

参照： http://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-interconnects.html

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 仮想環境について

**A.10.12.** Can I run MySQL Cluster nodes inside virtual machines (such as those created by VMWare, Parallels, or Xen)?

MySQL Cluster is supported for use in virtual machines beginning with MySQL Cluster NDB 7.2. We currently support and test using Oracle VM.
Some MySQL Cluster users have successfully deployed MySQL Cluster using other virtualization products; in such cases, Oracle can provide MySQL Cluster support, but issues specific to the virtual environment must be referred to that product's vendor.

SQLノードを仮想環境で利用するケースも増えてきておりますが、データノードに関しては定期的にディスクに書き込むGCPやLCP等の処理や、起動時にデータをメモリーにロードするまでの時間が許容出来るかの確認をしてください。また、データノードにおいては多くのトラフィックが発生しますので、仮想環境のようにリソースを共有している環境においてもネットワークの帯域が十分にある事も担保しておく必要があります。
上記の理由から、データノードに関しては物理サーバーでの利用を推奨しております。

参照: https://dev.mysql.com/doc/refman/5.6/en/faqs-mysql-cluster.html#faq-cluster-can-cluster-vm

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Agenda

1. MySQL Cluster概要
2. システムデザイン（構成とサイジング）
3. 構成パラメータと構成変更 (--reload, --init)
4. バックアップとリカバリー
5. スケールアウト(データノード追加とデータ再配置)
6. レプリケーション(NDB -> NDB,  NDB -> InnoDB )
7. その他、参考情報

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# MySQL Clusterインストールとパラメータ

MySQL Clusterのインストール方法に関しては、マニュアルページに手順含めて説明してありますので、具体的なインストール方法に関してはマニュアルを参照下さい。

## 18.2 MySQL Cluster のインストール


18.2.1 MySQL Cluster Auto-Installer
18.2.2 Linux での MySQL Cluster のインストール
18.2.3 Windows での MySQL Cluster のインストール
18.2.4 MySQL Cluster の初期構成
18.2.5 MySQL Cluster の初期起動
18.2.6 テーブルとデータを含む MySQL Cluster の例
18.2.7 MySQL Cluster の安全なシャットダウンと再起動
18.2.8 MySQL Cluster NDB 7.3 のアップグレードとダウングレード

```
cd /var/tmp
tar -zxvf mysql-5.6.22-ndb-7.4.4-linux-i686-glibc23.tar.gz
cd mysql-5.6.22-ndb-7.4.4-linux-i686-glibc23
cp bin/ndbd /usr/local/bin/ndbd
cp bin/ndbmtd /usr/local/bin/ndbmtd
```

参照: http://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-installation.html

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 構成パラメーター

- MySQL Cluster を構成するには、2 つのファイルを操作する必要があります。
- my.cnf: MySQL Cluster のすべての実行可能ファイルのオプションを指定します。このファイルは、(これまでの MySQL の使用経験からご存じのとおり) クラスタ内で実行されている個々の実行可能ファイルからアクセスできる必要があります。
- config.ini: このファイルは、グローバル構成ファイルとも呼ばれますが、MySQL Cluster管理サーバーからのみ読み取られます。管理サーバーに格納された情報は、クラスタに参加している全てのプロセスに配布されます。config.ini には、クラスタに関与する各ノードの記述が含まれています。これには、データノードに関する構成パラメータと、クラスタ内のすべてのノード間の接続に関する構成パラメータが含まれます。このファイルに含まれる可能性があるセクションと各セクションに配置される構成パラメータの種類を簡単に確認するには、「config.ini ファイルのセクション」を 参照してください。

※構成パラメータを変更した場合にパラメータの変更を反映させるために、ローリングリスタート(--initial, --reload)やクラスター全体の再起動が必要になる場合があります。約170の構成パラメータがあり、85%のパラメータはローリングリスタートに対応していて、残りの15%のパラメータに関しては、システム全体の再起動が必要になります。

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 初期パラメータ構成例

```
# TCP PARAMETERS
[tcp default]
SendBufferMemory=2M
ReceiveBufferMemory=2M
# MANAGEMENT NODE PARAMETERS
[ndb_mgmd default]
DataDir=path/to/management/server/data/directory
[ndb_mgmd]
HostName=management-server-A-hostname
NodeId=management-server-A-nodeid
[ndb_mgmd]
HostName=management-server-B-hostname
NodeId=management-server-B-nodeid
# DATA NODE PARAMETERS
[ndbd default]
NoOfReplicas=2
LockPagesInMainMemory=1
DataMemory=3072M
IndexMemory=384M
ODirect=1
NoOfFragmentLogFiles=300
DataDir=path/to/data/node/data/directory
MaxNoOfConcurrentOperations=100000
SchedulerSpinTimer=400
SchedulerExecutionTimer=100
RealTimeScheduler=1
TimeBetweenGlobalCheckpoints=1000
TimeBetweenEpochs=200
DiskCheckpointSpeed=10M
DiskCheckpointSpeedInRestart=100M
RedoBuffer=32M
CompressedLCP=1
CompressedBackup=1
MaxNoOfLocalScans=64
MaxNoOfTables=1024
MaxNoOfOrderedIndexes=256
[ndbd]
HostName=data-node-A-hostname
NodeId=data-node-A-nodeid
LockExecuteThreadToCPU=1
LockMaintThreadsToCPU=0
[ndbd]
HostName=data-node-B-hostname
NodeId=data-node-B-nodeid
LockExecuteThreadToCPU=1
LockMaintThreadsToCPU=0
# SQL NODE / API NODE PARAMETERS
[mysqld]
[mysqld]
[mysqld]
```

参考: https://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-config-starting.html

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# --initialize

**--initial オプション:** グローバル構成ファイルを再度読み込む時に、既存の全てのコンフィグレーションキャッシュファイルが削除され、構成ファイルから新しいキャッシュが作成されます。

複数の管理サーバーでのローリング再起動　複数の管理ノードを持つ MySQL Cluster のローリング再起動を実行すると、ndb_mgmd によって、その他の管理ノードが実行されているかどうかがチェックされ、実行されている場合は、そのノードの構成データの使用が試みられることに注意してください。これが発生することを回避し、ndb_mgmd にその構成ファイルを強制的に再読み取りさせるには、次のステップを実行します。

1. MySQL Cluster のすべての ndb_mgmd プロセスを停止します。
2. すべての config.ini ファイルを更新します。
3. 必要に応じて、--reload、--initial、またはその両方のオプションを付けて単一の ndb_mgmd を起動します。
4. 最初の ndb_mgmd を --initial オプションを付けて起動した場合は、残りの ndb_mgmd プロセスもすべて --initial を使用して起動する必要があります。最初の ndb_mgmd を起動するときに使用されたその他のオプションに関係なく、最初の ndb_mgmdプロセスの後に、--reload を使用して残りのプロセスを再起動しないようにしてください。
5. 通常どおりに、データノードおよび API ノードのローリング再起動を完了します。

ローリング再起動を実行してクラスタの構成を更新する際に、ndbinfo.nodes テーブルの config_generation カラムを使用して、新しい構成で正常に再起動されたデータノードを追跡できます。セクション18.5.10.18「ndbinfo nodes テーブル」を参照してください。

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# --reload

**--reload オプション:** 管理サーバーのキャッシュとグローバル構成ファイルが比較され、これらが異なる場合は、管理サーバーによって新しい構成キャッシュが作成 されます。既存の構成キャッシュは保持されますが、使用されません。管理サーバーのキャッシュとグローバル構成ファイルに同じ構成データが含まれている場合は、既存のキャッシュが使用され、新しいキャッシュは作成されません。

管理ノードの構成に変更を加えたあとは、新しい構成を有効にするために、クラスタのローリング再起動を実行する必要があります。

config_generation カラムは、各データノードで有効になっているクラスタ構成のバージョンを示します。
これは、構成パラメータの変更を行うために、クラスタのローリング再起動を実行する際に役立つことがあります。

```
mysql> SELECT * FROM ndbinfo.nodes;
+---------+--------+---------+-------------+------------------+
| node_id | uptime | status  | start_phase | config_generation |
+---------+--------+---------+-------------+------------------+
|       1 |    490 | STARTED |           0 |                6 |
|       2 |    486 | STARTED |           0 |                6 |
+---------+--------+---------+-------------+------------------+
```

このデータノードで使用中の
クラスタ構成ファイルのバージョン。

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 全体再起動が必要な構成パラメータ

| | | | | |
|---|---|---|---|---|
| BuildIndexThreads | 数値 | S | NDB 7.3.0 | データノードの構成パラメータ |
| CrashOnCorruptedTuple | ブール | S | NDB 7.3.0 | データノードの構成パラメータ |
| Diskless | true\|false (1\|0) | IS | NDB 7.3.0 | データノードの構成パラメータ |
| ExecuteOnComputer | name | S | NDB 7.3.0 | データノードの構成パラメータ |
| HeartbeatOrder | 数値 | S | NDB 7.3.0 | データノードの構成パラメータ |
| Id | 符号なし | IS | NDB 7.3.0 | データノードの構成パラメータ |
| IndexStatAutoCreate | ブール | S | NDB 7.3.0 | データノードの構成パラメータ |
| IndexStatAutoUpdate | ブール | S | NDB 7.3.0 | データノードの構成パラメータ |
| InitialLogFileGroup | 文字列 | S | NDB 7.3.0 | データノードの構成パラメータ |
| InitialTablespace | 文字列 | S | NDB 7.3.0 | データノードの構成パラメータ |
| MaxDiskWriteSpeed | 数値 | S | NDB 7.4.1 | データノードの構成パラメータ |
| MaxDiskWriteSpeedOtherNo | 数値 | S | NDB 7.4.1 | データノードの構成パラメータ |
| MaxDiskWriteSpeedOwnRes | 数値 | S | NDB 7.4.1 | データノードの構成パラメータ |
| MinDiskWriteSpeed | 数値 | S | NDB 7.4.1 | データノードの構成パラメータ |
| NodeGroup | | IS | NDB 7.3.0 | データノードの構成パラメータ |
| NodeId | 符号なし | IS | NDB 7.3.0 | データノードの構成パラメータ |
| NoOfReplicas | 整数 | IS | NDB 7.3.0 | データノードの構成パラメータ |
| StringMemory | % またはバイト | S | NDB 7.3.0 | データノードの構成パラメータ |
| MaxNoOfExecutionThreads | 整数 | IS | NDB 7.3.3 | データノードの構成パラメータ (マルチスレッド) |
| ThreadConfig | 文字列 | IS | NDB 7.3.0 | データノードの構成パラメータ (マルチスレッド) |
| ExecuteOnComputer | name | S | NDB 7.3.0 | 管理ノードの構成パラメータ |
| HeartbeatThreadPriority | 文字列 | S | NDB 7.3.0 | 管理ノードの構成パラメータ |
| Id | 符号なし | IS | NDB 7.3.0 | 管理ノードの構成パラメータ |
| NodeId | 符号なし | IS | NDB 7.3.0 | 管理ノードの構成パラメータ |
| DefaultOperationRedoProbl | 列挙 | S | NDB 7.3.0 | SQL ノードおよび API ノードの構成パラメータ |
| EventLogBufferSize | バイト | S | NDB 7.3.0 | SQL ノードおよび API ノードの構成パラメータ |
| ExecuteOnComputer | name | S | NDB 7.3.0 | SQL ノードおよび API ノードの構成パラメータ |
| HeartbeatThreadPriority | 文字列 | S | NDB 7.3.0 | SQL ノードおよび API ノードの構成パラメータ |
| Id | 符号なし | IS | NDB 7.3.0 | SQL ノードおよび API ノードの構成パラメータ |
| NodeId | 符号なし | IS | NDB 7.3.0 | SQL ノードおよび API ノードの構成パラメータ |

参照:
http://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-params-overview.html

ORACLE
Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# その他：構成ファイルオプション

--config-cache: “--config-cache=OFF”このオプションを使用すると、管理サーバーは構成キャッシュを完全にバイパスします。この場合、管理サーバーは存在する可能性がある構成ファイルを無視し、常に config.ini ファイルからその構成データを読み取ります。
※　既に、コンフィグレーションキャッシュがある場合は削除する必要があります。

構成キャッシュがみつかりません　この場合、管理サーバーはグローバル構成ファイルを読み取り、そのファイルと同じ構成データを含むキャッシュを作成します。

参照：http://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-config-file.html

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 参考) データノードIDについて

NodeId 一意のノード ID は、クラスタのすべての内部メッセージでノードのアドレスとして使用されます。データノードでは、これは 1-48 の (これらを含む) 範囲の整数です。クラスタ内の各ノードは、一意の識別子を持っている必要があります。データノードのID用に1～48を予め確保して、管理ノード, SQLノード, APIノードには50番以降のIDを割り当てる事をお勧め致します。

| 有効なバージョン | 型/単位 | デフォルト | 範囲/値 | 再起動タイプ |
|---|---|---|---|---|
| NDB 7.3.0～ | 符号なし | [none] | **1 - 48** | IS |

参考：https://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-ndbd-definition.html

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 参考)管理コマンド

データノードの停止

ndb_mgm> 3 STOP

データノードを一台ずつ再起動

ndb_mgm> ALL RESTART

クラスターのシャットダウン (SQL/APIノードは事前に停止)

ndb_mgm> SHUTDOWN

バックアップ(各データノードにバックアップファイルを作成)

ndb_mgm> START BACKUP

ヘルプコマンド

ndb_mgm> HELP

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Agenda

1. MySQL Cluster概要
2. システムデザイン（構成とサイジング）
3. 構成パラメータと構成変更 (--reload, --init)
4. バックアップとリカバリー
5. スケールアウト(データノード追加とデータ再配置)
6. レプリケーション(NDB -> NDB,  NDB -> InnoDB )
7. その他、参考情報

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 管理ノードにてBACKUPコマンドを実行

```
ndb_mgm> START BACKUP
Waiting for completed, this may take several minutes
Node 4: Backup 2 started from node 1
Node 4: Backup 2 started from node 1 completed
 StartGCP: 43696 StopGCP: 43699
 #Records: 2077 #LogRecords: 0
 Data: 53752 bytes Log: 0 bytes
ndb_mgm>
```

```
#ndb_mgm -e ”startup backup”
```

```
[root@CLSData01 admin]# ls -l /usr/local/mysql/mysql-cluster/BACKUP/
合計 0
drwxr-x---. 2 root root 72 10月  22 13:56 BACKUP-1
drwxr-x---. 2 root root 72  1月   3 22:35 BACKUP-2
[root@CLSData01 admin]#
```

```
[root@CLSData02 admin]# ls -l /usr/local/mysql/mysql-cluster/BACKUP/
合計 0
drwxr-x---. 2 root root 72 10月  22 13:56 BACKUP-1
drwxr-x---. 2 root root 72  1月   3 22:35 BACKUP-2
[root@CLSData02 admin]#
```

.Data: クラスタデータ
.ctl: クラスタのメタデータ
.log: クラスタのログファイル

```
[root@CLSData01 admin]# ls -l /usr/local/mysql/mysql-cluster/BACKUP/BACKUP-2/
合計 60
-rw-r--r--. 1 root root 28096  1月   3 22:35 BACKUP-2-0.4.Data
-rw-r--r--. 1 root root 27492  1月   3 22:35 BACKUP-2.4.ctl
-rw-r--r--. 1 root root    52  1月   3 22:35 BACKUP-2.4.log
[root@CLSData01 admin]#
```

参考: http://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-backup.html

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# RESTOREを実施

```
ndb_restore [-c <mgm_host>:<mgm_port>]
-n <node_id> [-m] -b <backup_id> -r [-e]
--backup_path=/path/to/backup/files
```

| オプション | 概要 |
|---|---|
| -c | 接続文字列 |
| -n | データノードID |
| -b | バックアップID |
| -r | リストア |
| -m | メタデータ (初回のみ) |

-e : スレーブがレプリケーションを開始する為にepochをリストアします(スレーブにリストアする場合)
--no-binlog : mysqldが起動していて、バイナリーログを設定している場合リストアログを記録しない。
--printにてバックアップファイルの中身を確認する事が可能。
--include-databases または --include-tables を使用すると、それらのオプションによって指定されたデータベースまたはテーブルのみがリストアされます。

NOTE: Restore from cluster backup by ndb_restore does not recover hidden PK

参照: http://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-programs-ndb-restore.html

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 参考) RESTOREの実施例

管理ノードのみ起動じた状態でデータノードを初期化
※事前にdatadir=/usr/local/mysql/mysql-cluster/dataで指定フォルダーをリネームしても問題ありません。

```
# ndbmtd --connect-string=192.168.56.114,192.168.56.115 --nostart --initial
# ndbmtd --connect-string=192.168.56.114,192.168.56.115 --nostart --initial
```

DATA NODE1の最新のバックアップをリストア。初回なので-mでメタデータもリストア

```
# ndb_restore -c 192.168.56.114 -n 1 -b 5 -m -r --backup_path=./restore/BACKUP1/BACKUP-5
<SNIP>
NDBT_ProgramExit: 0 - OK
```

DATA NODE2の最新のバックアップをリストア。

```
# ndb_restore -c 192.168.56.114 -n 2 -b 5 -r --backup_path=./restore/BACKUP2/BACKUP-5
<SNIP>
NDBT_ProgramExit: 0 - OK
```

リストアが完了したので、MySQLを起動してデータが戻っている事を確認したら完了。
MySQLをシングルユーザーモードで起動しておいて確認しながらも可能です。
NOTE: 最新のバックアップ以降のデータが必要な場合は、バイナリーログで差分をリストアする事も可能です。

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# mysqldumpを利用する場合

通常バックアップ

1.必要に応じて、Mysql instanceをシングルユーザーモードに設定

2. mysqldump –uroot –ppass db > dump_all.sql

NDBクラスターのデータをInnoDBで利用する場合

1.必要に応じて、Mysql instanceをシングルユーザーモードに設定

2. mysqldump -uroot -ppass --no-data db | sed 's/ndbcluster/InnoDB/g' > ddl_dump.sql

3. mysqldump -uroot -ppass --no-create-info db > dml_dump.sql

※レプリケーションで利用する場合は、--master-data=2 --hex-blob 等のオプションを付けると対応が容易。

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 参考) Single User Mode

【1】Singleユーザーモードに設定 ⟹ 【2】Backup取得 ⟹ 【3】SingleユーザーモードからEXIT

```
ndb_mgm> show
Connected to Management Server at: 192.168.56.114:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @192.168.56.114  (mysql-5.6.25 ndb-7.4.7, Nodegroup: 0, *)
id=2    @192.168.56.115  (mysql-5.6.25 ndb-7.4.7, Nodegroup: 0)
[ndb_mgmd(MGM)] 2 node(s)
id=100  @192.168.56.114  (mysql-5.6.25 ndb-7.4.7)
id=101  @192.168.56.115  (mysql-5.6.25 ndb-7.4.7)
[mysqld(API)]   3 node(s)
id=50   @192.168.56.114  (mysql-5.6.25 ndb-7.4.7)
id=51   @192.168.56.115  (mysql-5.6.25 ndb-7.4.7)
id=60 (not connected, accepting connect from 192.168.56.108)
ndb_mgm> enter single user mode 50
Single user mode entered
Access is granted for API node 50 only.
ndb_mgm>
```

```
ndb_mgm>show
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @192.168.56.114  (mysql-5.6.25 ndb-7.4.7, single user mode, Nodegroup: 0, *)
id=2    @192.168.56.115  (mysql-5.6.25 ndb-7.4.7, single user mode, Nodegroup: 0)
[ndb_mgmd(MGM)] 2 node(s)
id=100  @192.168.56.114  (mysql-5.6.25 ndb-7.4.7)
id=101  @192.168.56.115  (mysql-5.6.25 ndb-7.4.7)
[mysqld(API)]   3 node(s)
id=50   @192.168.56.114  (mysql-5.6.25 ndb-7.4.7)
id=51   @192.168.56.115  (mysql-5.6.25 ndb-7.4.7)
id=60 (not connected, accepting connect from 192.168.56.108)
ndb_mgm> exit single user mode
Exiting single user mode in progress.
Use ALL STATUS or SHOW to see when single user mode has been exited.
ndb_mgm>
```

参照: https://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-single-user-mode.html

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Agenda

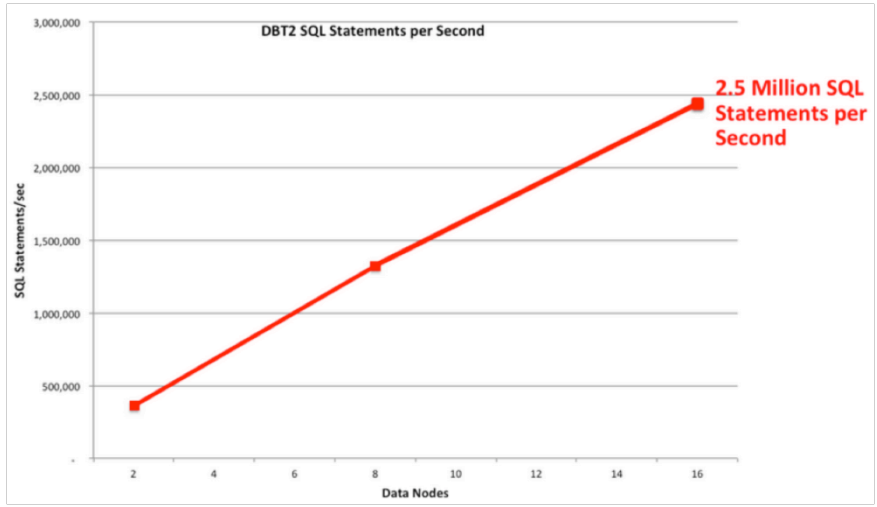
1. MySQL Cluster概要
2. システムデザイン（構成とサイジング）
3. 構成パラメータと構成変更 (--reload, --init)
4. バックアップとリカバリー
5. スケールアウト(データノード追加とデータ再配置)
6. レプリケーション(NDB -> NDB,  NDB -> InnoDB )
7. その他、参考情報

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# データノードの追加によるスケールアウト

1. config.iniの編集
2. 管理ノードのローリングリスタート (--reload)
3. 既存のデータノードのローリングリスタート
4. SQLノードのローリングリスタート
5. 新しいノードの起動
6. CREATE NODEGROUPの実行
7. テーブルのデータの再分散（ALTER TABLE ... REORGANIZE PARTITION）
8. 空いた領域の解放（OPTIMIZE TABLE）




参照： http://dev.mysql.com/doc/refman/5.6/ja/mysql-cluster-online-add-node-basics.html

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 1,2 新設定のロード (reload)

1. config.iniの編集
2. 管理ノードのローリングリスタート (--reload)

#設定ファイルに新規ノードを追加
[ndbd]
NodeId=3
HostName=192.168.56.110
datadir=/usr/local/mysql-cluster/mysql-cluster/data

[ndbd]
NodeId=4
HostName=192.168.56.111
datadir=/usr/local/mysql-cluster/mysql-cluster/data

■ 対象管理ノード
[ndb_mgmd(MGM)] 2 node(s)
id=100 @192.168.56.114 (mysql-5.6.25 ndb-7.4.7)
id=101 @192.168.56.115 (mysql-5.6.25 ndb-7.4.7)

```
[cluster74_01 cluster]# /usr/local/mysql/bin/ndb_mgm -e "100 STOP"
Connected to Management Server at: 192.168.56.114:1186
Node 100 has shutdown.
Disconnecting to allow Management Server to shutdown

[cluster74_01 cluster]# /usr/local/mysql/bin/ndb_mgmd --reload -f /usr/local/mysql/mysql-cluster/ndb-config.ini
MySQL Cluster Management Server mysql-5.6.25 ndb-7.4.7
```

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 3,4 ステータスと設定の反映

3. 既存のデータノードのローリングリスタート
4. SQLノードのローリングリスタート

【クラスターログの確認】

2016-03-23 21:55:50 [MgmtSrvr] INFO -- Starting configuration change, generation: 7
2016-03-23 21:55:50 [MgmtSrvr] INFO -- Node 100: Node 2 Connected
2016-03-23 21:55:50 [MgmtSrvr] INFO -- Configuration 8 commited
2016-03-23 21:55:50 [MgmtSrvr] INFO -- Config change completed! New generation: 8

```
ndb_mgm> show
<SNIP>
[ndbd(NDB)]     4 node(s)
id=1    @192.168.56.114  (mysql-5.6.25 ndb-7.4.7, Nodegroup: 0, *)
id=2    @192.168.56.115  (mysql-5.6.25 ndb-7.4.7, Nodegroup: 0)
id=3 (not connected, accepting connect from 192.168.56.110)
id=4 (not connected, accepting connect from 192.168.56.111)
```

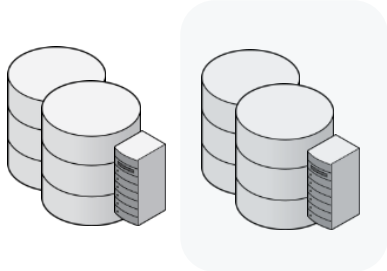
```
ndb_mgm> 1 restart
Node 1: Node shutdown initiated
Node 1: Node shutdown completed, restarting, no start.
Node 1 is being restarted
Node 1: Started (version 7.4.7)
```

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 5. 新しいノードの起動

[mysql-cluster]# /usr/local/mysql-cluster/bin/ndbmtd --connect-string=192.168.56.114 --nostart

```
ndb_mgm> show
Cluster Configuration
---------------------
[ndbd(NDB)]     4 node(s)
id=1    @192.168.56.114  (mysql-5.6.25 ndb-7.4.7, Nodegroup: 0)
id=2    @192.168.56.115  (mysql-5.6.25 ndb-7.4.7, Nodegroup: 0, *)
id=3    @192.168.56.110  (mysql-5.6.25 ndb-7.4.7, not started)
id=4    @192.168.56.111  (mysql-5.6.25 ndb-7.4.7, not started)

[ndb_mgmd(MGM)] 2 node(s)
id=100  @192.168.56.114  (mysql-5.6.25 ndb-7.4.7)
id=101  @192.168.56.115  (mysql-5.6.25 ndb-7.4.7)

[mysqld(API)]   3 node(s)
id=50   @192.168.56.114  (mysql-5.6.25 ndb-7.4.7)
id=51   @192.168.56.115  (mysql-5.6.25 ndb-7.4.7)
id=60 (not connected, accepting connect from 192.168.56.108)
```

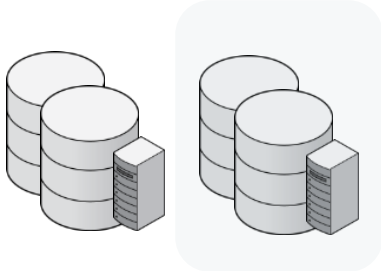


Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 5. 新しいノードの起動 - start

--nostartオプションで起動した場合のみ実行

```
ndb_mgm> 3 start
Database node 3 is being started.
ndb_mgm> 4 start
Database node 4 is being started.
ndb_mgm> show
Cluster Configuration
---------------------
[ndbd(NDB)]     4 node(s)
id=1    @192.168.56.114  (mysql-5.6.25 ndb-7.4.7, Nodegroup: 0)
id=2    @192.168.56.115  (mysql-5.6.25 ndb-7.4.7, Nodegroup: 0, *)
id=3    @192.168.56.110  (mysql-5.6.25 ndb-7.4.7, no nodegroup)
id=4    @192.168.56.111  (mysql-5.6.25 ndb-7.4.7, no nodegroup)
[ndb_mgmd(MGM)] 2 node(s)
id=100  @192.168.56.114  (mysql-5.6.25 ndb-7.4.7)
id=101  @192.168.56.115  (mysql-5.6.25 ndb-7.4.7)
[mysqld(API)]   3 node(s)
id=50   @192.168.56.114  (mysql-5.6.25 ndb-7.4.7)
id=51   @192.168.56.115  (mysql-5.6.25 ndb-7.4.7)
id=60 (not connected, accepting connect from 192.168.56.108)
```



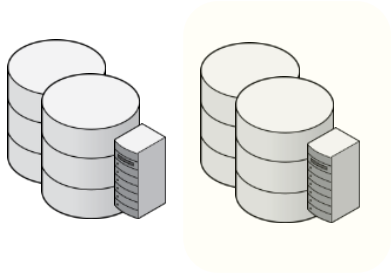
Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 6. CREATE NODEGROUPの実行

ノードグループは、1 つ以上のノードで構成され、パーティションまたはレプリカ のセットを格納します。

```
ndb_mgm> CREATE NODEGROUP 3,4
Nodegroup 1 created
ndb_mgm> show
Cluster Configuration
---------------------
[ndbd(NDB)]     4 node(s)
id=1    @192.168.56.114  (mysql-5.6.25 ndb-7.4.7, Nodegroup: 0)
id=2    @192.168.56.115  (mysql-5.6.25 ndb-7.4.7, Nodegroup: 0, *)
id=3    @192.168.56.110  (mysql-5.6.25 ndb-7.4.7, Nodegroup: 1)
id=4    @192.168.56.111  (mysql-5.6.25 ndb-7.4.7, Nodegroup: 1)

[ndb_mgmd(MGM)] 2 node(s)
id=100  @192.168.56.114  (mysql-5.6.25 ndb-7.4.7)
id=101  @192.168.56.115  (mysql-5.6.25 ndb-7.4.7)

[mysqld(API)]   3 node(s)
id=50   @192.168.56.114  (mysql-5.6.25 ndb-7.4.7)
id=51   @192.168.56.115  (mysql-5.6.25 ndb-7.4.7)
id=60 (not connected, accepting connect from 192.168.56.108)
```

NoOfReplicas= 2



Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

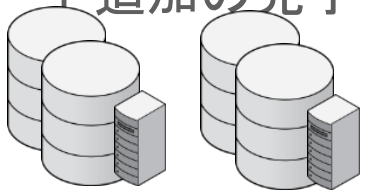
# 7. テーブルのデータの再分散（ALTER TABLE ... REORGANIZE PARTITION）

```
ndb_mgm> ALL REPORT MEMORY
Node 1: Data usage is 1%(55 32K pages of total 4096)
Node 1: Index usage is 0%(58 8K pages of total 8352)
Node 2: Data usage is 1%(55 32K pages of total 4096)
Node 2: Index usage is 0%(58 8K pages of total 8352)
Node 3: Data usage is 1%(81 32K pages of total 4096)
Node 3: Index usage is 0%(0 8K pages of total 8352)
Node 4: Data usage is 1%(81 32K pages of total 4096)
Node 4: Index usage is 0%(0 8K pages of total 8352)
```

```
mysql> ALTER ONLINE TABLE T20150319 REORGANIZE PARTITION;
Query OK, 0 rows affected, 1 warning (8.27 sec)
Records: 0  Duplicates: 0  Warnings: 1
```

```
ndb_mgm> ALL REPORT MEMORY
Node 1: Data usage is 1%(52 32K pages of total 4096)
Node 1: Index usage is 0%(58 8K pages of total 8352)
Node 2: Data usage is 1%(52 32K pages of total 4096)
Node 2: Index usage is 0%(58 8K pages of total 8352)
Node 3: Data usage is 2%(90 32K pages of total 4096)
Node 3: Index usage is 0%(4 8K pages of total 8352)
Node 4: Data usage is 2%(90 32K pages of total 4096)
Node 4: Index usage is 0%(4 8K pages of total 8352)
```

7. テーブルのデータの再分散（ALTER TABLE ... REORGANIZE PARTITION）
8. 空いた領域の解放（OPTIMIZE TABLE）
9. データノード追加の完了



NOTE:
ndbinfo.memory_per_fragmentを参照すると、
新規ノードへのデータの分散状況が確認可能

```
mysql> select fq_name, table_id, node_id, fragment_num, fixed_elem_alloc_bytes, hash_index_alloc_bytes
    -> from ndbinfo.memory_per_fragment where fq_name='world/def/City';
+----------------+----------+---------+--------------+-----------------------+-----------------------+
| fq_name        | table_id | node_id | fragment_num | fixed_elem_alloc_bytes | hash_index_alloc_bytes |
+----------------+----------+---------+--------------+-----------------------+-----------------------+
| world/def/City |       35 |       1 |            0 |                196608 |                 32768 |
| world/def/City |       35 |       1 |            1 |                196608 |                 32768 |
| world/def/City |       35 |       2 |            0 |                196608 |                 32768 |
| world/def/City |       35 |       2 |            1 |                196608 |                 32768 |
+----------------+----------+---------+--------------+-----------------------+-----------------------+
```

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# 参考: ALTER ONLINE TABLE実行時のClusterログ

**【サンプルテーブル】**

```
CREATE TABLE `City` (
 `ID` int(11) NOT NULL AUTO_INCREMENT,  `Name` char(35) NOT NULL DEFAULT '',
 `CountryCode` char(3) NOT NULL DEFAULT '',  `District` char(20) NOT NULL DEFAULT '',
 `Population` int(11) NOT NULL DEFAULT '0',  PRIMARY KEY (`ID`), KEY `CountryCode` (`CountryCode`),
 CONSTRAINT `city_ibfk_1` FOREIGN KEY (`CountryCode`) REFERENCES `Country` (`Code`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=ndbcluster AUTO_INCREMENT=4080 DEFAULT CHARSET=latin1
```

**【コマンド実行】**

```
mysql> ALTER ONLINE TABLE City REORGANIZE PARTITION;
Query OK, 0 rows affected, 1 warning (9.35 sec)
```

**【クラスターログ】** Default: /DataDir/ndb_Node ID_cluster.log

```
2016-03-23 22:29:08 [MgmtSrvr] INFO     -- Node 3: reorg-copy table 35 processed 0 rows
2016-03-23 22:29:08 [MgmtSrvr] INFO     -- Node 4: reorg-copy table 35 processed 0 rows
2016-03-23 22:29:08 [MgmtSrvr] INFO     -- Node 2: reorg-copy table 35 processed 1995 rows
2016-03-23 22:29:08 [MgmtSrvr] INFO     -- Node 1: reorg-copy table 35 processed 2084 rows
2016-03-23 22:29:16 [MgmtSrvr] INFO     -- Node 3: reorg-delete table 35 processed 0 rows
2016-03-23 22:29:16 [MgmtSrvr] INFO     -- Node 4: reorg-delete table 35 processed 0 rows
2016-03-23 22:29:17 [MgmtSrvr] INFO     -- Node 1: reorg-delete table 35 processed 1026 rows
2016-03-23 22:29:17 [MgmtSrvr] INFO     -- Node 2: reorg-delete table 35 processed 977 rows
```

ALTER ONLINE TABLE
実行時のクラスターログ

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

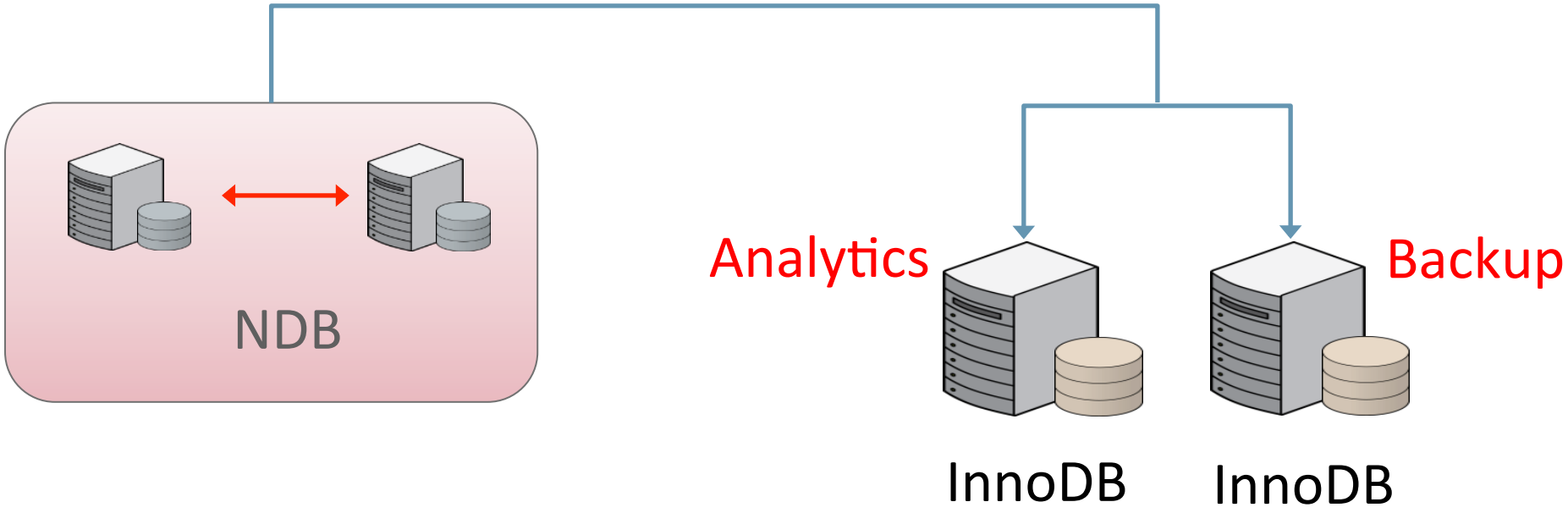
# Agenda

1. MySQL Cluster概要
2. システムデザイン（構成とサイジング）
3. 構成パラメータと構成変更 (--reload, --init)
4. バックアップとリカバリー
5. スケールアウト(データノード追加とデータ再配置)
6. レプリケーション(NDB -> NDB,  NDB -> InnoDB )
7. その他、参考情報

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# MySQL Cluster(NDB)とMySQL(InnoDB)の連携




Synchronous replication

Asynchronous replication

NOTE: NDBからInnoDBにレプリケーションを貼る時は、ストレージエンジンが異なりますので
DDL実行時の運用を調整して下さい。

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# レプリケーションの設定

## 1) SQLノードにてmy.cnfを変更

server_id =  xx
log-bin=binlog
binlog_format=ROW

## 2) SQLノードにてレプリケーション用アカウント作成

GRANT REPLICATION SLAVE ON *.* TO ‘replication_user’@‘%’ IDENTIFIED BY ‘password';

## 3) スレーブ用サーバーにデータリストア

## 4) レプリケーションの開始

CHANGE MASTER TO
MASTER_HOST = '192.168.56.114',
MASTER_USER = ' replication_user ',
MASTER_PASSWORD='password',
MASTER_PORT=3306,
MASTER_LOG_FILE='mysql-bin.000020',
MASTER_LOG_POS=120;

NOTE: ClusterはGTIDに対応して いないので、ログのポジションベースのレプリケーションを利用。
必要に応じて、MaxBufferedEpochs , TimeBetweenEpochsを調整して下さい。

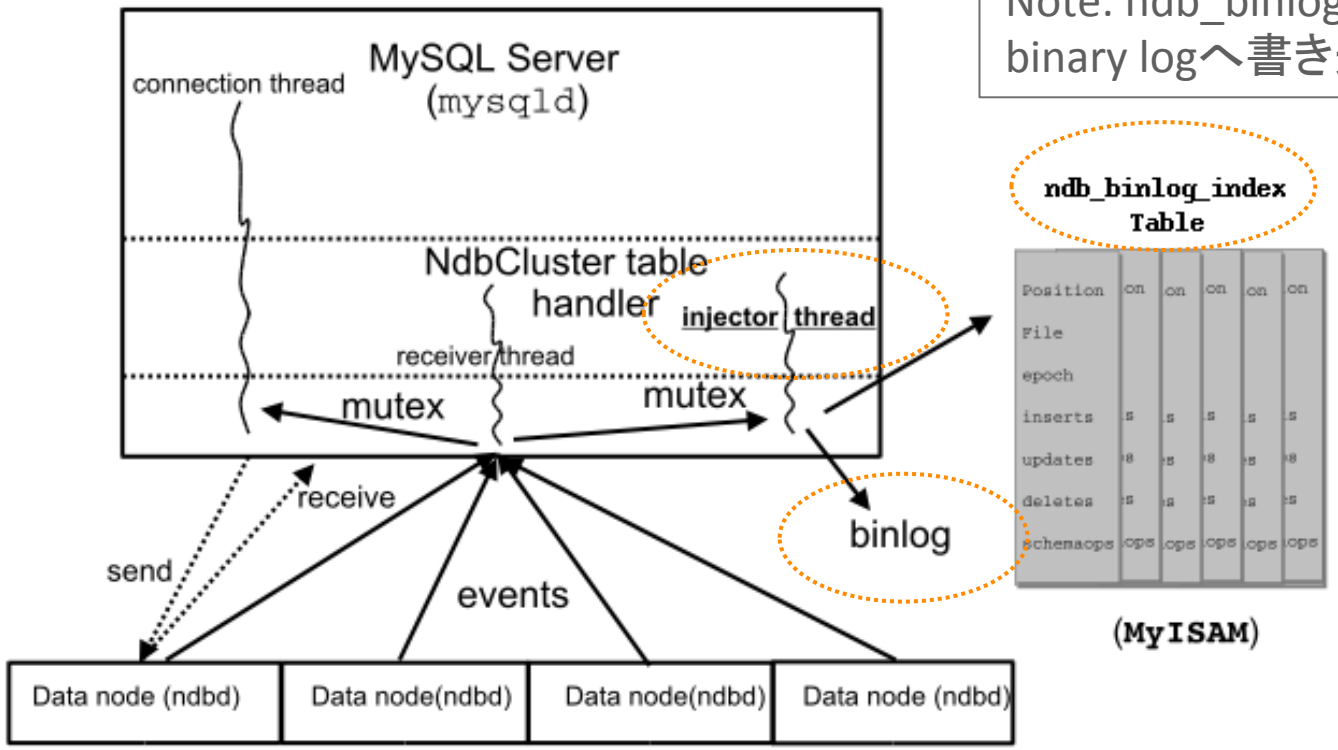
ORACLE
Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Binlog Injection

NDB インジェクタは、クラスタ内のすべてのデータイベントを取得する役割を担い、データを変更、挿入、または削除するすべてのイベントがndb_binlog_index テーブルに記録されたかを確認します。スレーブの I/O スレッドは、マスターのバイナリログからスレーブのリレーログにイベントを転送します。

**MySQL Replication Between Clusters, Injecting into Binlog**

Note: ndb_binlog_indexへの記録と同時にbinary logへ書き込み、スレーブに送信




Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# ndb_binlog_index

ndb_binlog_index と ndb_apply_status の両方とも NDB バイナリログのインジェクタスレッドによって維持されます。

```
mysql> SHOW MASTER STATUS;
+-----------------+----------+--------------+------------------+-------------------+
| File            | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+-----------------+----------+--------------+------------------+-------------------+
| mysql-bin.000078 |     3465 |              |                  |                   |
+-----------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)

mysql> select Position,File,epoch,gci,next_position,next_file from ndb_binlog_index order by epoch desc limit 1;
+----------+-------------------+-----------------+-------+---------------+-------------------+
| Position | File              | epoch           | gci   | next_position | next_file         |
+----------+-------------------+-----------------+-------+---------------+-------------------+
|     3085 | ./mysql-bin.000078 | 278292405944338 | 64795 |          3465 | ./mysql-bin.000078 |
+----------+-------------------+-----------------+-------+---------------+-------------------+
1 row in set (0.00 sec)
```

NOTE: ndb_apply_status というテーブルは、マスターからスレーブに複製された操作の記録を取るために使用されます。
ndb_apply_status テーブルはスレーブでのみ移入されます。

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Agenda

1. MySQL Cluster概要
2. システムデザイン（構成とサイジング）
3. 構成パラメータと構成変更 (--reload, --init)
4. バックアップとリカバリー
5. スケールアウト(データノード追加とデータ再配置)
6. レプリケーション(NDB -> NDB,  NDB -> InnoDB )
7. その他、参考情報

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# その他、設計上の考慮事項

1.明示的にすべてのテーブルの主キーを指定
- Clusterは、PKの無いテーブルに対してHidden PKを指定します。
- Hidden PK はクラスター間で複製されません。
- APIノードはランダムに選択されたデータノードに接続し、トランザクションはデータを持っているノードに転送される為、追加のネットワークレーテンシーが発生します。
- INSERT時、ユニークな自動インクリメント値をFETCH し全てのINSERTの為にグローバル・ロックを取得
- ndb_restoreでリストアした場合、Hidden PKはリストアされません

2. 主にSPJ(Push Down Join) の為に、ANALYZE TABLEを実行する事をお勧めします
3. OPTIMIZE tablesを実行時、ディスクでは無くメモリーベースのテーブルからスペースを取得
4. No Query Cache – expensive to invalidate it across SQL Nodes
5. クラスターに適切なアプリケーションやスキーマ設計を選択
- 短く・小さなトランザクション, 一括更新は10000を上限として全てのデータが更新されるまで繰り返す
- 多くの並列トランザクション
- シンプルなアクセスパターン
- 最も重要な処理に対してネットワークのラウンドトリップを最小限に抑える
- 大きく複雑なJOIN処理を避ける (even with SPJ from 7.2)
- 必要に応じてスキーマを非正規化
- Request H/W and /W resources to support MySQL Cluster

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Commercial Edition

## MySQL Cluster Carrier Grade Edition

データ量の爆発的な増加が続いています。その原因となるのは、イン
ク、ますます高機能化の進むスマート・デバイスを相互に接続する高
M2M（Machine to Machine）相互通信などです。

MySQL Clusterは、リレーショナル・データベースの柔軟性とオープン
可用性データベースです。MySQL Clusterを使用すれば、次世代の
におけるデータベース関連の課題に、妥協のないスケーラビリティ、ア

### 特長

- **書込み処理の拡張性を高める自動シャーディング**: MySQL Clu
ディング（パーティショニング）されるため、アプリケーションに対する
ディティ・ハードウェア上でデータベースを水平方向に拡張することが
- **99.999% の可用性**: MySQL Cluster は、シェアードナッシング型
性を発揮できるよう設計されています。これにより、障害回復力が
ンス作業が可能になります。
- **SQLおよびNoSQLのAPI**: MySQL Cluster を使用すれば、リレ
方の長所を組み合わせて、コストとリスク、複雑さを軽減するソリュ
- **リアルタイムのパフォーマンス**: MySQL Cluster はリアルタイムのレ
厳しいWebアプリケーションや通信アプリケーション、エンタープライス
- **アクティブ/アクティブ遠隔地レプリケーション**: 遠隔地レプリケーシ
スのスケーラビリティ確保の目的で複数のクラスタを地理的に分散
- **オンラインでのスケーリングとスキーマのアップグレード**: 継続的な
オンラインでのノードの追加や稼動中のデータベース・スキーマの更
ダイナミックなワークロードにも対応できます。
- **MySQL Cluster Manager**: MySQL Cluster Managerは一般
Cluster Carrier Grade Editionデータベースの作成と管理を簡素
- **24時間365日のテクニカル・サポート**: Oracle Premier Support
MySQLサポート・エンジニアに24時間365日、直接お問い合わせ
ションの開発、運用、管理に関してお客様をいつでも支援できる能

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# MySQL Cluster Manager

## DevOpsの柔軟性と効率の向上、停止時間の削減

### 自動管理

- 個別のノードやクラスタ全体の起動/停止
- オンラインでのノード追加
- オンラインでの設定変更
- オンラインでのアップグレード
- オンラインバックアップ & リカバリ
- 既存のクラスタのインポート

### Self-Healing

- ノード監視
- SQLノードや管理ノードを含めた自動リカバリ

### 高可用性の管理

- クラスタ全体の設定の一貫性を管理
- 設定の永続化
- HA エージェント

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# MySQL Cluster Managerの導入効果

## MySQL Cluster 7.1から7.4へのアップグレード

### MySQL Cluster Manager 導入前

- 1 x クラスタの事前状態チェック
- 8 x ssh コマンド/サーバー
- 8 x 停止コマンド/プロセス
- 4 x 構成ファイルの転送 - scp
  (2 x mgmd & 2 x mysqld)
- 8 x プロセスごとの開始コマンド
- 8 x 開始または再参加プロセスの確認
- 8 x 完了確認処理
- 1 x クラスタ全体の完了確認
- 各構成ファイルの手動による編集を除く

**合計: 46 コマンド-**
**2.5 時間の作業**

### MySQL Cluster Manager 導入後

upgrade cluster --package=7.4 mycluster;

**合計: 1 コマンド -**
**完全自動処理**

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# MySQL Enterprise Support

- 最大のMySQLのエンジニアリングおよびサポート組織
- MySQL開発チームによるサポート
- 29言語で世界クラスのサポートを提供
- メンテナンス・リリース、バグ修正、パッチ、アップデートの提供
- 24時間x365日サポート
- 無制限サポート・インシデント
- MySQL コンサルティング・サポート

Get immediate help for any MySQL issue, plus expert advice

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# MySQL Supportの特徴

- 「パフォーマンス・チューニング」や「SQLチューニング」まで通常サポートの範囲内
  - コンサルティングサポートが含まれており、「クエリ・レビュー」、「パフォーマンス・チューニング」、「レプリケーション・レビュー」、「パーティショニング・レビュー」などに対応可能
  - 詳細はこちらを参照下さい http://www-jp.mysql.com/support/consultative.html
- ソースコードレベルでサポート可能
  - ほとんどのサポートエンジニアがソースを読めるため、対応が早い開発エンジニアとサポートエンジニアも密に連携している

サポート、
コンサルテーティブ・サポート
は共に回数制限が無い為、
リモートDBAとして活用頂く事で、
自社内の調査・検証工数を大幅
に削減する事が可能。TCO削減
が可能です。

環境に応じた、MySQL Cluster
の最適なパラメータに関しても、
お問い合わせ頂けます。

ORACLE

Copyright © 2019 Oracle and/or its affiliates. All rights reserved. |

# Integrated Cloud

## Applications & Platform Services

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

ORACLE®