MySQL Cluster 技術入門

松信 嘉範 (MATSUNOBU Yoshinori)
MySQL株式会社
シニアコンサルタント
ymatsunobu@mysql.com

Agenda

- MySQL社の紹介
- MySQL Cluster技術解説

MySQL社の紹介

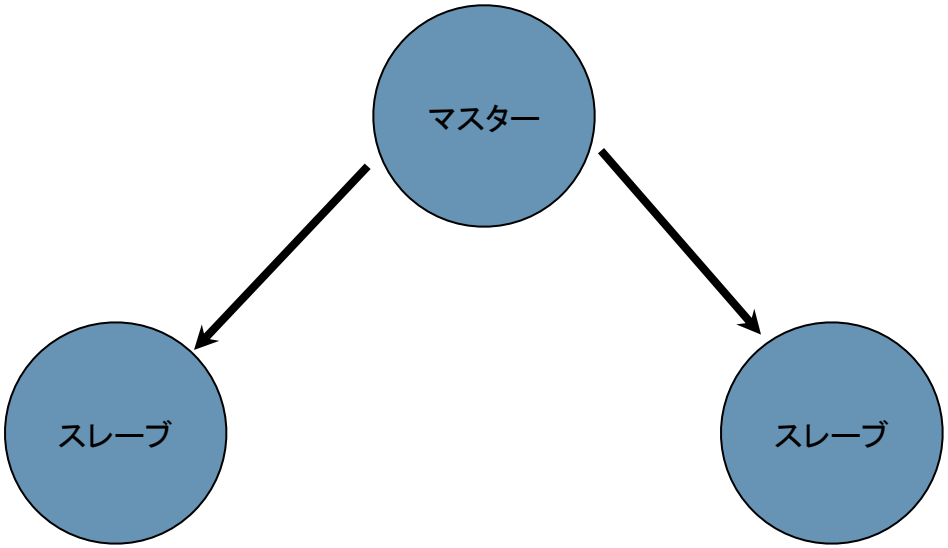
- 1995年に設立、日本法人は2006年2月に設立
- 社員数約400名、世界24カ国に在籍
- 海外での圧倒的な人気
  - 1,000万以上のインストールベース
  - 5万ダウンロード/日
- TCOの削減を実現
  - 浮いた費用を用いて品質を向上させるアプローチも可能
- 豊富な周辺ツール
- 多数のパートナー企業
  - 住商情報システム様、HP様、NTTコムウェア様、スマートスタイル様、NRI様他

MySQLを利用している企業

- **High Volume Websites**
  - Web 2.0
  - Dynamic content
  - eCommerce
  - ”Look to Book”
  - Session Management
  - Gaming & entertainment
  - Scale Out
- **Enterprise**
  - Data Warehousing
  - High-Volume OLTP
  - Scale Out
- **Embedded**
  - Bundled in software applications & hardware components

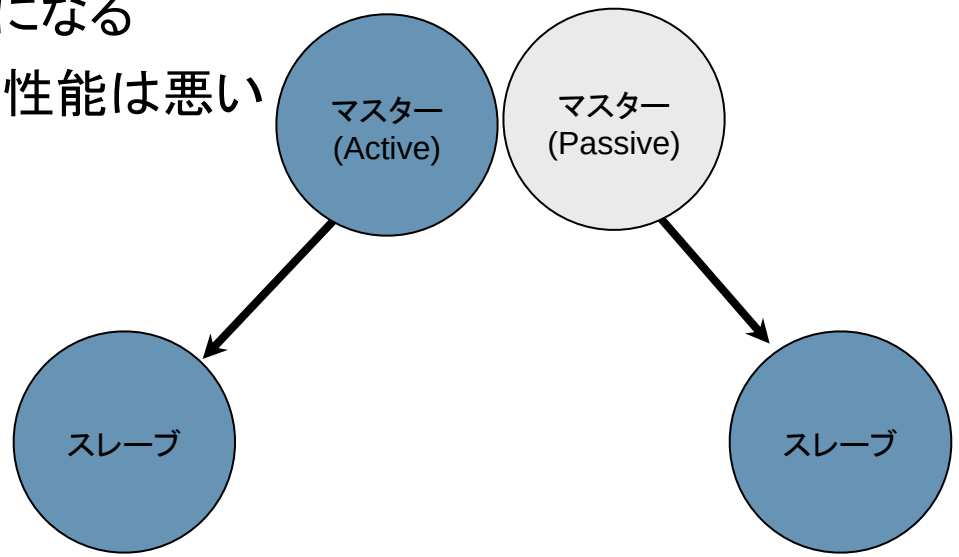
先のセッションの復習(1)

- MySQL Replication
  - MySQLが標準機能として持っている
  - マスター/スレーブ型の非同期レプリケーション
  - 参照のスケールアウトが可能 (L/Bによるスレーブ分散が一般的)
  - セットアップが非常に簡単
  - マスターのHAの仕組みを標準では持っていない
  - マルチマスターは可能だが、多少複雑で一般的ではない
    - 特にフェイルバックは難しい



先のセッションの復習(2)

- Heartbeat + DRBD/SAN + MySQL Replication
  - マスターのHAを実現
  - Active/Standby型クラスタ構成
  - DRBDを使えば共有ディスクは不要
  - アプリケーションから見て透過的にフェイルオーバー可能
  - スレーブは別途追加可能
  - 停止検知→完全停止→fsck→マウント
    →mysqld起動までダウンタイムになる
  - 起動後、キャッシュに乗るまでの性能は悪い
  - 99.99%の可用性(目安)



MySQL Clusterの特徴(1)

- MySQLのストレージエンジンの1つ「NDB」
  - InnoDBやMyISAMなどのかわりに使う。共存も可能
  - CREATE TABLE xxx (xxx) ENGINE=NDB;
- 最初のバージョンは、2004年にMySQL4.1としてリリース

- インメモリで動作するため非常に高速
  - テーブルデータをメモリ上に置くということ
  - 5.1からはデータ領域をディスクに置くことも可能
    - インデックス領域は必ずメモリ上に置かれる(高速化のため)
- mysqldを数多く並べることでスケールアウト可能
- 「NDB API」を直接使うことで、超高速化が可能

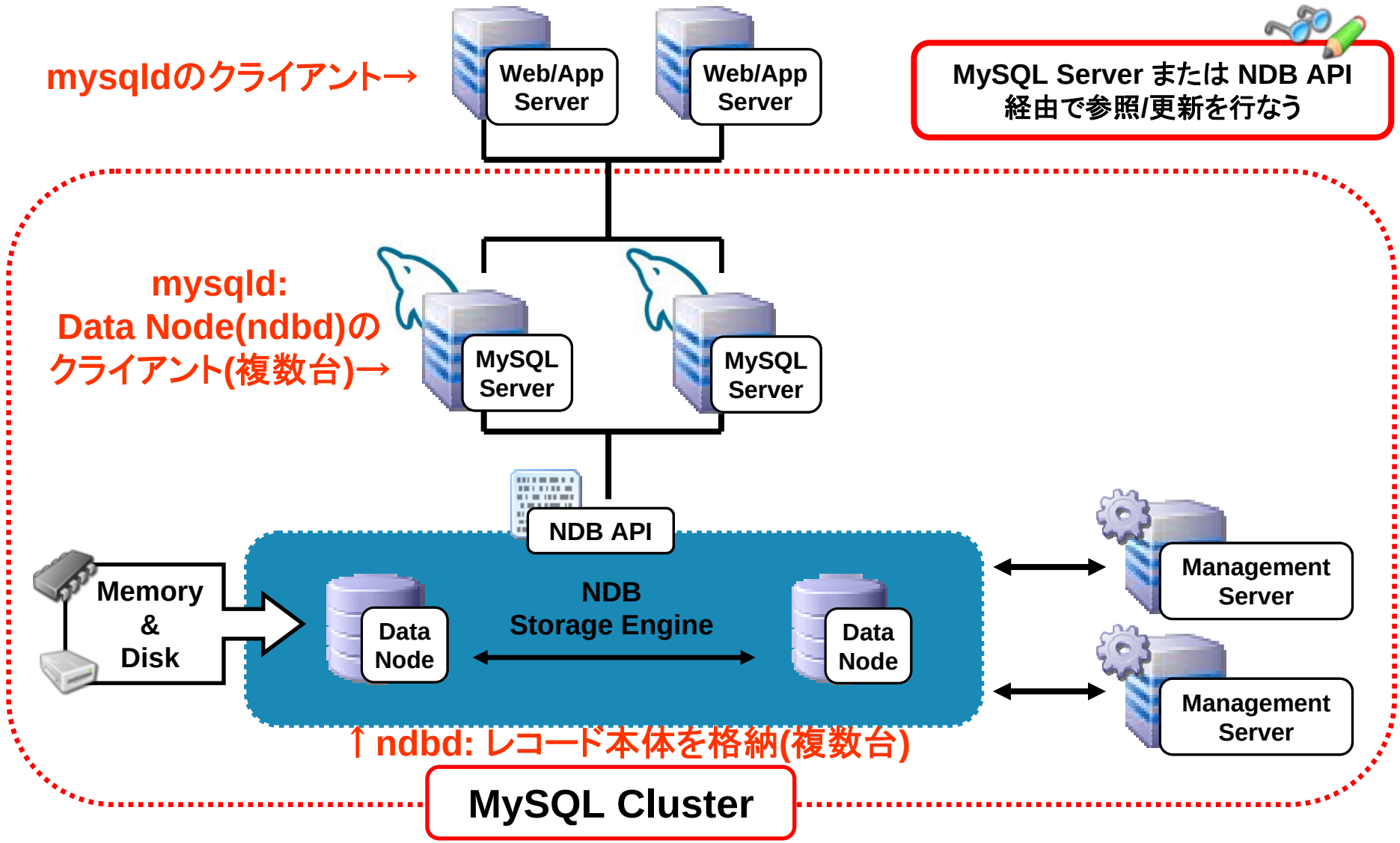
- トランザクション対応
- 行レベルロックとMVCCをサポート

MySQL Clusterの特徴(2)

- シェアードナッシング型構成
- 特別なハードウェアは不要
  - Data Nodeは64bit機を強く推奨
  - 共有ディスクは不要

- 同期レプリケーションをしている
- 高速なフェイルオーバー(秒単位のレベル)
- オンラインバックアップが可能

MySQL Clusterの基本アーキテクチャ



MySQL Clusterの構成要素

- データノード(Data Node)
  - レコードの保管場所
  - 複数台で同期レプリケーションされる
  - レプリケーションの仕組みは、mysqldが持っているレプリケーションとは別物
  - 複数台並べて冗長性を高める
- SQLノード(SQL Node)
  - SQL文の構文解析等を担当するノード
  - 通常のmysqldプロセスが該当
    - NDB APIを用いて自作プログラムを書くことも可能
  - NDBストレージエンジンでは、レコードをここでは持たない
  - 複数台並べてスケールアウト可能
- 管理ノード(Management Nodes)
  - Data NodeとSQLノードの起動時やフェイルオーバー時などに必要

ハードウェア/OS要求スペック

論理ノード数	・最低3ノード ・最大64ノード、48Data Node
CPU	・x86_32またはx86_64、PowerPC、SPARC ・メモリ要求から、64ビットを強く推奨
メモリ	・データサイズやData Node数、レプリケーション設定に依存 ・Data Node1個あたり16GB～32GB程度が一般的
ディスク	・Data Nodeでも書き込みはある。RAID5は避ける
ネットワーク	・Gigabit Ethernetを推奨 ・Dolphin Expressにより高速化が可能
OS	・Linux、Solaris、HP-UX、AIX等 ・Windowsは現在未サポート

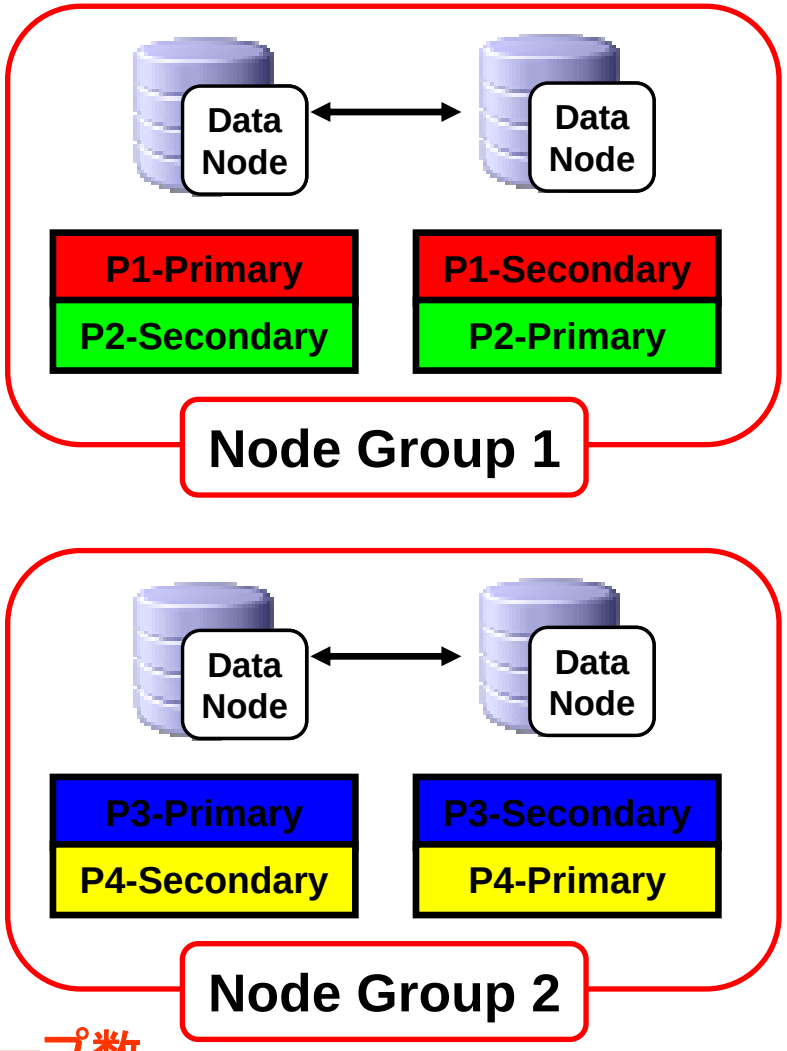
アプリケーションとの相性

テーブルサイズ	・メモリ上におさまる範囲 ・漸増型テーブルはハウスキーピングを考慮 ・5.1ではディスク上への保存も可能
フェイルオーバー	・Data Nodeは自動でフェイルオーバー ・数秒単位でのフェイルオーバーが可能 (99.999%の可用性)
リカバリ	・起動時に自動的に最新の状態に復旧
SQL文との相性	・主キー検索、一意キー検索が非常に高速 ・ジョインは避ける
データ型	・固定長型(INTEGER等)は良い ・TEXT/BLOB型は避ける ・VARCHAR型は5.0まで固定長として扱われる ・キャラクタセットと占有バイト数を考慮 (VARCHAR(100) CHARSET latin1/utf8)

Data Nodeのアーキテクチャ

テーブルデータは分散配置される
(主キー値からの線形ハッシュ計算による割当)

ID	Capital	Country	UTC	
1	Copenhagen	Denmark	2	Partition 1
2	Berlin	Germany	2	
3	New York City	USA	-5	Partition 2
4	Tokyo	Japan	9	
5	Athens	Greece	2	Partition 3
6	Moscow	Russia	4	
7	Oslo	Norway	2	Partition 4
8	Beijing	China	8	



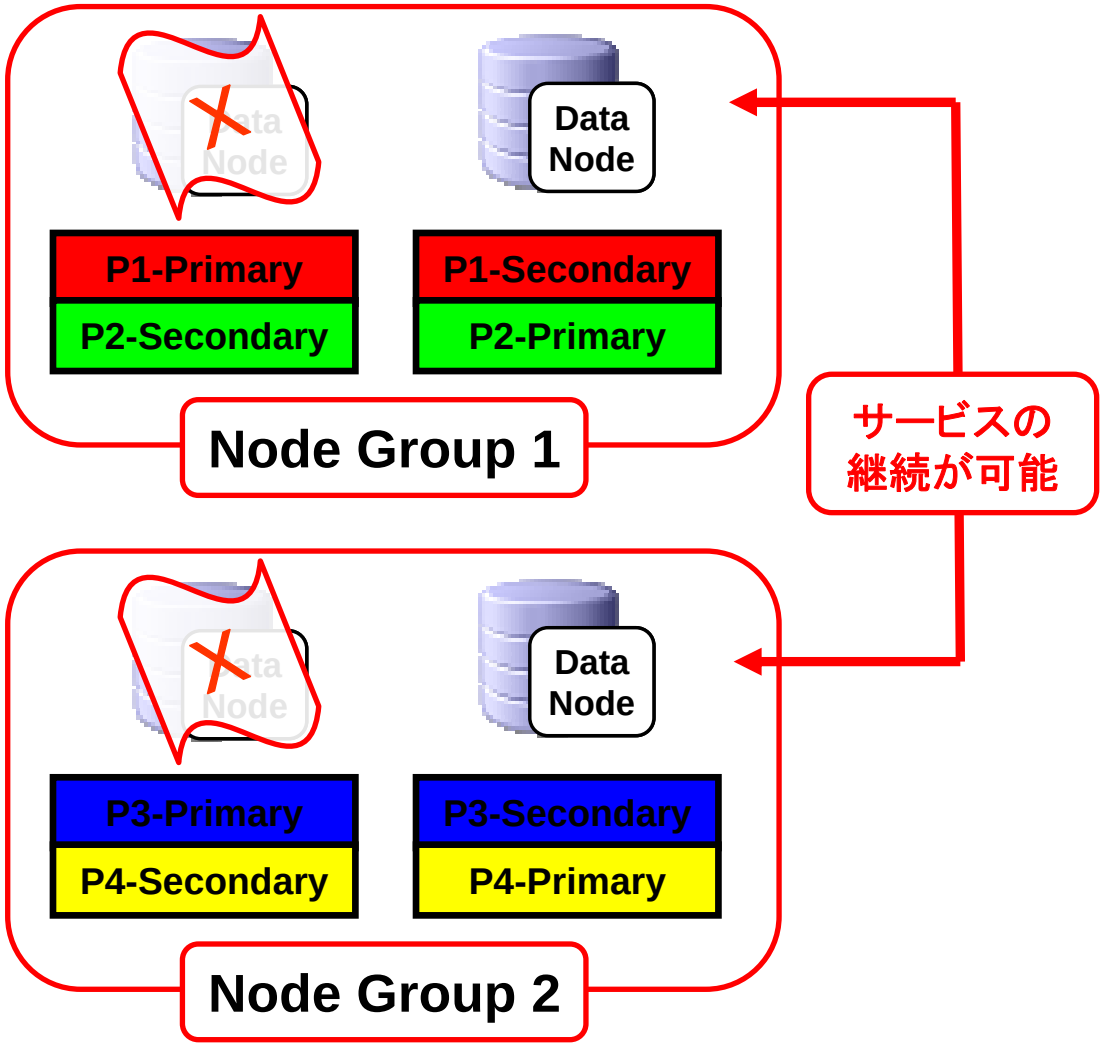
- Data Node(4つ)
- レプリカ単位(2つ)
- ノードグループ(2つ)

Data Node数 = レプリカ単位 × ノードグループ数

Data Nodeのアーキテクチャ(2)

ID	Capital	Country	UTC
1	Copenhagen	Denmark	2
2	Berlin	Germany	2
3	New York City	USA	-5
4	Tokyo	Japan	9
5	Athens	Greece	2
6	Moscow	Russia	4
7	Oslo	Norway	2
8	Beijing	China	8

Partition 1
Partition 2
Partition 3
Partition 4



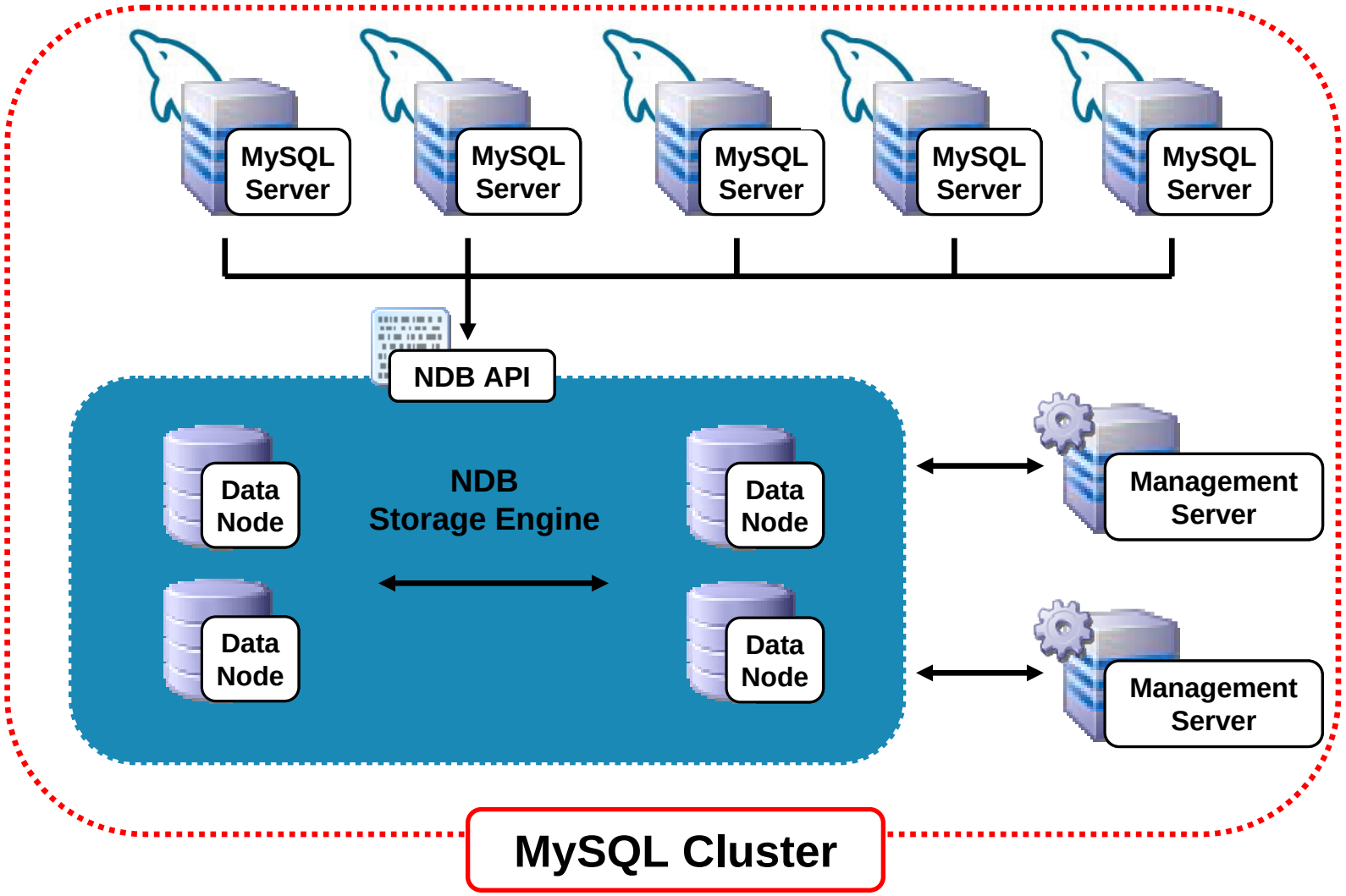
ノードグループ内の全ノードが
全滅しない限り
サービスの継続が可能

設定方法

- config.ini
  - MySQL Clusterの設定ファイル
  - 全体で1個
  - my.cnfとは別物 (my.cnfは別途SQL Nodeごとに作成する)
  - Management Nodeの起動時に読み込まれる

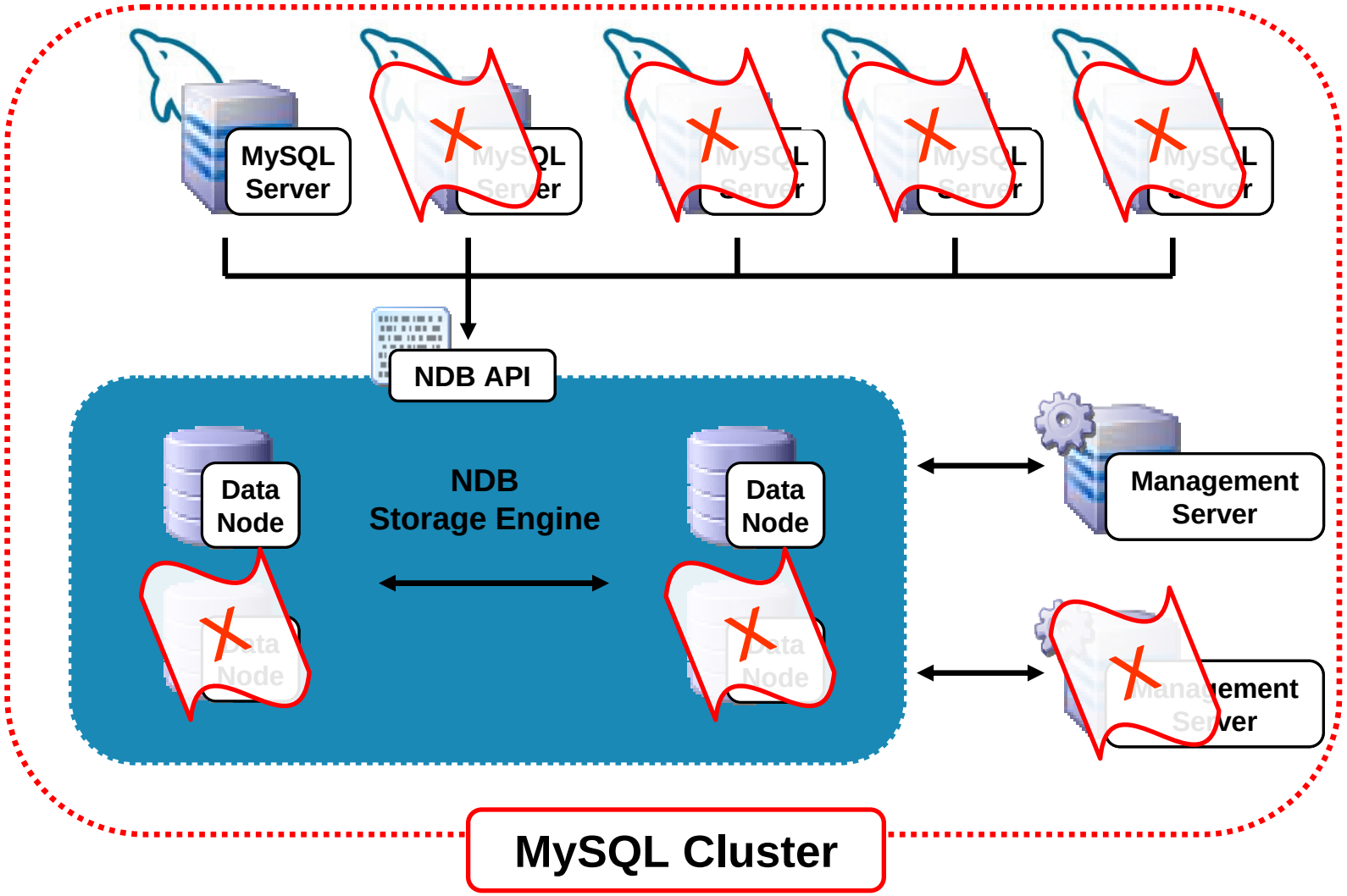
```
[NDBD DEFAULT]
NoOfReplicas=2
DataMemory=10G
IndexMemory=7G

[NDBD]
Id=1
HostName=ndbd_ip
DataDir=/path/to/ndbd

[NDBD]
Id=2
HostName=ndbd_ip
DataDir=/path/to/ndbd
…
```

MySQL Clusterによるスケールアウト



アプリケーションパーティショニングよりも簡単にスケールアウト可能

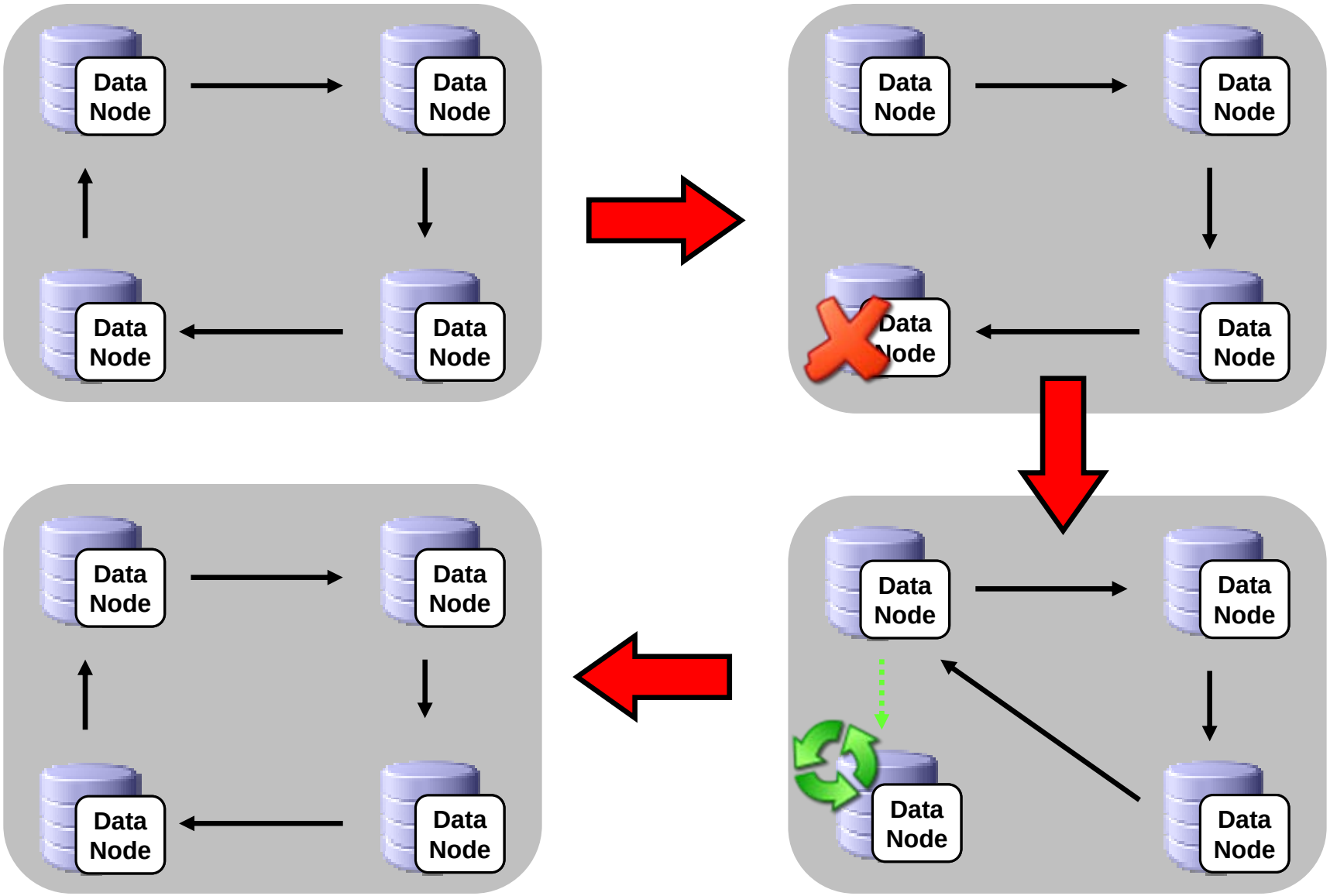
MySQL Clusterによるスケールアウト(フェイルオーバー時)



耐障害性

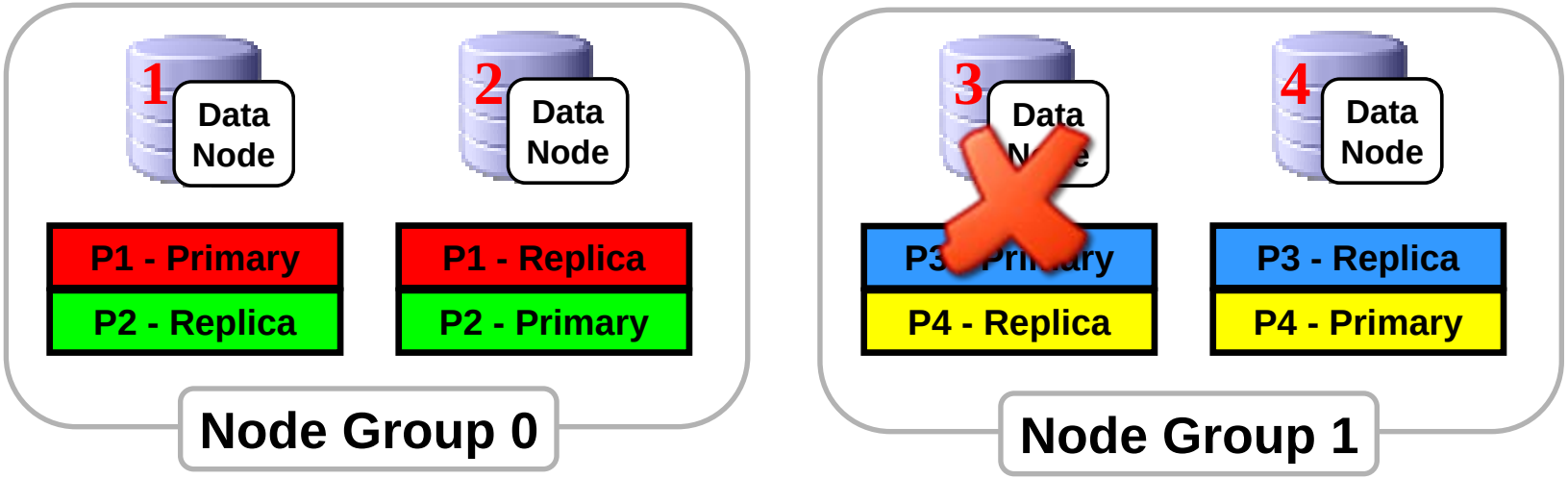
	サービス継続	コネクション継続	トランザクション継続
SQL Node障害	○	△	△
Data Node障害	○	○	○
Management Node障害	○	○	○
SQL-Data Node間障害	○	○	×
Data-Data Node間障害	○	○	×

Data Node障害の検知メカニズム



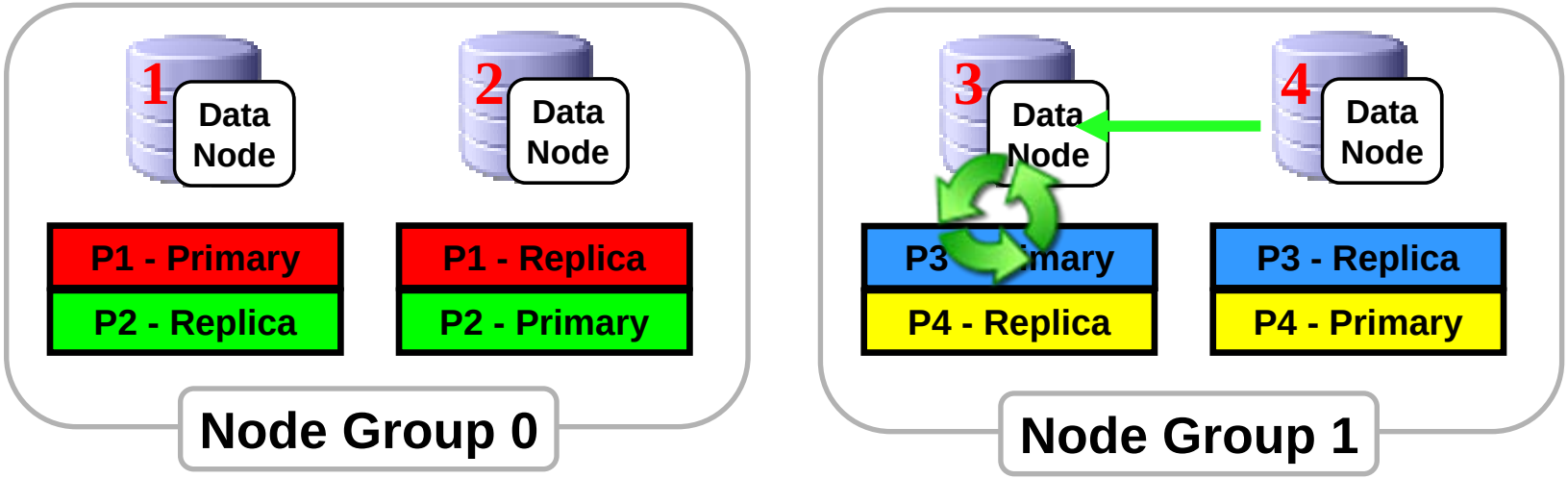
Data Nodeの障害とリカバリ

- *Server 3* の障害が検知される
- *Server 4* が *Server 3* への処理要求も処理する



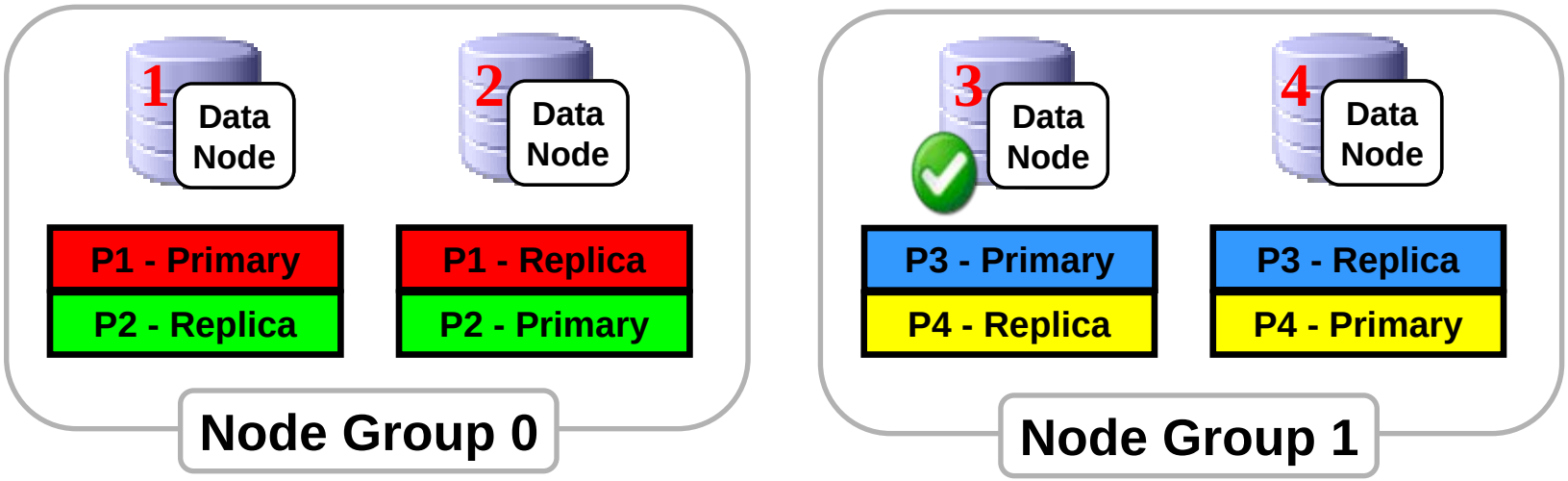
Data Nodeの障害とリカバリ

- *Server 3 は Server 4 をコピーする*
- 障害後の差分をコピー



Data Nodeの障害とリカバリ

- リカバリが終われば、通常通り処理要求をさばくようになる



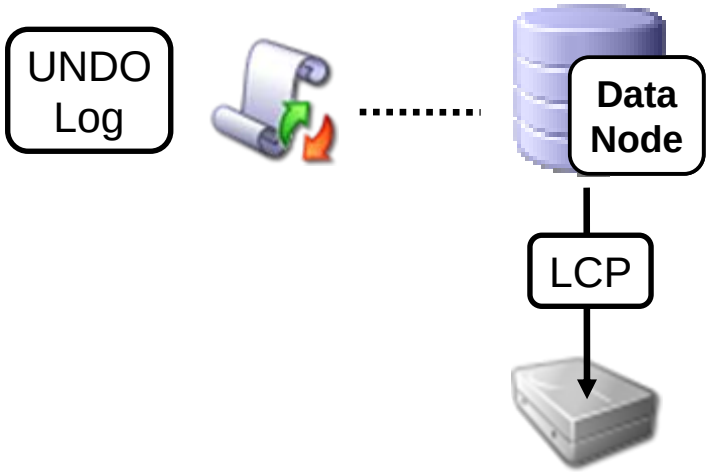
データの永続性

- インメモリデータベース
- 本体プロセスが停止したらデータは消失する?
  - MEMORYストレージエンジンなら消失する
- 消失したら困るケースがほとんど
- 消失しない仕組みを持っている
- 定期的にディスクに書き込む
  - チェックポイント処理
    - ローカルチェックポイント(LCP)
    - グローバルチェックポイント(GCP)

チェックポイント処理

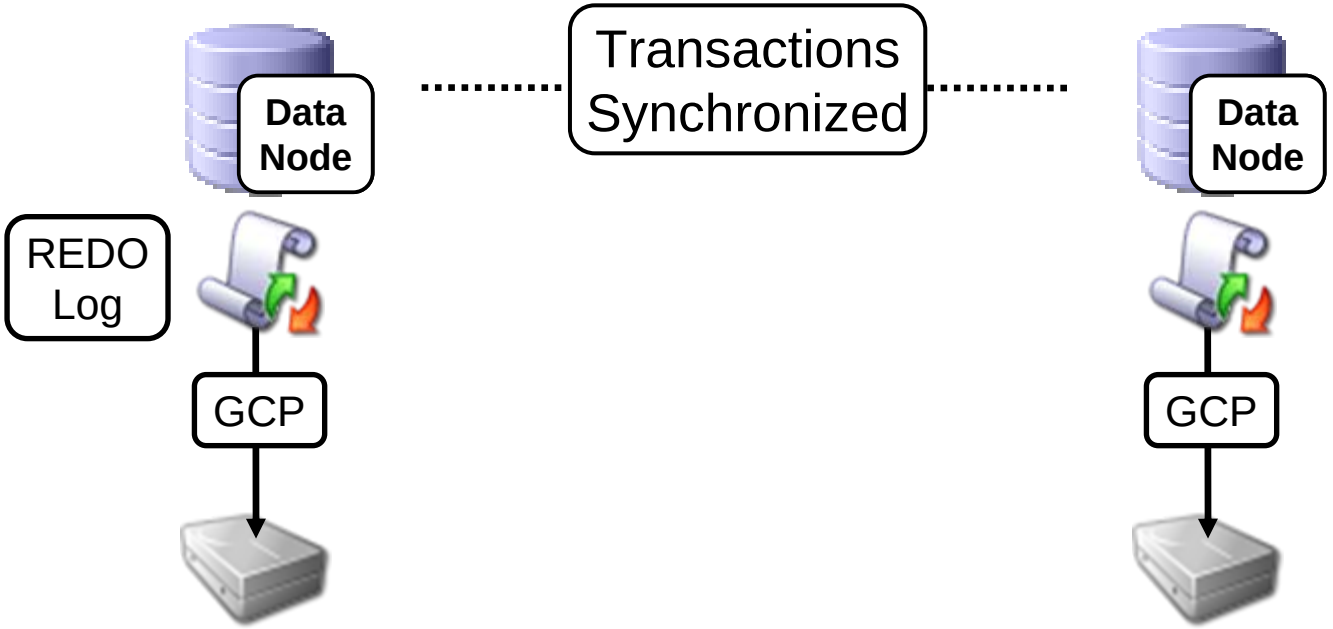
メモリの中身をディスクに書き出す処理

- ローカルチェックポイント (LCP)
  - 全テーブルデータをディスクに書き出す
  - 全Data Node上で実行
  - 更新/参照処理を一切ブロックしない
  - 設定パラメータで間隔を制御可能
  - シャットダウン時にも実行される
  - 起動時はLCPによって生成されたファイルを読み込む



チェックポイント処理

- グローバルチェックポイント (GCP)
  - トランザクションのコミット情報をディスクに書く
  - コミット時同期書き込みではない
    - 別ノードに同期レプリケーションすることで代用
  - 設定パラメータで間隔を制御可能 (デフォルト2秒)



オンラインバックアップ・リカバリ

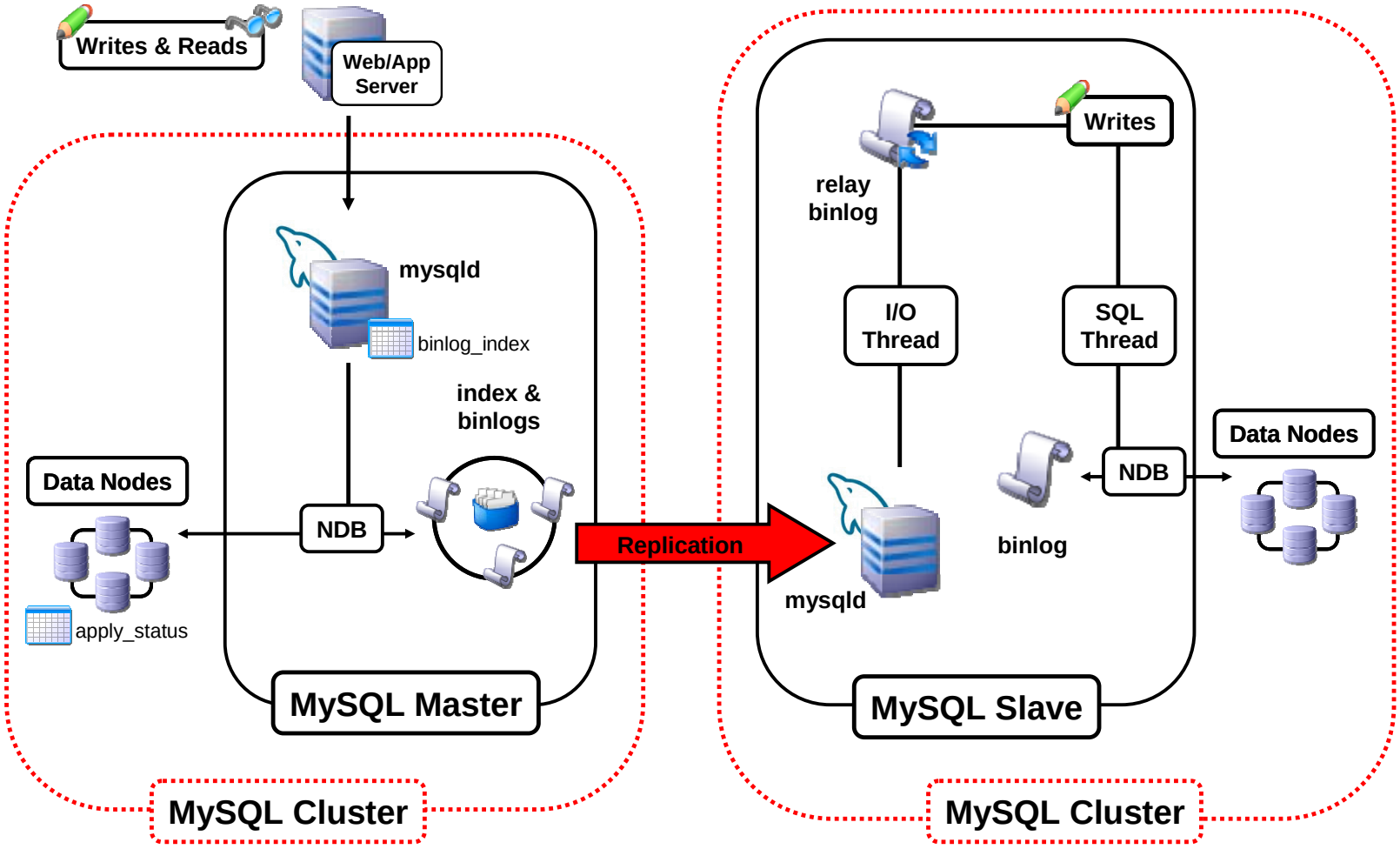
バックアップ

- オンラインバックアップが可能
- Management Nodeで実行
- 各Data Nodeにバックアップ指示が行く
- 各Data Nodeのローカルディスク上にバックアップ
- 参照/更新ともブロックしない

リストア

- `ndb_restore -m` でメタデータをリストア
- `ndb_restore -r` で実データをリストア
- 全Data Nodeで繰り返す

MySQL Cluster間のレプリケーション(5.1-)



主にディザスタリカバリーの用途

MySQL Cluster NDB API

- NDBストレージエンジン(Data Node)に直接アクセスするためのC++ API
- Java版もある (NDB/J)
- ODBCやJDBCなどの標準には準拠していない。独自仕様
- mysqldを必要としない (mysqldはNDB APIを用いてData Nodeにアクセスしている)
- Web上でAPIが公開されている

http://dev.mysql.com/doc/ndbapi/en/index.html

NDB APIの例

```
myTransaction = myNdb->startTransaction();
myOperation = myTransaction->getNdbOperation(myTable);

myOperation->readTuple();
myOperation->equal("ATTR1", i);

myRecAttr = myOperation->getValue("ATTR2", NULL);
if (myRecAttr == NULL) APIERROR(myTransaction->getNdbError());

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

printf(" %2d    %2d¥n", i, myRecAttr->u_32_value());
myNdb->closeTransaction(myTransaction);
```

MySQL Clusterのチューニング

- サーバ構成
  - Data NodeとSQLノードの台数分散
  - Data Nodeの動的追加は運用上難しい
- ジョインは避ける
- my.cnf
  - table_cache
  - ndb_force_send
  - ndb_use_exact_count
  - engine_condition_pushdown
- config.ini
  - DataMemory
  - IndexMemory
  - NoOfReplicas
  - NoOfFragmentLogFiles
  - TimeBetweenLocalCheckpoints
  - TimeBetweenGlobalCheckpoints
  - DiskCheckpointSpeed (5.1)
  - MaxNoOfConcurrentTransactions
  - MaxNoOfConcurrentOperations
  - MaxNoOfAttributes
  - MaxNoOfTables
  - MaxNoOfOpenFiles
  - RedoBuffer
  - SendBufferMemory

DBT-2ベンチマークの結果

- TPC-Cと類似したOLTP型ベンチマーク
- ベンチマーク結果 (Ethernet, 中間報告)
  - 2台のマシンにndbd, mysqld同居　→ 22,000tpm
  - 2台にndbd、4台にmysqld　→70,000tpm
  - mysqldの配置によってはもっと高い値を出せると思われる
  - InnoDBやMyISAM1台構成で70,000tpmは不可能 →スケールアウト
- 変動要因は多い
  - アプリケーションの性質
  - テーブル設計
  - SQL文
  - パラメータ設定
  - サーバ台数と配置
  - ネットワークカード
- チューニングは重要
  - コンサルティングやトレーニングは有効な手段

MySQL Clusterに向かないもの

- NDBテーブル間のジョイン
  - ジョインは極力避けるようにテーブル設計とクエリを考慮

- レコードサイズが急増し、退避もできないもの
  - Data Nodeの追加は、長時間のシステム全面停止を伴うので難しい

  ※Data Node数固定で、メモリを増設するのであればサービス停止無しで DataMemory,IndexMemoryの増加が可能
    - メモリの動的追加が可能なハードウェアが比較的安価で購入可能
  - ノード数が多ければ、その分1ノードあたりのメモリ消費量は少なくなる
  - 最初の見積が大事
    - サイジングをする
    - ndb_size.pl
  - サイズの大きな列を排除できればかなり多くのレコードを扱える
  - ハウスキーピングが理想

- 巨大なTEXT/BLOBデータ
  - テーブルを分けて別のストレージエンジンで扱うと良い

MySQLコンサルティングサービス

- MySQLアーキテクチャデザイン
- MySQL Cluster Jumpstart
- MySQL Scale-out, HA and Replication Jumpstart
- MySQLパフォーマンスチューニング
- MySQLデータベース管理
- MySQLマイグレーション
- MySQL Time Hire
- その他、MySQLの専門知識が必要なもの全般

http://www-jp.mysql.com/consulting
consulting-jp@mysql.com

MySQL User Conference Tokyo

- 2007年9月11日(火)、12日(水)
- http://www-jp.mysql.com/news-and-events/news/article_1325.html
- 日本科学未来館(東京・お台場)にて開催
- Marten Mickos、Brian Akerらが来日予定
- 事前申込は7月下旬～8月上旬頃より開始予定

Case Study: Alcatel-Lucent

Leader in fixed, mobile and converged broadband networking, IP technology, applications and services

Products	Requirements
HLR/AuC **IM-HSS** **UMA** **AAA**	• Performance of read/write queries • ATCA hardware support • Small footprint • Flexible architecture for administration of application and database • Scalable to millions of subscribers • ACID transactions • Local and geographic redundancy • Ability to modify schemas online • Low TCO

http://www.mysql.com/why-mysql/case-studies/mysql_cs_alcatel.pdf