
ORACLE®

MySQLの高可用性ソリューション

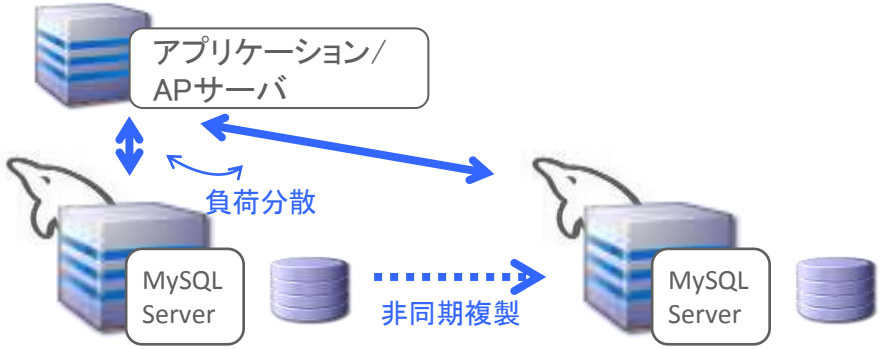
Yoshiaki Yamasaki / 山崎 由章
MySQL Senior Sales Consultant, Asia Pacific and Japan

Safe Harbor Statement

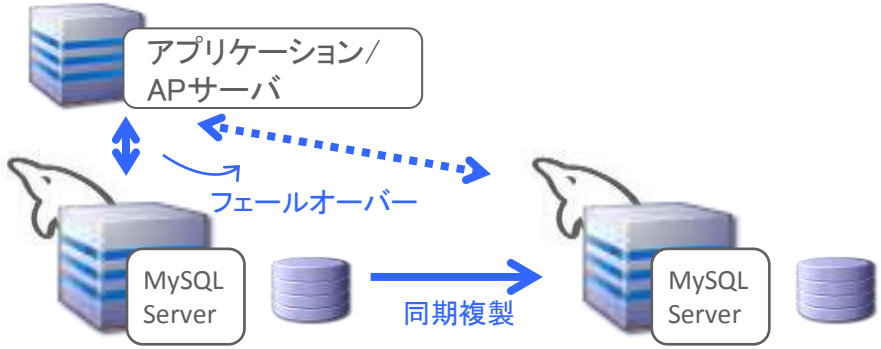
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

MySQLの高可用性構成のパターン

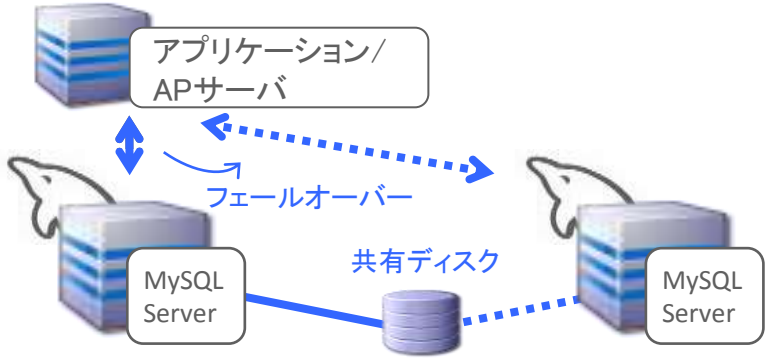
- レプリケーション（標準機能）
  非同期&準同期データレプリケーション



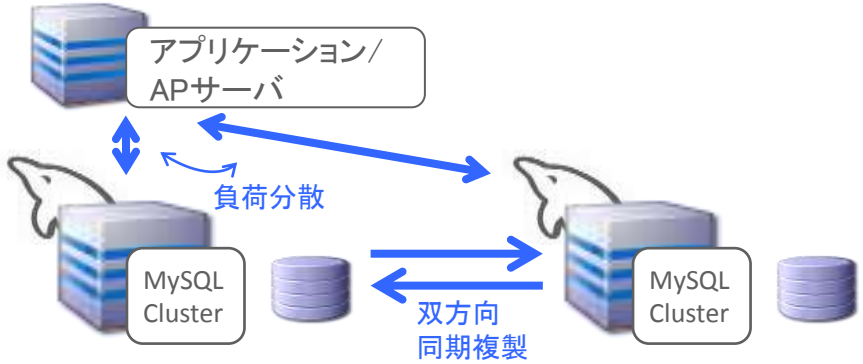
- MySQL+DRBD
  Linux用のノード間データコピー



- Oracle Clusterwareなど
  共有ディスクにデータを格納

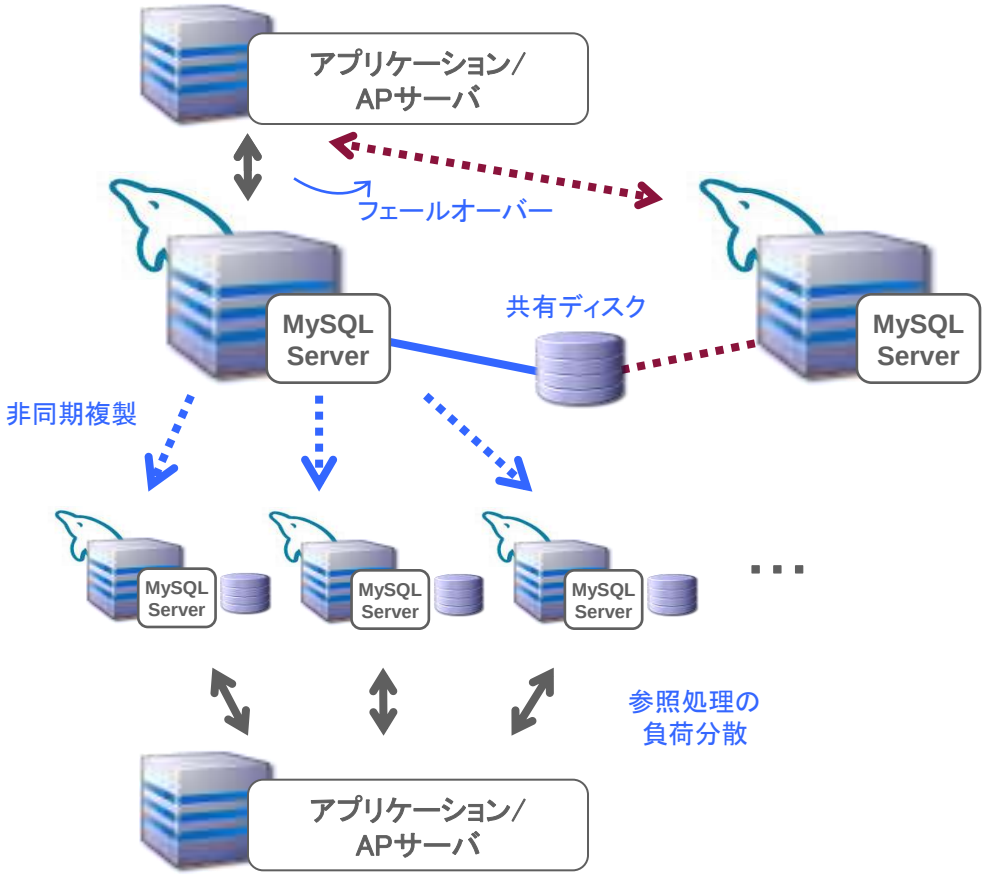


- MySQL Cluster
  シェアードナッシング型高性能クラスタ

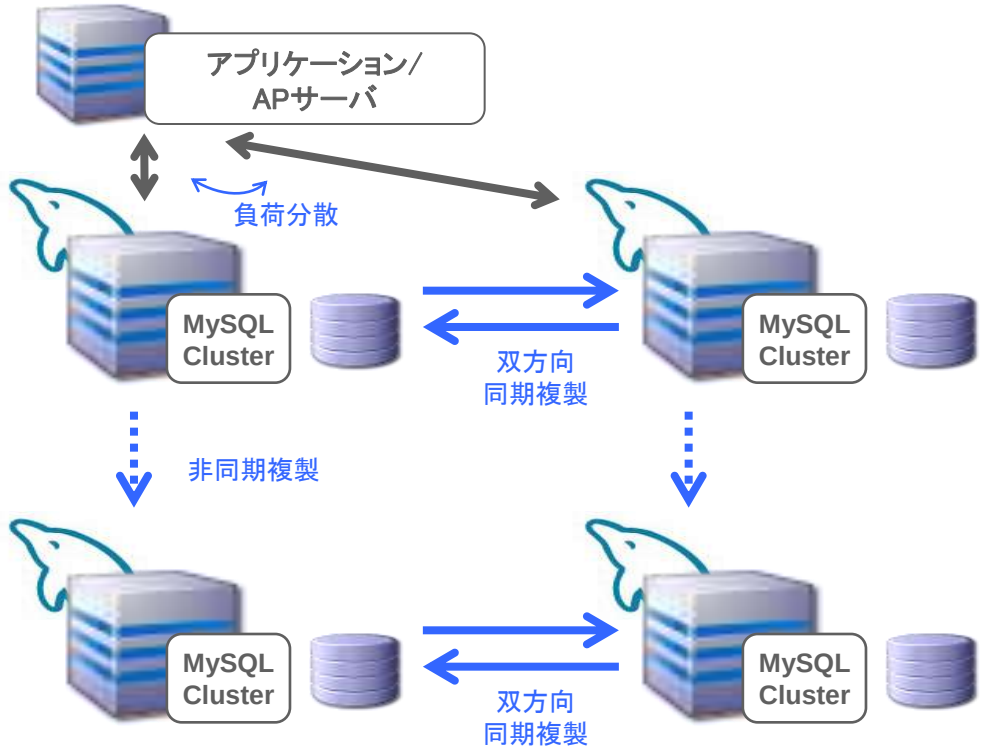


複合型の高可用性構成例

- 共有ディスク型構成＋レプリケーション



- MySQL Cluster＋レプリケーション



レプリケーションによる高可用性構成

- メリット
  - MySQLの標準機能だけで実現でき、共有ディスクや特別なソフトウェアも不要であるため、コスト(H/Wコスト、ソフトウェアコスト)が低い
  - 参照処理の負荷分散と高可用性構成を同じ仕組みで実現できる
- デメリット
  - 比較的、運用上コストがかかる
    - 障害発生時に、どのようにしてフェイルオーバー処理を実行するか？
    - フェイルオーバーによってMySQLサーバーの構成が変わった場合、アプリケーションからMySQLサーバーへの接続先を切り替える必要がある
    - (MySQL 5.5以前の場合) スレーブがクラッシュセーフでないため、スレーブに障害が発生するとスレーブを再構築しないといけない場合がある

レプリケーションによる高可用性構成

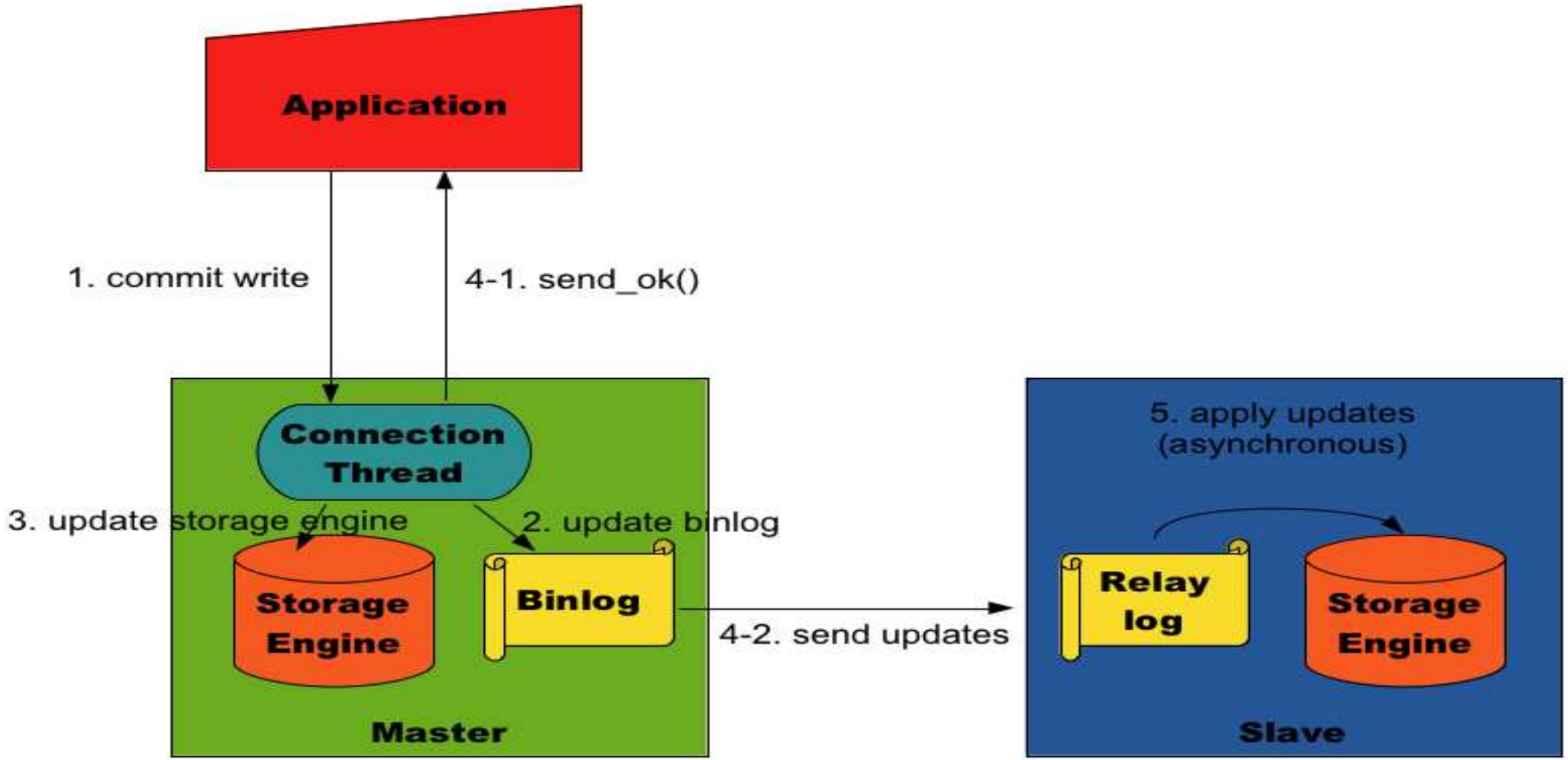
- デメリットに対する改善機能
  - フェイルオーバー処理の自動化
    - GTIDモードでレプリケーションを構成すると、MySQL Utilities内のmysqlfailoverを使用することで、自動フェイルオーバーが可能になる(mysqlfailoverの実体はPythonスクリプト)
  - アプリケーションからの接続先切り換えの必要性
    - MySQL Utilities内のMySQL Fabricを使用することで、フェイルオーバー処理を自動化でき、フェイルオーバー後にもアプリケーションからの接続先変更が不要になる(内部的に、GTIDモードによるレプリケーションを使用)
  - (MySQL 5.5以前の場合) スレーブがクラッシュセーフでないため、スレーブに障害が発生するとスレーブを再構築しないといけない場合がある
    - MySQL 5.6で以下のパラメータを設定することで、クラッシュセーフなスレーブが実現できる(※)
      - relay_log_recovery = ON
      - relay_log_info_repository = TABLE

※マスター/スレーブ共に、InnoDBを使用する必要あり

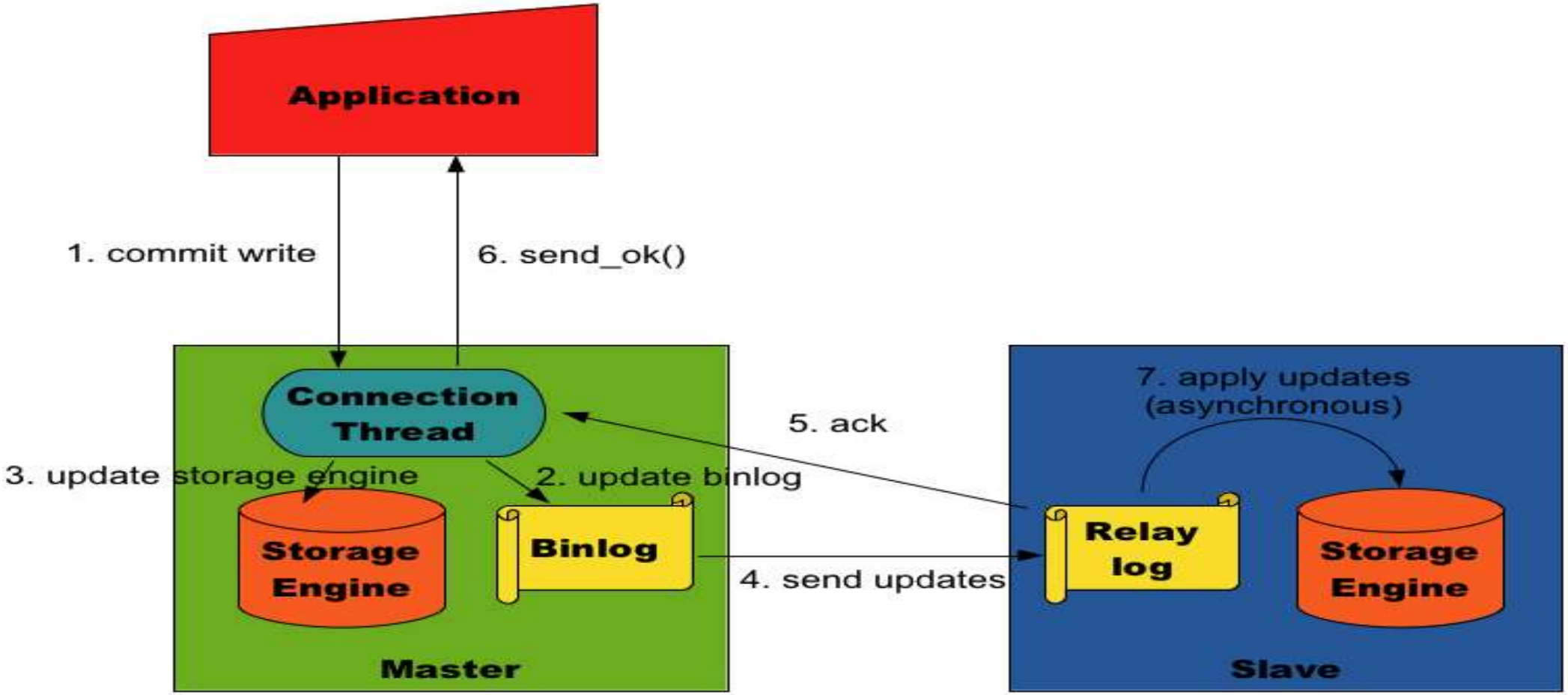
レプリケーションによる高可用性構成

- 非同期と準同期の違い
  - 非同期(デフォルト)
    - マスターでの更新処理と、更新内容をスレーブに伝搬する処理は非同期で行われる
      ⇒マスターに障害が発生した際、障害発生直前の更新内容がスレーブに
      伝搬できていない可能性がある
    - マスターでの更新処理は、準同期よりも早く完了する
  - 準同期
    - 更新内容がスレーブに伝搬されてから、マスターでの更新処理が完了する
      ⇒マスターに障害が発生した際、障害発生直前までの更新内容がスレーブに
      伝搬されていることが保証される
    - マスターでの更新処理は、非同期より遅くなる

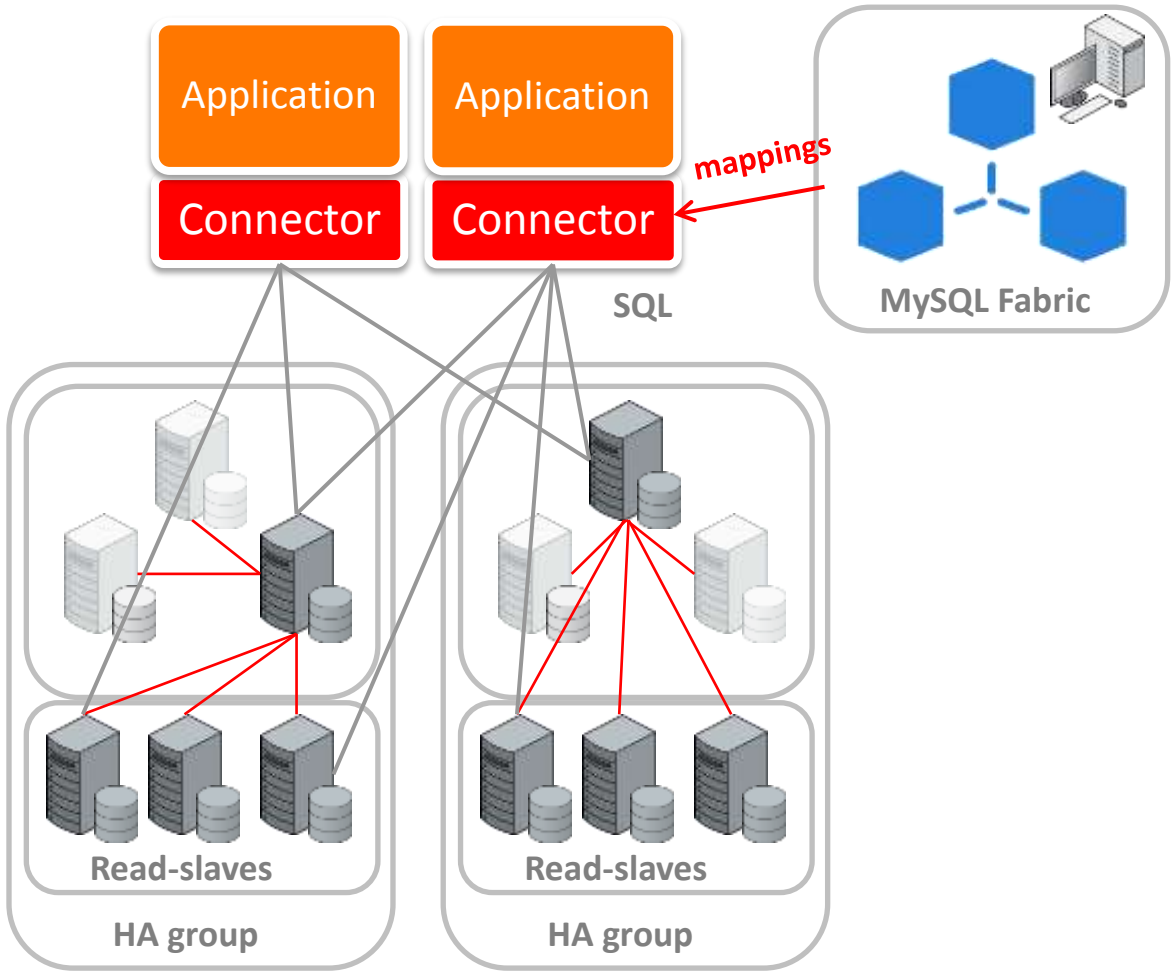
非同期レプリケーション



準同期レプリケーション



MySQL Fabric 1.5: 高可用性 & シャーディング



- OpenStack との統合
- 高可用性
  - サーバの監視; スレーブの自動昇格と透過的なレプリケーション切り替え
- シャーディングによる拡張性
  - アプリケーションがシャードのキーを提供
    - 整数型、日付型、文字列型
  - レンジまたはハッシュ
  - シャード再構成可能
- Fabric対応コネクタ利用: Python, Java, PHP, .NET, C (labs)
  - プロキシを使わないので低レイテンシ、ボトルネック無し

MySQL+DRBDによる高可用性構成

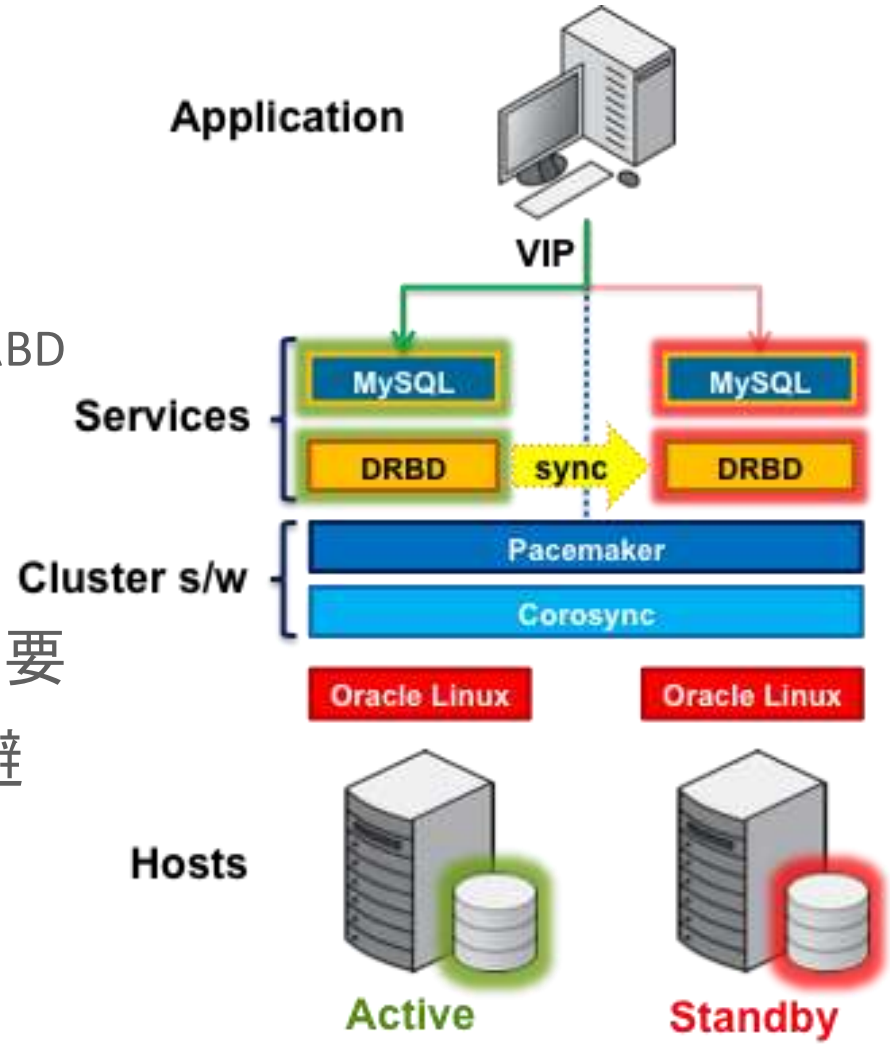
- メリット
  - 共有ディスクが不要であり、比較的安価にアクティブ/スタンバイの高可用性構成が組める
- デメリット
  - 比較的、運用コストがかかる
    - 障害発生時に、どのようにしてフェイルオーバー処理を実行するか？
      - DRBDで同期しているディスク領域と同期していないディスク領域が混在することにも注意が必要
    - プライマリ/スタンバイで同期が取れなくなった場合の対応
      - MySQL以外に、DRDBについても知識が必要

High Availability with Oracle Linux

Oracle Linux ＋ DRBD Stack

- 認定構成だからこそ実現できる、
  Oracleによるフルスタックサポート
  - Oracle Linux Unbreakable Enterprise Kernel R2 に統合されたDRBD
  - Oracle Linux 6.2以上で使用可能
  - クラスタリングとフェイルオーバーのために、
    PacemakerとCorosyncを使用
- 分散ストレージを利用するため、共有ディスクやSAN不要
- 同期レプリケーションによってデータを失うリスクを回避
- オープンソースで実績の多いソリューション

※ホワイトペーパー：DRBD - Configuration and Deployment Guide
http://www.mysql.com/why-mysql/white-papers/mysql_wp_drbd.php



共有ディスク＋クラスタウェア(Oracle Clusterwareなど)を使った高可用性構成

- メリット
  - フェイルオーバー処理をクラスタウェアで自動制御できるため、運用コストが低い
  - 共有ディスクにデータがあるため、プライマリ/スタンバイでデータの不整合が起きない
- デメリット
  - 比較的、コスト(H/Wコスト、ソフトウェアコスト)がかかる
    - 共有ディスクが必要
    - クラスタウェアが必要
      ⇒Oracle Clusterwareを使用することで、ソフトウェアコストを削減可能

High Availability with Oracle Linux

Oracle Clusterware ＋ MySQL Enterprise Edition

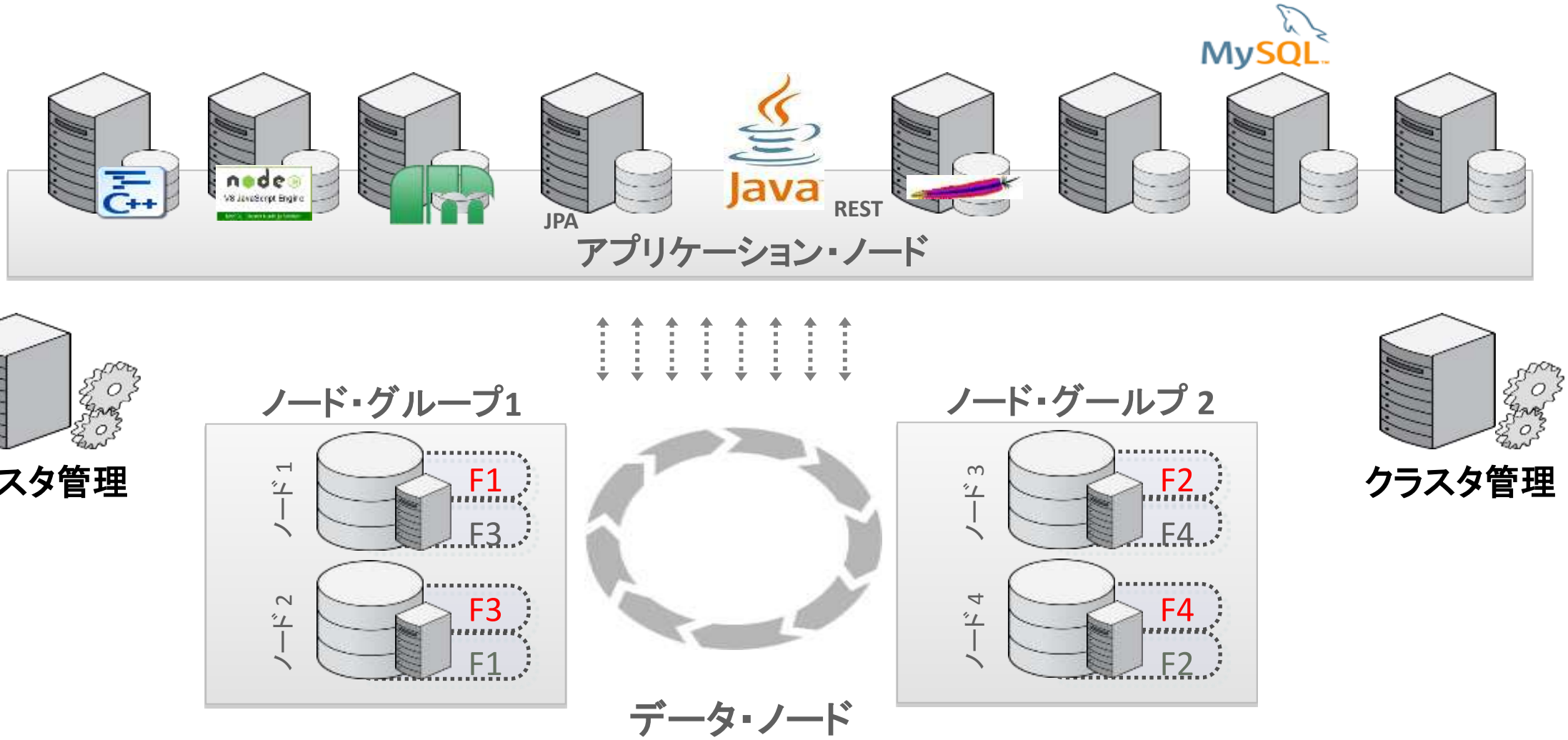
- Oracle Clusterware 12cに、MySQL対応のエージェントが追加
  - MySQL対応エージェントを使用するためには、MySQL EEが必要
  - Oracle Linuxのサポート契約があれば、Oracle Clusterwareについてもサポートを受けることが可能
    ⇒Oracle Clusterwareを使った高可用性構成が、安価に構築可能

http://www.oracle.com/technetwork/database/database-technologies/clusterware/overview/index.html
http://www.oracle.com/technetwork/database/database-technologies/clusterware/downloads/ogiba-2189738.pdf

MySQL Clusterによる高可用性構成

- メリット
  - 単一障害点が無く、フェイルオーバー時間も極めて短いため、可用性が非常に高い
  - 参照処理だけでなく、更新処理についても負荷分散できる
  - SQLだけでなく、豊富なNoSQLインターフェースも持っている(両方の利点を活かせる)
    ※MySQL Clusterは、元々はNoSQLのデータストアでした。
- デメリット
  - 通常のMySQLサーバーとはアーキテクチャが異なる(データの持ち方が異なる)ため、アプリケーションの処理内容によって、向き/不向きがある
    - 主キーやユニークインデックスベースの処理が得意
    - スキャン系の処理は、主キー/ユニークインデックスベースの処理に比べるとオーバーヘッドが大きい
  - 運用方法が、通常のMySQLサーバーと異なる

MySQL Clusterのアーキテクチャ概要



1,000億ドル以上の取引を守るMySQL Cluster

PayPal™

アプリケーション

世界最大級のオンライン決済サービス。Paypalの口座間やクレジットカードでの送金や入金が可能。アクティブアカウント1億以上、20以上の通貨に対応し、203の国と地域で利用可能。年率30%の成長。

MySQL導入の効果

MySQL ClusterをAWSの5拠点に導入し、全世界で1/3秒未満のレイテンシを実現。リアルタイムでの不正検知が可能に。

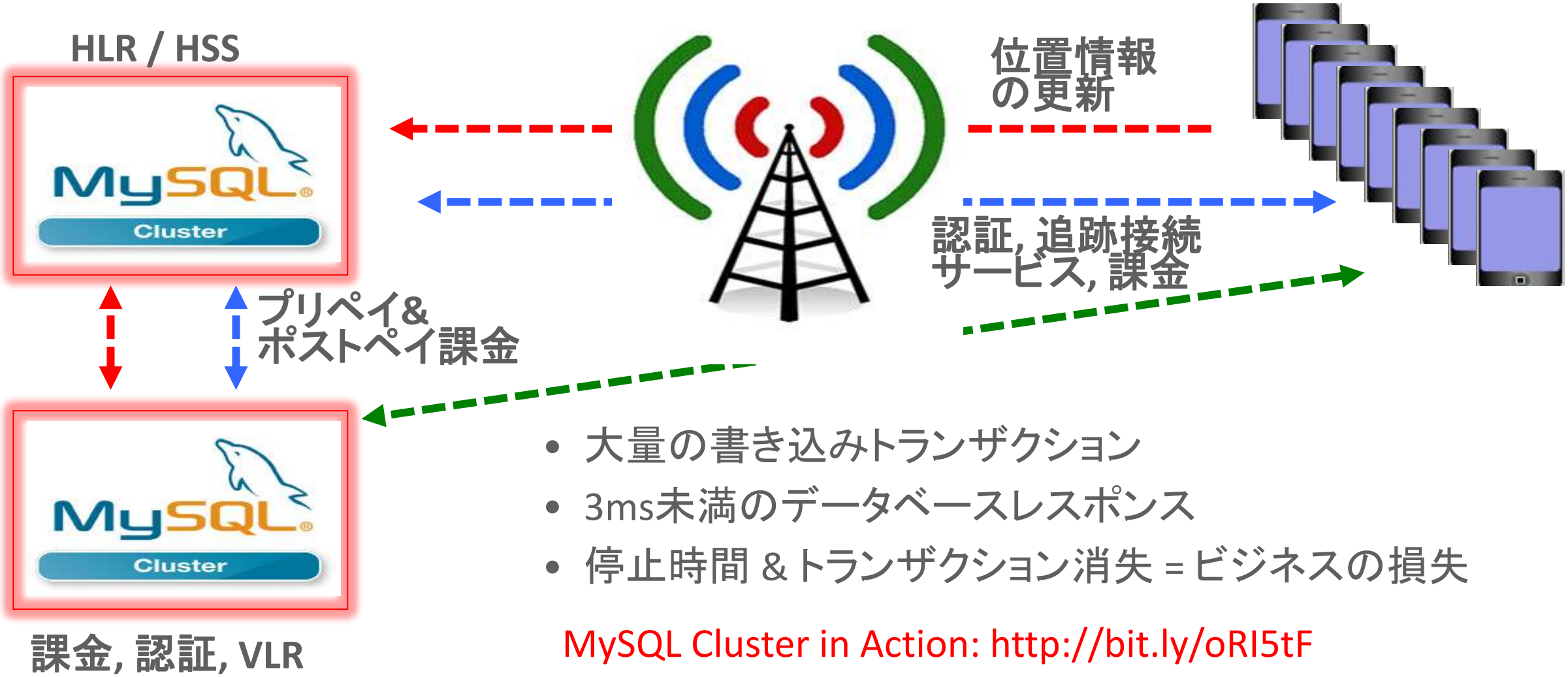
MySQL導入の理由

“NoSQLの特徴である迅速な開発とSQLモデルの信頼性の両方のメリットを実装してるため”
Daniel Austin, Chief Architect, PayPal

ORACLE

Who's Using MySQL Cluster?

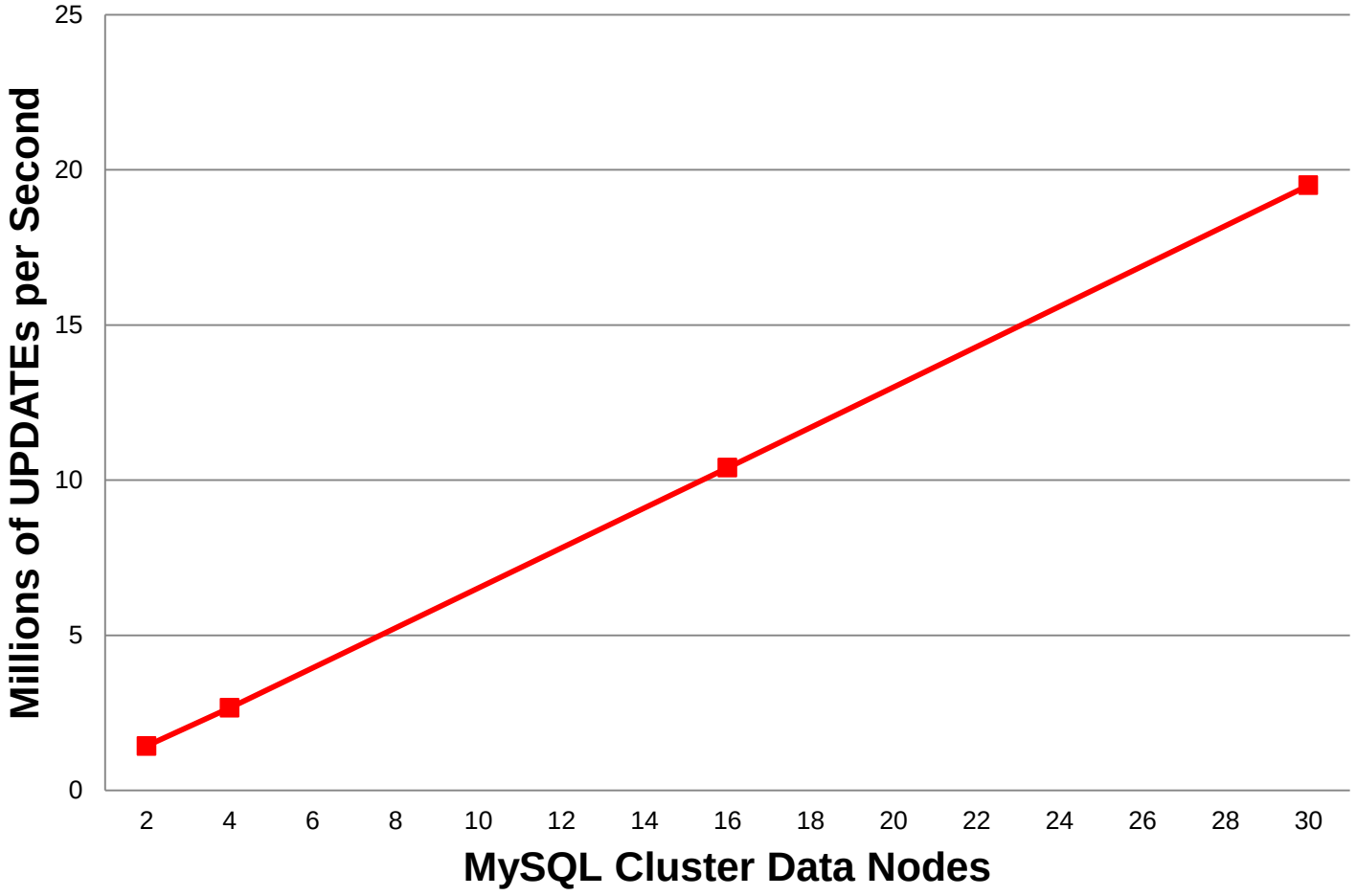
導入事例：携帯電話ネットワーク



- 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

1分間に12億回の書込み(UPDATE)



- NoSQL C++ API, flexaSynch benchmark
- 30 x Intel E5-2600 Intel Servers, 2 socket, 64GB
- ACID Transactions, with Synchronous Replication

MySQL Cluster 7.4　GA(製品版)リリース間近！！

- 2015年1月22日にRC(リリース候補版) リリース済み
- MySQL Cluster に特化したセミナーを3/25(水)に開催予定
  - 詳細は、日本オラクルのイベントページにて近日公開予定

MySQL HA & Scaling Solutions

	MySQL Replication	MySQL Fabric	Oracle VM Template	Oracle Clusterware	Solaris Cluster	Windows Cluster	DRBD	MySQL Cluster
App Auto-Failover	✖	✔	✔	✔	✔	✔	✔	✔
Data Layer Auto-Failover	✖	✔	✔	✔	✔	✔	✔	✔
Zero Data Loss	MySQL 5.7	MySQL 5.7	✔	✔	✔	✔	✔	✔
Platform Support	All	All	Linux	Linux	Solaris	Windows	Linux	All
Clustering Mode	Master + Slaves	Master + Slaves	Active/Passive	Active/Passive	Active/Passive	Active/Passive	Active/Passive	Multi-Master
Failover Time	N/A	Secs	Secs +	Secs +	Secs +	Secs +	Secs +	< 1 Sec
Scale-out	Reads	✔	✖	✖	✖	✖	✖	✔
Cross-shard operations	N/A	✖	N/A	N/A	N/A	N/A	N/A	✔
Transparent routing	✖	For HA	✔	✔	✔	✔	✔	✔
Shared Nothing	✔	✔	✖	✖	✖	✖	✔	✔
Storage Engine	InnoDB+	InnoDB+	InnoDB+	InnoDB+	InnoDB+	InnoDB+	InnoDB+	NDB
Single Vendor Support	✔	✔	✔	✔	✔	✖	✔	✔

Hardware and Software
Engineered to Work Together


ORACLE®