

大規模Webサイトにおける MySQLの運用・管理

松信 嘉範(MATSUNOBU Yoshinori)

MySQL株式会社

シニアコンサルタント

ymatsunobu@mysql.com

昨今のWebアプリの特徴(1)

- ダウンタイム短縮化への要求がそれなりにある
 - 秒単位レベルではない。数分程度でも良い
 - 数時間とかダウンしていると困る
- 同時接続数が多い
 - 1つのMySQLサーバあたり、1000を超えることも珍しくない
 - スレッドベースのMySQLの利点
- データ量が多い
 - 全部合わせるとテラバイト級になることも珍しくない

昨今のWebアプリの特徴(2)

- 複数のMySQLサーバ
 - 1台では処理しきれないため
- Readが多く、Writeが少ない (90:10とか)
 - MySQLレプリケーションとの相性が良い
 - Write負荷が高いタイプに向くのはMySQL Cluster
- PHPとPerlが主流。Javaが続き、Rubyがその次くらい?

セッション内容

- HA構成
- オンラインバックアップ
- リカバリ
- パーティショニング
- MySQL4.0からのアップグレード
- 稼動監視

「マスター障害時はスレーブをマスターに昇格」の考慮事項

- レプリケーションは非同期なので、バイナリログの転送が追いついていない可能性がある
- 複数台のスレーブがある場合、各スレーブごとにレプリケーション状況がまちまちになっている可能性がある
 - マスターが100、スレーブAが99、スレーブBが97、スレーブCが98、など
- 最も新しいスレーブを選んで昇格させたとしても、残りのスレーブとの状態のずれを解消する必要がある(場合がある)
- 新しいマスターは接続パラメータを変えて起動する必要がある
- 残りのスレーブは接続先のマスターを変える操作が必要
- 全スレーブのスペックが悪い場合、後でマスターを良いスペックのものに置き換える必要が出てくるが、どうやるか

- 考えなければならないことは多く、自動化は難しい
 - ダウンタイムが長くなれば、High Availability構成とは言えない
- まずマスターが壊れないようにベストを尽くす。その先の課題
- 「障害時に一部のデータを失っても良い」という場合には向いている

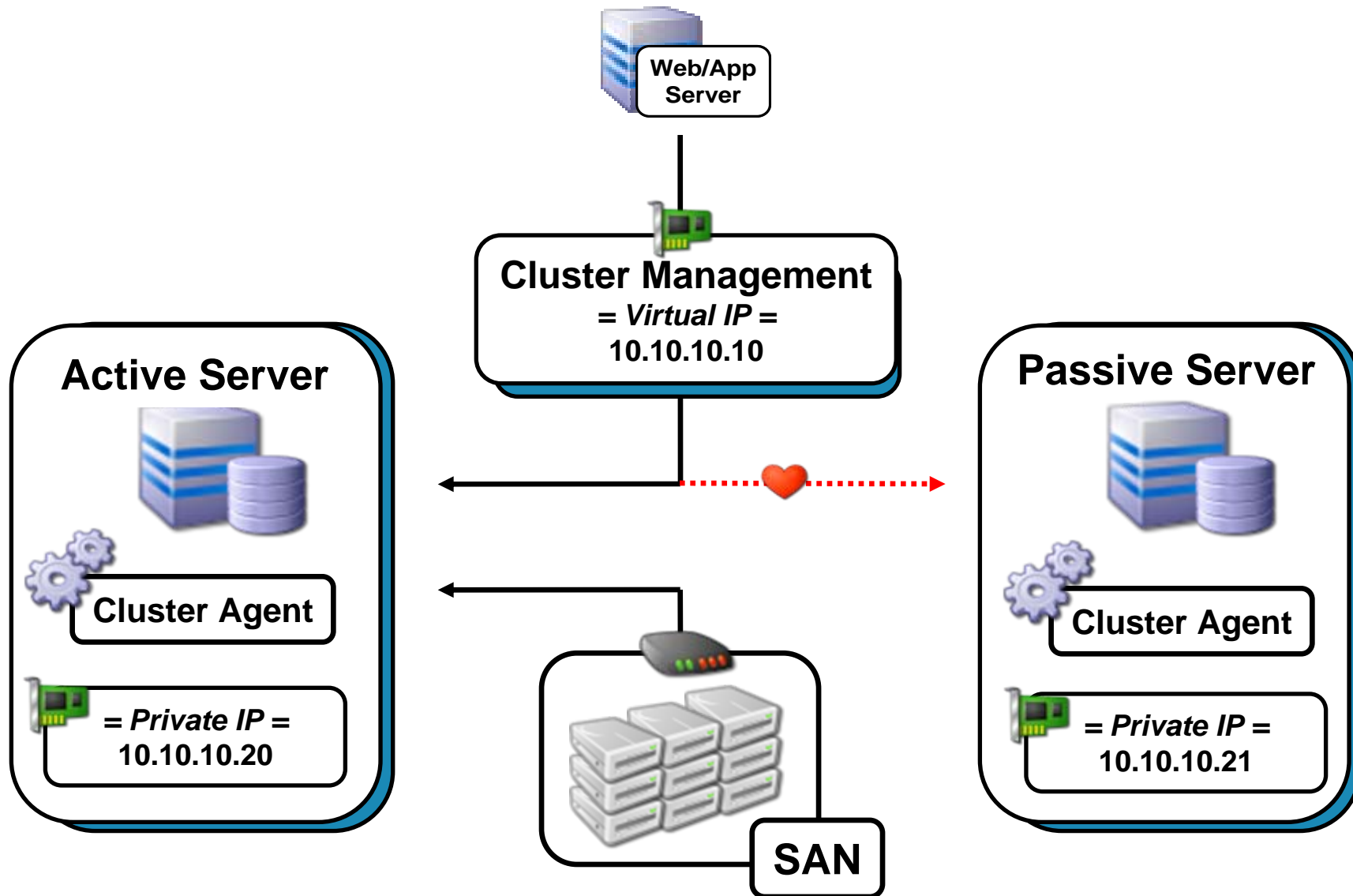
マスターのHigh Availability構成 (1)

- どのようなものか
 - 冗長構成を組む
 - 片方がクラッシュしたら、それを検知してもう片方が引き継ぐ処理を「自動で行う」
 - Active/Passive型とActive/Active型がある
 - MySQLでのActive/Active型のHAソリューションが「MySQL Cluster」
 - マルチマスターも一応Active/Activeだが注意点が多い
- 効果
 - ダウンタイムの短縮
 - オペレーションの単純化 (運用費の削減につながる)

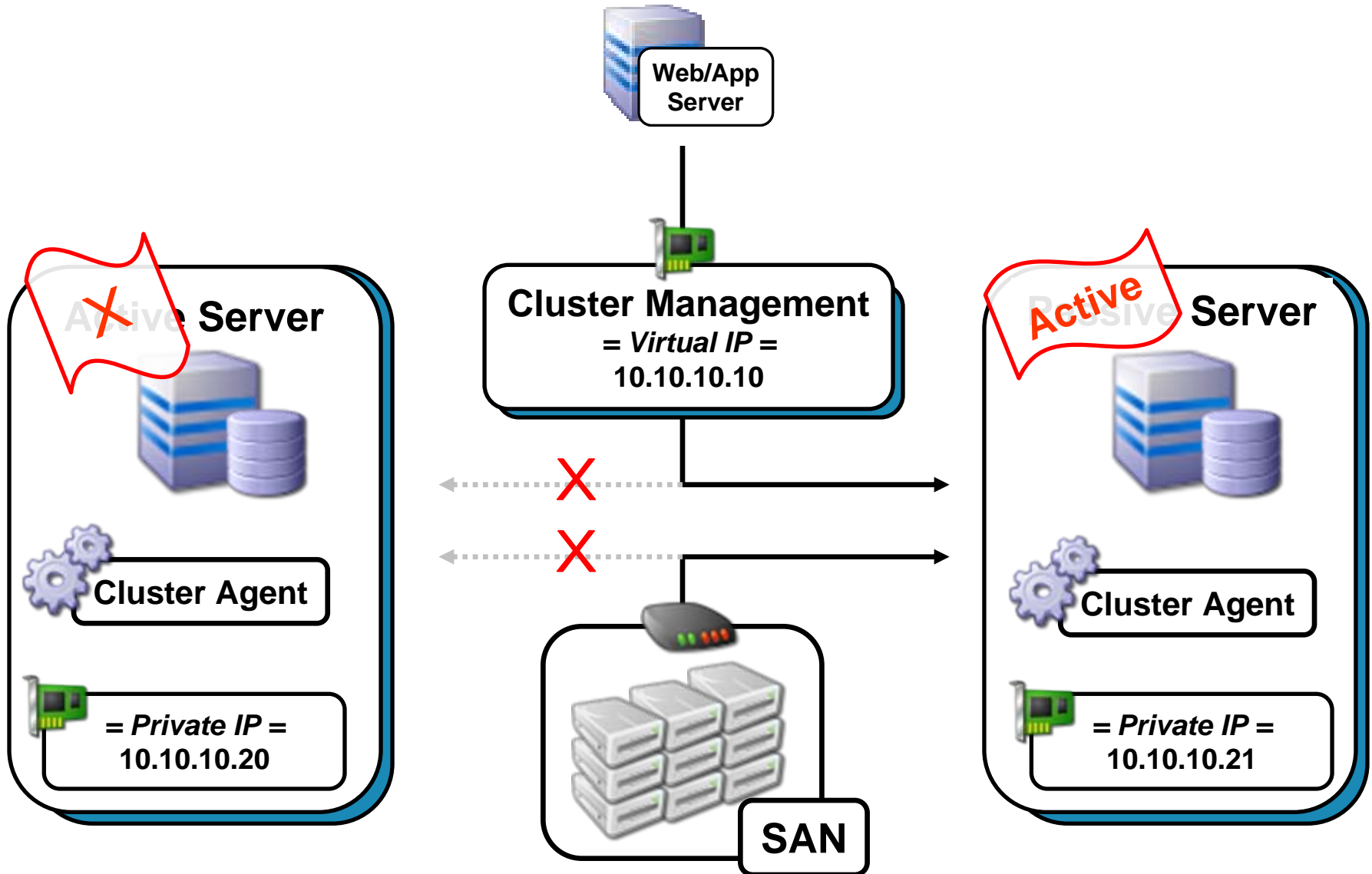
マスターのHigh Availability構成 (2)

- 定番はActive/Passive構成
- 構成
 - クラスタソフト+共有ディスク型
 - (例:Heartbeat + SAN + MySQL)
 - クラスタソフト+ディスクミラー型
 - (例:Heartbeat + DRBD + MySQL)
- 考慮事項
 - 障害の検知方法
 - フェイルオーバー時間
 - Split Brainの回避方法
 - 性能の上げ方 (Active/Passiveは負荷分散にはならない)
 - Passiveサーバの使い道

HA構成の種類 – Active/Passive型



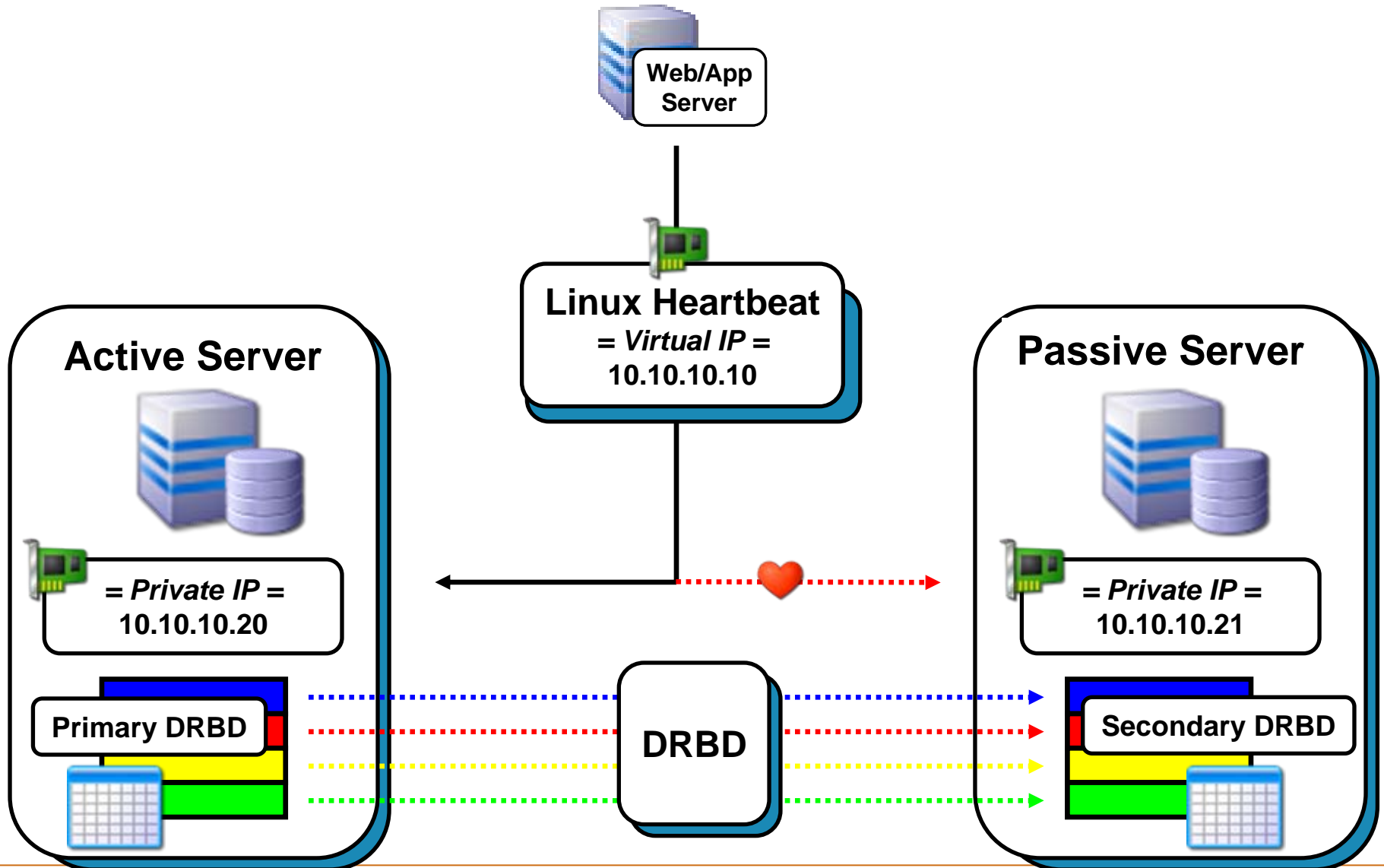
フェイルオーバー時



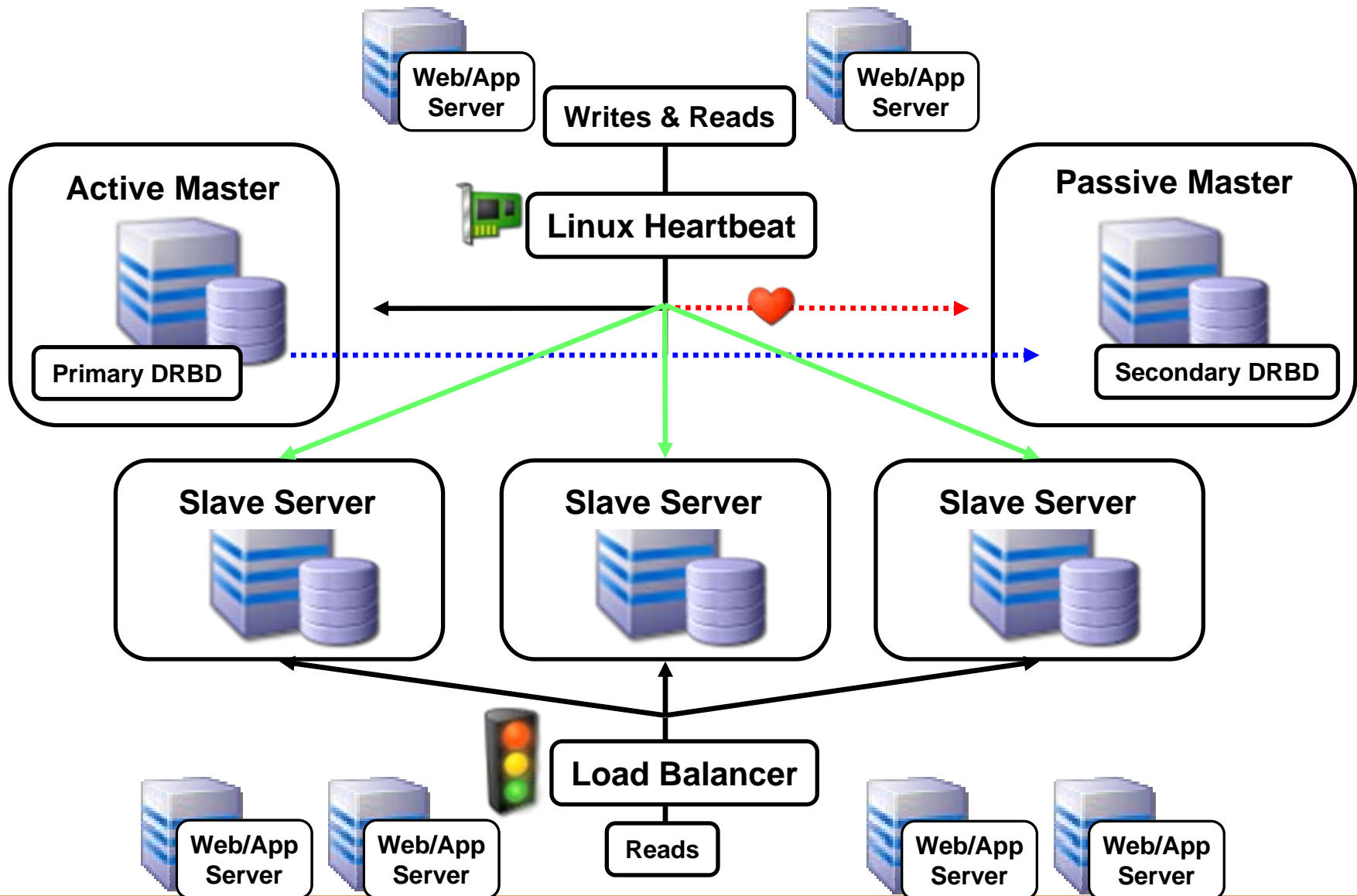
Heartbeat + DRBD + MySQL

- ネットワークディスクミラー(DRBD)を用いたActive/Standby構成
 - <http://www.drbd.org>
 - ディスクをマシンごとに分散配置し、片方向物理レプリケーション (ネットワーク RAID1)
 - 特別なハードウェアが不要(通常のIPネットワーク上で動作)
 - Linux上で動作
 - 同期レプリケーション (性能は良い(SQL文のレイヤーを回避))
 - オープンソース、MySQL Enterpriseではオプションでサポート
- Active/Passive型クラスタ構成と同等で、負荷分散にはならない
 - ミラーされている側ではファイルシステムのマウントすらできない
- 自動フェイルオーバーが可能
- 同期レプリケーションなので不整合の問題が発生しない
 - フェイルバックもより簡単になる

Heartbeat + DRBD + MySQL

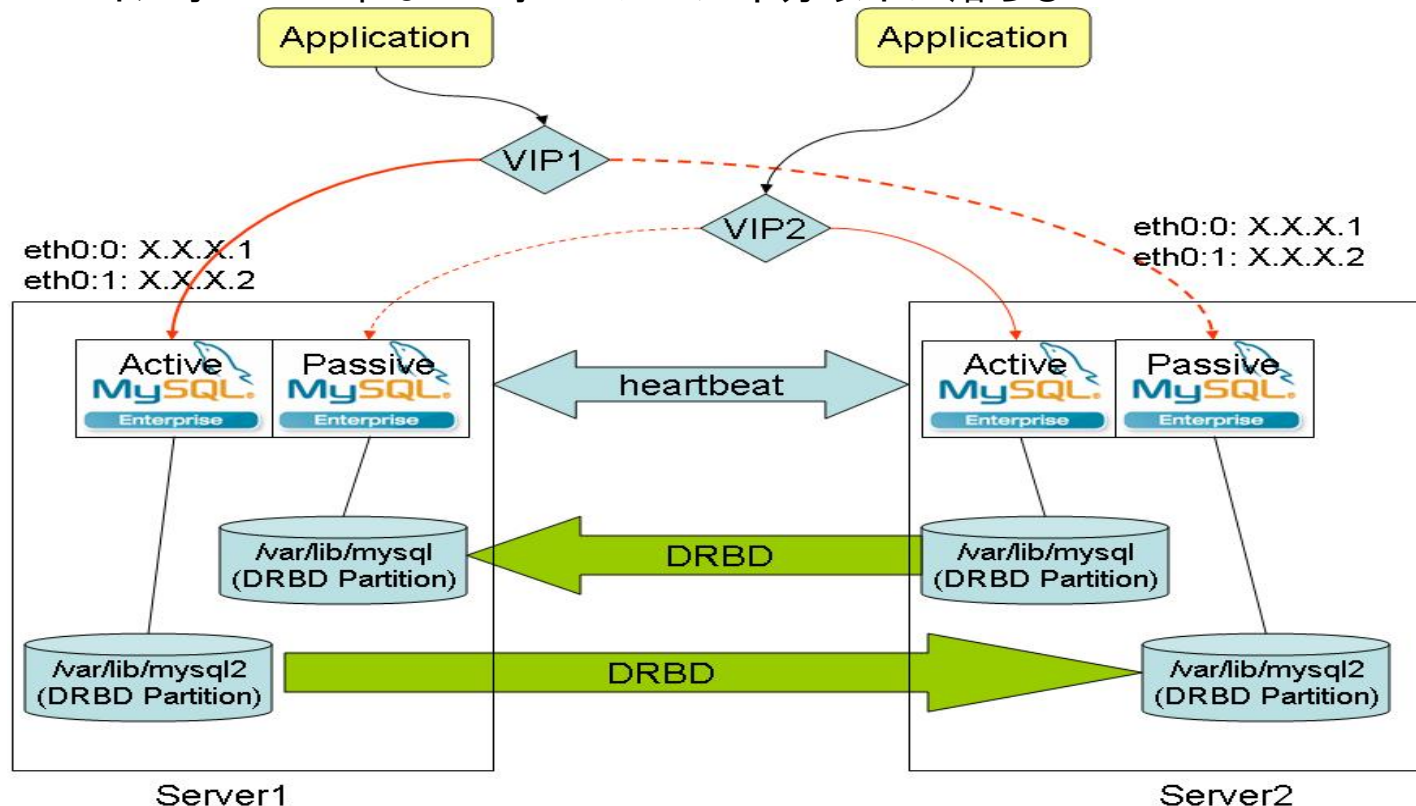


Heartbeat + DRBD + MySQLレプリケーション + L/B



HA構成とハードウェア選定

- ディスクミラー型の場合、待機系も同等のスペックを用意
 - 同期ミラーの速度が、アプリケーションの更新速度に追いつかない危険があるため
- 双方向HA
 - ハードウェアを有効活用できる (DRBDならNICは3枚以上必要)
 - フェイルオーバー中はパフォーマンスが半分以下に落ちる



バックアップ・リカバリ

バックアップの種類 (1)

- 稼動/停止の観点
 - コールドバックアップ
 - オンラインバックアップ
 - 共有ロックをかける (参照はでき、更新はできない)
 - ロックを一切かけない (参照も更新もできる)
- バックアップ時間の観点
 - フルコピー
 - バックアップ時間はデータ量に比例
 - スナップショット
 - バックアップ時間はデータ量と関係なく一瞬
- 取得場所の観点
 - マスター上 (1台構成ならほかに選択の余地がない)
 - スレーブ上

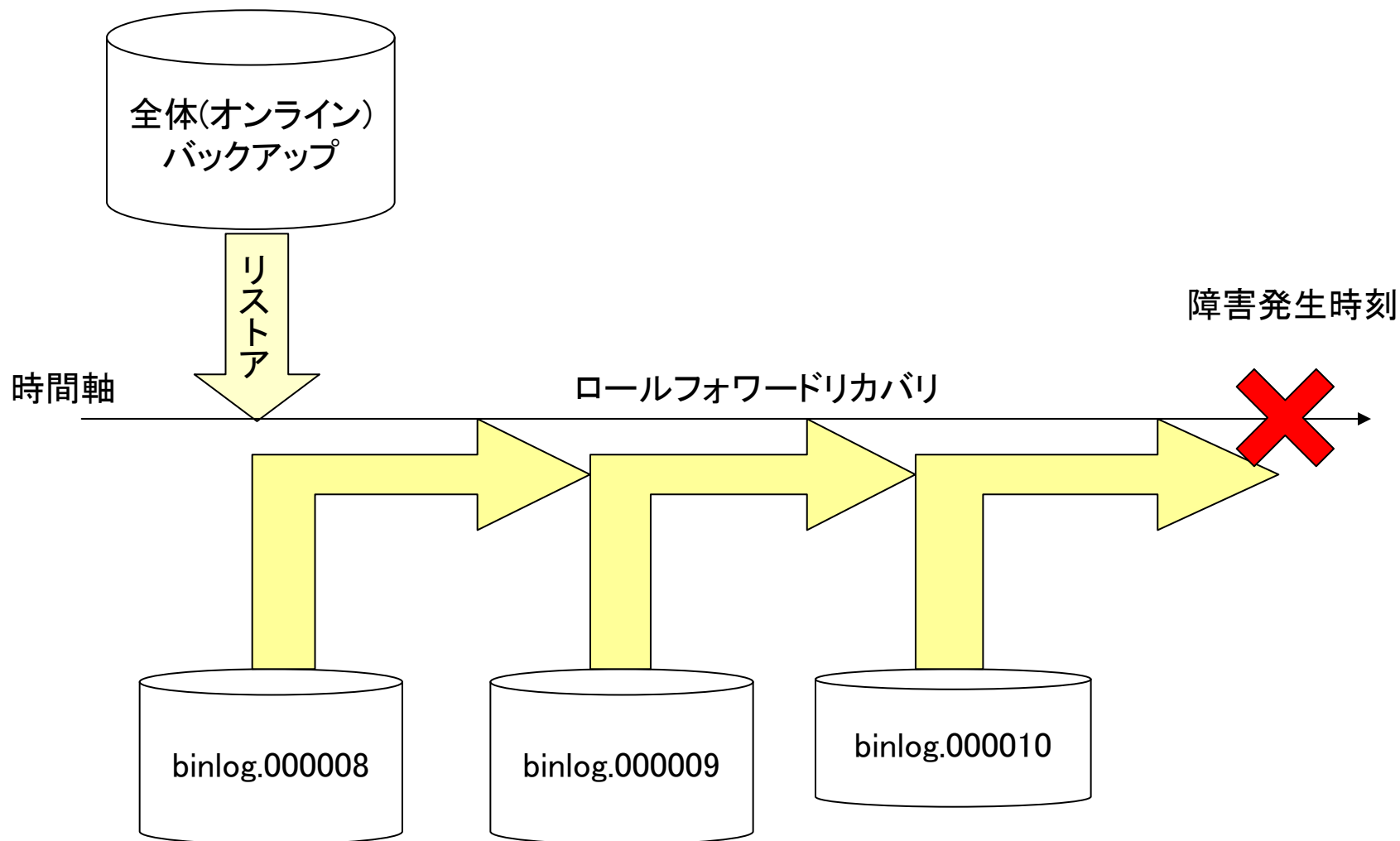
バックアップの種類 (2)

- フォーマットの観点
 - 物理バックアップ
 - バックアップサイズはデータファイルと基本的に同じ
 - 論理バックアップ
 - データサイズが(相対的に)大きい = 書き込みがパフォーマンス上ボトルネックになりやすい
 - 中身を書き換えることもできる
- 範囲の観点
 - 全体バックアップ
 - 差分/増分バックアップ

MySQLのバックアップ

- B2C系のWebアプリでは、ほとんどの場合サービスを止めずにバックアップをする要求がある
- オンラインバックアップ時に参照/更新をブロックするかどうかはストレージエンジンに依存する
 - トランザクション非対応の場合、更新は必ずブロックされる
 - トランザクション対応であれば大丈夫 (InnoDB, NDB, Falcon)
- バイナリログの位置を記録する必要がある
 - ロールフォワードリカバリに必要
- バックアップの主な目的
 - 障害発生時からの復旧
 - スレーブ(場合によってはマスター)の新規セットアップ

ロールフォワードリカバリ



物理バックアップ

- スレーブの1台をバックアップとして使う
 - マスターを一切ロックせずにバックアップできる
- LVMスナップショット
 - FLUSH TABLES WITH READ LOCK; によるDB全体の共有ロック
 - SHOW MASTER STATUS; によるバイナリログの位置記録
 - LVMスナップショットによるバックアップ取得
 - UNLOCK TABLES;
 - InnoDBはクラッシュリカバリによって復旧できる
 - innodb_flush_log_at_trx_commit=1が必須
 - スナップショットはOSのファイルキャッシュを迂回
 - MyISAMでは一貫性のあるバックアップが取れる保証がない (sync必須)
- InnoDB Hot Backup
- MySQL6.0からはオンラインバックアップAPIが登場
 - ストレージエンジンごとに最適化された方法で物理バックアップをオンラインで取れる

論理バックアップ

- `mysqldump --single-transaction --master-data=2`
 - InnoDBでのみ有効
 - ロックを(ほとんど)かけずに一貫性のあるバックアップが取れる
 - 内部的にはFLUSH TABLES WITH READ LOCKをしている
 - 先頭にバイナリログの位置がコメントとして記録される
 - リカバリするときは、どの位置からバイナリログをあてれば良いかが分かる

```
-- CHANGE MASTER TO MASTER_LOG_FILE='binlog.000001', MASTER_LOG_POS=1285;
```

増分バックアップ

- 全体バックアップは重たい処理なので、毎日実行したくない
→増分バックアップと併用する
- バイナリログの切り替え→バックアップ
 - mysqladmin flush-logs
 - OSコマンドによるコピー
- 全体バックアップの間隔が長い場合、リカバリに時間がかかる
 - 全体バックアップ結果のリストア→バイナリログの適用
 - バイナリログには更新系SQL文がそのまま記録されているのでリカバリの時間効率は良くない
 - バイナリログの量が多ければそれだけ時間がかかる
 - スレーブをバックアップとして使うと、バイナリログの適用時間を極小化できる

リカバリ

- 障害の種類
 - システムダウン
 - 電源断、カーネルパニック
 - mysqldプロセスの異常終了/ハングアップ
 - データ消失
 - ディスク障害
 - オペレーションミス

システムダウンからのリカバリ

- リカバリ方法はストレージエンジンによって違う
 - 単純に再起動
 - InnoDB、NDB
 - リカバリ時間はデータ量に比例しない
 - 実用上必須
 - REPAIR TABLE
 - MyISAM
 - リカバリ時間はデータ量に比例する
 - REPAIR TABLEをせずに捨てる、という考え方もある
(クラッシュしたら捨てるというタイプのテーブルでMyISAMを使う)

データ消失からのリカバリ

- ロールフォワードリカバリ
 - 最終バックアップをリストアし、それから最終バックアップ以降のバイナリログを適用
 - スレーブのうち1台をマスターに昇格し、未適用のバイナリログを適用
- バイナリログはリカバリに必須
 - バイナリログを消失すると、最新の状態へのリカバリはできない
 - ハードウェア障害に対しては、RAID1、DRBDなどによる冗長化

リストア/リカバリ時間

- リストア
 - 物理バックアップからのリストアが最も速い
 - mysqldumpはSQL文の羅列だが、バルクINSERTになっているため高速
- ロールフォワードリカバリ
 - バイナリログはSQL文が逐次記録されているため実行効率が悪い
 - 適用量を減らせるよう、全体バックアップの頻度を調整
- 高速化の定石
 - log-bin無効化、innodb_flush_log_at_trx_commit=0など
 - 失敗すればやり直せば良い場合は耐障害性は意識しなくて良い

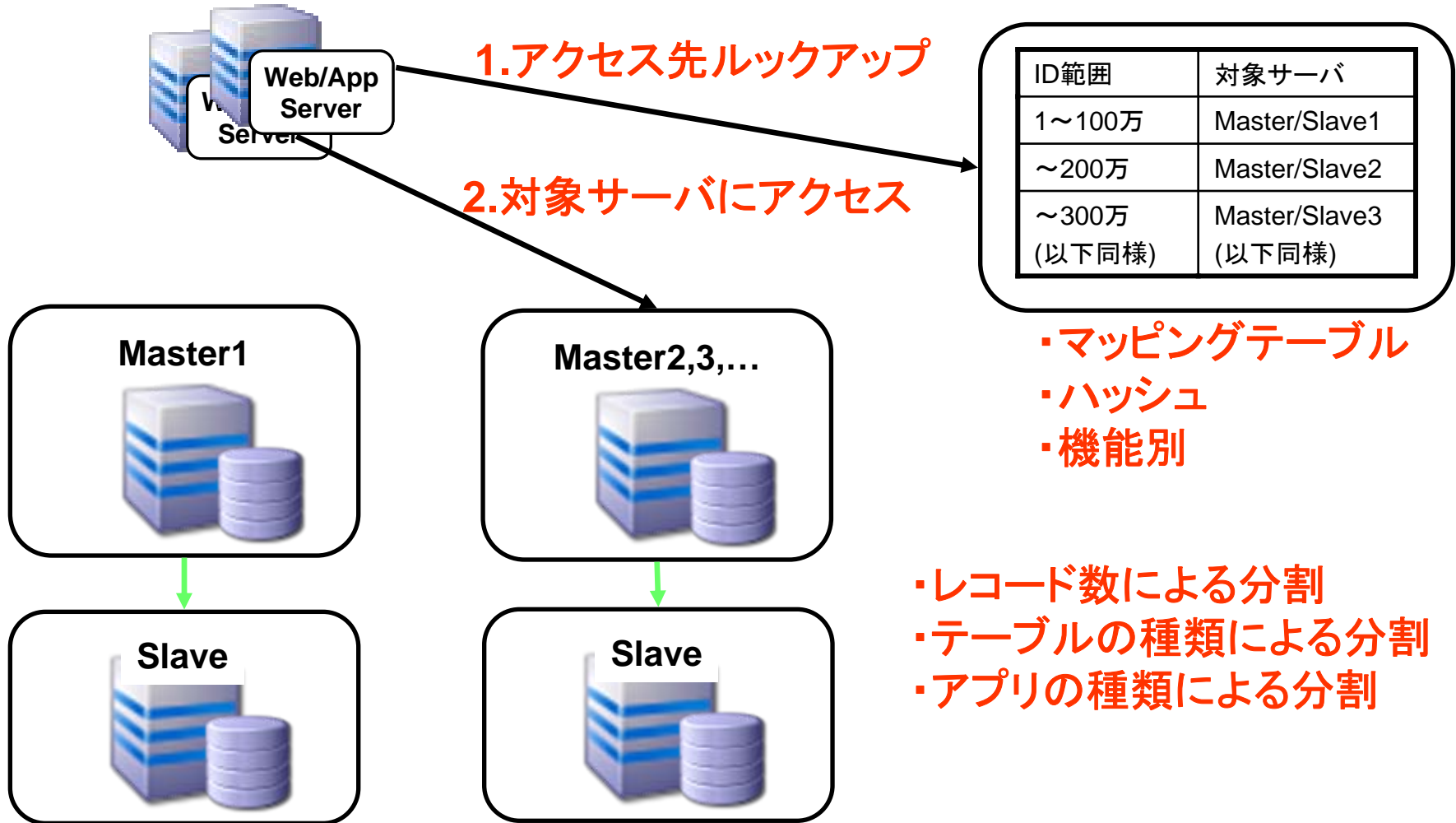
レプリケーションとテンポラリテーブル

1. マスターでテンポラリテーブルを作成
 - バイナリログに記録、レプリケーションされる
2. スレーブをシャットダウンする
 - 定期バックアップなどの目的で
3. マスターでテンポラリテーブルを更新
 - バイナリログに記録
4. スレーブを再起動する
 - テンポラリテーブルは無くなっている
 - スレーブでテンポラリテーブルを更新しようにも、その定義が無い
5. スレーブがエラーで止まる
 - スレーブを止めるときは、テンポラリテーブルが無いタイミングをみはかって止める
 - STOP SLAVEは大丈夫
 - `show status like 'slave%'`で、`Slave_open_temp_tables`を見る

オンラインバックアップとテンポラリテーブル

1. テンポラリテーブルを作成
 - バイナリログに記録
2. オンラインバックアップを取得
 - mysqldumpなどのオンラインバックアップ結果には、テンポラリテーブルの作成構文は含まれない
3. テンポラリテーブルを更新
 - バイナリログに記録
4. このオンラインバックアップ結果を用いてスレーブを新規セットアップ
 - テンポラリテーブルの定義は無い
5. レプリケーションを開始
 - スレーブでテンポラリテーブルを更新しようにも、その定義が無い
6. スレーブがエラーで止まる
 - マスターでオンラインバックアップを取るときは、テンポラリテーブルが無いタイミングをみはからって取る

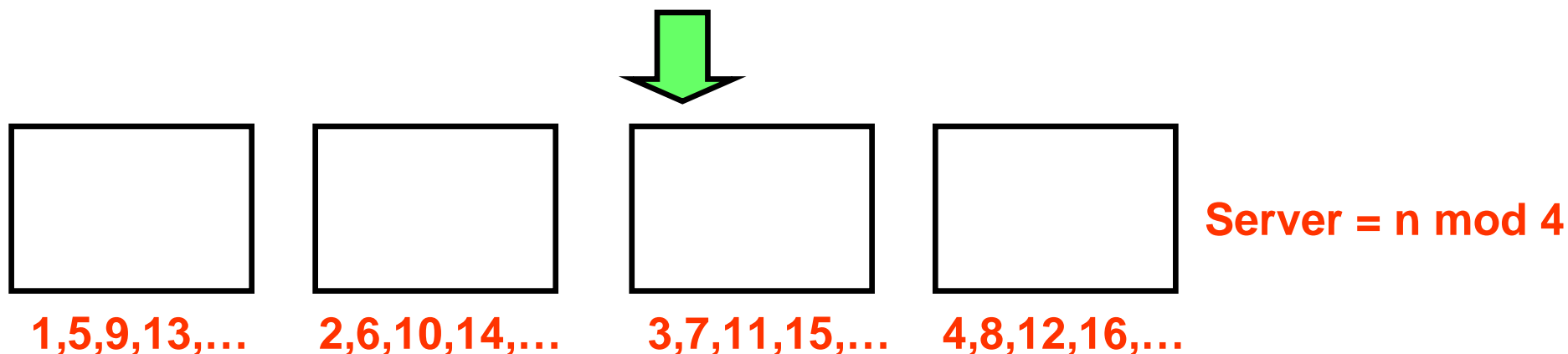
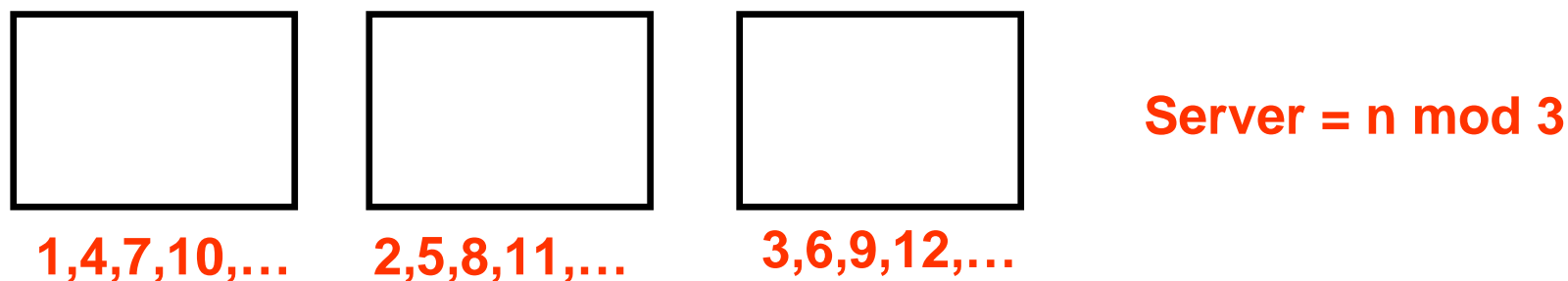
アプリケーションパーティショニング



パーティショニングの種類

- 機能別
 - 日記、ブックマーク、掲示板...
 - アプリケーションとしては元々別物なので影響が少ない
 - 各機能ごとに、やがてパーティショニングが必要になってくる
- グローバルカタログ(共通マスタ:ルックアップ表)によるパーティショニング
 - ルックアップ結果をベースにアクセス先DBを決定
 - MySQLテーブルとしてルックアップ表を持つ
 - アプリケーションから共通に使われるテーブルも置く
 - 再配置が容易
- 計算式によるパーティショニング
 - ハッシュ値をベースにアクセス先DBを決定
 - 処理が単純
 - 再配置が難しい

計算式型と再配置(re-balance)の問題



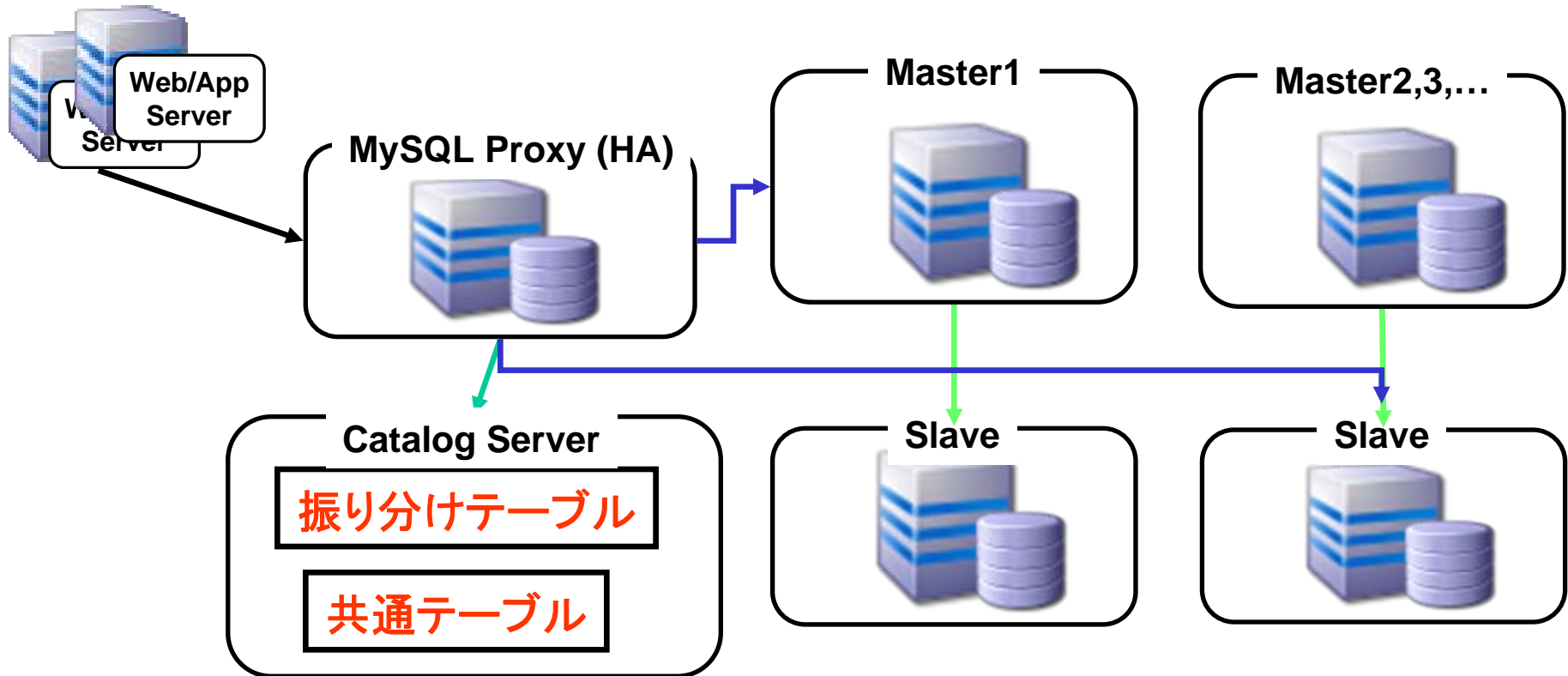
- データを別のノードに移すのは困難
- MemcachedやMySQL Clusterも同様の問題を抱える
- 「Consistent Hashing」によって軽減、
あるいはアプリケーション特有のルールで対処

グローバルカタログ型

ID範囲	対象サーバ
1~100万	Master/Slave1
~200万	Master/Slave2
~300万 (以下同様)	Master/Slave3 (以下同様)

- 柔軟だが、欠点は面倒なこと
 - ルックアップ表もHA構成が必要
 - 専用のテーブル定義、クエリを書かないといけない
 - 専用のデータベース接続が必要
- 出来合いのツールが登場しているのでそれを活用すると良い
 - Hibernate Shards
 - HiveDB
 - MySQL Proxyとの組み合わせ

MySQL Proxy



- 別名はMySQL Load Balancerで、L/Bの代替となるソフトウェア
- 参照/更新の自動振り分け
- 対象パーティションの自動認識
 - アプリケーションから見ると接続先はMySQL Proxyサーバだけになる
- LUA言語によるスクリプトで記述
 - MySQL Enterprise購入ユーザには汎用的に使えるスクリプトを提供/サポート予定

パーティショニングの考慮事項

- いつパーティショニングするか、タイミングを決める
 - レスポンスタイムの平均値
 - ロードアベレージ
- タイミングを遅らせる工夫
 - アプリが複雑になり、障害切り分けも難しくなるので、パーティショニングを避けられるのであれば避ける(遅らせる)
 - ある程度のスケールアップをする (特にメモリ)
 - 32bit機の2GB Memory <<<< 64bit機の16GB Memory
 - RAID 1+0
 - パフォーマンスチューニングをする
 - 50%~100%の性能向上は珍しくない
 - my.cnfのパラメータによる影響は大きい
 - SQL文の処理効率
- 5.1のパーティショニング機能
 - ハッシュ、主キー、レンジ、リスト、複合型
 - ディスクI/Oボトルネックの軽減に効果的

MySQL 4.0 → 5.0/5.1 の主なメリット (InnoDB)

- Row format = compact の導入
 - 20%程度の省スペース化
 - 使用ファイルサイズが小さくなる
 - I/O量が減るため高速化
- マルチコアCPUスケーラビリティの向上
 - 5.0では8コア程度までスケール。実用上問題ない
 - 4.0では改善されることはない
- クエリキャッシュがトランザクション内でも動作する
 - ORM製品の多くはトランザクションを暗黙的に使っているため効果は大きい

MySQL 4.0 → 5.0/5.1 の主なメリット (InnoDB)

- マスターの障害発生時に、スレーブと状態がずれる可能性が消える
 - 4.0は構造上避けられない
 - 5.0は2相コミットの導入により回避
 - 現在ではグループコミットが崩れる。近い将来改善
- mysqldumpでオンラインバックアップが取れる
 - `mysqldump --single-transaction --master-data=2`

MySQL 4.0 → 5.0/5.1 の主なメリット(全般)

- バイナリログのコミット時同期書き込みができるようになった
 - OSのクラッシュ時にも、マスターとスレーブで状態がずれることが無い

- 検索時により多くの索引が使えるようになった
 - index_merge (FROM t1 WHERE c1 = 'abc' OR c2='def'など)

- Unicodeのサポート
 - 3バイト文字まで。4バイト文字は、MySQL5.2からサポート

- 範囲外の文字をエラー扱いはできる
 - INSERT INTO t1 (数値型の列) VALUES('abc');
 - システム変数sql_mode='TRADITIONAL';

MySQL 4.0→5.0/5.1の主なメリット(全般)

- プリペアドステートメント
- クエリキャッシュがプリペアドステートメントを使っても機能する(5.1)
- サーバーサイドカーソル
 - プログラム側で、`SELECT * FROM large_table;`を一度に処理できる
- ストアドプロシージャ、ビュー、トリガー、サブクエリ
- 使えるストレージエンジンの数が増えた
 - Memcached(5.1), Archive, Federated, Blackhole, etc..
 - 5.1からはMySQL本体ソースを変えずにストレージエンジンを入れ替えできる
 - データウェアハウス系(Nitro, Infobright)や全文検索に特化したストレージエンジンなど

4.0と5.0の仕様差異

- 4.1以降では文字コード変換の考慮が必要
 - クライアント側とサーバ側の文字コードを合わせることで、変換は発生しなくなる
 - `character-set-server=cp932/eucjpms/utf8`
 - `skip-client-character-set-handshake`
- CHAR(n), VARCHAR(n), BINARY(n)等の意味が変化
 - 4.0まではバイト数、4.1以降は文字数
 - InnoDBであれば、全部可変長なので消費バイト数は気にしなくて良い
- 文字列型は範囲チェックを行うようになった
 - CHAR/VARCHAR/TEXT型に、0x00などのバイナリ値(文字でない値)は入れられなくなった
 - UTF-8の4バイト文字も入れられない。バイナリ型を使うと良い

アップグレード方法

- 論理バックアップ→リストアによるアップグレード
 - バイナリ互換は事実上無いと考えて良い
 - `mysqldump --all-databases`
 - `mysql --default-character-set=cp932 < /path/to/dump.sql`
- レプリケーション環境の場合
 - 全部一斉に上げるか、スレーブ→マスターの順に上げるかのどちらか
 1. どこかでmysqldumpで全体バックアップを取る。バイナリログの位置も記録
 2. 各スレーブごとにバージョンを5.0.xに上げる
 - まっさらにする
 - 5.0.xで新規セットアップする
 - 全体バックアップをリストアする
 - スレーブを再開
 3. マスターのバージョンを5.0.xに上げる
 - 例: アプリケーションを止め、全スレーブが同期を取れたことを確認した上で、スレーブの1個をマスターに昇格する。現マスターは別の用途に使う
- マスター4.0→スレーブ5.1は(現状では)動作保証外
 - レプリケーションが止まってしまう
 - アプリケーションを止め、ダンプしてリストアする

稼動監視

- 稼動状況を定期的にモニタリングし、条件を満たしたら
 - メールやSNMPなどで通知
 - HA構成の場合はフェイルオーバー
- 条件
 - mysqldプロセスの存在
 - PIDファイル
 - 接続確認
 - mysqlコマンドでのログイン可否
 - クエリの応答速度
 - レプリケーション遅延
 - SHOW SLAVE STATUSのSeconds_behind_master
 - リソースの使用状況
- 考慮事項
 - 「クラッシュ」と「高負荷によるハングアップ状態」で対処を変える
 - 平常時とフェイルオーバー時で条件を変える

MySQL Enterprise Monitor

MySQL Network Enterprise Dashboard | Monitor - Microsoft Internet Explorer

ファイル(E) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)

アドレス(D) http://demo1.mysql.com:10080/index.php/dashboard/?action=index

MySQL Enterprise Dashboard Refresh: Every 1 Minute Help Log Out

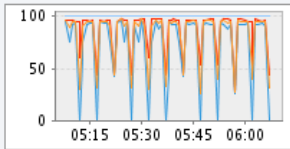
Servers

- All Servers (6)
 - LA_DataCenter:13306
 - Montreal_DataCenter:13306
 - NY_development1:13306
 - NY_development2:13306
 - QA:3306
 - Staging:3306
- Development (2)
- Production (2)
- QA (1)
- robsgroup (2)

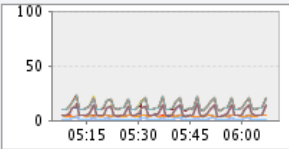
Monitor | Advisors | Events | Graphs | Settings

All Servers Graphs

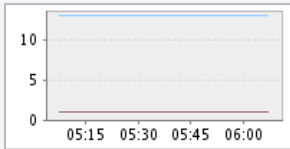
Hit Ratios



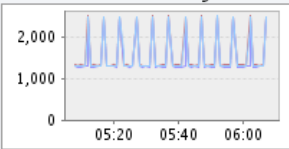
CPU Utilization



Connections



Database Activity



edit favorites configure graphs

All Servers Heat Chart

	Server Status	CPU Usage	IO Usage	RAM usage	Lock Connections	MySQL Cache	Temp Tables to Disk	Table Scans	Critical Alerts	Warnings	Info	
All Servers (6)	●	○	○	○	○	○	○	○	●	42	38	37
Development (2)	●	●	○	○	○	○	○	○	●	12	12	11
Production (2)	●	●	○	○	○	○	○	○	●	12	12	9
QA (1)	●	○	○	○	○	○	○	○	○	9	7	9
robsgroup (2)	●	●	○	○	○	○	○	○	●	12	12	9

Show Legend Standalone Heat Chart

All Servers Critical Events [1 to 10 of 42] Page 1 2 3 4 5 next » last »

Server	Advisor	Rule	Time	
Montreal_DataCenter:13306	Memory Usage	Table Cache Not Optimal	08/04/07 13:06	close
LA_DataCenter:13306	Memory Usage	Table Cache Not Optimal	08/04/07 13:06	close
NY_development2:13306	Memory Usage	Table Cache Not Optimal	08/04/07 13:06	close
NY_development1:13306	Memory Usage	Table Cache Not Optimal	08/04/07 12:56	close

インターネット

その他のツール

- Nagios
- innotop
- mytop
- mtstat
- mysqladmin extended

Innotop

```

ps-vm1.mysql.com - Tera Term VT
ファイル(E) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
InnoDB Buffers (? for help) server1, 02:48:05, InnoDB 11s :-), 2.22k QPS, 53 thd, 6.0
----- Buffer Pool -----
CXN  Size  Free Bufs  Pages  Dirty Pages  Hit Rate  Memory  Add'l Pool
server1 125.00k  120463   7536      6988 1000 / 1000  2.15G   11.25M
----- Page Statistics -----
CXN  Reads  Writes  Created  Reads/Sec  Writes/Sec  Creates/Sec
server1   75    543    7461      0.00      5.82      35.36
----- Insert Buffers -----
CXN  Inserts  Merged Recs  Merges  Size  Free List Len  Seg. Size
server1     0         0         0     1         0         2
----- Adaptive Hash Index -----
CXN  Size  Cells Used  Node Heap Bufs  Hash/Sec  Non-Hash/Sec
server1 4.41M      1         1         0.00    2406.51
  
```

- InnoDBを中心とした統計情報を見れる
- SHOW INNODB STATUSがベースで、見やすくなっている
- <http://sourceforge.net/projects/innotop/>

mtstat

```

ps-vm1.mysql.com - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
$mtstat -Mmysqlinnodb -a
-dsk/total- -net/total- ---system-- ----total-cpu-usage---- ---paging-- -----mysql-----
_read _writ _recv _send _int_ _csw_ usr sys idl wai hiq siq _in_ _out_ _idirty _iflush
 91k 249k 0 0 1043 1355 1 1 98 1 0 0 2023B 2572B 15183 0
 0 0 3666B 3564B 1034 1172 1 1 98 0 0 0 0 0 15183 0
 0 392k 990B 190B 1051 997 1 1 98 0 0 0 0 0 15183 0
 0 2084k 640B 190B 1053 1028 2 1 96 1 0 0 0 0 15119 64
4096B 14M 1664B 190B 1130 1028 2 1 76 21 0 0 0 0 14735 384
 0 19M 896B 190B 1186 1104 3 1 73 23 0 0 0 0 14075 660
 0 17M 1527B 190B 1165 1016 2 1 73 23 0 0 0 0 13563 512
 0 12M 1664B 254B 1229 1063 2 2 73 24 0 0 0 0 13174 389
 0 9248k 1152B 190B 1279 1011 2 1 73 24 0 0 0 0 12918 256
 0 16M 1152B 254B 1164 1092 4 1 72 24 0 0 0 0 12405 513
 0 19M 1370B 190B 1166 1053 3 1 73 23 0 0 0 0 11764 641

```

- vmstatのインターフェースで統計情報を見れる
- <https://launchpad.net/mtstat>

まとめ

- マスターのHA構成はサービスレベルや運用負担の観点で重要
 - Linux+DRBD+Heartbeat+MySQLが主流
- アプリケーションパーティショニングやマスター/スレーブの振り分けは、今後はMySQL Proxyを使うと効果的だろう
- 稼動監視のためのツールは豊富に揃っているので上手く使う
 - MySQL Enterprise Monitor
- MySQL 4.0の開発は終了しており、バグフィックスも行われな
い。5.0または5.1(GA後)を推奨
- MySQL社では、MySQL Enterpriseサブスクリプションでこれらをサポート。またコンサルティングも提供
 - <http://www-jp.mysql.com/consulting/>

MySQLのコンサルティングサービス

- アーキテクチャ設計
- HA環境の構築
 - Heartbeat + DRBD + MySQL
 - 様々な条件化でのフェイルオーバーの動作確認
 - フェイルオーバー時間の短縮化
- パーティショニング
 - キャパシティプランニング、パフォーマンスチューニング
 - MySQL Proxyとの連携
- データベース管理
 - Remote DBA
- ベストプラクティス
- consulting-jp@mysql.com
- <http://www-jp.mysql.com/consulting>