Linux Conference 2004

heartbeat と MySQL を組み合わせたデータベース環境の構築

平成16年6月3日

株式会社NTTデータ
第四公共ビジネスユニット
岩田　浩
<iwatahrs@nttdata.co.jp>

目次

1. 筆者が所属する担当の紹介
2. 紹介事例システムの業務内容
3. 紹介事例のシステム開発の方針
4. オープンソース採用の判断ポイント
5. 実際に使用したプロダクト
6. heartbeat とは
7. heartbeat に不足する機能
8. psmonitord の実装
9. 障害検知時の動作イメージ
10. psmonitord のまとめ
11. 苦労したこと
12. 終わりに

1．筆者が所属する担当の紹介

株式会社NTTデータ 第四公共ビジネスユニット オープン技術統括部

●第四公共ビジネスユニット ≠ 研究開発部門

→ システム構築 (SI) 部門

→ 主に大規模システムを中心に取り組む

●オープン技術統括部

→ オープンソースソフトウェアを用いたSIを積極的に推進

今回は、事例紹介として、Linux, heartbeat, MySQL, Apache を使用した小規模分野におけるデータベースクラスタリング環境の方式面からみた構築の過程と、追加した機能を紹介する。

2．紹介事例システムの業務内容

業務内容

サービス利用状況管理システム

- サービス利用状況を検索し、今回の利用状況を登録
- 登録頻度は 2,000件 ～ 8,000件/日 程度
- 登録者は 40名 ～ 80名
- 将来的に数百万件の利用状況の管理を想定
- 24時間365日、システムを利用

システム条件

- 専属のシステムオペレータは配属せず、自動運転が基本
- インターネットには接続しない、独立したネットワークを構成
- 出来る限り低価格

3．紹介事例のシステム開発方針

紹介事例のシステム開発方針

1. OSSを用いたシステム構築ノウハウの流用と短期開発
2. Webアプリケーション方式で実現
3. サーバ、ネットワークを含めて Single Point Of Failure なし
   - サーバはクラスタリング
   - ネットワークは完全冗長化
4. ハードウェアはIAサーバを使用
5. 不足する機能があっても多少の工夫をすることで対応可能であれば、積極的にオープンソースソフトウェアを採用する

→ 結果的にオープンソースを全面的に採用した

4. オープンソース採用の判断ポイント(1)

1. **ミッションクリティカルなシステムではない**
   - 24時間365日のサービス利用である為、サーバやネットワークの冗長化は必須であるが、障害時のサーバ切換に数分程度の時間は許容されている

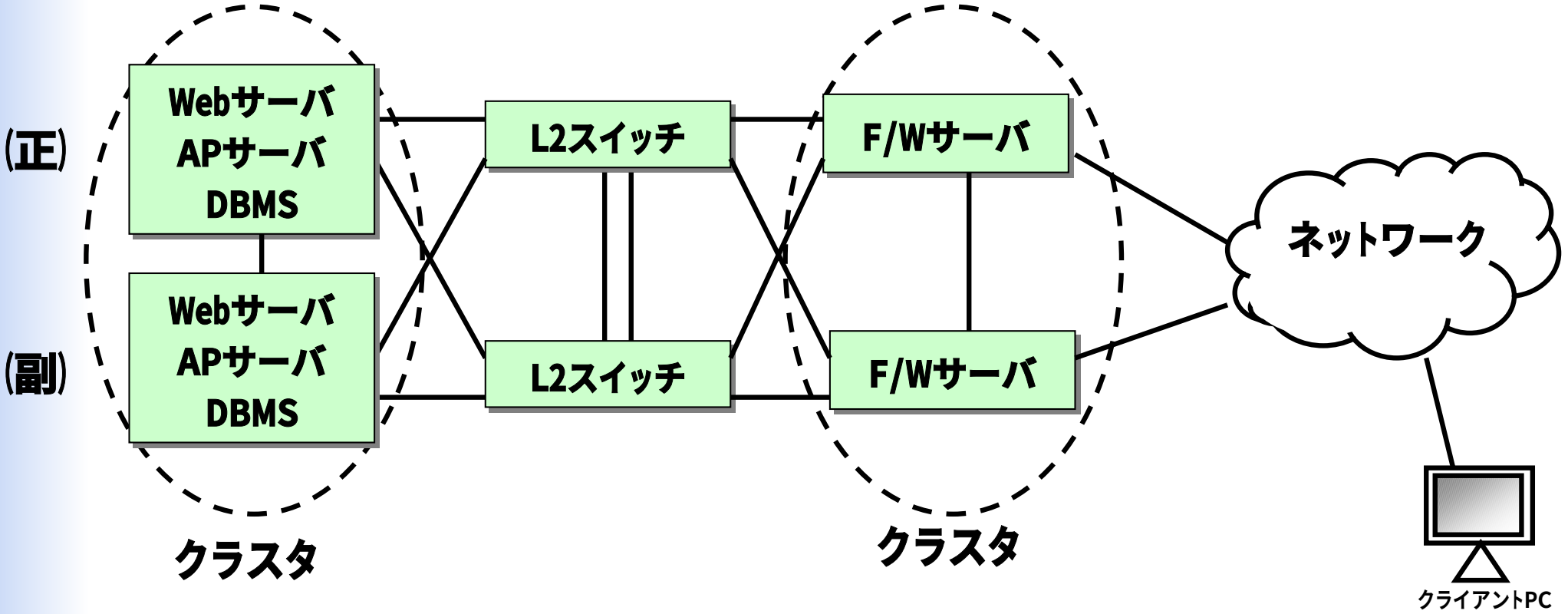
2. **クラスタ対応させるミドルウェアが限定されている**
   - 本事例で用いるもの(Webサーバ、DBサーバ)だけに対応すればいい

3. **レプリケーション型のクラスタ**
   - 業務体系および業務量を考慮すると、共有ディスクを使用しないレプリケーション構成で十分
   - 安価にデバッグ環境が用意できるため、確実に実証することができた

4．オープンソース採用の判断ポイント（2）

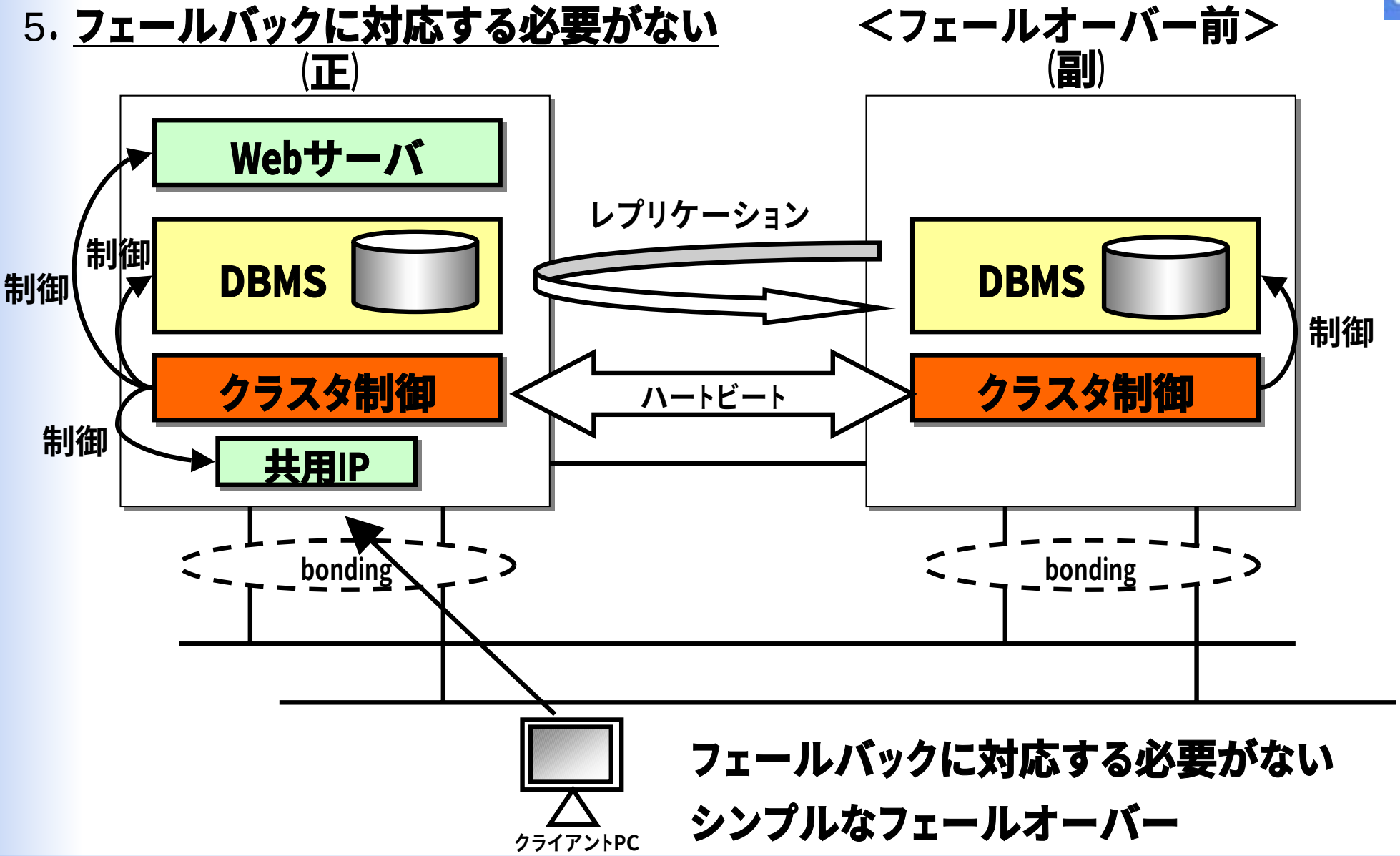
4．シンプルなシステム構成

・サーバ、ネットワークともにSingle Point Of Failure がない構成
・性能的、処理方式的に、Webサーバ、APサーバ、DBMSは同一ハードで問題ない



4．オープンソース採用の判断ポイント（3）

5．フェールバックに対応する必要がない

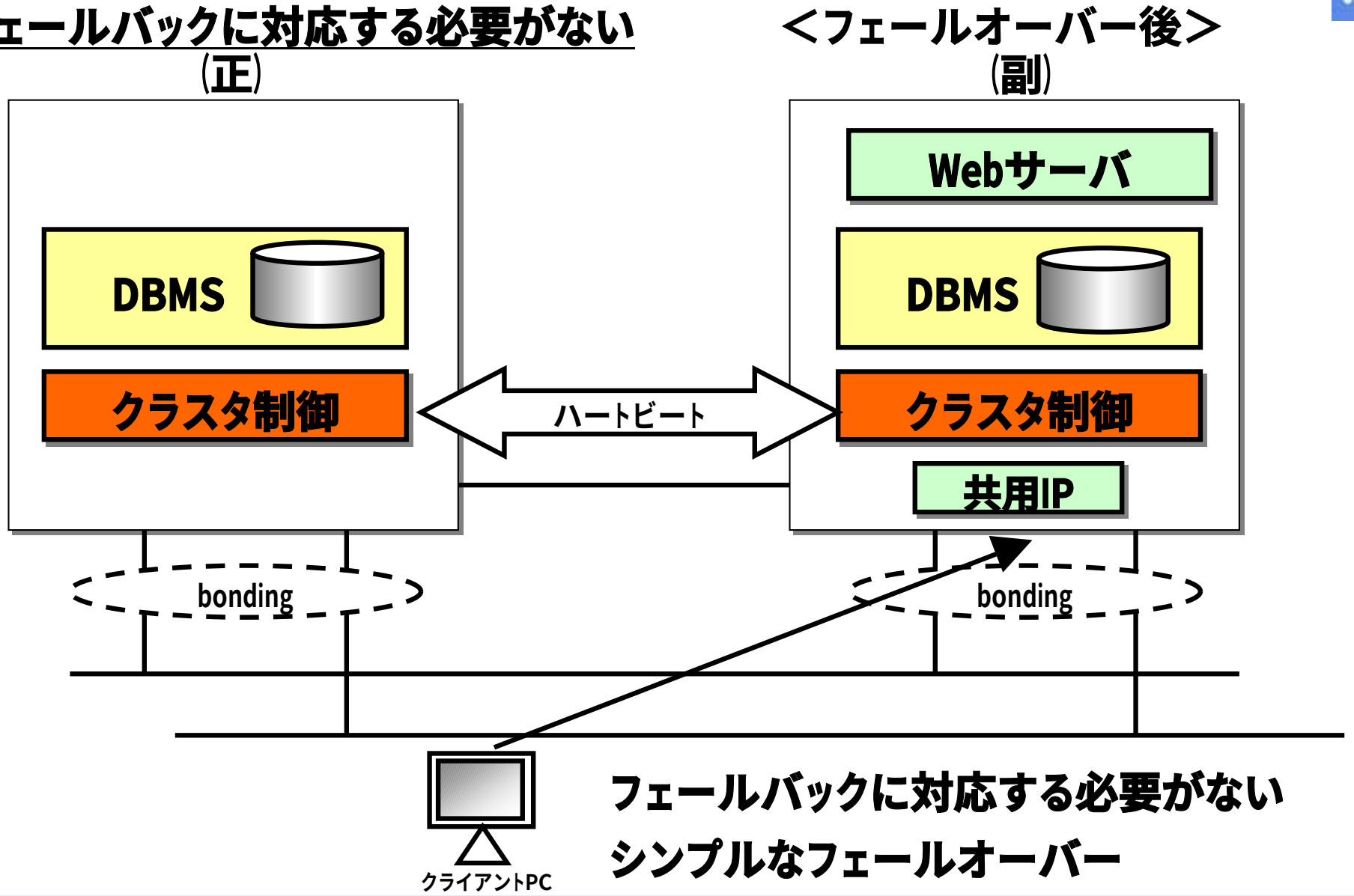


フェールバックに対応する必要がない

シンプルなフェールオーバー

4．オープンソース採用の判断ポイント（3）

5．**フェールバックに対応する必要がない**



5．実際に使用したプロダクト(1/2)

ここまでのポイントを考慮し、全てオープンソースで問題がないと判断し、以下のプロダクトを選択した。

業務AP記述言語

→ ***Perl (CGI)***

・**担当に経験豊富なリーダーが存在**

・**少数精鋭によるアプリケーション開発体制**

Webサーバ

→ ***Apache***

・Perl **で記述された** CGI **の実行環境としては最も一般的**

DBMS

→ ***MySQL***

・**標準でクラスタリング構成(レプリケーション)に対応**

5．実際に使用したプロダクト(2/2)

ここまでのポイントを考慮し、全てオープンソースで問題がないと判断し、以下のプロダクトを選択した。

クラスタリング

→ ***heartbeat***

・**オープンソースなクラスタリングソフトウェア**

NIC**冗長化**

→ ***Linux NIC Bonding***

・Linux**標準のネットワークインターフェイス冗長化機構**

スイッチの冗長化

→ ***機器(NETGEAR)固有のトランク接続機能***

6．heartbeat とは

- **クラスタ構成の根幹部分を構成するソフトウェア**
  ノード間でステータスを確認する通信（*heartbeat*）を行い、ノードの生死確認や共有リソースの制御を行う。

- **オープンソースプロダクト**
  - Linux HA Project **の成果物の一部** <http://www.linux-ha.org/heartbeat/>
    （**最新版は** ver.1.2.2 **、本事例では** ver.1.2.0 **を使用**）
  - UltraMonkey **などでも利用されている**

- **各種テンプレートが用意されている**
  - Apache
  - Idirectord
  - **共有**IP**アドレス**
  - SCSI**デバイス**（ServerRAID）

    など

7. heartbeat に不足する機能

heartbeat だけで全ての要求事項を満たすことは出来ない

- 障害検知の機能が不十分
  - ネットワーク障害の検知 (NIC片系障害)
  - プロセス障害の検知ができない
  - サービス障害の検知ができない
- プロセス障害時の動作が不十分
  - ゴミプロセスが残る場合がある

とはいえ、本事例で不足しているのは上記ぐらい。

監視対象を Apache、MySQL、heartbeat に限定すれば、既存技術の組み合わせと多少の工夫で要求条件をクリアできると判断

→ 不足部分を補うために、psmonitord を開発

8．psmonitord の実装 (1)

psmonitord

→ heartbeat と連動し、機能を補完するスクリプト

- **プロセス監視**
  - heartbeat、Apache、MySQL、ldirectord
  - ※ **監視対象は本事例で使用するものに限定した**
- **サービス監視**
  - Apache、MySQL
  - ※ **プロセス監視だけでなく、サービスも同時に監視する。**
- **障害発生時における、共有リソースのきれいな終了**
  - **プロセス障害時発生時に、ゴミが残らないようにクリーンアップする**
- Active / Standby **構成にのみ対応、自動的な切り戻しには未対応**

※**クラスタ構成の復旧には、手動で** heartbeat **および** psmonitord **の再起動が必要になる**

8. psmonitord の実装 (2)

実装言語:

- **全て** Perl **で記述**

heartbeat **との連動:**

- hb_standby **コマンド**

※ 細かな制御には適さないが、本事例ではそれで十分。何より手軽。

共有リソースの制御:

- /etc/ha.d/resource.d/apache ← Apache**制御ファイル**
  ldirectord ← ldirectord **制御ファイル**
  IPaddr ← **共用**IP**制御ファイル**

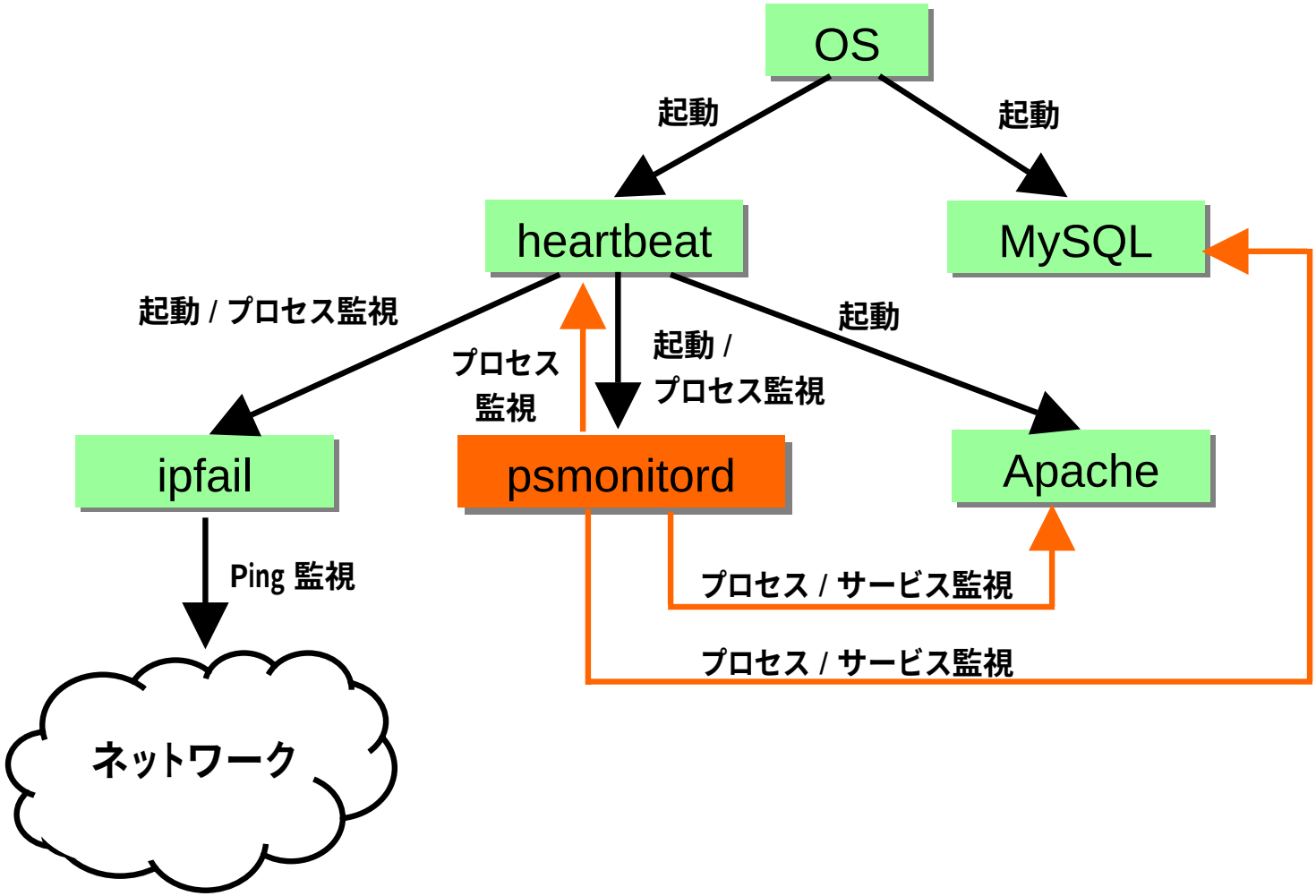
psmonitord **の起動方法:**

- heartbeat **から直接起動する**
- heartbeat **が** psmonitord **自身を監視している**
- heartbeat **の設定ファイル** (/etc/ha.d/ha.cf) **に記述**

```
respawn   root   /usr/lib/heartbeat/psmonitord
```


8. psmonitord の実装 (3)

psmonitord **の構成**:



8. psmonitord の実装 (4)

ファイル構成:

/etc/ha.d/psmonitord.conf	← **設定ファイル**
psmonitord_apache.conf	← **設定ファイル** (Apache**関連**)
psmonitord_mysql.conf	← **設定ファイル** (MySQL**関連**)
/usr/lib/heartbeat/psmonitord	← **実行ファイル**

設定ファイル例: (psmonitord.conf)

```
MCP="heartbeat: heartbeat: master control process"
FIFO="heartbeat: heartbeat: FIFO reader"
INIT_WAIT="10"
TIMEOUT="10"
INTERVAL="5"
RETRY="4"
SYSLOG="local0.info"
HALT_COMMAND=""
```


8．psmonitord の実装 (5)

プロセス監視 (heartbeat)

heartbeat **を構成する各種プロセスを監視する。**

- master control process
- FIFO reader

(1) ps**コマンドでプロセスが発見されなかった場合に障害発生とみなす**

(2) **障害検知すると**heartbeat**関連プロセスを消去**

(3) /etc/ha.d/haresource **の内容を解析し、所有する共有リソースを全て開放**

(4) **結果、フェールオーバー**

8．psmonitord の実装 (6)

プロセス・サービス監視 (Apache)

Apache **を構成する各種プロセス、サービスを監視する。**

- **設定ファイルで指定された文字列を監視**
  /usr/local/apache/bin/http –DSTATUS –f /usr/local/conf/httpd.conf
- TCP**ポート**80**番に対して通信を行い、返答文字列を監視**

(1) **障害を検知すると** Apache**に関連するプロセスを全て消去**

(2) hb_standby**コマンドを実行** (**設定ファイルで任意のコマンドを指定可能**)

(3) **結果、フェールオーバー**

8. psmonitord の実装 (7)

設定ファイル例: (psmonitord_apache.conf)

```
PS="/usr/local/apache/bin/httpd -DSTATUS -f /usr/local/apache/conf/httpd.conf"
CLEANUP_PS="/usr/local/apache/bin/httpd -DSTATUS -f /usr/local/apache/conf/httpd.conf"
HOST="localhost"
PORT="80"
METHOD="HEAD"
URL="/"
MATCH="401 Authorization Required"
INTERVAL="5"
RETRY="4"
FAILOVER="/usr/lib/heartbeat/hb_standby"
FAILOVER_RETURN="Going standby \[all\]"
```


8．psmonitord の実装 (8)

プロセス･サービス監視 (MySQL)

MySQL **を構成する各種プロセス、サービスを監視する。**

- **設定ファイルで指定された文字列を監視**
  /usr/local/mysql/bin/mysqld_safe
- TCP**ポート**3306**番に対して通信を行い、返答文字列を監視**
- **レプリケーション状態を監視**

(1) **障害を検知すると** MySQL **に関連するプロセスを全て消去**

(2) hb_standby**コマンドを実行 (設定ファイルで任意のコマンドを指定可能)**

(3) **結果、フェールオーバー**

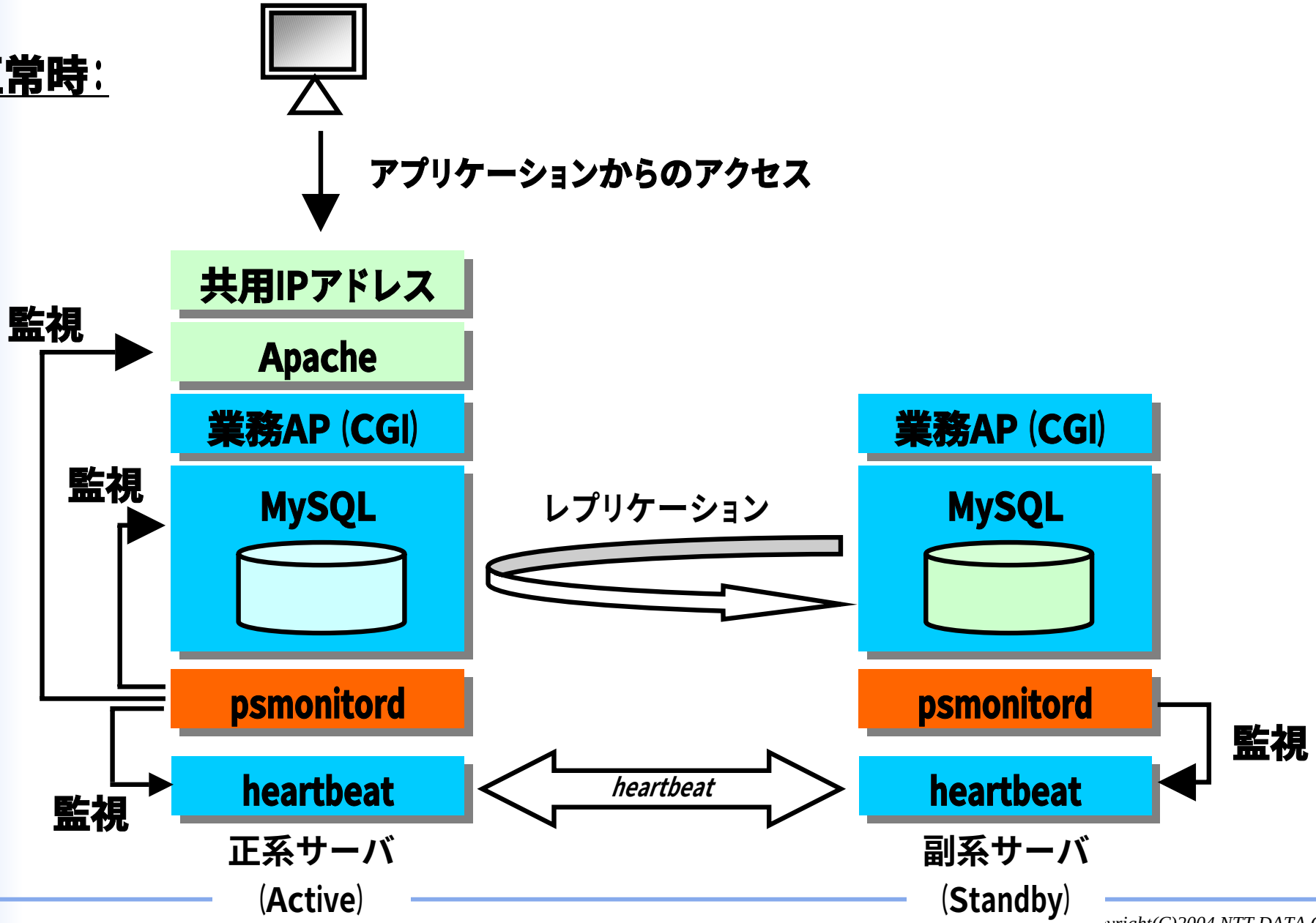
8. psmonitord の実装 (9)

設定ファイル例: (psmonitord_mysql.conf)

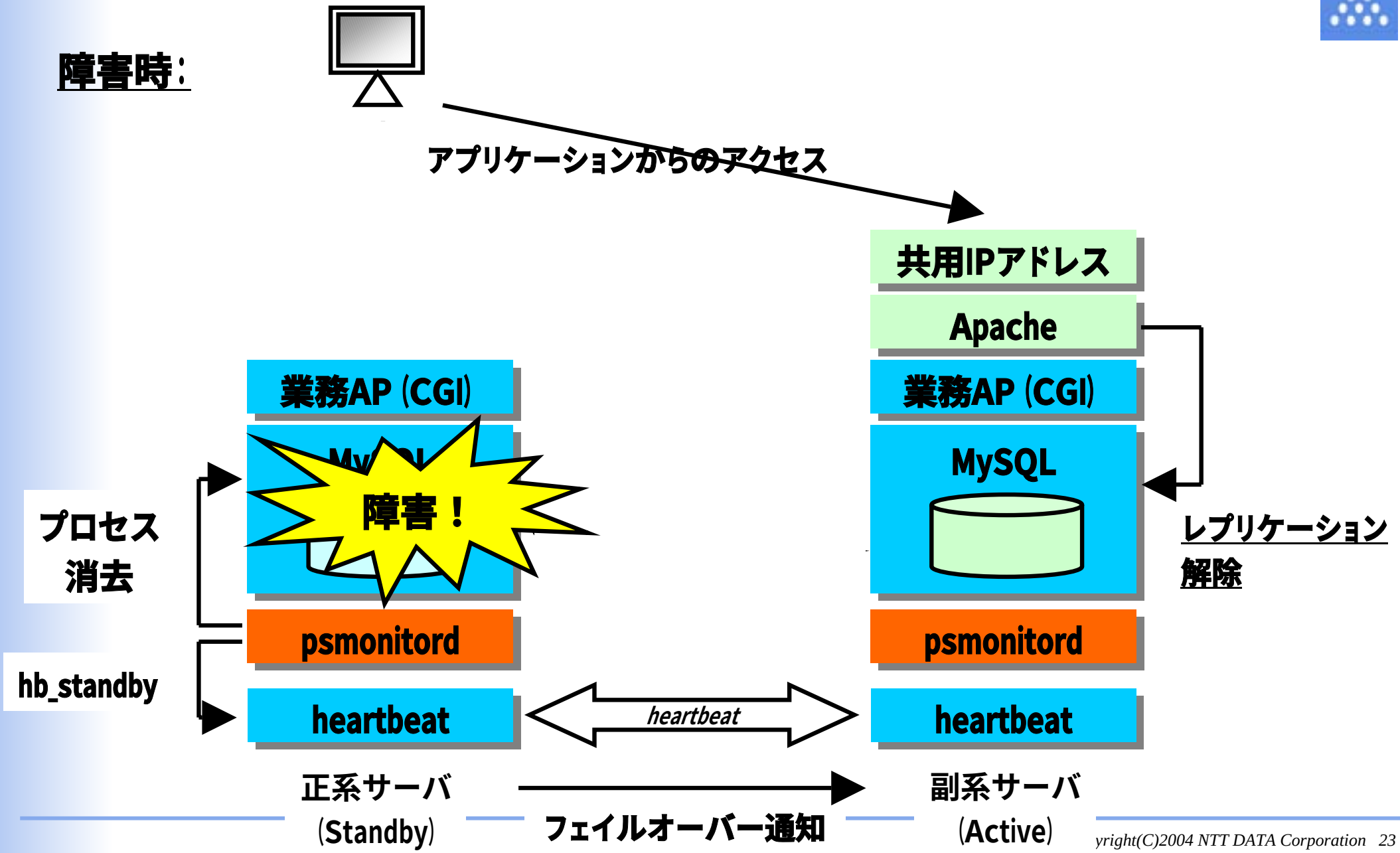
```
PS="/usr/local/mysql/bin/mysqld_safe"
CLEANUP_PS="/usr/local/mysql/libexec/mysqld"
HOST="scwd01"
PORT="3306"
USER="admin"
PASSWD=""
SLAVE_HOST="scwd02"
SLAVE_PORT="3306"
SLAVE_USER="admin"
SLAVE_PASSWD=""
INTERVAL="5"
RETRY="4"
LOG_RETRY="3"
FAILOVER="/usr/lib/heartbeat/hb_standby"
FAILOVER_RETURN="Going standby \[all\]"
```


9．障害検知時の動作イメージ（1/4）

正常時：



9．障害検知時の動作イメージ (2/4)



9．障害検知時の動作イメージ (3/4)

MySQL レプリケーション構成の解除方法

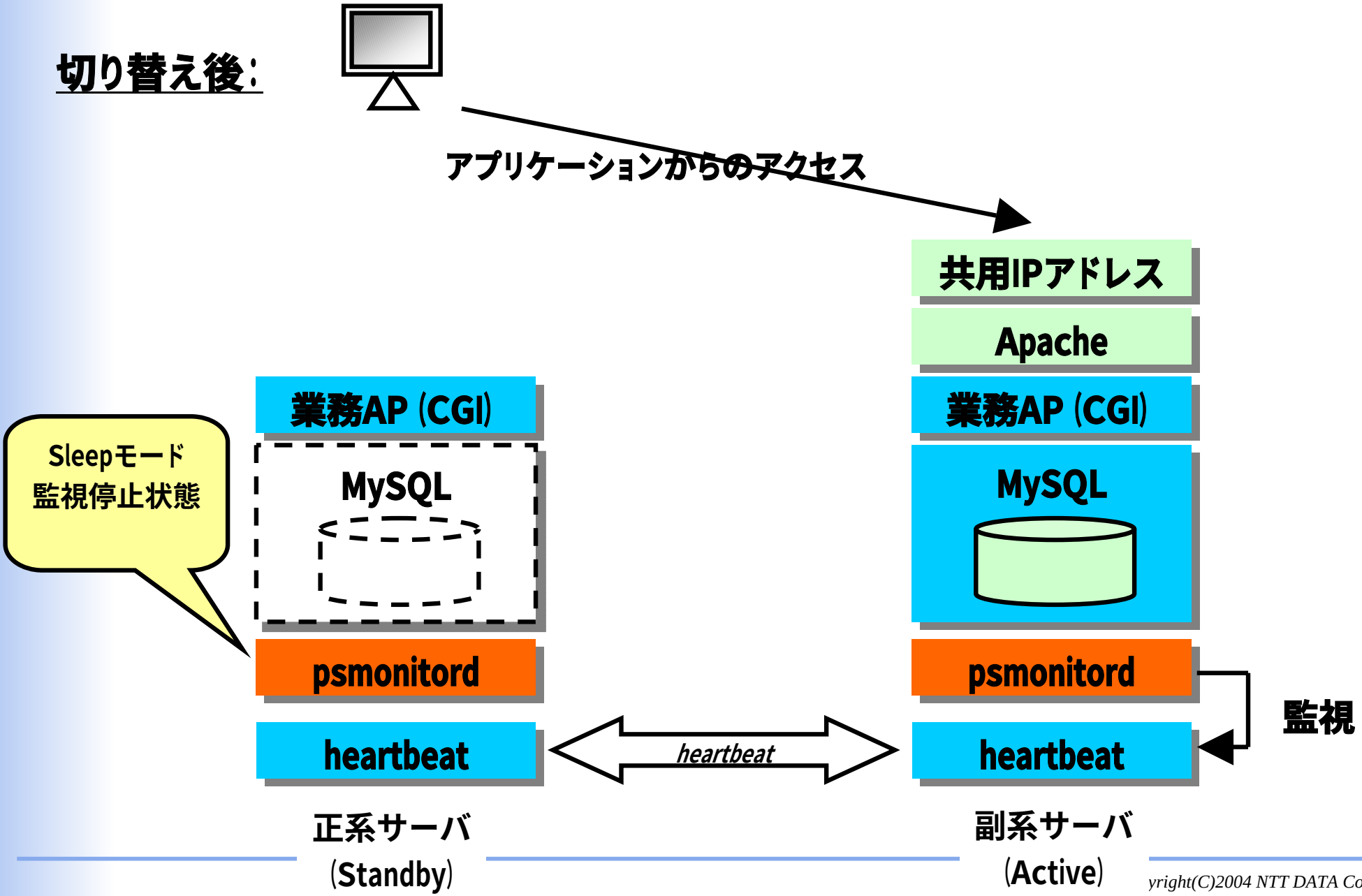
副系サーバ上で Apache が起動すると同時に、MySQL のレプリケーション構成を解除する。そのため、

/etc/ha.d/resource.d/apache に下記内容を追記する。

```
# Start MySQL Replication
/usr/local/mysql/bin/mysql -u admin -e "SLAVE STOP;"
/usr/local/mysql/bin/mysql -u admin -e "RESET MASTER;"
/usr/local/mysql/bin/mysql -u admin -e "RESET SLAVE;"
/usr/local/mysql/bin/mysql -u admin -e "CHANGE MASTER TO MASTER_HOST='', MASTER_PORT=33
06, MASTER_USER='';"
```


9．障害検知時の動作イメージ（4/4）

切り替え後：



10. psmonitord のまとめ (1/2)

新たに作り込んだ部分の整理 (psmonitord の実装範囲)：

障害要因	動作概要	実装分類
クラスタデーモン(heartbeat) 障害	フェールオーバー	psmonitord (一部追加実装)
IP監視デーモン (ipfail) 障害	フェールオーバーせず プロセス自動再生成	heartbeat 標準
プロセス監視デーモン (psmonitord) 障害	フェールオーバーせず プロセス自動再生成	heartbeat 標準
NIC片系障害	フェールオーバーせず 自動切替	Linux NIC Bonding
NIC両系障害	フェールオーバー	heartbeat 標準
ノード障害	フェールオーバー	heartbeat 標準
MySQL障害	フェールオーバー	psmonitord (新規実装)
Apache障害	フェールオーバー	psmonitord (新規実装)

10. psmonitord のまとめ (2/2)

今後の課題:

特定のお客様に必要な機能に限定して psmonitord **を実装している為、**

一般的には不足している機能もある。

1. **フェールバック**
   - **障害からの復旧時のフェールバック機能**
2. **heartbeat API の直接呼出し**
   - **フェールバックの実装には、お互いの** psmonitord **で状態を把握する必要があり、そのためには** heartbeat API **を直接呼出す必要がある**
3. **heartbeat 最新版へのキャッチアップ**
   - ver.1.2.0 **を使用しているが、**ver.1.2.1/1.2.2 **で便利な機能が追加されている**
     (hb_takeover **コマンド、**cl_status **コマンド、**client status APIs **など**)

これらの機能については、要件を考慮し検討を進める予定。

11. 苦労した部分

1. **全てオープンソースで構成**
   - 複数のプロダクト（heartbeat, ipfail, NIC Bonding そして psmonitord）を組み合わせる際の苦労
     - それぞれで独立した設定方法、同じ意味でも異なる書式。

     ＃ 商用のクラスタリング製品では NIC冗長化まで含めてパッケージング化されている。
   - 日本語での情報の少なさ

2. **閾値の決定**
   - 誤検知との戦い
     - 特にL2スイッチやNICの製品が違うだけで異なる挙動

       → 開発環境（デバッグ環境）と商用環境で機器を揃えないと厳しい

3. **監視機能の作りこみ**
   - 本構成では二重障害に未対応である為、最初の障害発生を必ず検知し、運用者に伝える仕組みが必須。本事例では、この部分も全てオープンソースで構成。

12. 終わりに

全てオープンソースを使用した、商用システム開発が成功した理由

1. **必須要件（業務、運用、システム）を明確に出来た**
   - 一般的な構成では必要な機能ではあっても、本事例に不必要な機能を明確化したため、必要最低限の実装にとどめることが出来た

2. **成功を確信することが出来た**
   - 実際に検証を行い、仮説を確信に変える事が出来た
   - オープンソースだから、中身を理解する事が出来た
   - オープンソースだから、不足する機能に対して、自己解決する事が出来た

ご清聴、ありがとうございました。

