KLab 勉強会#3

by hamano

目次

- Erlang とは
- 平行指向
- 関数型言語の話
- Erlang のプロセス
- プロセス間通信
- プロセスの性能
- プロセスの監視
- DBMS Mnesia
- 活用事例
- 参考文献

Erlang とは

「並列指向関数型言語」とその実行環境。

- 特徴
  - 並列指向、関数型言語、動的形付け、耐障害性
- 型
  - 整数、浮動小数、文字(列)、リスト、タプル、アトム、record(構造体)
- 歴史
  - 1982年 エリクソン社、電話交換機の実装に適した言語の模索を開始
  - 1988年 Erlang を開発
  - 1998年 Erlang をオープンソースとして公開

並列指向

平行プログラミングを言語レベルでサポートしている

- 並列プログラミングモデル
  - 共有メモリモデル
  - メッセージパッシングモデル

- メッセージパッシングモデル
  - MPI
  - PVM

- 共有メモリ方式でマルチプロセスプログラムを書くと
  - 排他制御が必要 -> ロックする -> デッドロックする。

代入は２度出来ない

他の関数型言語同様、Erlang では変数への破壊的代入を許しません。

```
1> A=1.
1
2> B=2.
2
3> A=3.

=ERROR REPORT==== 26-Sep-2007::14:13:55 ===
Error in process <0.30.0> with exit value: {{badmatch,3},[{erl_eval,expr,3}]}

** exited: {{badmatch,3},[{erl_eval,expr,3}]} **
```

- 代入というよりも束縛

ループはどう書くの？

再帰で書きます

```
loop(0) -> ok;
loop(N) ->
    io:fwrite("~w~n", [N]),
    loop(N - 1).
```

- スタックメモリを食いつぶすんじゃないの？
  - 末尾再起していれば大丈夫

- やってはいけない

```
loop(0) -> ok;
loop(N) ->
    loop(N - 1),
    io:fwrite("~w~n", [N]).
```

Erlang のプロセス

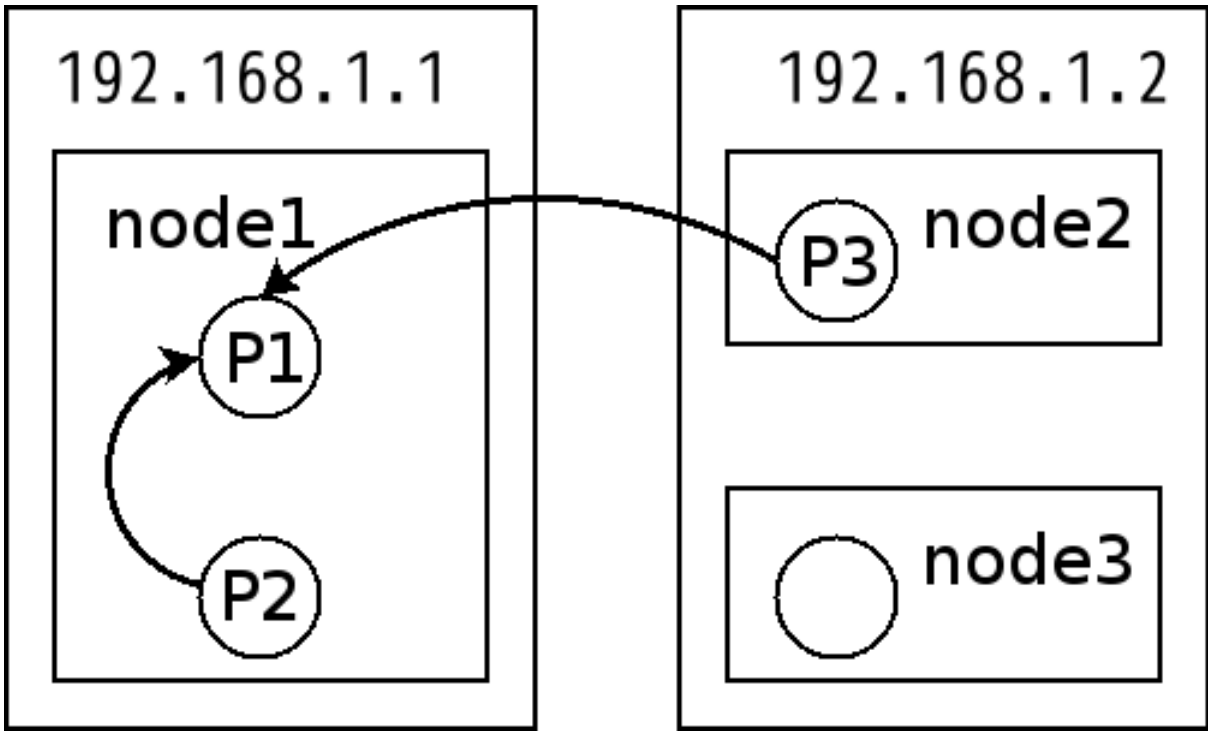
- Erlang のプロセスは OS のプロセスとは別物
- 軽くて早い
- たくさん起動できる。
- メッセージによって通信を行う
- プロセスはお互いに独立し、何も共有しない
- プロセス間リンク
  - プロセスはお互いに監視することが出来る
- ネットワーク透過性
  - プロセスは物理的に離れていても通信できる

プロセス間通信

3つの関数

- erlang:spawn
  - プロセスを生成する
- erlang:send
  - プロセスにメッセージを送る
- erlang:receive
  - メッセージを受け取る

プロセス間通信(例)



プロセス間通信の流れ

- P1 でプロセスを生成、メッセージループを行う
  - Pid = spawn(Func).
- P2 -> P1 へメッセージを送る
  - erlang:send(Pid, hello).
  - もしくは
  - Pid ! hello.
- P3 -> P1　へメッセージを送る
  - net_adm:ping('node1@192.168.1.1').
  - {Pid, 'node1@192.168.1.1'} ! hello.

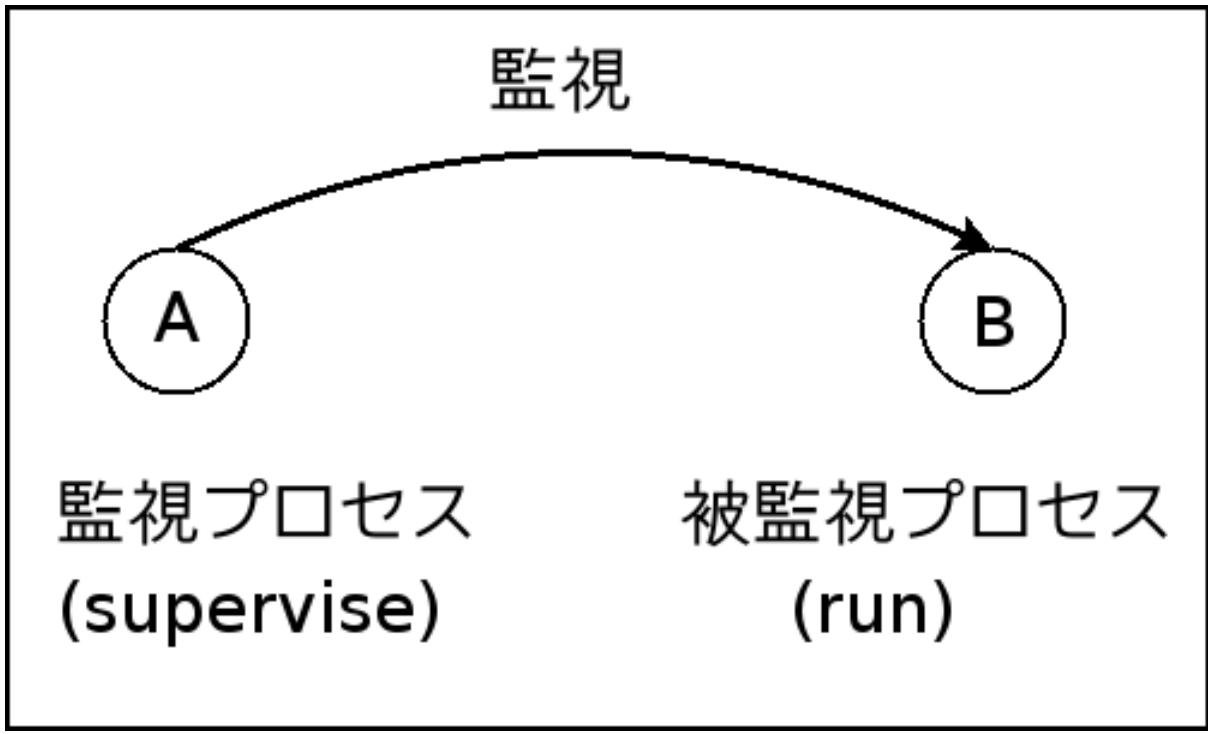
プロセスの性能(NPTL との比較)

スペック: CPU core2duo 1.86GHz メモリ 2G

	erlang プロセス	NPTL スレッド
起動速度	3.5 usec	12 usec
メモリ使用量 (1プロセス)	1272 byte (312 words) 最小476 byte	スタックサイズの最小 16k PTHREAD_STACK_M
最大 プロセス数	2G のメモリで 400万 プロセス起動した	最大スレッド数 PTHREAD_THREADS_ 16385 頑張っても10万が限度

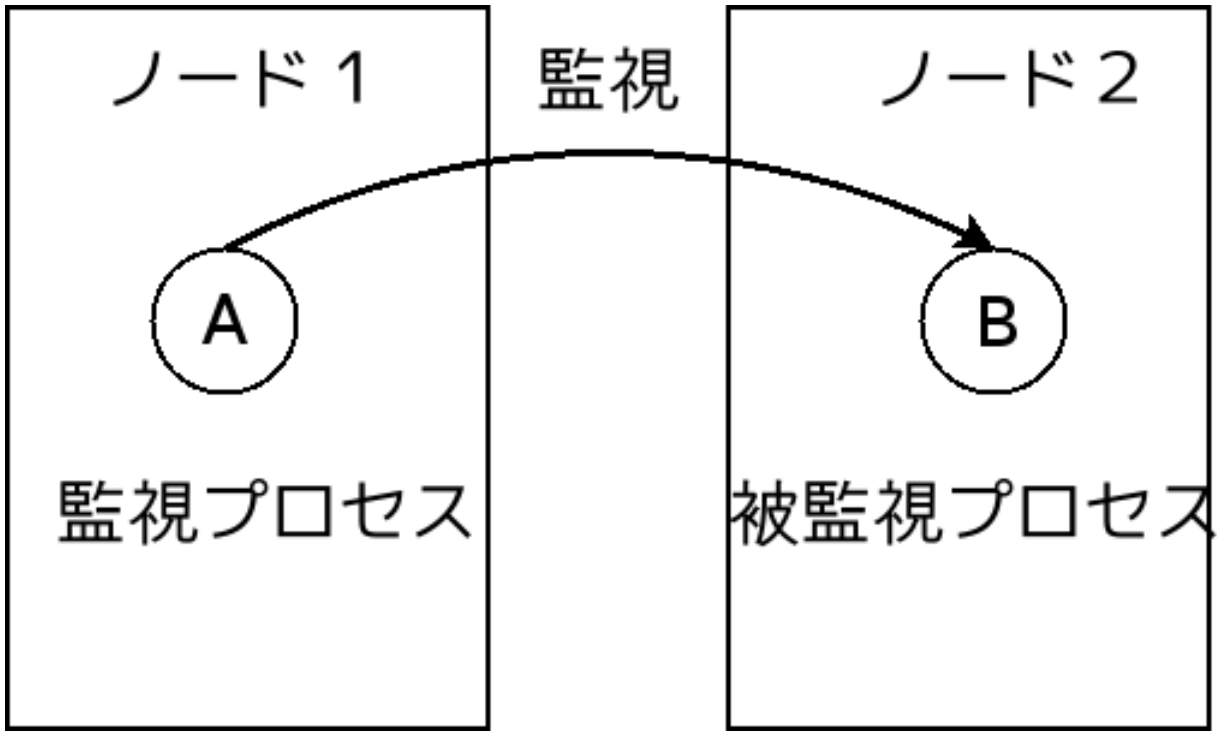
プロセスの監視(daemontools)

- スーパーバイザーモデル(daemontools の場合)



プロセスの監視(erlang)

- スーパーバイザーモデル(Erlang の場合)



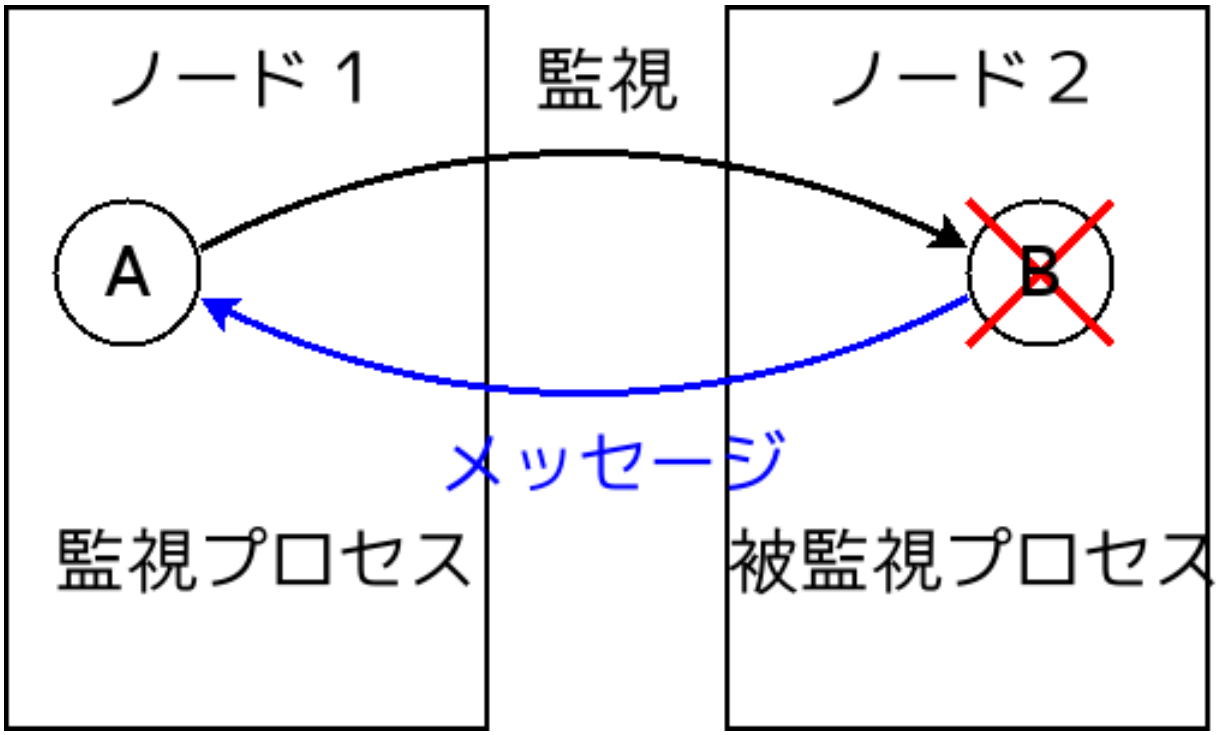
- 物理的に異なるサーバーで動作するプロセスを遠隔から監視できる。

プロセスの監視(検証)

3 種類の障害を想定して検証してみる

- プロセスが落っこちた
- ノード(VM) が落っこちた
- サーバーが落っこちた or ネットワークケーブルが切れた

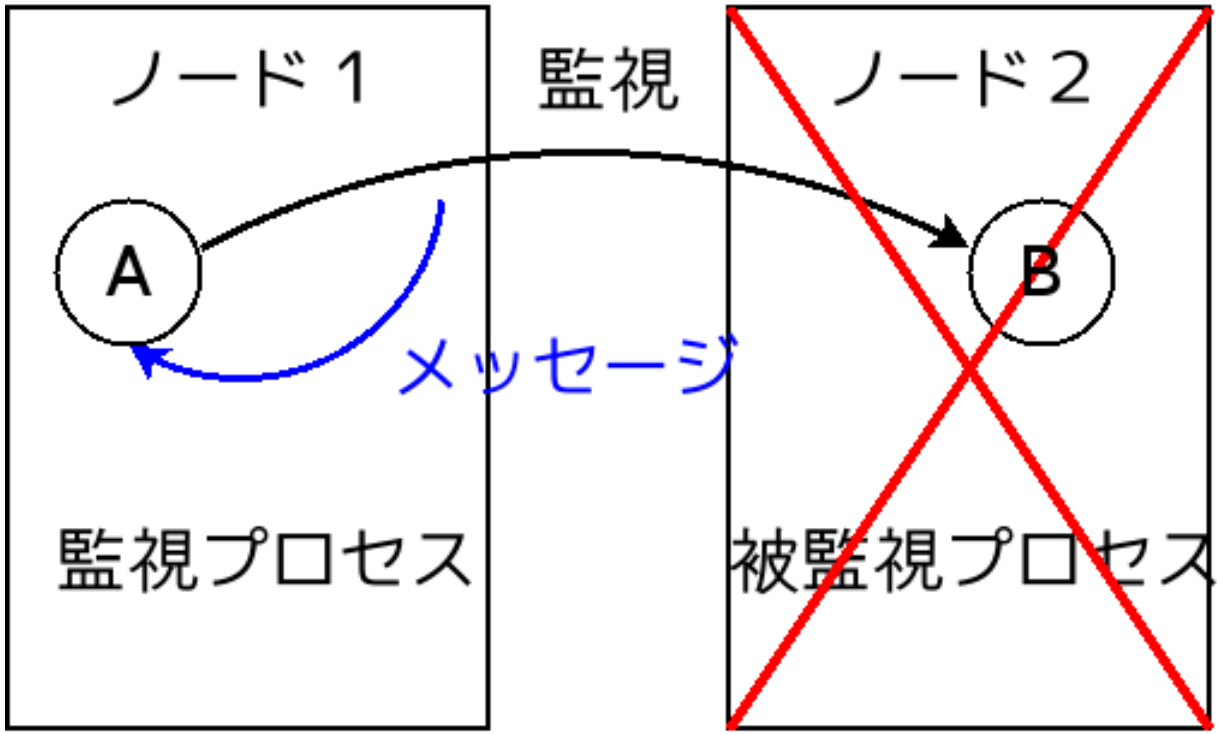
プロセスが落っこちた



メッセージの内容

** exited: {badarith,[{sv,loop,1}]} **

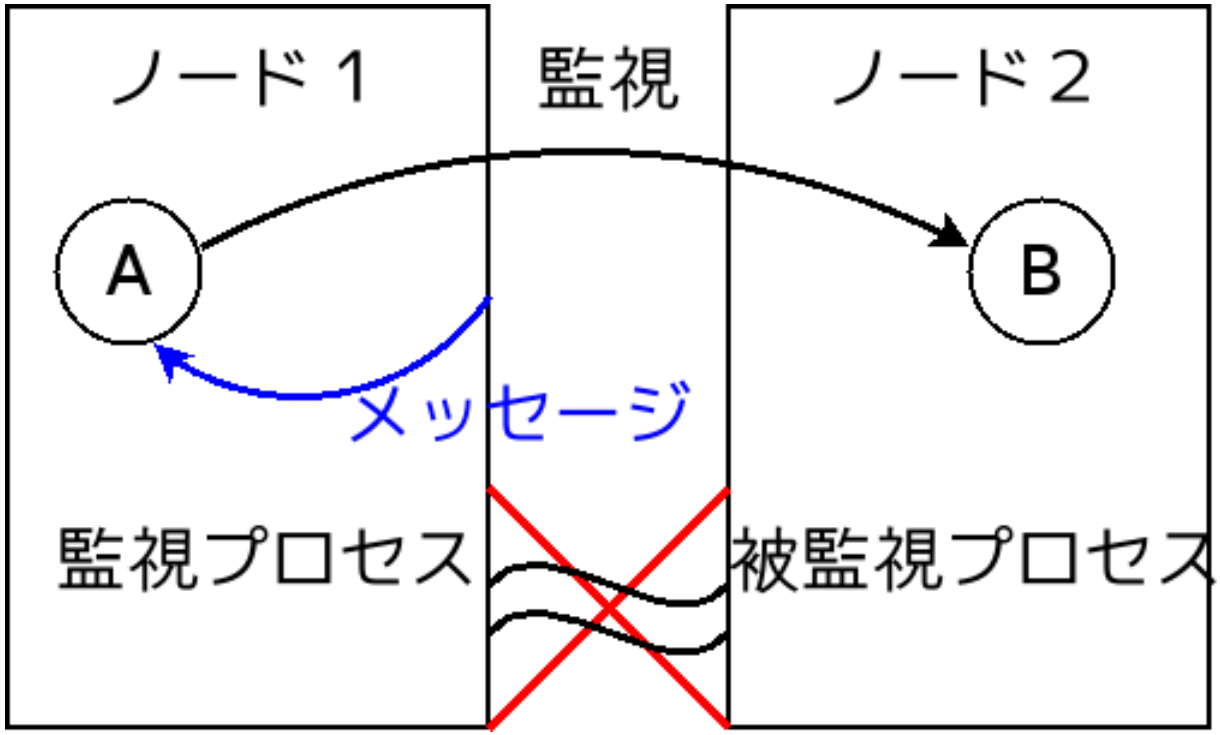
ノード(VM) が落っこちた



メッセージの内容

** exited: noconnection **

サーバーが落っこちた or ネットワークケーブルが切れた



50秒後にこんなメッセージが飛んできた

** Node 'node1@bento.klab.org' not responding **
** Removing (timedout) connection **
** exited: noconnection **

net_kernel:set_net_ticktime/1(デフォルトで60秒)で設定変更可能

DBMS Mnesia

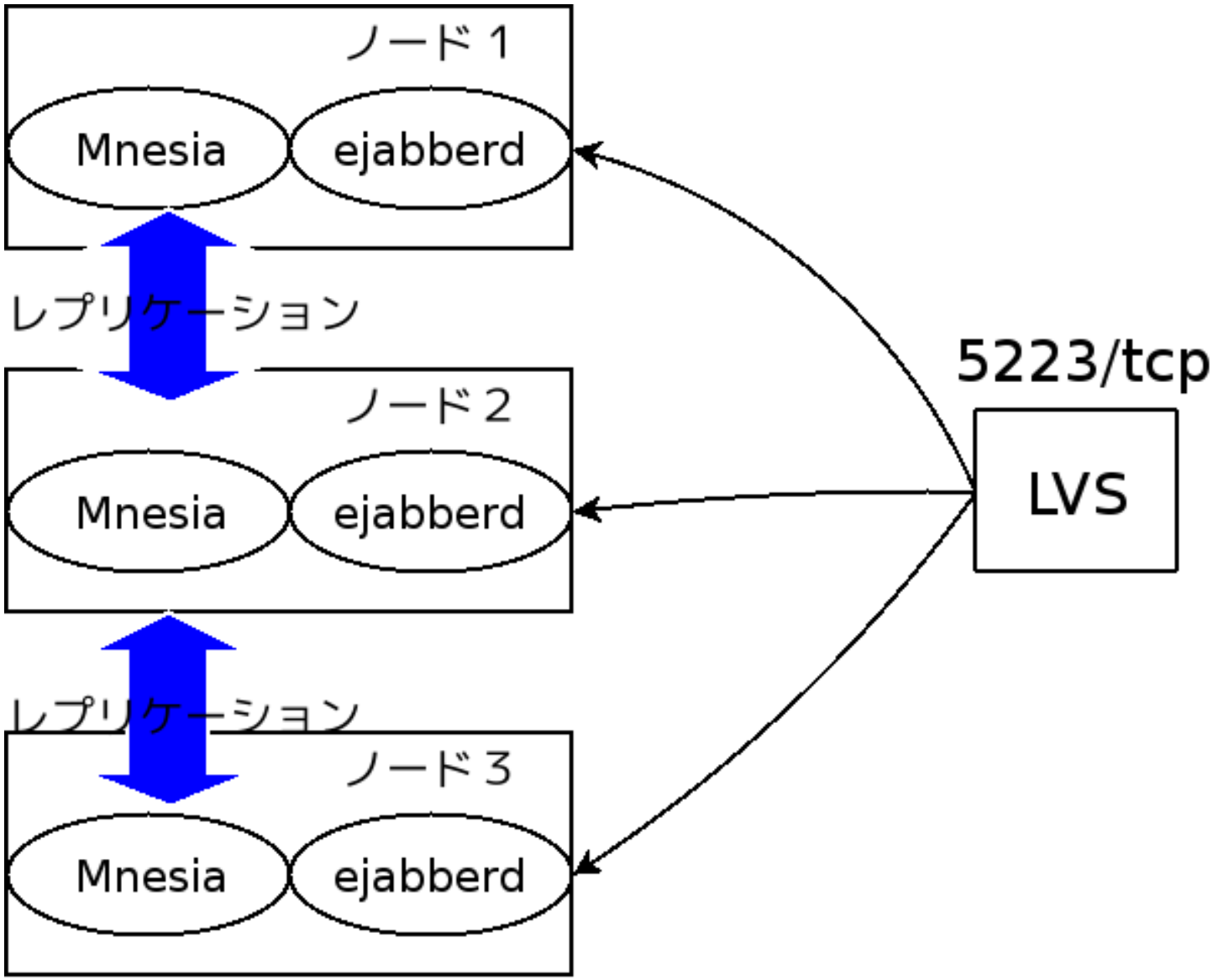
Erlang の実行環境 に付いてくる分散データベースマネジメントシステム

- 特徴
  - オブジェクトデーターベース
  - トランザクション
  - レプリケーション(マルチマスター!)
- ３種類のテーブルタイプ
  - ram_copies
  - disc_copies
  - disc_only_copies

活用事例

- ememcached
  - erlang で書いた memcached
- jabber.jp
  - KLab が運営しているメッセンジャーサービス
  - http://www.jabber.jp/

jabber.jp の場合



参考文献

- 公式サイト
  - http://www.erlang.org/

- Programming Erlang
  - http://www.pragmaticprogrammer.com/titles/jaerlang/

おしまい