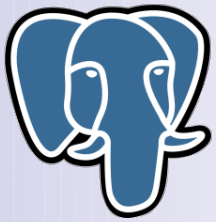
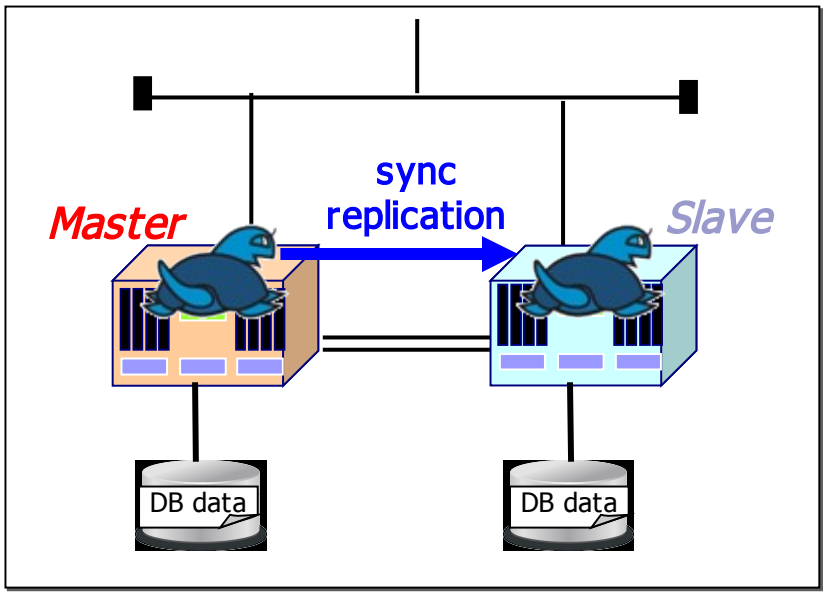
Clustering of PostgreSQL(replication) using Pacemaker

2012/9/29

Takatoshi MATSUO



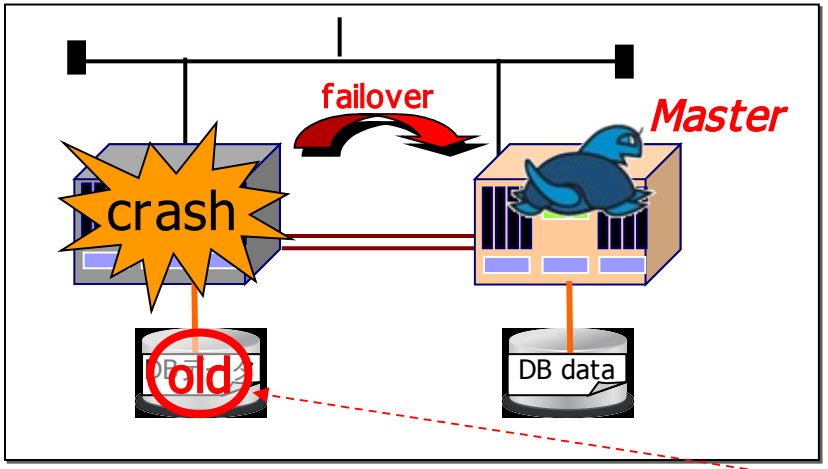
3 major functions



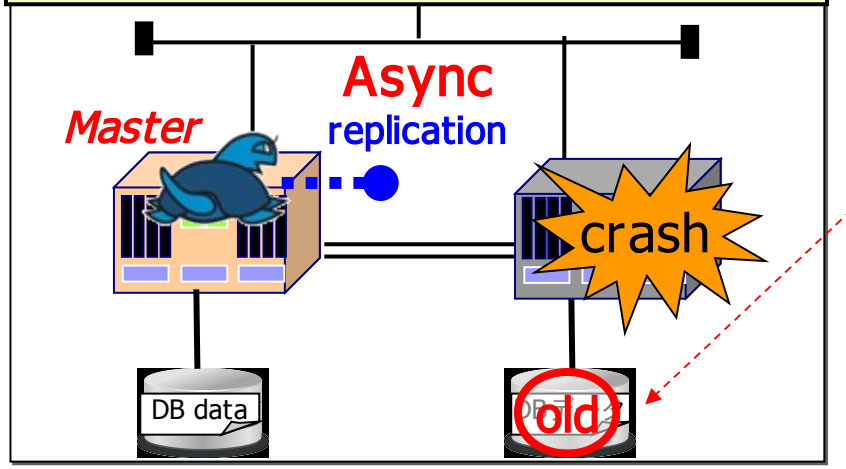
crash of Master

crash of Slave

1. failover



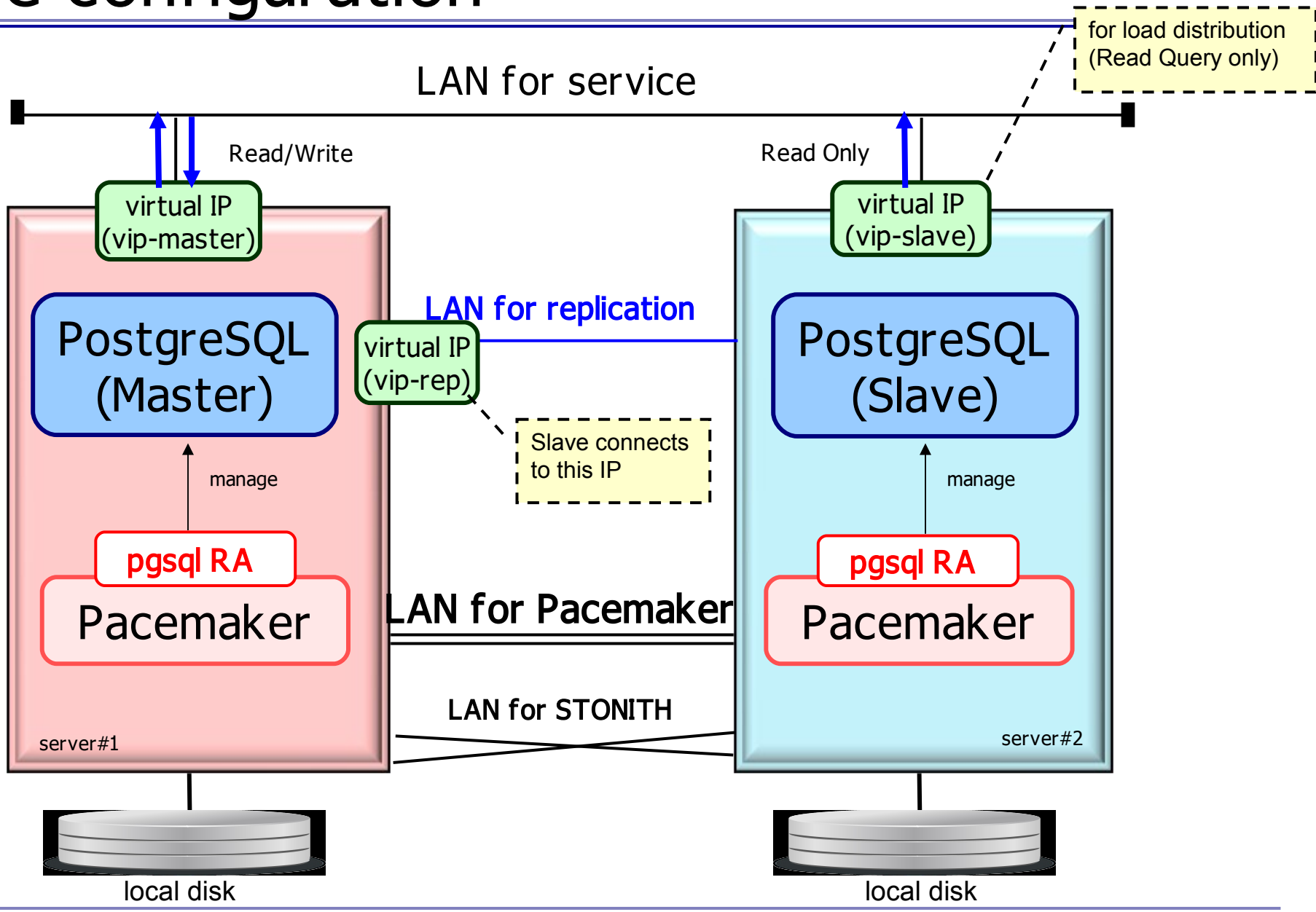
2. switch setting between sync and async



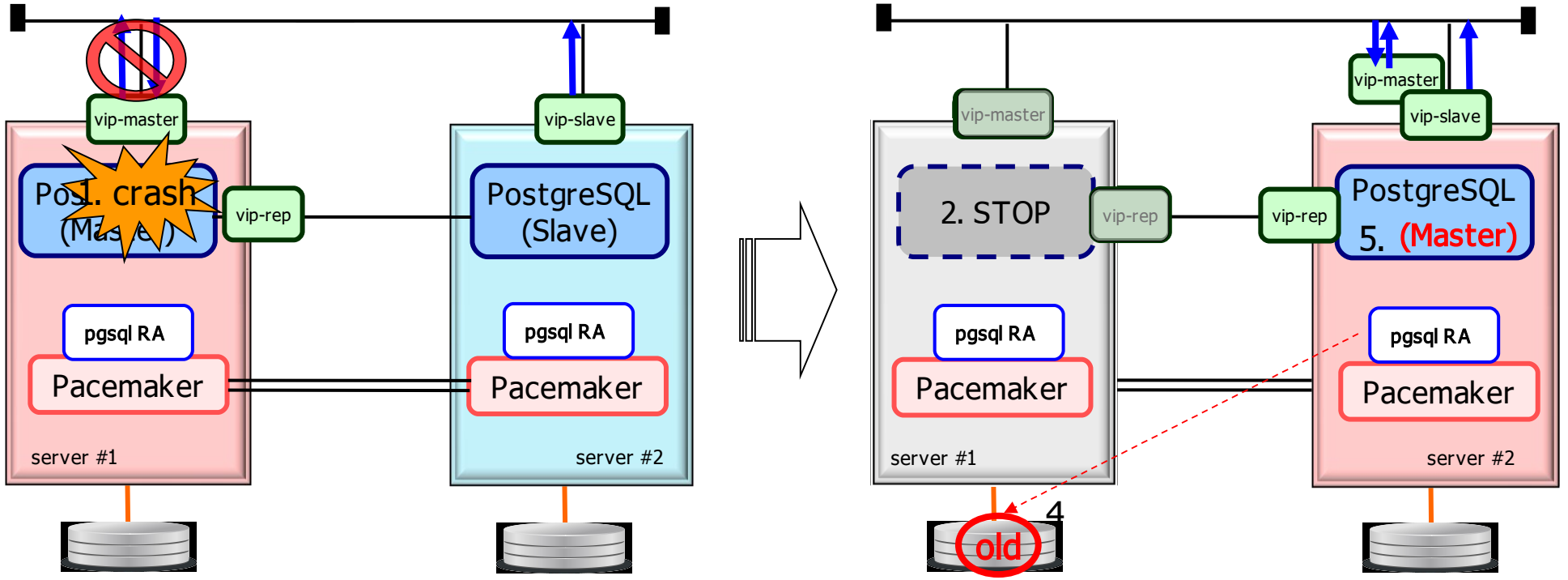
3 manage old data

 <- Japanese PostgreSQL mascot

Base configuration



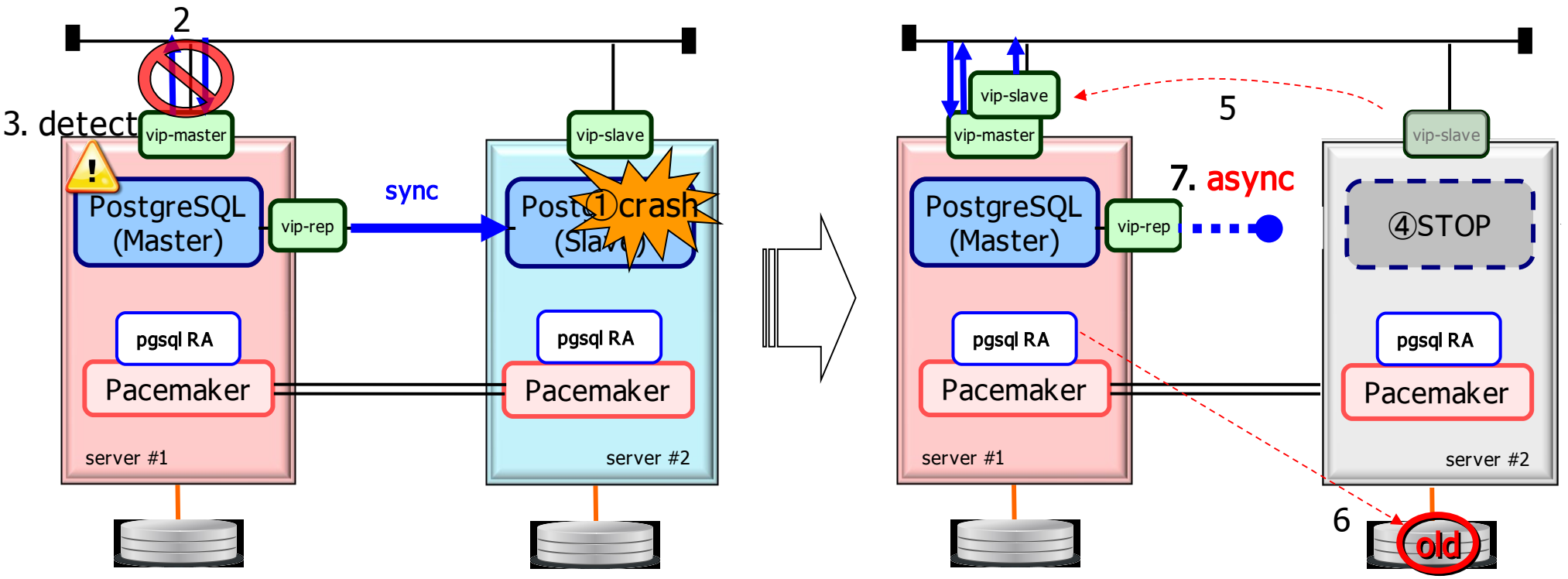
Basic action 1 : Master's failover



1. detect PostgreSQL crash

2. stop PostgreSQL on #1
3. stop virtual IPs(vip-master, vip-rep, vip-slave)
4. record that #1's data is old
5. promote #2's PostgreSQL
6. start IPs (vip-master, vip-rep, vip-slave) on #2

Basic action 2 : switch setting between sync and async



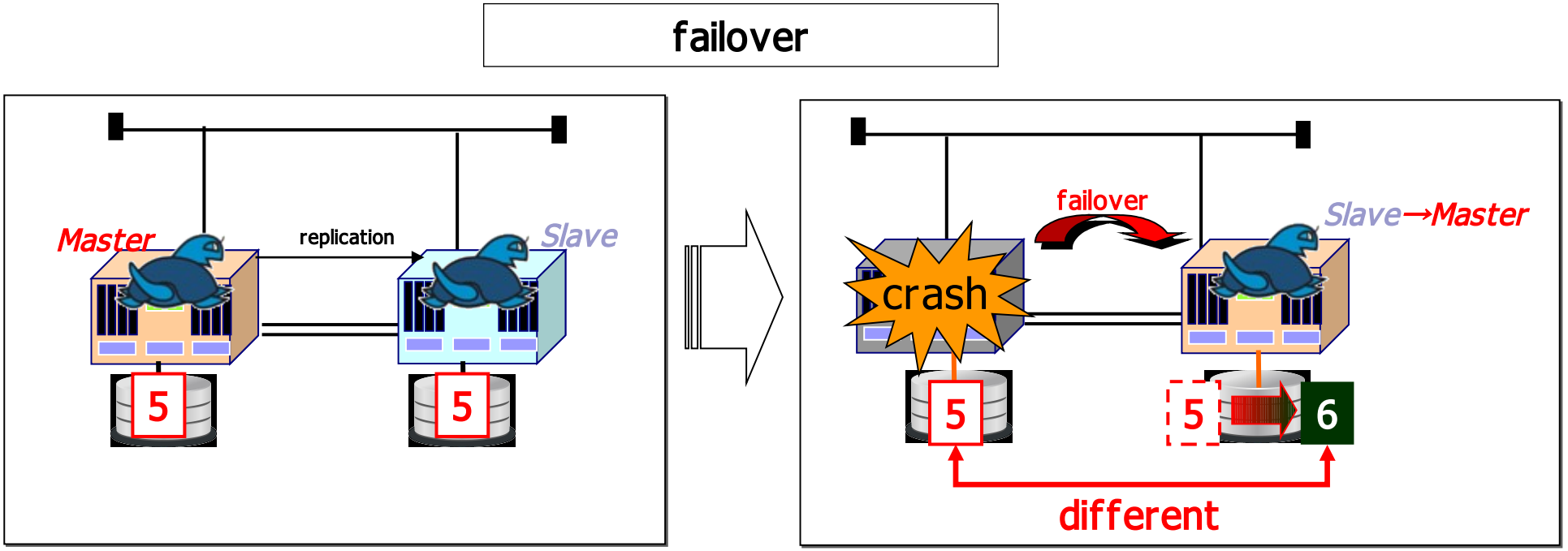
1. crash of Slave (ex : kill walreceiver)
2. Master's transaction is stopped
   ~ wait replication timeout ~
3. detect cutoff of replication

```
SELECT * from pg_stat_replication
```

4. stop PostgreSQL on #2
5. move vip-lave from #1 to #2
6. record that #2's data is old
7. swith setting from sync to async on #1
   -> resume transaction

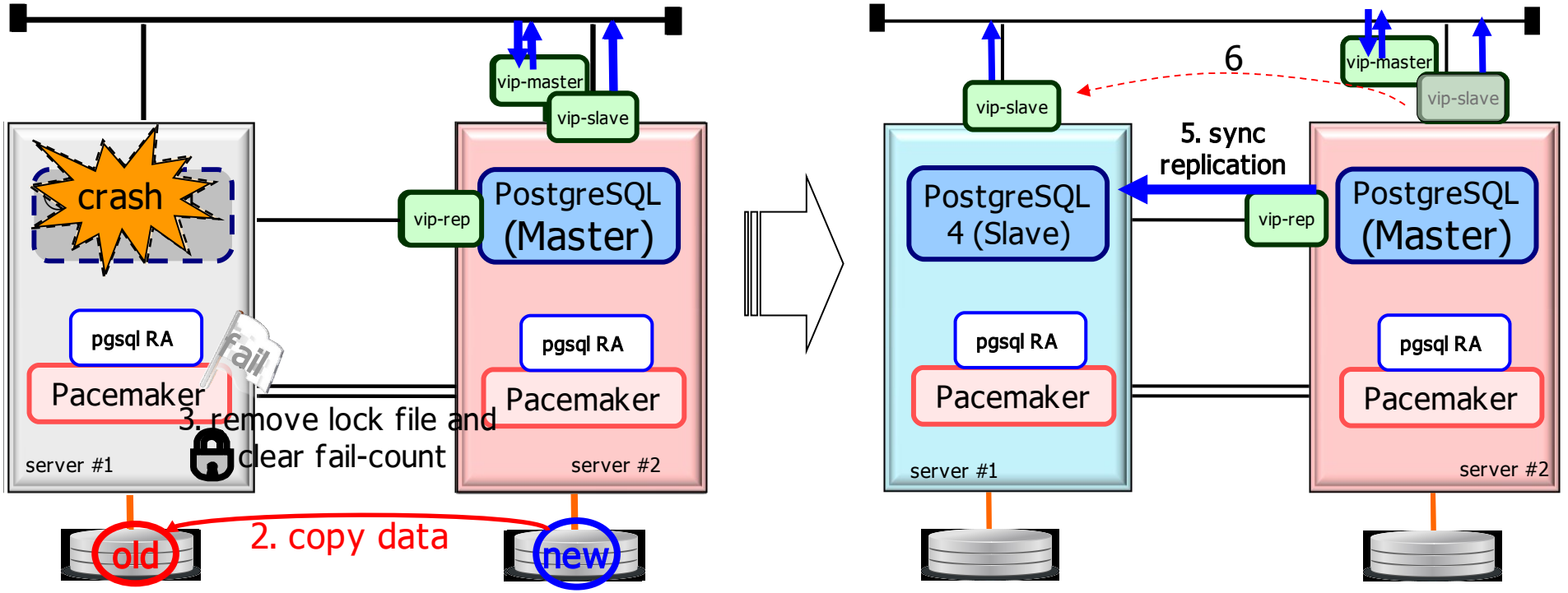
TimelineID

- TimelineID is incremented when promote is called
- Slave can't connect to Master if TimelineID is different



Need to copy DB data from new Master to old Master
to make up the number

operation 1: recovery after Master's failover

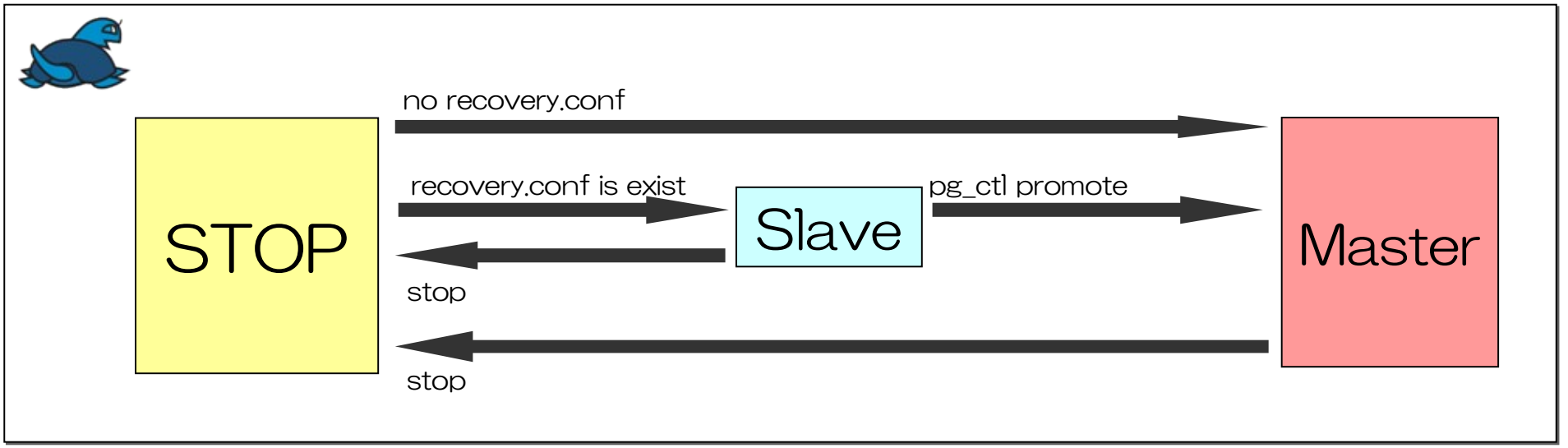


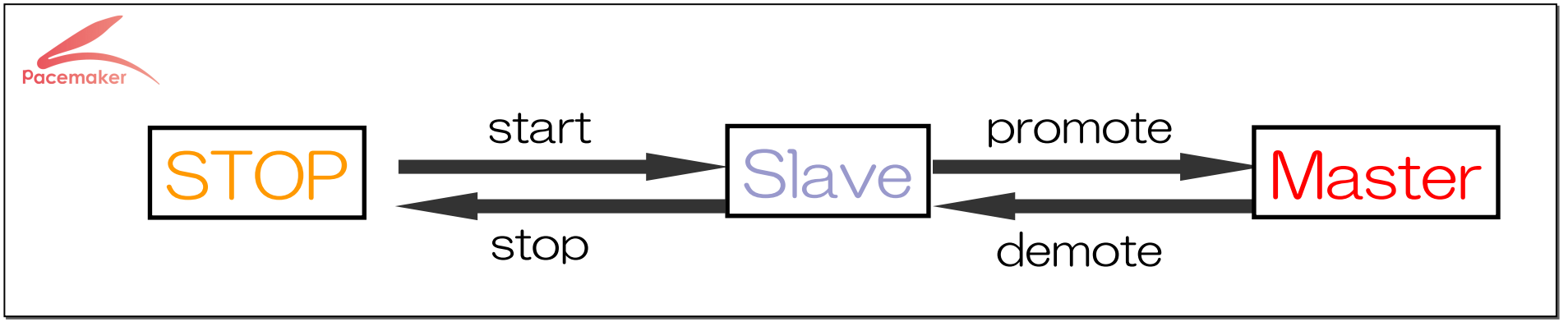
1. failure restoration
2. copy data from #2 to #1
   → make up the number
3. remove lock file and clear
   Pacemaker's fail-count on #1

(manually)

4. start PostgreSQL as slave on #1
5. start replication
   -> connect as async at first
   -> swith from async to sync
6. move vip-slave from #2 to #1

Transition of PostgreSQL and Pacemaker

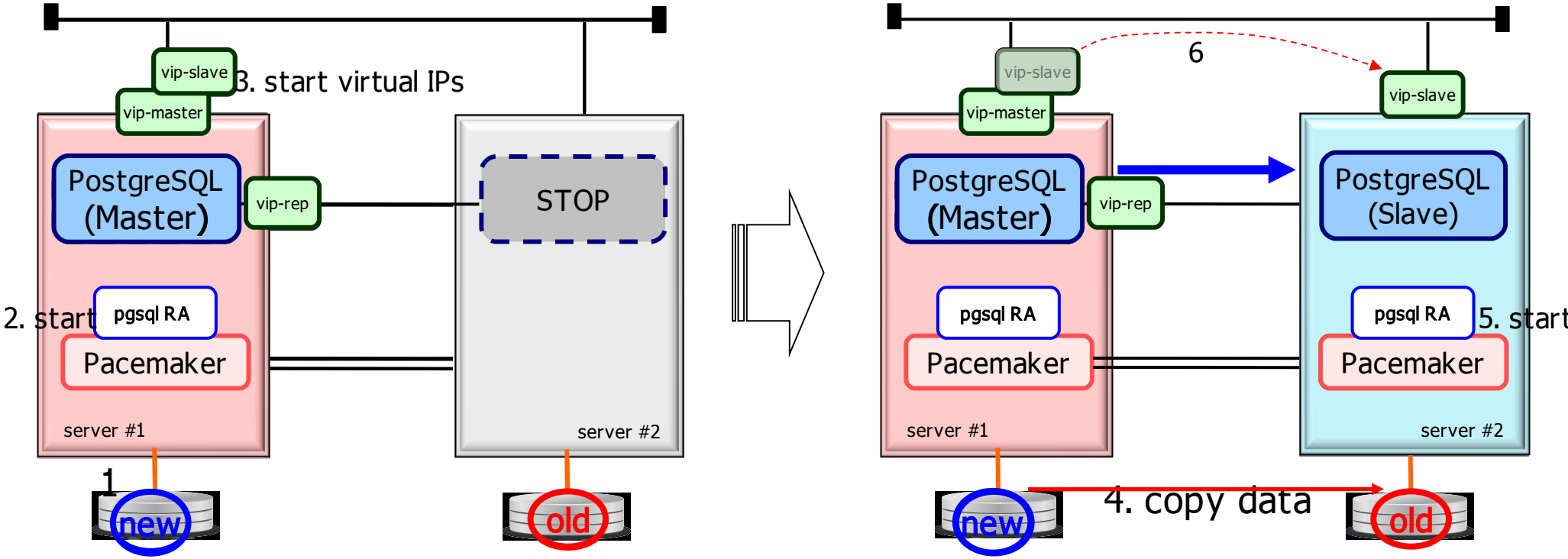




Pacemaker starts Master through Slave invariably

-> TimeineID is incremented when Master is started.

operation 2 : start



1. select server that has new data
2. start Pacemaker on selected server
   → start as Slave → promote
   → occur gap of TimelineID
3. start virtual IPs

(manually)

4. copy data from #1 to #2
   → make up the number
5. start Pacemaker
   → start replication
6. move vip-slave from #1 to #2

(manually)

detail

Term

To distinguish precisely between PostgreSQL's status and Pacemaker's status, I will use these terms.

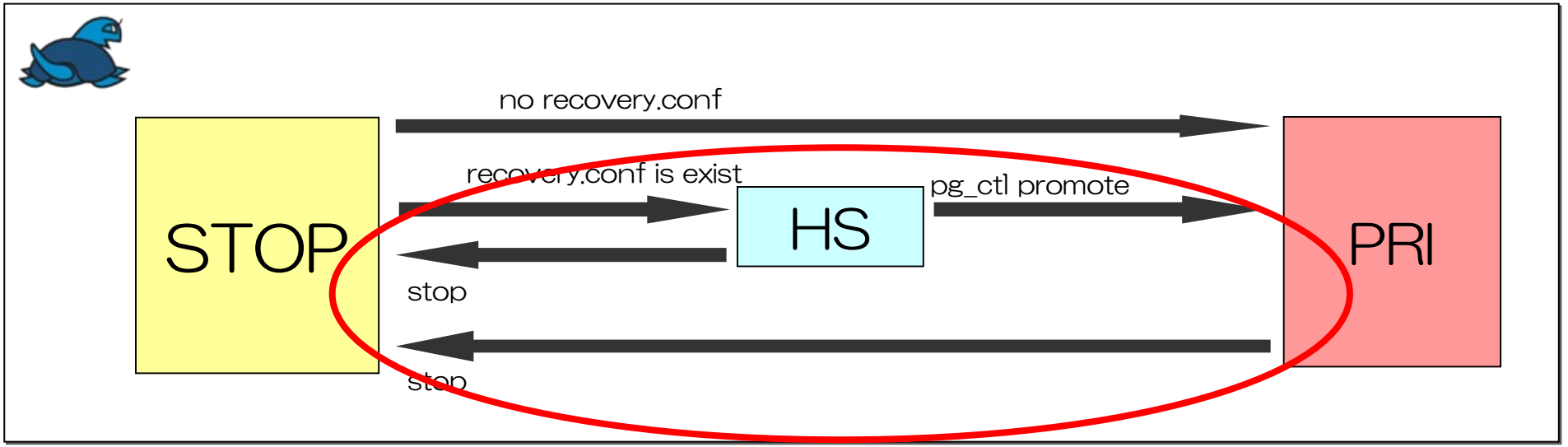
PostgreSQL's status

- PRI : running as Primary(Master)
- HS : running as Hot Standby(Slave)
- STOP : stopped

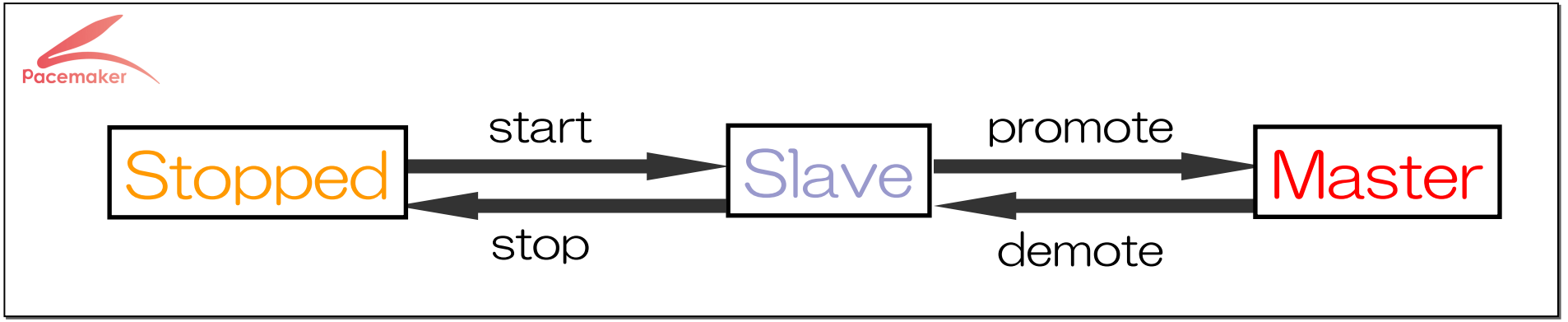
Pacemaker's status

- Master : manage PostgreSQL as Master
- Slave : manage PostgreSQL as Slave
- Stopped : manage PostgreSQL as Stopped

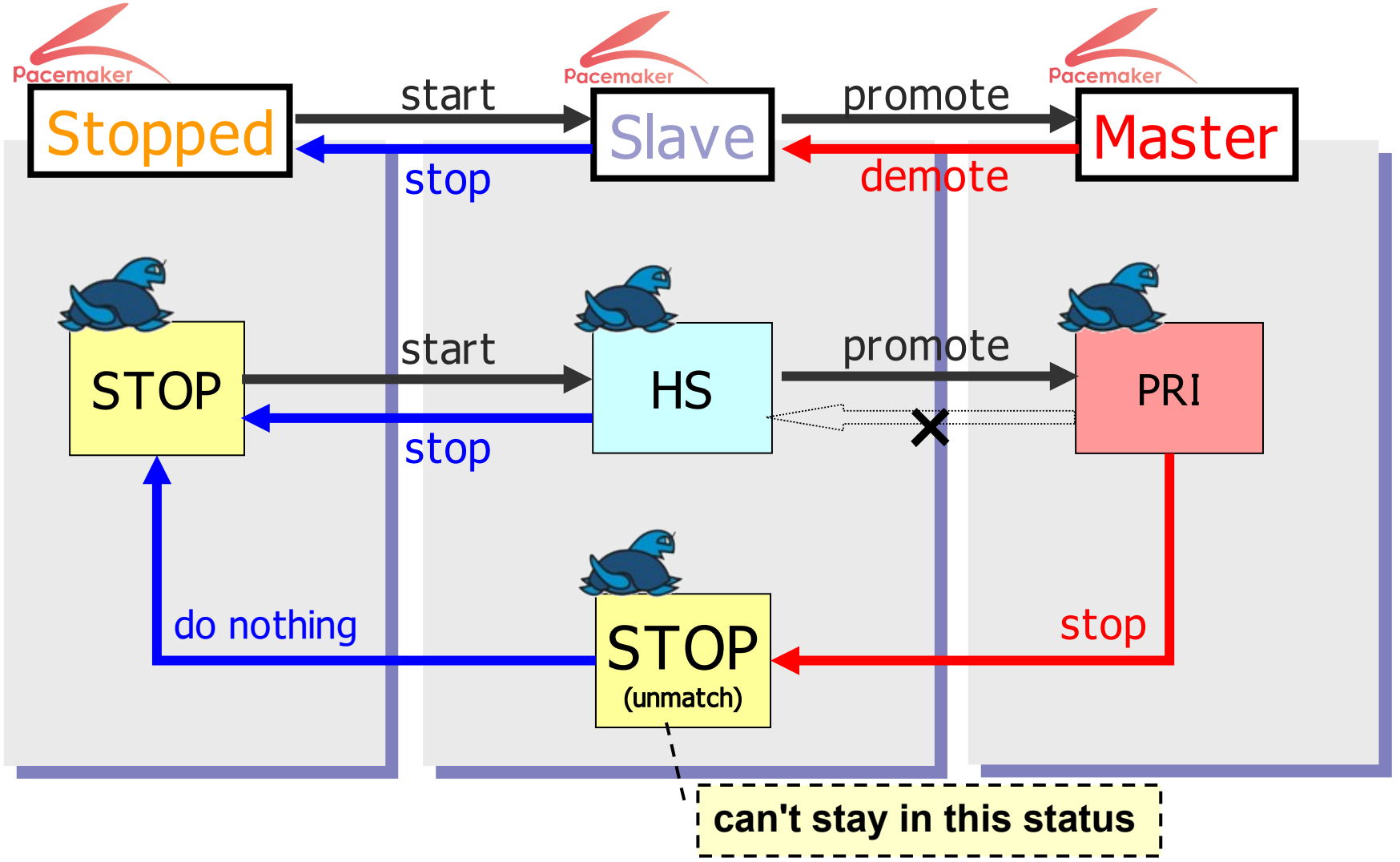
after fixing terms



RA uses these transitions



Status Mapping between Pacemaker and PostgreSQL



multiple status in HS

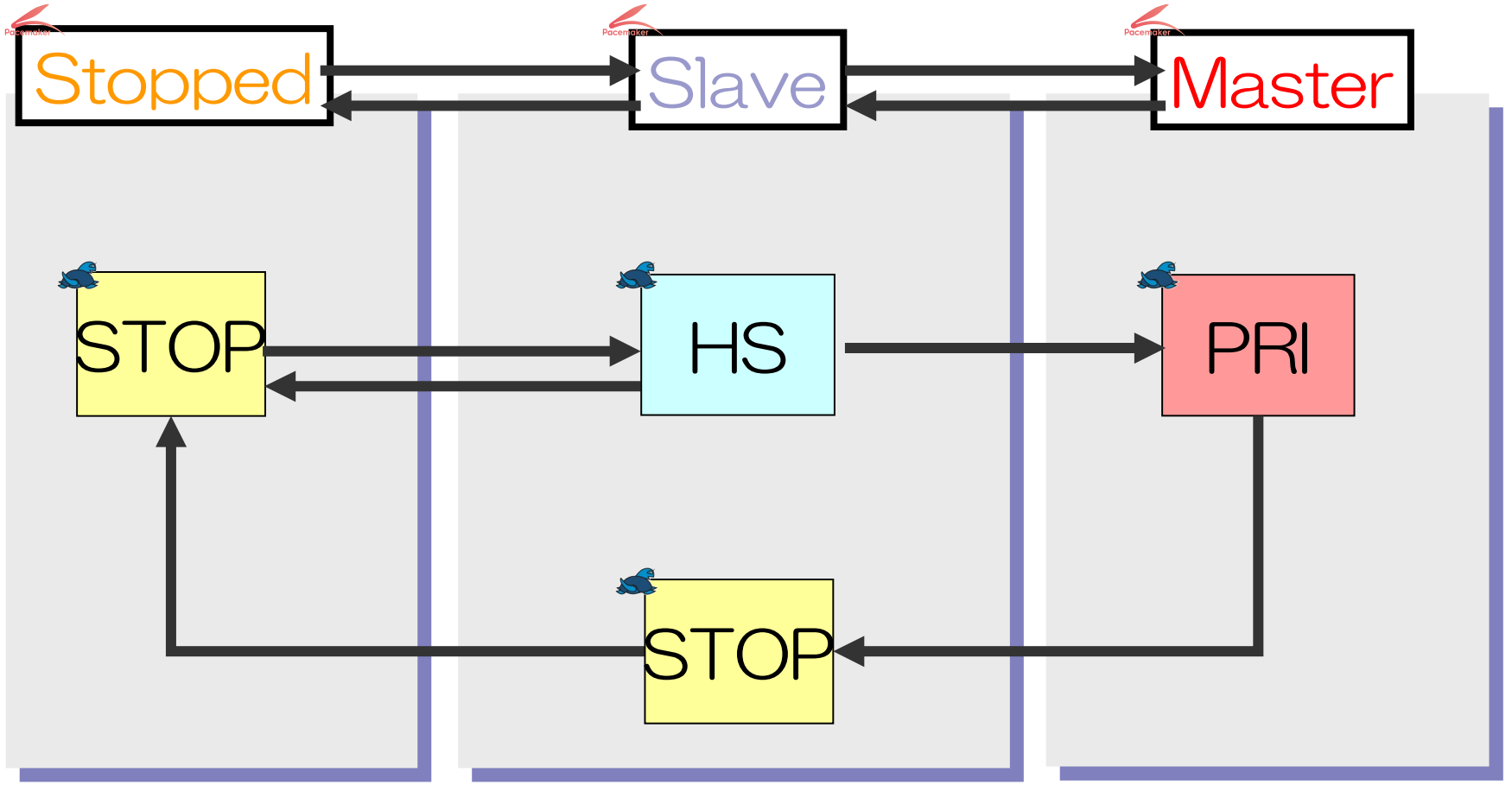
- There are multiple status in HS
  1. no connection ・・・ HS:alone
  2. connect to PRI
     a. except sync replication ・・・ HS:(others)
     b. sync replication ・・・ HS:sync

HS:(others)
- HS:connected
- HS:async

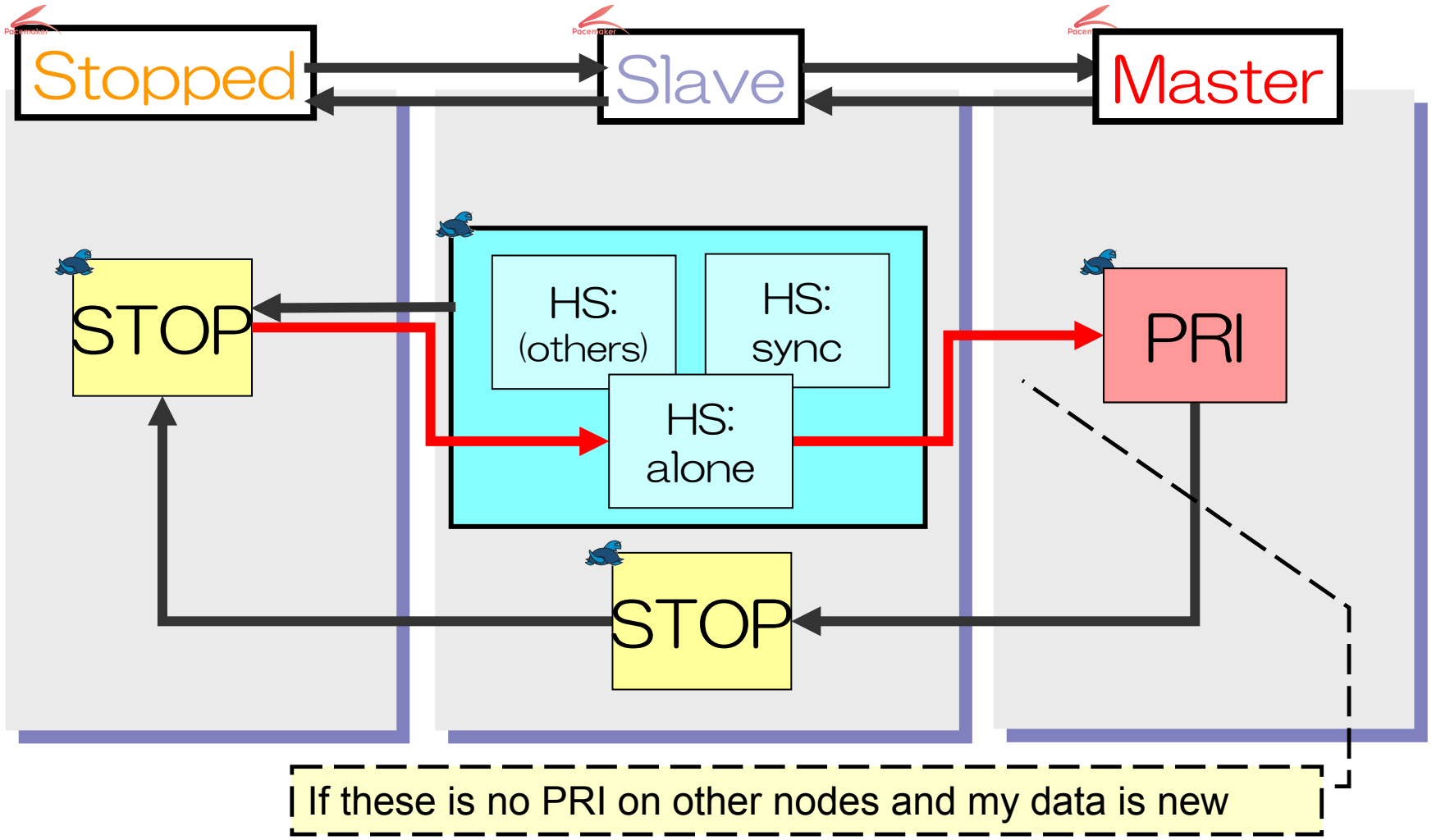
Pacemaker manages these status as ″pgsql-status″ attribute

assume that resource ID is pgsql

Status Mapping (before)



Status Mapping (after)

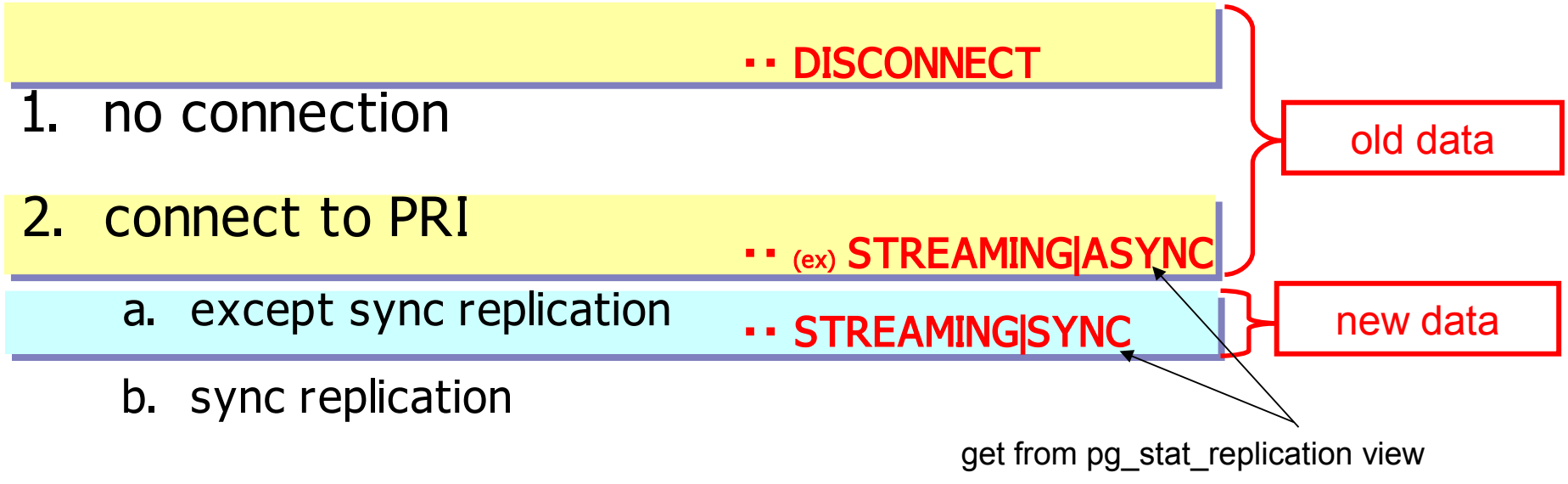


Judgment whether HS's data is new

- when PRI exists
  - record my data status based on connection status
    - If PRI is exist, HS can't be PRI, so RA only record the status.
- when PRI crashes
  - use the status that was recorded just before PRI is broken
- Initial start (PRI has never even existed)
  - If there is HS in other nodes
    - compare data
  - If there is no HS in other nodes
    - judge that my data is new

status of HS's data

- recorded status when PRI exists on other nodes



- In PRI ・・ LATEST — new data

Pacemaker manages these status as "pgsql-data-status" attribute

assume that resource ID is pgsql

difference "pgsql-status" "pgsql-data-status"

pgsql-status

- running status of PostgreSQL
  - STOP
  - HS:alone, HS:async, HS:sync
  - PRI
  - UNKNOWN

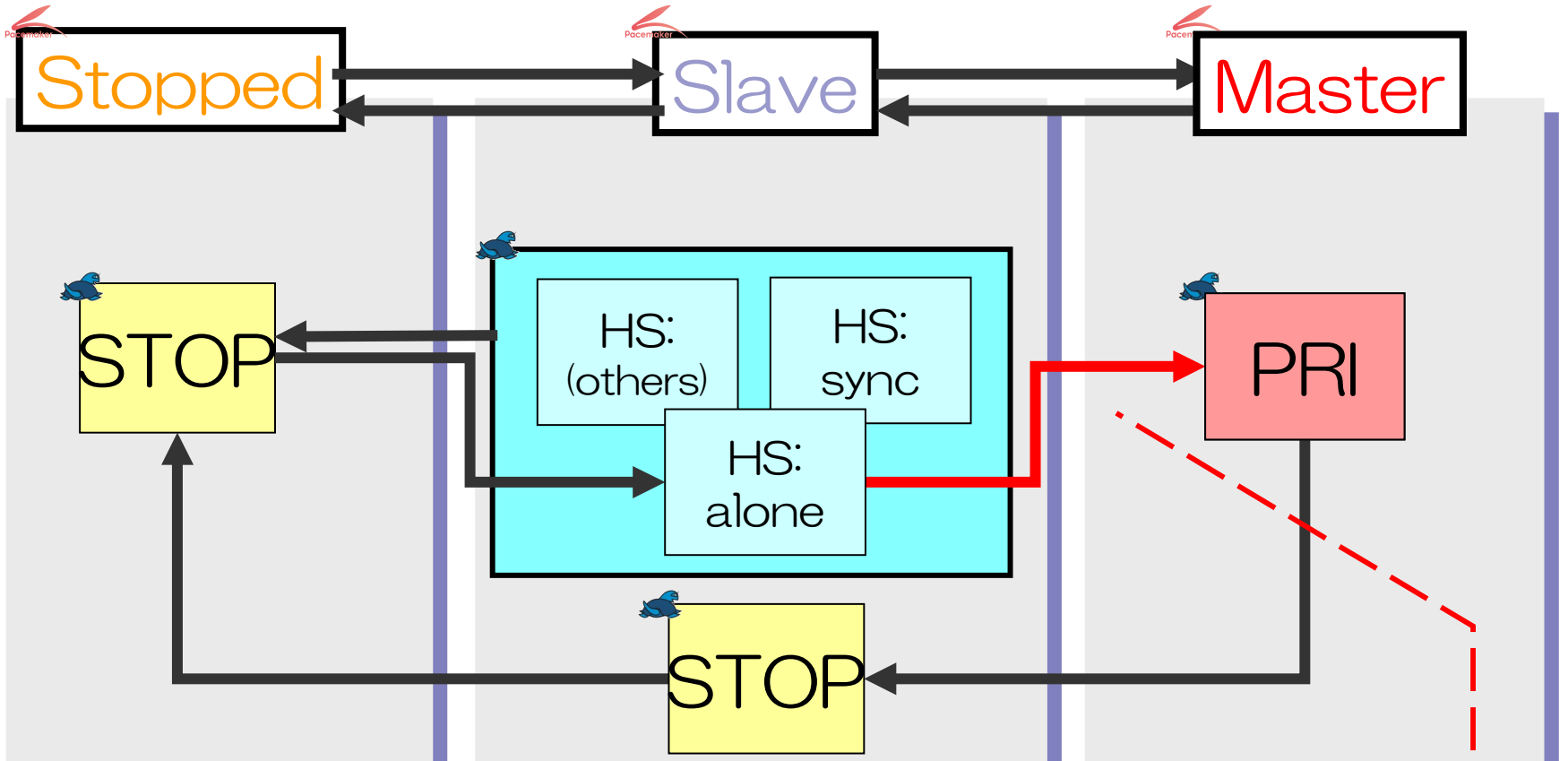
Use
- know running status of PostgreSQL
- work together with other resources

pgsql-data-status

- data status based on relationship to PRI
  - DISCONNECTED
  - STREAMING|ASYNC, STREAMING|SYNC and so on
  - LATEST
- status that was recorded just before PRI crashes remains
  - not always correspond with running status
    - It's not importable that pgsql-status=HS:alone and pgsql-data-status = LATEST

Use
- record data's old and new
- judge wheter Pacemaker can be Master

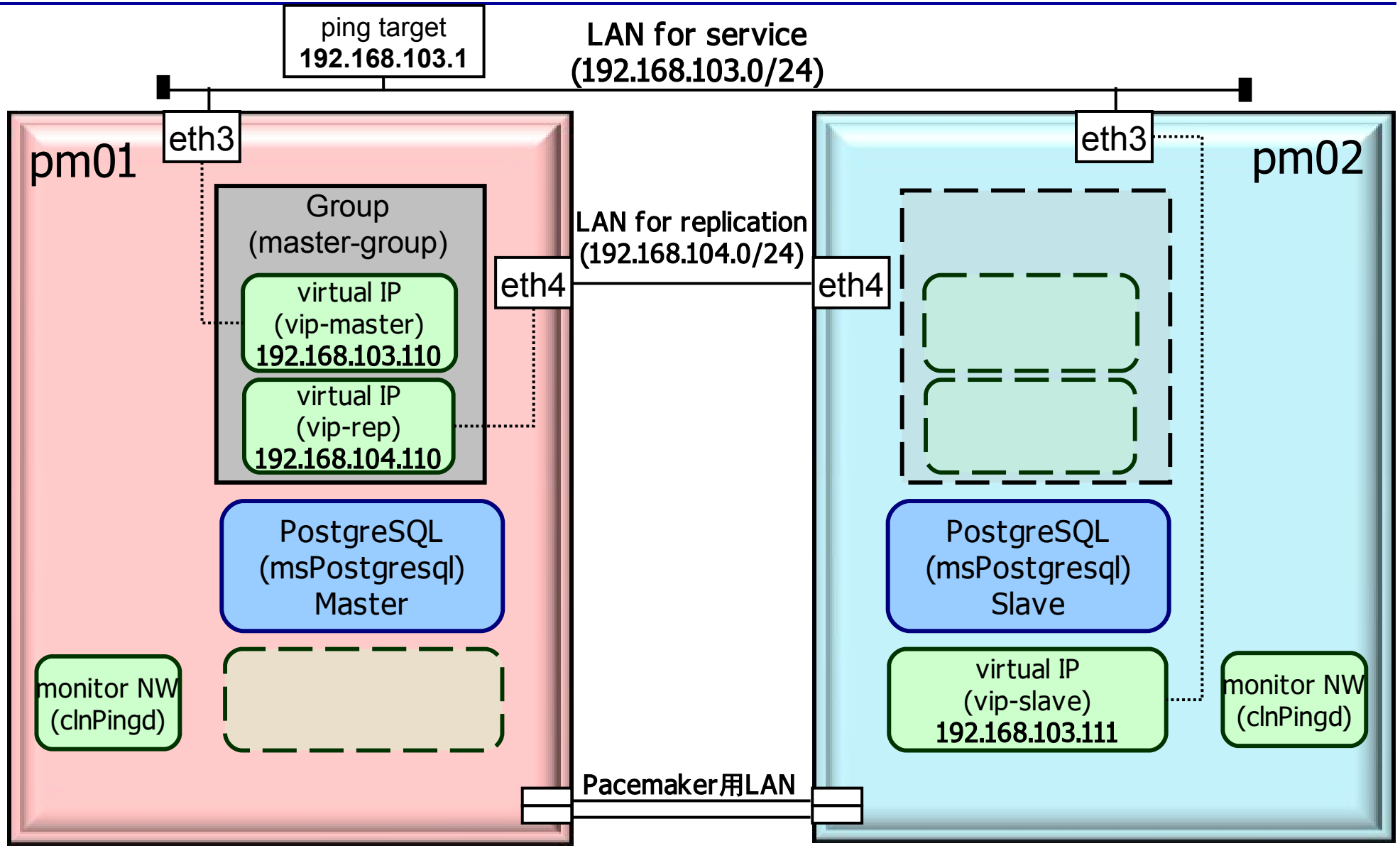
Status Mapping (final)



When there is no PRI on other nodes and my pgsql-data-status is new (=LATEST or STREAMING|SYNC).
Or I have newer data than other node's one. (initial start-up)

Sample configuration

Example (simple ver.)



skip the STONITH because of simple version

Sample Pacemaker configuration 1/2

```
property \
   no-quorum-policy="ignore" \
   stonith-enabled="false" \
   crmd-transition-delay="0s"

rsc_defaults \
   resource-stickiness="INFINITY" \
   migration-threshold="1"

ms msPostgresql pgsql \
   meta \
      master-max="1" \
      master-node-max="1" \
      clone-max="2" \
      clone-node-max="1" \
      notify="true"

group master-group \
   vip-master \
   vip-rep \
   meta \
      ordered="false"

clone clnPingd \
   pingCheck

primitive vip-master ocf:heartbeat:IPaddr2 \
   params \
      ip="192.168.103.110" \
      nic="eth3" \
      cidr_netmask="24" \
   op start   timeout="60s" interval="0s"  on-fail="restart" \
   op monitor timeout="60s" interval="10s" on-fail="restart" \
   op stop    timeout="60s" interval="0s"  on-fail="block"
```

use Master/Slave

```
primitive vip-rep ocf:heartbeat:IPaddr2 \
   params \
      ip="192.168.104.110" \
      nic="eth4" \
      cidr_netmask="24" \
   meta \
      migration-threshold="0" \
   op start   timeout="60s" interval="0s"  on-fail="stop" \
   op monitor timeout="60s" interval="10s" on-fail="restart" \
   op stop    timeout="60s" interval="0s"  on-fail="block"

primitive vip-slave ocf:heartbeat:IPaddr2 \
   params \
      ip="192.168.103.111" \
      nic="eth3" \
      cidr_netmask="24" \
   meta \
      resource-stickiness="1" \
   op start   timeout="60s" interval="0s"  on-fail="restart" \
   op monitor timeout="60s" interval="10s" on-fail="restart" \
   op stop    timeout="60s" interval="0s"  on-fail="block"

primitive pgsql ocf:heartbeat:pgsql \
   params \
      pgctl="/usr/pgsql-9.1/bin/pg_ctl" \
      psql="/usr/pgsql-9.1/bin/psql" \
      pgdata="/var/lib/pgsql/9.1/data/" \
      rep_mode="sync" \
      node_list="pm01 pm02" \
      restore_command="cp /var/lib/pgsql/9.1/data/pg_archive/%f %p" \
        primary_conninfo_opt="keepalives_idle=60 \
                              keepalives_interval=5 keepalives_count=5" \
      master_ip="192.168.104.110" \
      stop_escalate="0" \
   op start   timeout="30s" interval="0s"  on-fail="restart" \
   op stop    timeout="30s" interval="0s"  on-fail="block" \
   op monitor timeout="30s" interval="11s"  on-fail="restart" \
   op monitor timeout="30s" interval="10s"  on-fail="restart" role="Master" \
   op promote timeout="30s" interval="0s"  on-fail="restart" \
   op demote  timeout="30s" interval="0s"  on-fail="block" \
   op notify  timeout="60s" interval="0s"
```

Main setting

Sample Pacemaker configuration 2/2

```
primitive pingCheck ocf:pacemaker:pingd \
   params \
      name="default_ping_set" \
      host_list="192.168.103.1" \
      multiplier="100" \
   op start   timeout="60s" interval="0s"  on-fail="restart" \
   op monitor timeout="60s" interval="2s"  on-fail="restart" \
   op stop    timeout="60s" interval="0s"  on-fail="ignore"

### Resource Location ###
location rsc_location-1 msPostgresql \
   rule -inf: not_defined default_ping_set or default_ping_set lt 100

location rsc_location-2 vip-slave \
   rule  200: pgsql-status eq HS:sync \
   rule  100: pgsql-status eq PRI \
   rule -inf: not_defined pgsql-status \
   rule -inf: pgsql-status ne HS:sync and pgsql-status ne PRI

### Resource Colocation ###
colocation rsc_colocation-1 inf:  msPostgresql       clnPingd
colocation rsc_colocation-2 inf:  master-group       msPostgresql:Master

### Resource Order ###
order rsc_order-1 0:   clnPingd            msPostgresql        symmetrical=false
order rsc_order-2 inf: msPostgresql:promote  master-group:start   symmetrical=false
order rsc_order-3 0:   msPostgresql:demote   master-group:stop    symmetrical=false
```

I recommend adding STONITH resource on business.

pick up

Main setting of pgsql RA

```
primitive pgsql ocf:heartbeat:pgsql ¥
    params ¥
        pgctl="/usr/pgsql-9.1/bin/pg_ctl" ¥
        psql="/usr/pgsql-9.1/bin/psql" ¥
        pgdata="/var/lib/pgsql/9.1/data/" ¥
        rep_mode="sync" ¥
        node_list="pm01 pm02" ¥
        restore_command="cp /var/lib/pgsql/9.1/data/pg_archive/%f %p" ¥
        primary_conninfo_opt="keepalives_idle=60 ¥
                              keepalives_interval=5 keepalives_count=5" ¥
        master_ip="192.168.104.110" ¥
        stop_escalate="0" ¥
    op start   timeout="30s" interval="0s"  on-fail="restart" ¥
    op stop    timeout="30s" interval="0s"  on-fail="block" ¥
    op monitor timeout="30s" interval="11s"  on-fail="restart" ¥
    op monitor timeout="30s" interval="10s"  on-fail="restart" role="Master" ¥
    op promote timeout="30s" interval="0s"  on-fail="restart" ¥
    op demote  timeout="30s" interval="0s"  on-fail="block" ¥
    op notify  timeout="60s" interval="0s"
```

use sync replication

all hostname that get into repliation

restore_command of recovery.conf

primary_conninfo of recovery.conf (can't set application_name)

vip-repのIP

use immediate shutdown to speed up failover

start-up setting of vip-slave

```
location rsc_location-2 vip-slave \
    rule 200: pgsql-status eq HS:sync \
    rule 100: pgsql-status eq PRI \
    rule -inf: not_defined pgsql-status \
    rule -inf: pgsql-status ne HS:sync and pgsql-status ne PRI
```

if pgsql-status = HS:sync, start vip-slave

if pgsql-status = PRI, start vip-lave

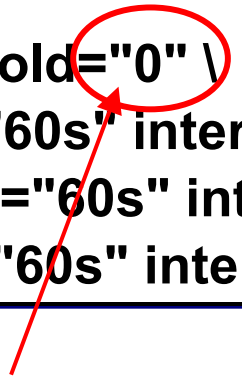
Pacemaker starts vip-slave on Slave (HS:sync) preferentially

You need to set resource-stickiness not to fasten vip-slave.

```
primitive vip-slave ocf:heartbeat:IPaddr2 \
    params \
        ip="192.168.103.111" \
        nic="eth3" \
        cidr_netmask="24" \
    meta \
        resource-stickiness="1" \
```

restart setting of vip-rep

```
primitive vip-rep ocf:heartbeat:IPaddr2 \
    params \
        ip="192.168.104.110" \
        nic="eth4" \
        cidr_netmask="24" \
    meta \
        migration-threshold="0" \
    op start   timeout="60s" interval="0s"  on-fail="stop" \
    op monitor timeout="60s" interval="10s" on-fail="restart" \
    op stop    timeout="60s" interval="0s"  on-fail="block"
```



If vip-rep is broken, Pacemaker attempts to restart it.

start and stop order setting between Master and master-group (vip-master,vip-rep)

start virtual IPs afiter promote

```
order rsc_order-2 inf: msPostgresql:promote  master-group:start   symmetrical=false
order rsc_order-3 0:   msPostgresql:demote   master-group:stop    symmetrical=false
```

This incllude virtual IPs

stop virtual IPs after demote
not to cut a replication connection.

display example of crm_mon command

```
Online: [ pm01 pm02 ]

vip-slave   (ocf::heartbeat:IPaddr2):       Started pm02
 Resource Group: master-group
     vip-master (ocf::heartbeat:IPaddr2):   Started pm01
     vip-rep    (ocf::heartbeat:IPaddr2):   Started pm01
 Master/Slave Set: msPostgresql
     Masters: [ pm01 ]
     Slaves: [ pm02 ]
 Clone Set: clnPingd
     Started: [ pm01 pm02 ]

Node Attributes:
* Node pm01:
    + default_ping_set                  : 100
    + master-pgsql:0                    : 1000
    + pgsql-data-status                 : LATEST
    + pgsql-status                      : PRI
    + pgsql-master-baseline             : 00000000150000B0
    + pm02-eth1                         : up
    + pm02-eth2                         : up
* Node pm02:
    + default_ping_set                  : 100
    + master-pgsql:1                    : 100
    + pgsql-data-status                 : STREAMING|SYNC
    + pgsql-status                      : HS:sync
    + pm01-eth1                         : up
    + pm01-eth2                         : up

Migration summary:
* Node pm01:
* Node pm02:
```

virtual IPs

state of PostgreSQL

pgsql-status, pgsql-data-status on pm01

RA shows xlog location when promote is called

pgsql-status, pgsql-data-status on pm02

Setting of PostgreSQL

postgresql.conf (Only an important point)

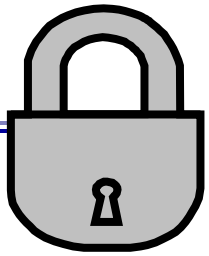
- listen_address = *
  - PostgreSQL can't listen particular IP because Slave dosen't have vip-master
- delete synchronous_standby_names
  - RA uses this parameter to switch between sync and async
    - RA insert "include /var/lib/pgsql/tmp/rep_mode.conf" into postgresql.conf
- restart_after_crash = off
  - Pacemaker manages state of PostgreSQL
- replication_timeout = 20s (about?)
  - If replication LAN or Slave are broken, Master's transaction is suspended until PostgreSQL cuts connection as this timeout
- hot_standby = on
  - to monitor Slave
- max_standby_streaming_delay = -1
  max_standby_archive_delay = -1
  - prevent cancel of monitor on Slave

postgresql.conf (その他)

- wal_level = hot_standby
- wal_keep_segments = 64 (about?)
- wal_receiver_status_interval = 5s (about?)
  - this parameter < replication_timeout
- hot_standby_feedback = on
- archive_mode = on

lock-file

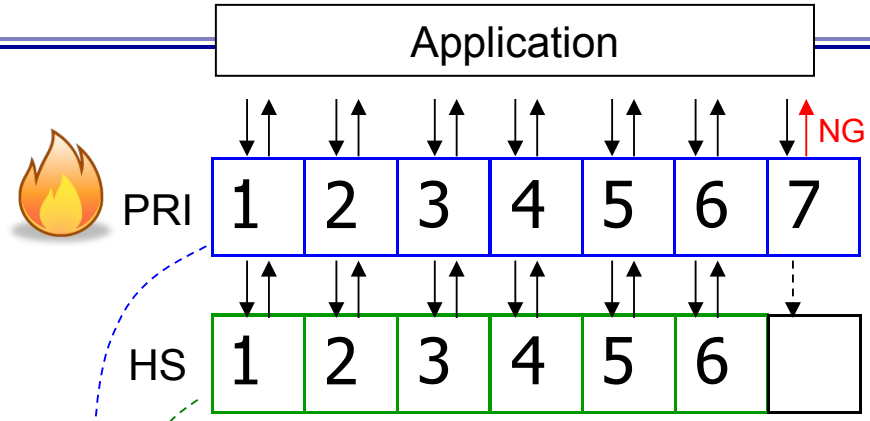
lock-file



- This file is created when promote is called
  - To avoid data inconsistency
    - Pacemaker can't start PostgreSQL if this file exists
    - path(default) : /var/lib/pgsql/tmp/PGSQL.lock
  - The file is removed when demote is called and HS dosen't exist.

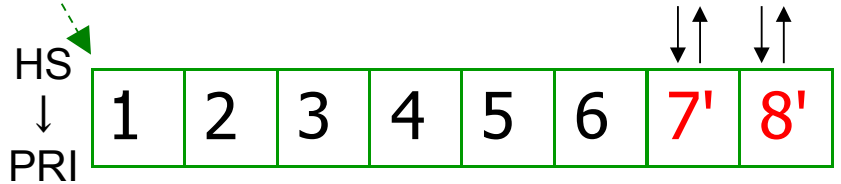
flaw example

- write a data at location 7 on PRI
- PRI is broken during replication

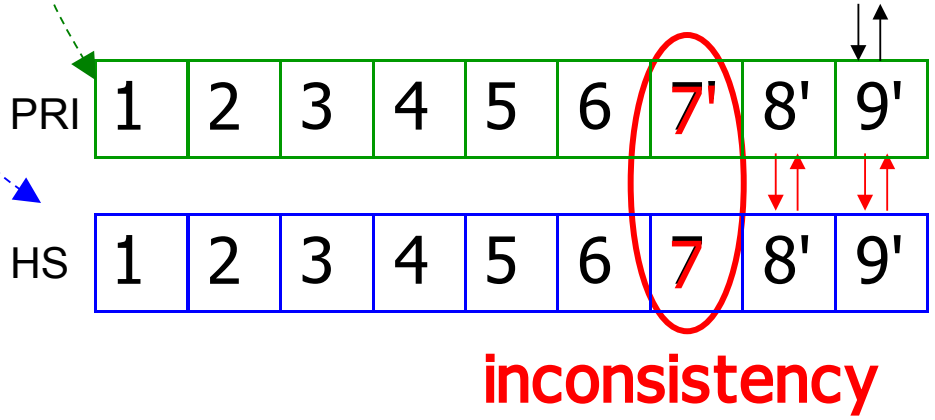


fail-over is occurred

- Application writes a data at location 7', 8' .... on new PRI

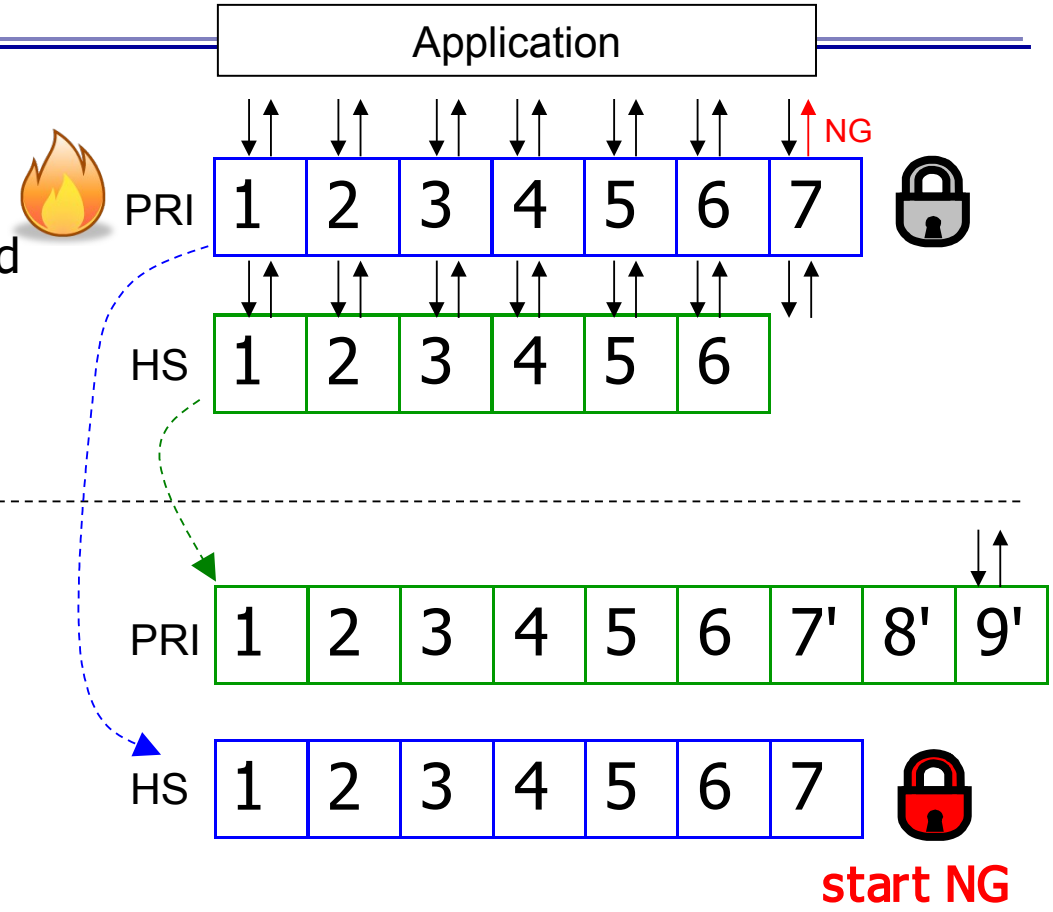


- recover old PRI as HS
  - data is replicated from 8'
- data inconsistency is occured



After

- create lock file during promote
- write a data at location 7
- PRI is broken and fail-over is occured



- recover old PRI as HS
  → start ERROR

You need to copy data from new PRI to old PRI
and remove the file

Conclusion

- 3 major functions
  - failover of Master
  - switch setting between sync and async
  - manage old data
- Setting of Pacemaker
  - need a virtual IP (vip-rep) for replication
  - if necessary you can use vip-slave to handle read-only query
- Causion
  - can't connect to PRI if TimelineID is defferent
  - data inconsistency may be occured after fail-over
    - need to copy data from new PRI to old PRI and remove lock file

reference : my environment to develop

- Pacemaker 1.0.12
  - 1.0.11's Master/Slave function has a bug
- PostgreSQL 9.1
  - non-PostgreSQL9.0-compliant