More reliability and support for PostgreSQL 10: Introducing Pgpool-II 3.7

PGConf.ASIA 2017

SRA OSS, Inc Japan
Tatsuo Ishii

Who am I?

- Working on OSS activities and businesses
  - OSS activities
    - PostgreSQL committer
    - Pgpool-II developer
  - OSS Businesses
    - Support for OSS including PostgreSQL
    - Developping and selling PowerGres
    - Training for PostgreSQL

SRA OSS, INC.

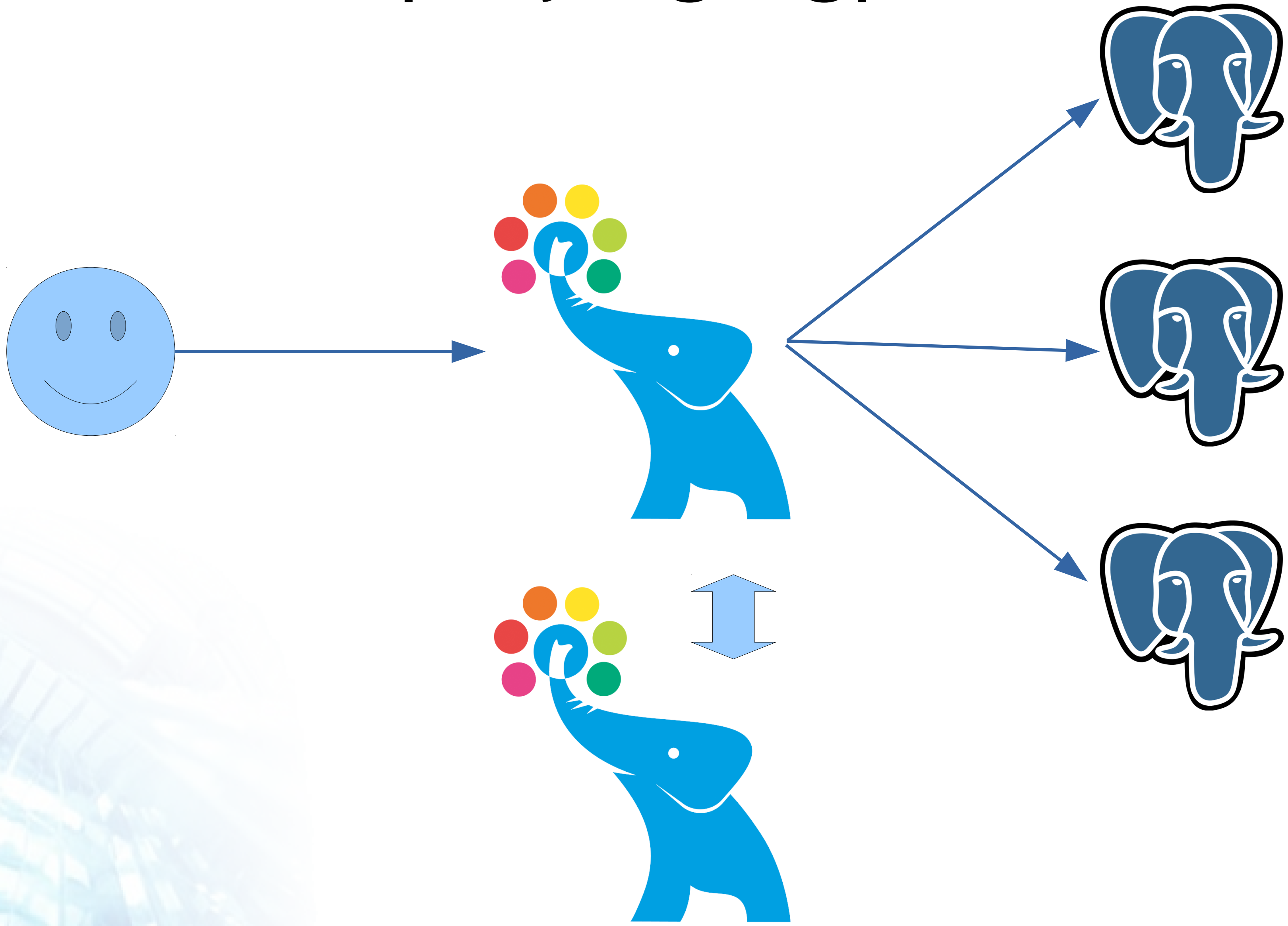
Today's agenda

- What is Pgpool-II?
- About Pgpool-II 3.7
- Feature plans

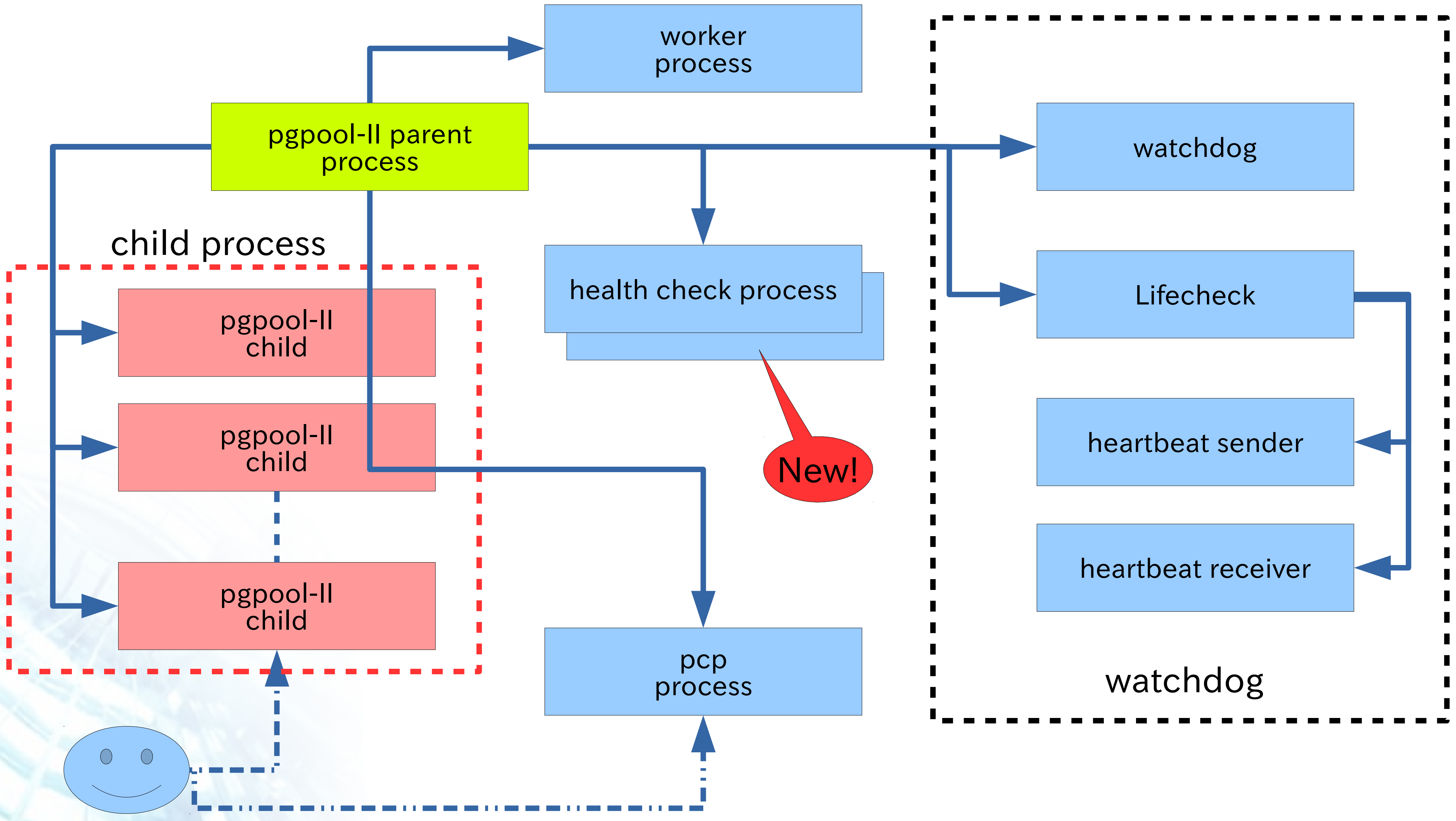
What is Pgpool-II?

- A cluster management tool dedicated to PostgreSQL
  - Support wide range of PostgreSQL versions: from 7.4 to 10
  - Support for PostgreSQL derived products including Amazon Aurora and RedShift
- Employing the same license as PostgreSQL
- Managing multiple PostgreSQL instances as a “cluster”
  - Detaching failed server automatically (automatic fail over)
  - New server or repaired server can be reattached to the cluster after re-syncing it (“on line recovery, fail back)
  - Dispatching queries by recognizing whether they are writing ones or reading ones automatically
  - Dispatching read queries to multiple servers (load balancing)
- Adding features missing in PostgreSQL
  - Connection pooling
  - Query cache

Deploying Pgpool-II

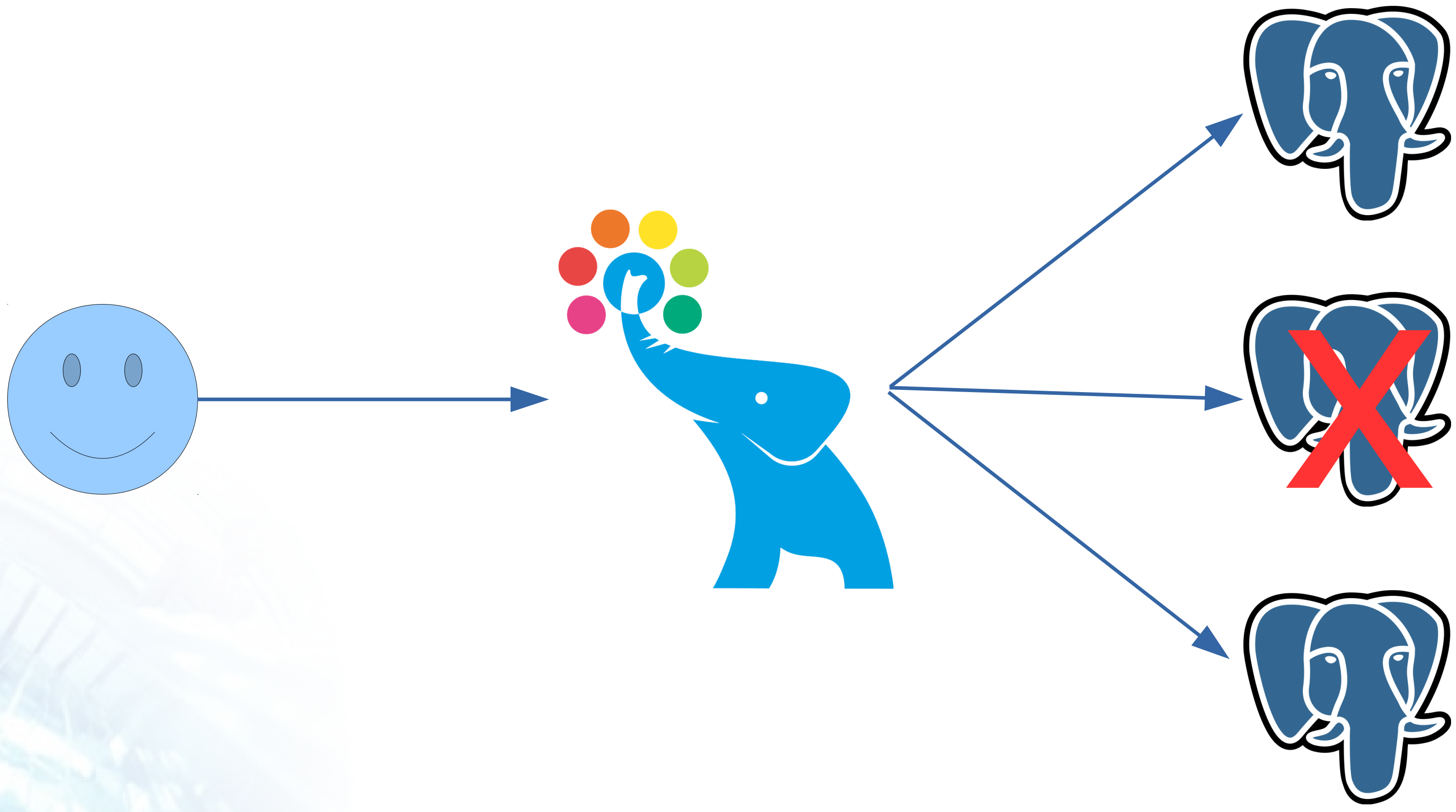


Process structure of Pgpool-II



Enhancements in Pgpool-II 3.7 : Health check

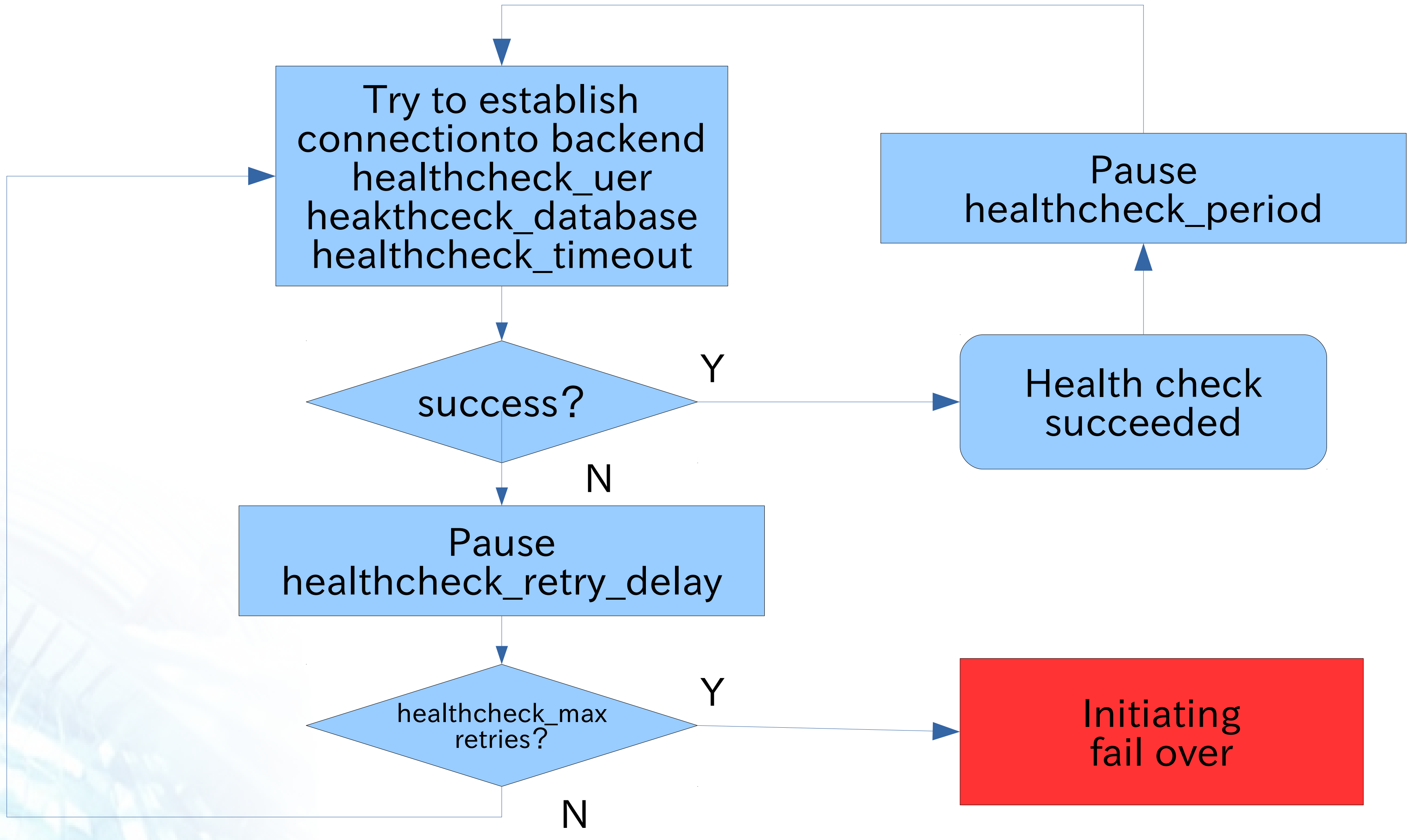
Automatic fail over : Dealing with PostgreSQL failure



Fail over

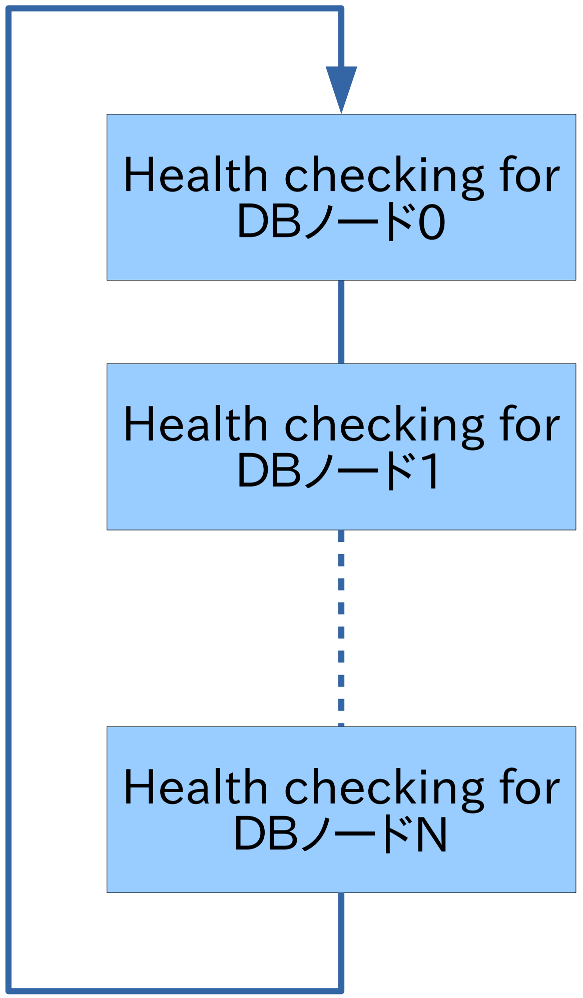
- A fail over is triggered by:
  - Health checking
    - Trying to connect to backend. Ok if succeeded
    - Per node health checking process (New in Pgpool-II 3.7)
  - Notice of backend shutdown
    - Checking backend error codes
  - Communication errors with backend
    - Detecting Socket communication failure
    - Does not care if the error is coming communication or actual backend failure
  - Errors detected while establishing connection to backend
    - Socket communication error detection
    - Does not care if the error is coming communication or actual backend failure
  - Forcing fail over using PCP commands

Health checking and parameters


Try to establish connectionto backend healthcheck_uer heakthceck_database healthcheck_timeout
success?
Y
Health check succeeded
Pause healthcheck_period
N
Pause healthcheck_retry_delay
healthcheck_max retries?
Y
Initiating fail over
N

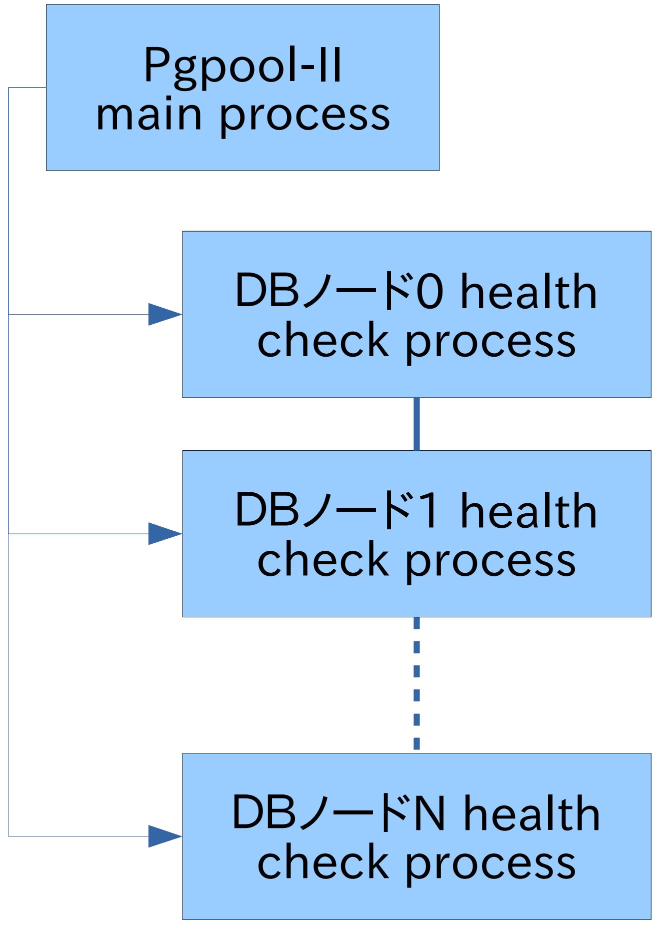
Issue with health checking

- Possible delay of checking due to sequential processing of health checking
- A cycle of health checking could be longer if a failure happens with one of DB nodes
- Pgpool-II main process may be affected by the health checking because it is executed in the same process


Health checking for DBノード0
Health checking for DBノード1
Health checking for DBノードN

Enhanced health checking in Pgpool-II 3.7

- Forking dedicated health check process by Pgpool-II main process
- Pgpool-II main process is not affected by healh checking thus more stable
- No delay in error detection
- Separate health check parameters for each DB node


Pgpool-II main process
DBノード0 health check process
DBノード1 health check process
DBノードN health check process

How to set per DB node health check parameters

- In addition to the previous way, it is possible to add DB node suffix to the parameters. For example, “healthcheck_period0”
- Also the previous way which does not use the node suffix. If there's no per node parameter is specified, then the no node parameter value is used
- Per node health check parameters can be viewed by using “pgpool show health_check”

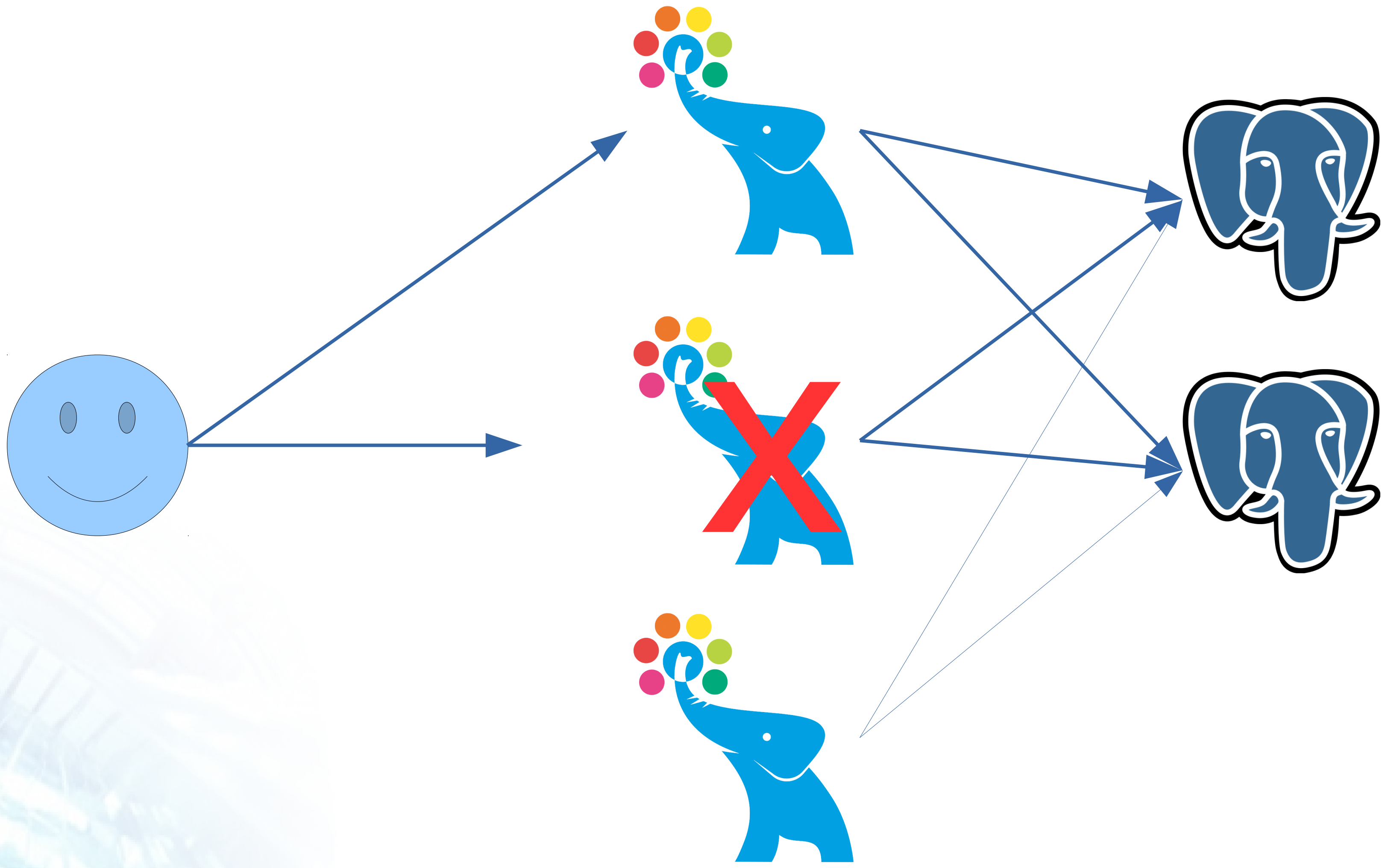
```
test=# pgpool show health_check;
            item             |  value   |                                               description
-----------------------------+----------+------------------------------------------------------------------------------------------------------
 health_check_period         | 10       | Time interval in seconds between the health checks.
 health_check_timeout        | 20       | Backend node health check timeout value in seconds.
 health_check_user           | t-ishii  | User name for PostgreSQL backend health check.
 health_check_password       | *****    | Password for PostgreSQL backend health check database user.
 health_check_database       |          | The database name to be used to perform PostgreSQL backend health check.
 health_check_max_retries    | 5        | The maximum number of times to retry a failed health check before giving up and initiating failover.
 health_check_retry_delay    | 1        | The amount of time in seconds to wait between failed health check retries.
 connect_timeout             | 1000     | Timeout in milliseconds before giving up connecting to backend.
 health_check_period0        | 0        | Time interval in seconds between the health checks.
 health_check_timeout0       | 20       | Backend node health check timeout value in seconds.
 health_check_user0          | nobody   | User name for PostgreSQL backend health check.
 health_check_password0      | *****    | Password for PostgreSQL backend health check database user.
 health_check_database0      |          | The database name to be used to perform PostgreSQL backend health check.
 health_check_max_retries0   | 0        | The maximum number of times to retry a failed health check before giving up and initiating failover.
 health_check_retry_delay0   | 1        | The amount of time in seconds to wait between failed health check retries.
 connect_timeout0            | 10000    | Timeout in milliseconds before giving up connecting to backend.
 health_check_period1        | 10       | Time interval in seconds between the health checks.
 health_check_timeout1       | 20       | Backend node health check timeout value in seconds.
 health_check_user1          | t-ishii  | User name for PostgreSQL backend health check.
 health_check_password1      | *****    | Password for PostgreSQL backend health check database user.
 health_check_database1      |          | The database name to be used to perform PostgreSQL backend health check.
 health_check_max_retries1   | 5        | The maximum number of times to retry a failed health check before giving up and initiating failover.
 health_check_retry_delay1   | 1        | The amount of time in seconds to wait between failed health check retries.
 connect_timeout1            | 1000     | Timeout in milliseconds before giving up connecting to backend.
(24 rows)
```

Enhancements in Pgpool-II 3.7 : Watchdog

What is Watchdog?

- Watchdog detects Pgpool-II failure
- Avoding Pgpool-II being single point of failure using backup Pgpool-II
- If a Pgpool-II fails, other Pgpool-II inherits the VIP
- If odd number of Pgpool-II (and more than 1) are deployed, it is possible to avoid the split brain syndrome caused by a network splitting

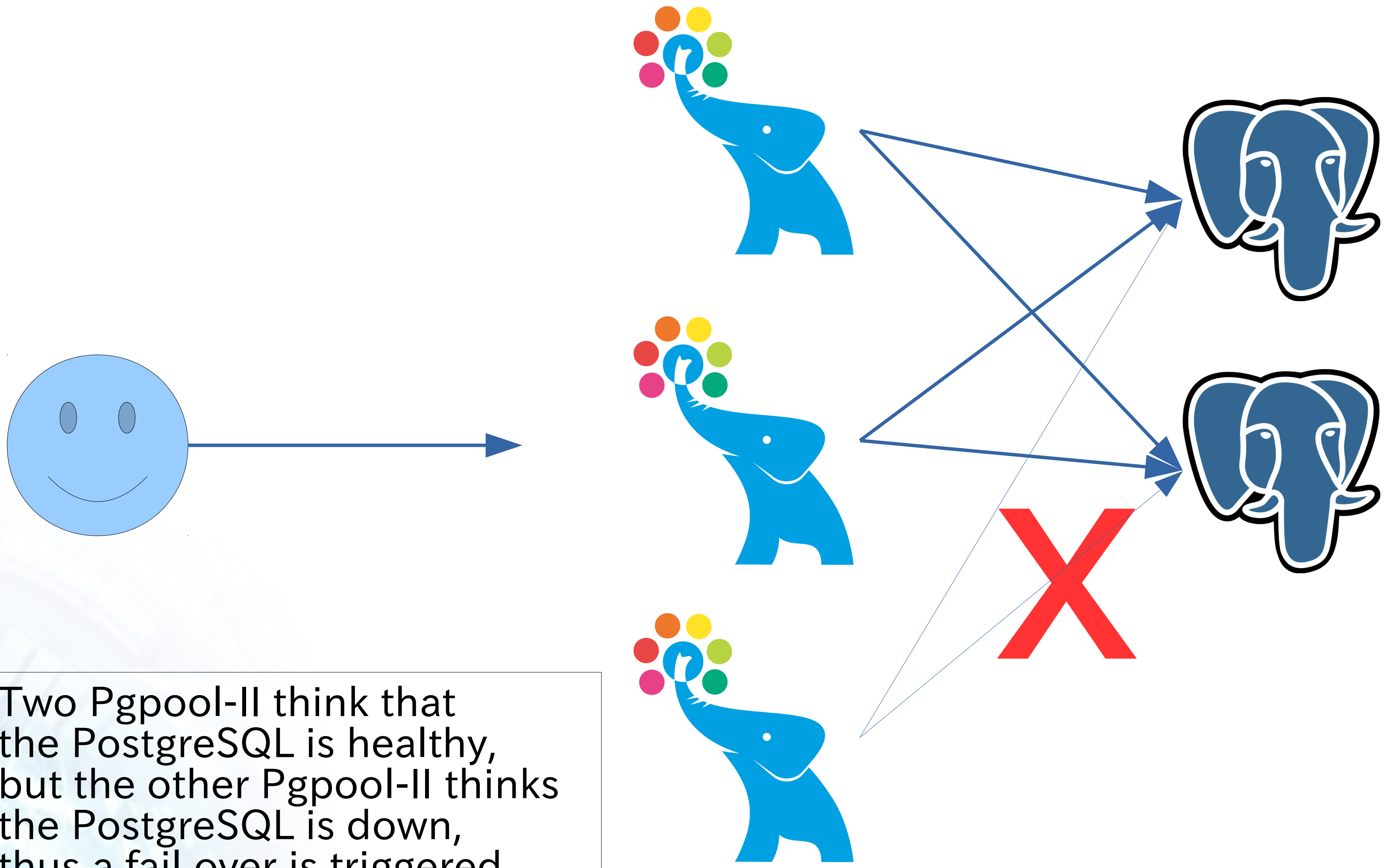
Switching VIP with Watchdog



Issue with backend failure detection

- Because Pgpool-II checks the backend failure by attempting to connect to the backend, it is not possible to distinguish whether the backend is really dead or it' s just a network problem
- This is not an issue if there's only one Pgpool-II
  - PostgreSQL can not be used anyway regardless the failure
- If there are multiple Pgpool-II instances, and a part of the network fails, then a fail over would be triggered (false positive error)

False positive error detection of PostgreSQL



Two Pgpool-II think that
the PostgreSQL is healthy,
but the other Pgpool-II thinks
the PostgreSQL is down,
thus a fail over is triggered.

Quorum consensus fail over

- If quorum exists, each watchdog is requested a vote if a backend failure is reported from a watchdog
- Master watchdog collects the votes and triggers a fail over only if the number of votes exceeds the half of the number of nodes. This could avoid the false positive error detection of PostgreSQL by a partial network failure
- The rejected watchdog marks the backend in “quarantine” state, and detaches the backend locally
  - Note that even if the primary PostgreSQL is in quarantine state, no promotion of a standby will be performed
  - Executing pcp_attach_node on the quarantine backend will be ignored
- It is still possible to force to trigger a fail over manually using pcp_detatch_node which ignores the quarantine state

What does it mean “a quorum exists” anyway?

- If the total number of live watchdog cannot be majority, we can regard it as “there's no quorum”. Examples:
  - Number of watchdog nodes = 5 and live nodes is less than or equal to 2
  - Number of watchdog nodes = 3 and live node is less than or equal to 1
- We regard that a quorum exists if there's one or more live watchdog nodes, and the total number of watchdog node is even

New parameters in watchdog (1)

- failover_when_quorum_exists
  - If a quorum exists, trigger a fail over
  - If a quorum does not exist, then the backend is in the "quarantine" state and is detached locally
    - Note that even if the primary PostgreSQL is in quarantine state, no promotion of a standby will be performed
    - It is still possible to force to trigger a fail over manually using pcp_detatch_node which ignores the quarantine state
    - Executing pcp_attach_node on the quarantine backend will be ignored
  - The default is ON. If OFF, behaves as previous release (executes fail over even if a quorum does not exist)

New parameters in watchdog (2)

- failover_require_consensus
  - Only enabled when failover_when_quorum_existsis ON and a quorum exists
  - Trigger a fail over if majority of watchdog node vote
  - The default is ON

New parameters in watchdog (3)

- allow_multiple_failover_requests_from_node
  - If ON, a watchdog node can make multiple votes
    - For exaple, health check detected an error, but a vote did not win. In this case if more errors happen, health check detects it and another vote can be made. So there would be 2 votes
  - Multiple votes could trigger a fail over
    - It is useful if a permanent failure is not found by other watchdog nodes but it is desirable to trigger a fail over
  - Only enabled if failover_when_quorum_exists and failover_require_consensus is ON
  - The default is OFF

Dealing with Amazon Auroa

- What is Amazon Aurora?
  - Official name：“PostgreSQL Compatibility for Amazon” Aurora
  - High write performance（2-3 times faster than RDS), read replicas, automatic fail over
- Pgpool-II supports load balancing of read queries, query cache and connection pooling. Automatic fail over is suppressed because Aurora does it.
- Since in Auora the writer's IP is fixed, new flag “ALWAYS_MASTER” is added to not move primary node
- Detailed instructions how to set up is included in the docs

Dealing with logical replication

- What is logical replication?
  - Newly added in PostgreSQL 10
  - By sending logical transactions from “publisher” to “subscriber”, it is possible to replicate particular tables
- Pgpool-II's logical replication mode
  - It is assumed that all publishers are in the “master node”
  - It is assumed that all subscribers are in the “slave node”
  - Send write queries to the master node
  - Read queries will be load balanced
  - It is user's responsibilty to replicate tables which are load balanced by Pgpool-II
  - pgpool_setup can handle logical replication (-m l)

PostgreSQL 10 parser imported

- Now it is possible to parse correctly new syntaxes added to PostgreSQL 10 including “CREATE SUBSCRIPTION”

Enhancing pool_hba.conf

- In addition to IP address host names can be used
- Updated loading process enhances performance

Other enhancements and changes

- Add “role” column to pcp_node_info
  - Print “primary”, “standby” etc.
- Now the default value for master_slave_sub_mode is changed to “stream” from “slony”
- “Slony” mode is added to pgpool_setup
- New test suits using “pgproto” added
  - To test extended queries

Feature plans

- Planned in “3.8” or later (my personal wishes are included)
  - Enhance primary node detection
    - Pgpool-II automatically detects primary node but...
    - Once a primary is found, others are regarded as standby
      - This may regard standalone backends or standbys following other primary as standby
      - More strict test is necessary（checking primary node connected from those standbys for example)
  - Enhance load balancing
    - Allow to specify in finer granularity
  - Plug-ins?
- Comments, requests and suggestions are welcome!

Reference URL

- PostgreSQL
  - http://www.postgresql.org
- Pgpool-II
  - http://pgpool.net
- Pgproto
  - https://github.com/tatsuo-ishii/pgproto

Thank you!