

How to manage a herd  
of elephants:

# PostgreSQL clusters using streaming replication and pgpool-II

SRA OSS, Inc. Japan  
Tatsuo Ishii

# About me

- Came from Tokyo, Japan
- PostgreSQL committer
- Original author of pgpool-II
- Working for SRA OSS for 10 years



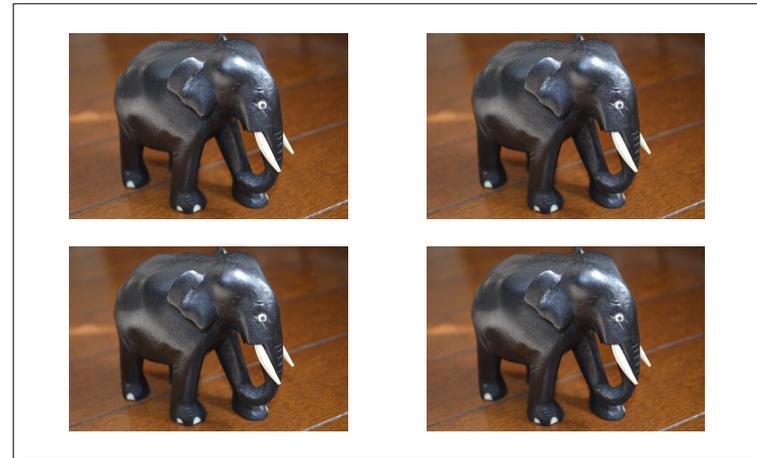
Mt. Fuji, sunset and Enoshima-island

# About SRA OSS, Inc. Japan

- One of the oldest PostgreSQL companies in the world
- Doing PostgreSQL and other OSS related business for 10 years
- Main business is support. Has over 600 support contracts
- Our contributions to PostgreSQL include:
  - Multi byte support
  - pgbench, pgstattuple
  - Recursive queries (WITH RECURSIVE)
  - 64 bit large objects

# A herd of PostgreSQL

- An elephant is powerful
- However, a herd of elephants is even more powerful!
- Problem is, how to manage the herd of elephants?



||

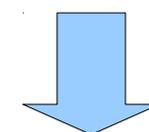
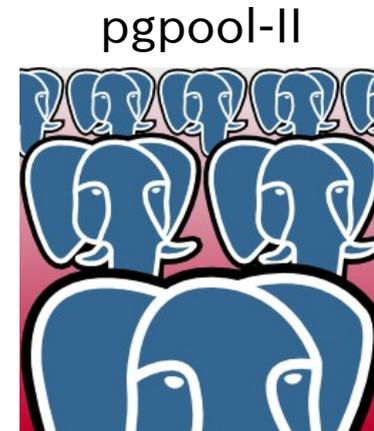


# Problems of the herd of elephants include:

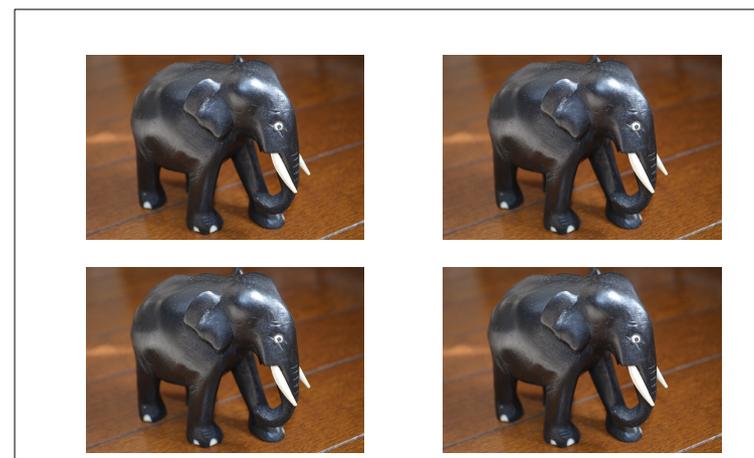
- It needs a leader (a primary server)
- If the leader retires, a new leader must take over its role (fail over, promote)
- Other elephants must follow the new leader
- If a non leader elephant cannot continue to work, it must retire and leave the herd (standby fail over)
- If a new elephant wants to join the herd, it should be accepted without disturbing the herd
- Elephants should help each other to perform a task in an efficient way (load balancing)
- There are tasks which can only be performed by the leader (write queries – needs query dispatching)

# Solving the problems by using pgpool-II

- By using pgpool-II, a streaming replication PostgreSQL cluster almost looks like a single PostgreSQL server
- User supplied “fail over script” could define which standby server should take over when the primary server goes down
- User supplied “follow master command” allow other standbys follow the new primary server
- If a standby server goes down, it is removed from the cluster definition and clients can continue to use the cluster
- pgpool-II examines each query. If the query is a read only one or not. If not, it is forwarded to one of servers (load balancing).
- If a query is a write query, it will be forwarded to the primary server

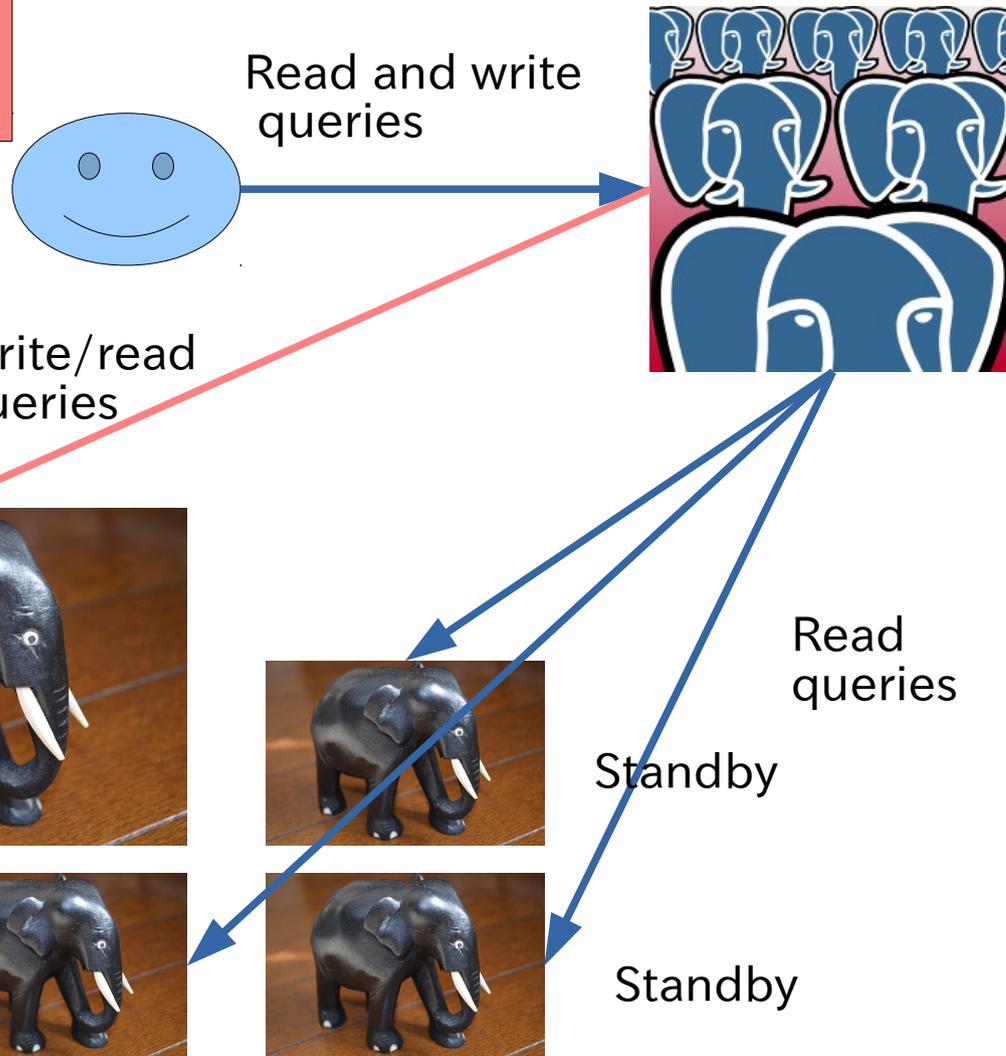


A herd of elephants looks like single big elephant



# Query dispatching/routing

Minimize modifications to existing single PostgreSQL applications



Primary



Standby



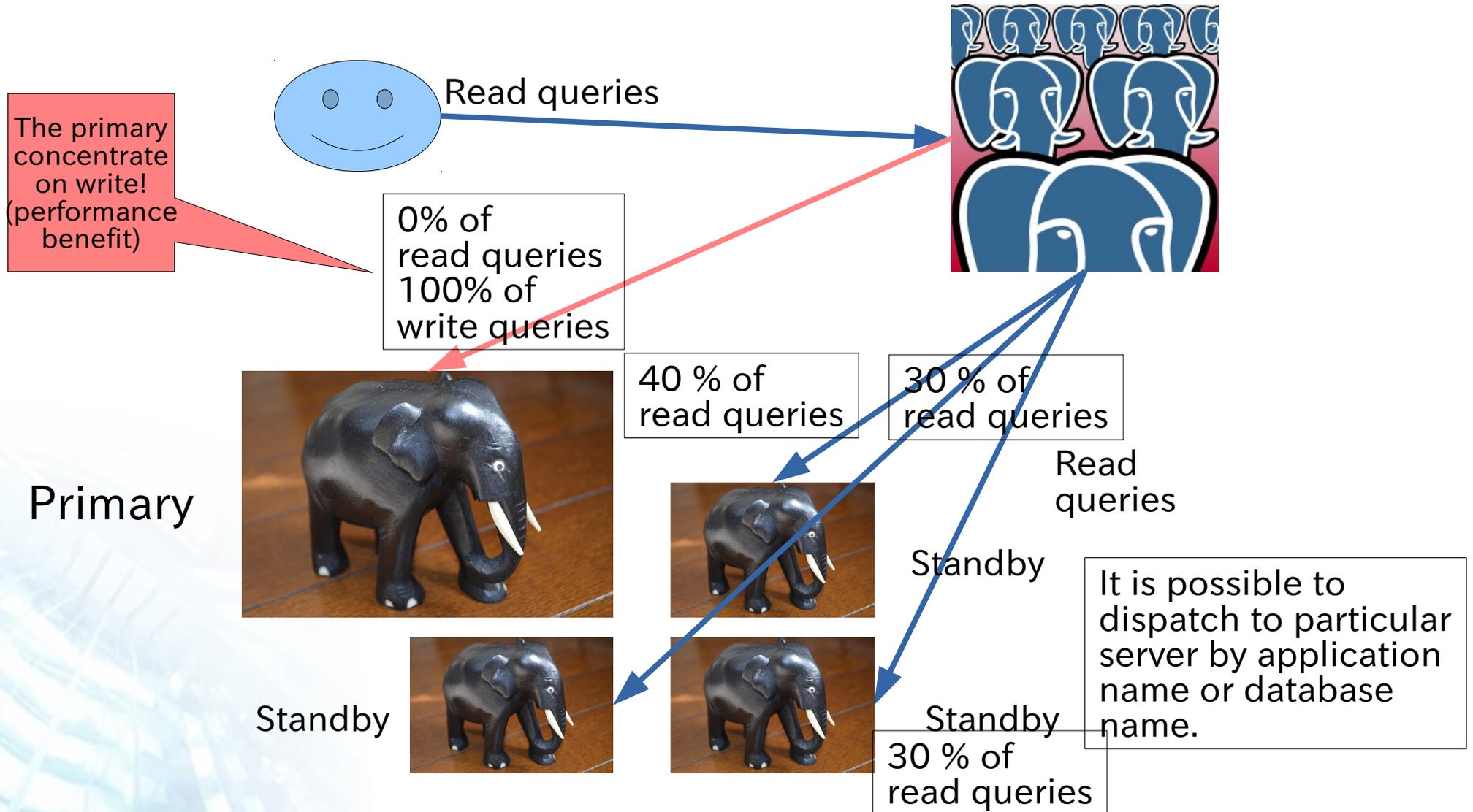
Standby



Standby



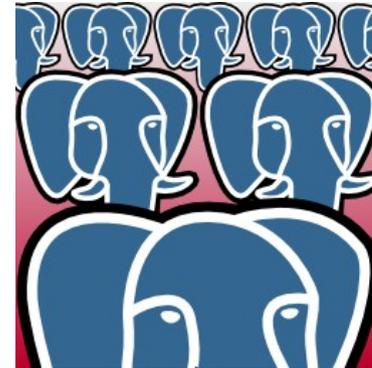
# Load balancing



# When the standby server goes down

If a standby goes down, it is simply took off from the cluster and user can continue to use the service using other servers

Existing session need to restart, however.



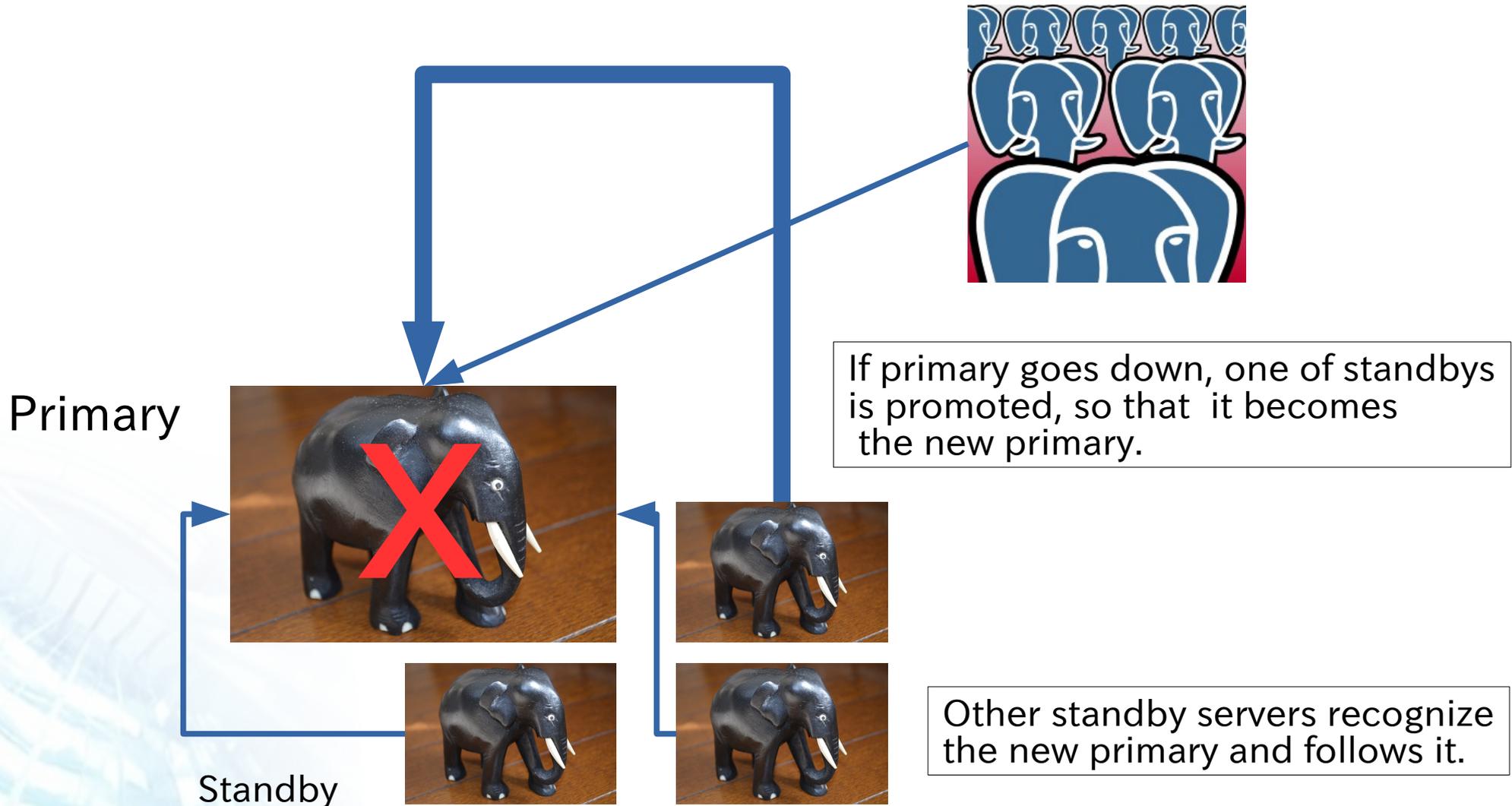
Primary



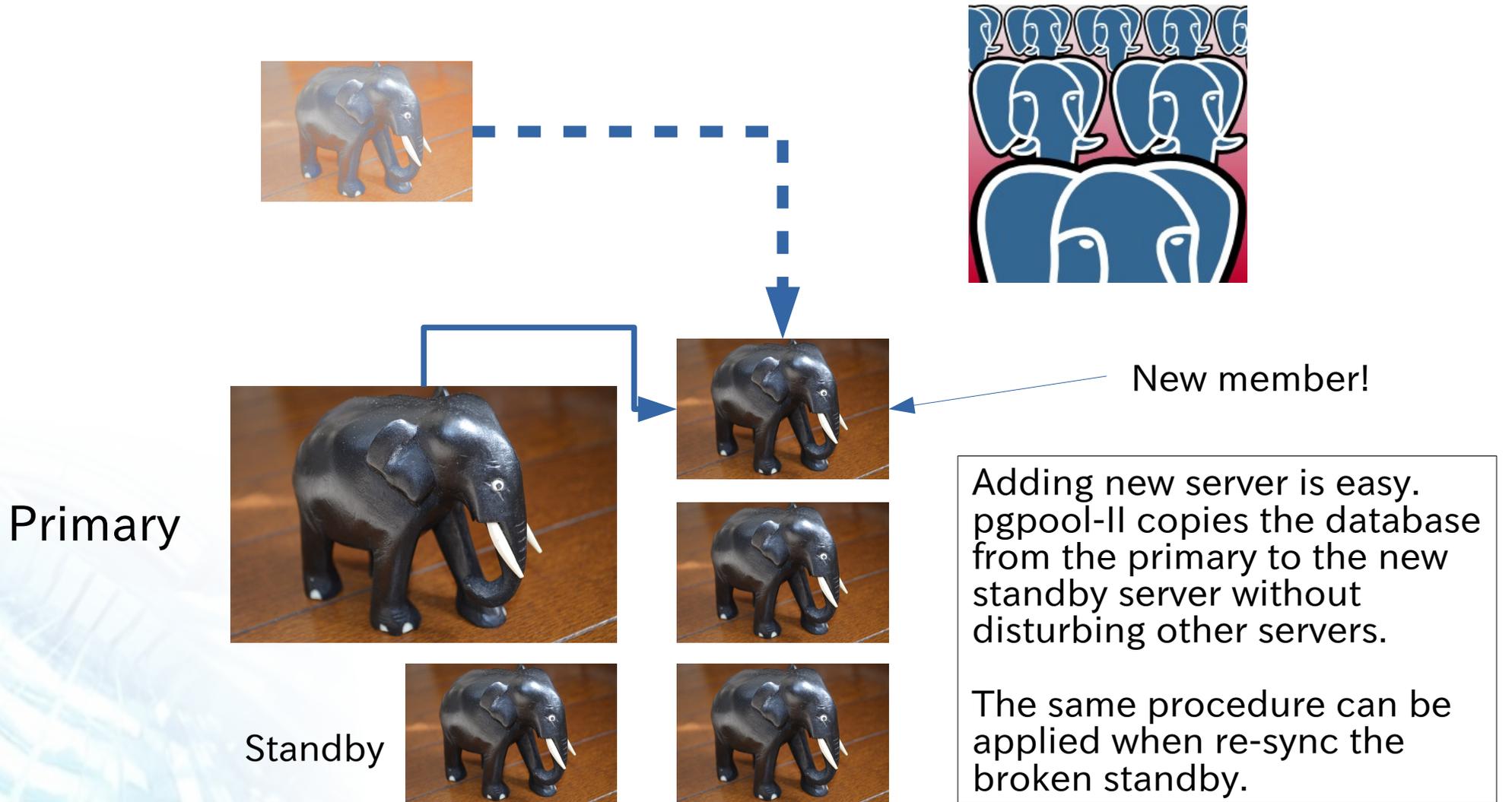
Standby



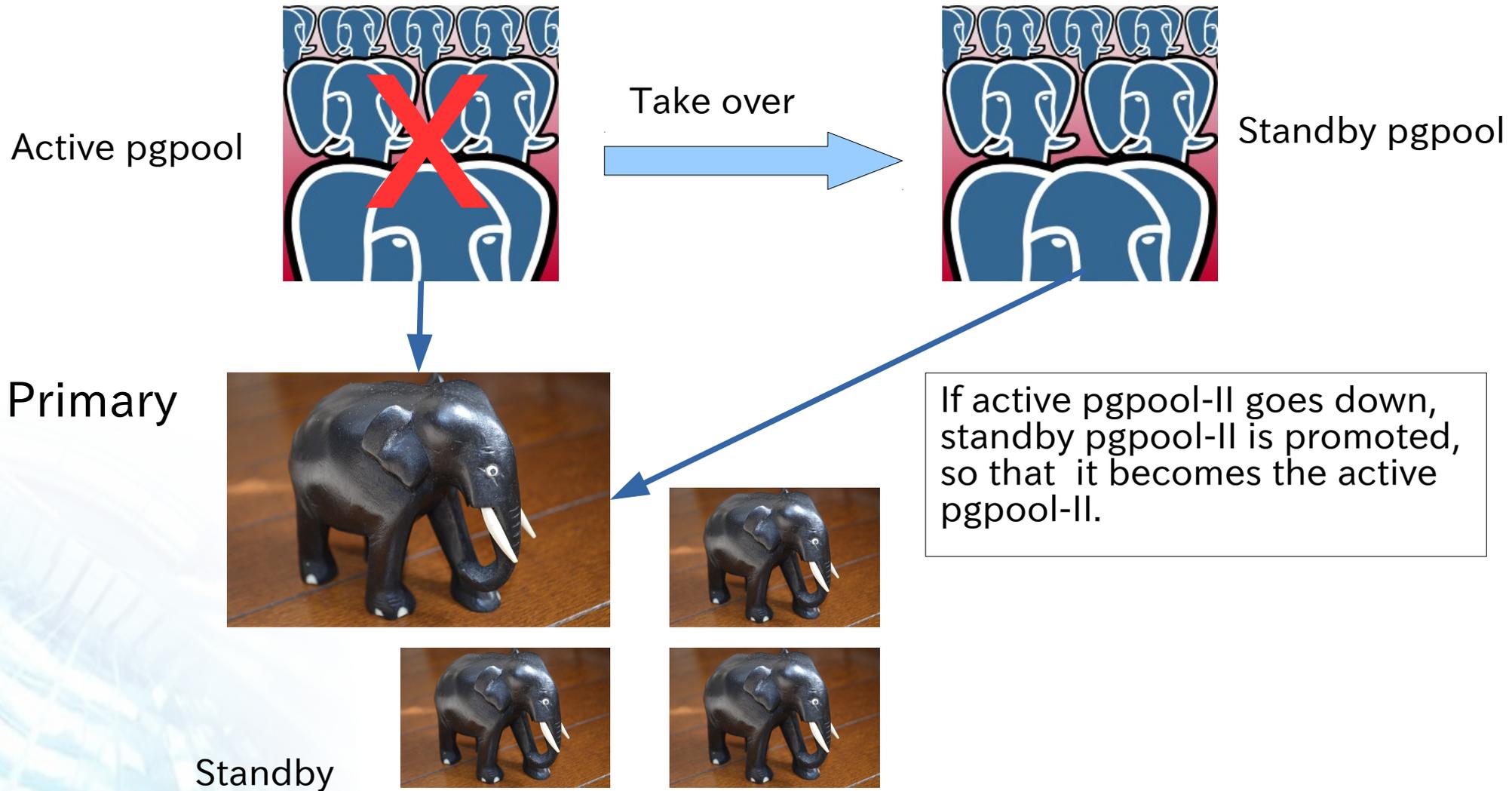
# When the primary server goes down



# Adding new server

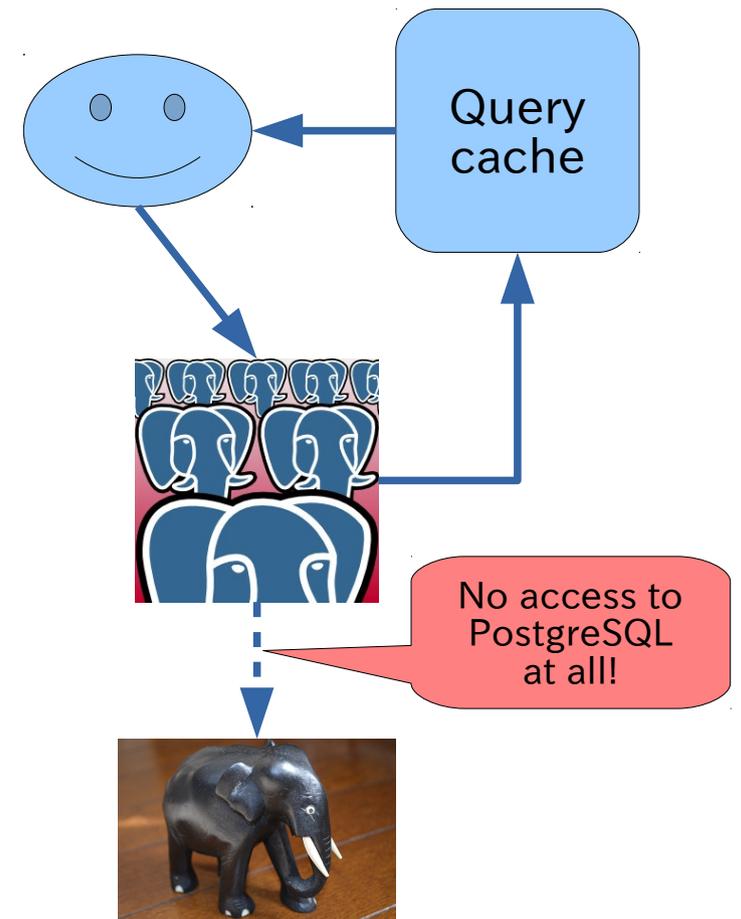


# Watchdog: built-in High Availability for pgpool-II



# In-memory query cache

- Pgpool-II can reuse query results using “query cache”
- Since the query cache is placed on memory, it's extremely fast
- No access to PostgreSQL
- The cache storage can be placed on either the shared memory, or memcached
- When a table is updated, all of related cache entries are invalidated
- Also time-based invalidation is possible



# Support policy

- Pgbpool-II hire “major releases” and “minor releases” method like PostgreSQL
  - 3.x.y – x: major version, y: minor version
    - example: “3.4.3”: “3.4” is the major version, “3” is the minor version
  - 1 major release per year
  - 3-4 minor releases per year
  - Each minor releases keep compatibility
  - Each major releases change API/usage (may not incompatible to earlier versions)
- Keep on providing back patches for five years after initial release
- Because there's one major release each year, we keep on supporting last 5 to 6 versions
- If you need longer support period, please contact to commercial support providers

# The latest version: pgpool-II 3.5 just has been released!

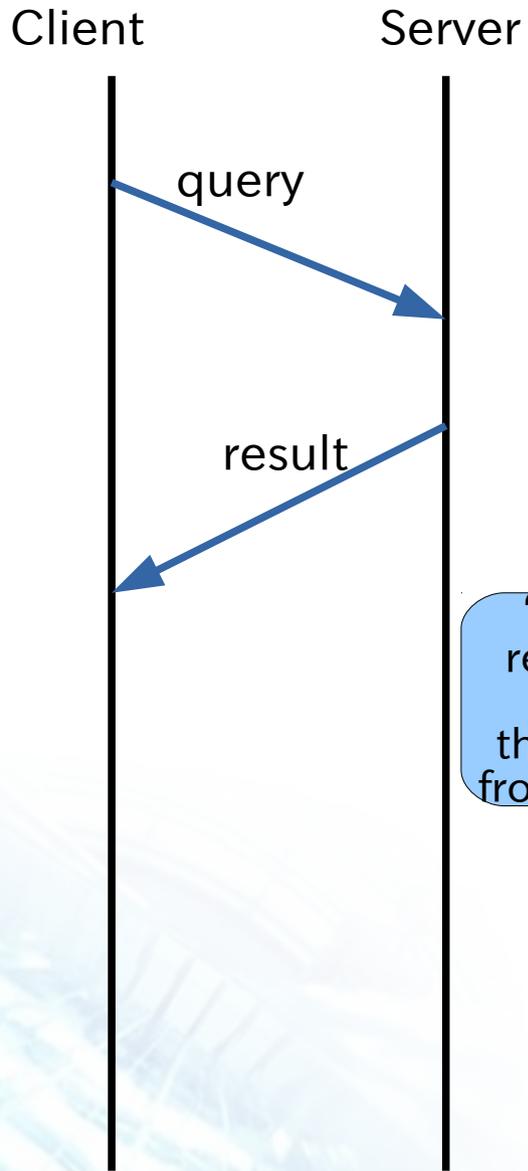
- Improved performance
- Improved watchdog
- Ready for PostgreSQL 9.5
- Enhanced pcp commands
- and more

# Improved performance

# Improving extended query performance

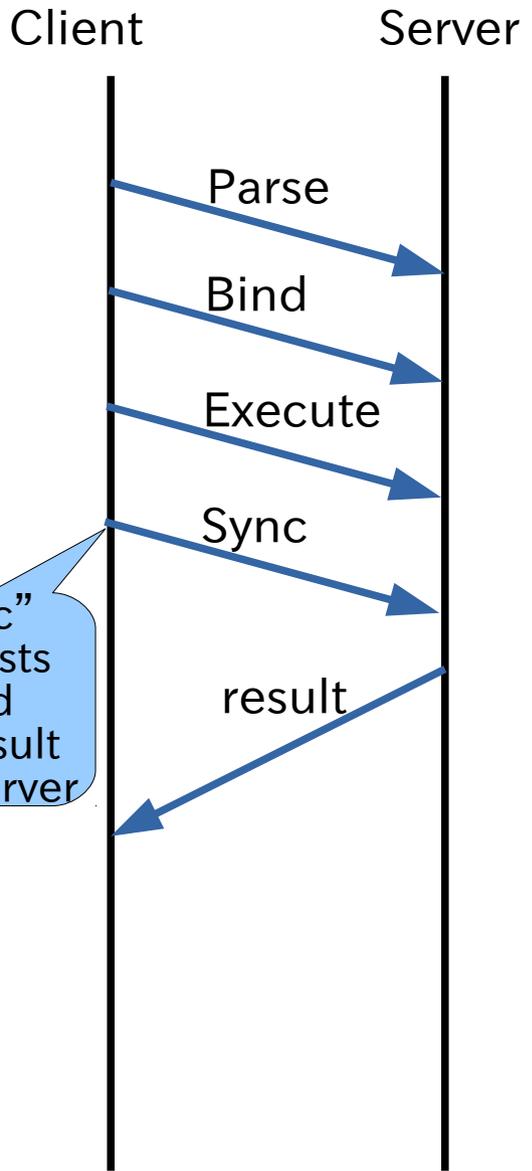
- Using extended protocol (typically used in Java) in pgpool-II is slow (as slow as half of simple protocol)
- Current implementation of pgpool-II for extended protocol is not so great: it requires additional flush messages, and this is the source of poor performance
- Ok, so how the extended protocol is handled?

Some details are omitted

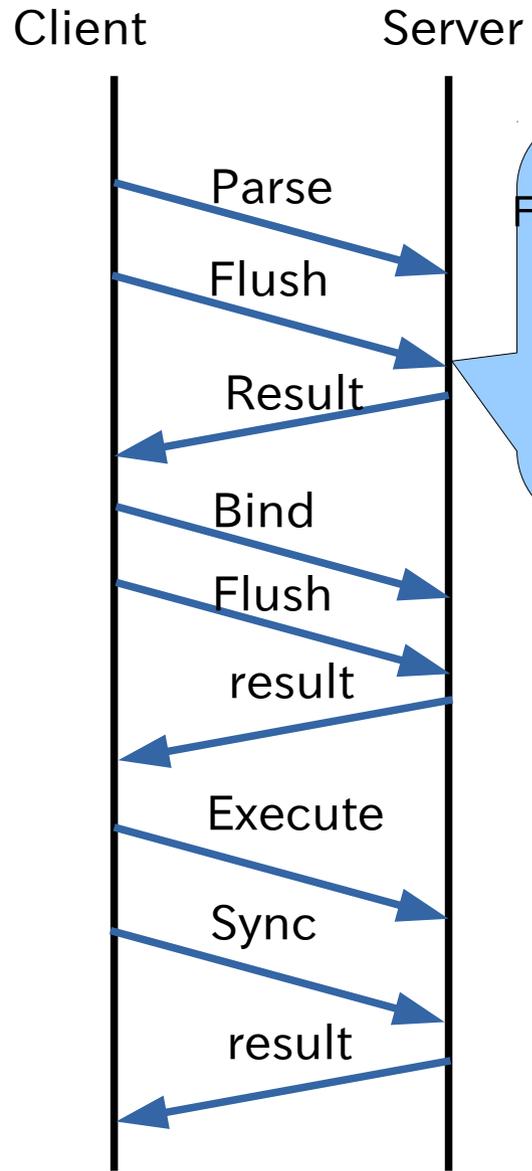


simple protocol

“Sync” requests send the result from server



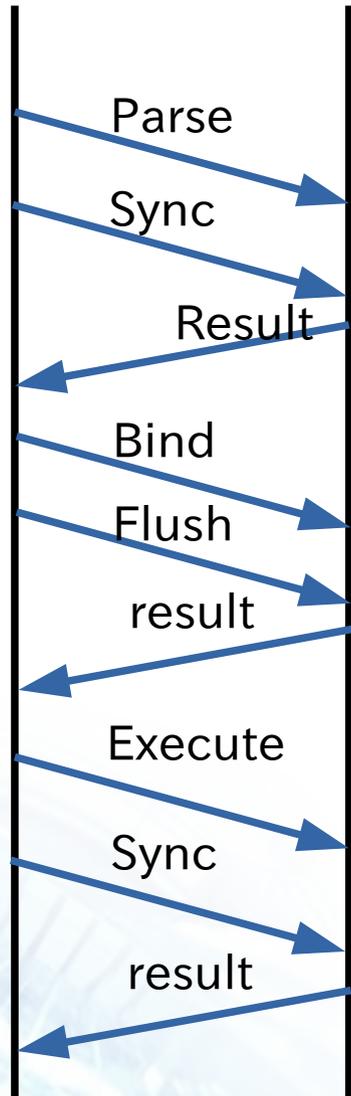
extended protocol



extended protocol with pgpool-II

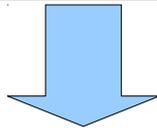
Flush is needed to handle multiple PostgreSQL more traffic

Client Server

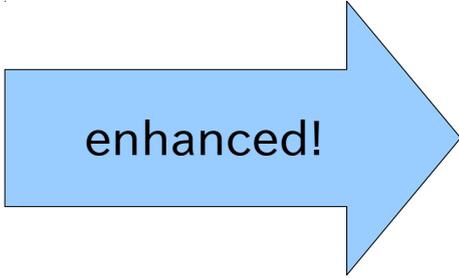


extended protocol with pgpool-II

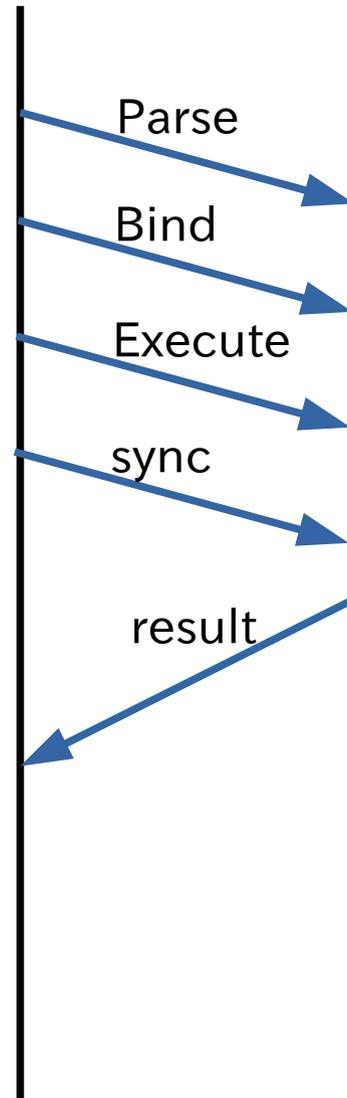
Too many Flush messages



In streaming replication we could omit some of Flush messages

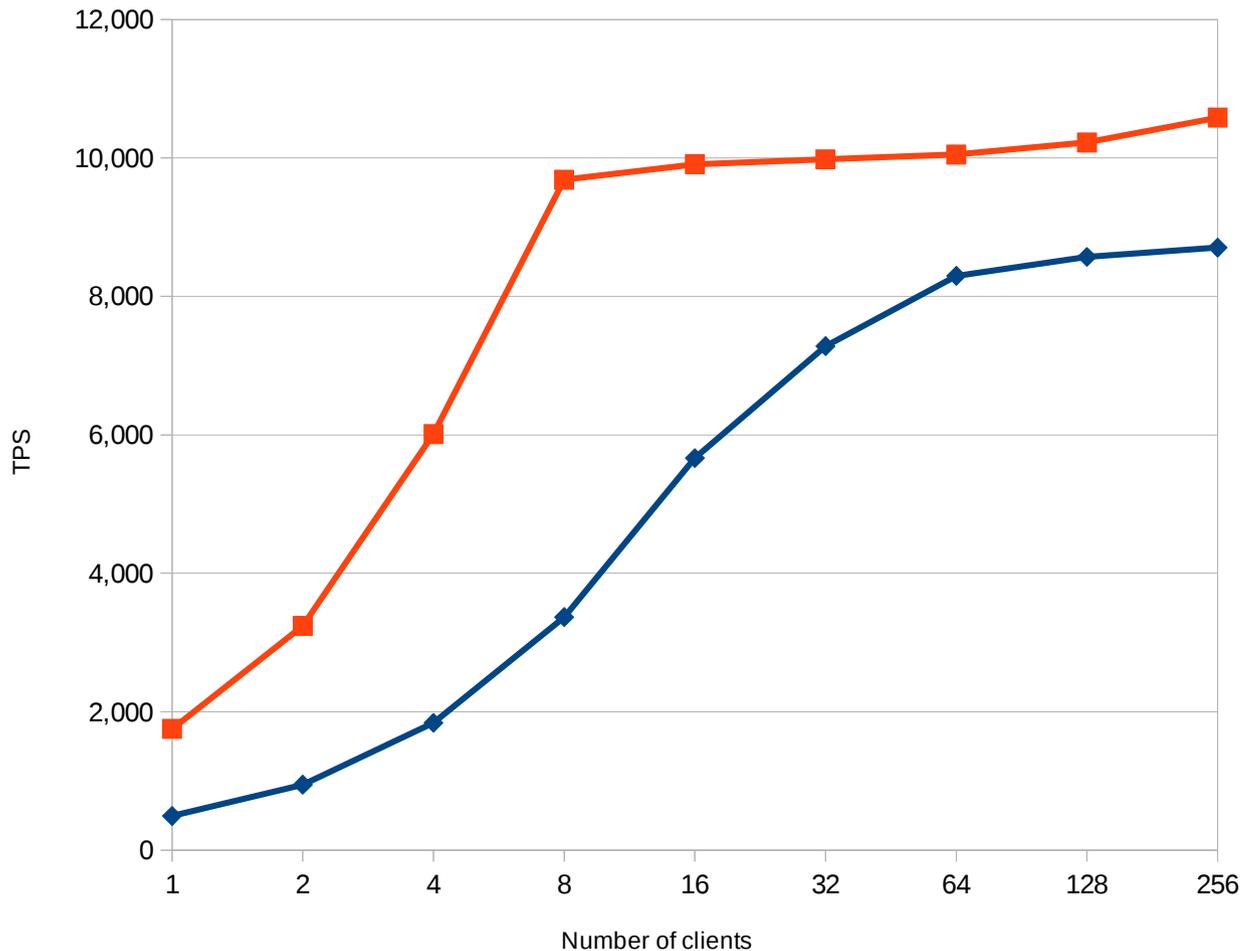


Client Server



extended protocol with pgpool-II in 3.5

# Benchmarking Extended protocol query performance



pgpool-II 3.5 is  
20% to 250%  
faster than  
pgpool-II 3.4!

—■— pgpool-II 3.5  
—◆— pgpool-II 3.4

AWS m4.large instance  
CentOS 6  
PostgreSQL 9.4 x2  
(streaming replication)  
pgbench -S

# Overcoming Thundering herd problem

- What is “the thundering herd problem”?

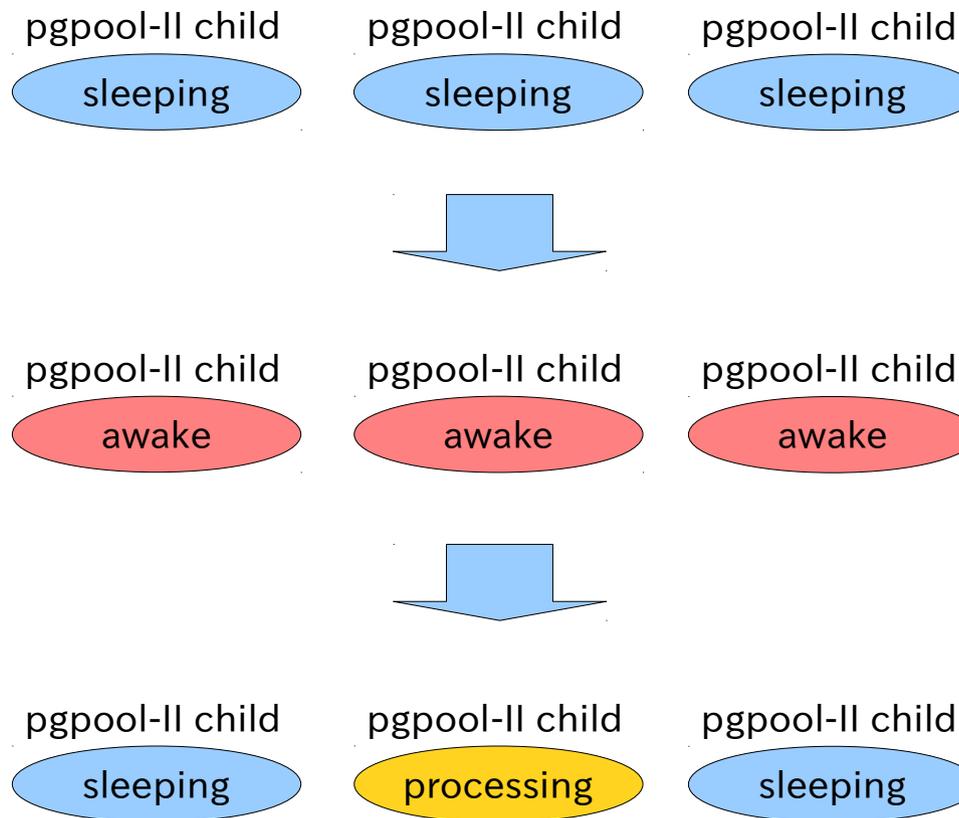
- “The thundering herd problem occurs when a large number of processes waiting for an event are awoken when that event occurs, but only one process is able to proceed at a time. After the processes wake up, they all demand the resource and a decision must be made as to which process can continue. After the decision is made, the remaining processes are put back to sleep, only to all wake up again to request access to the resource.”

From Wikipedia

- pgpool-II forks off many child process and they are waiting for connection requests from clients
- If a connection request arrives, all of the child process are awoken but only one of them is allowed to accept the request
- Other child process start sleeping again, to wait for next connection request
- This leads to an excessive context switching and results in poor performance

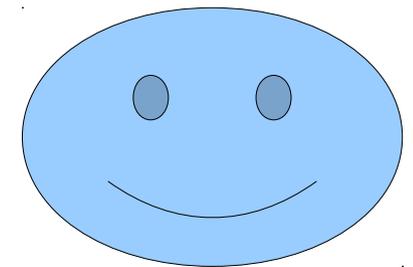
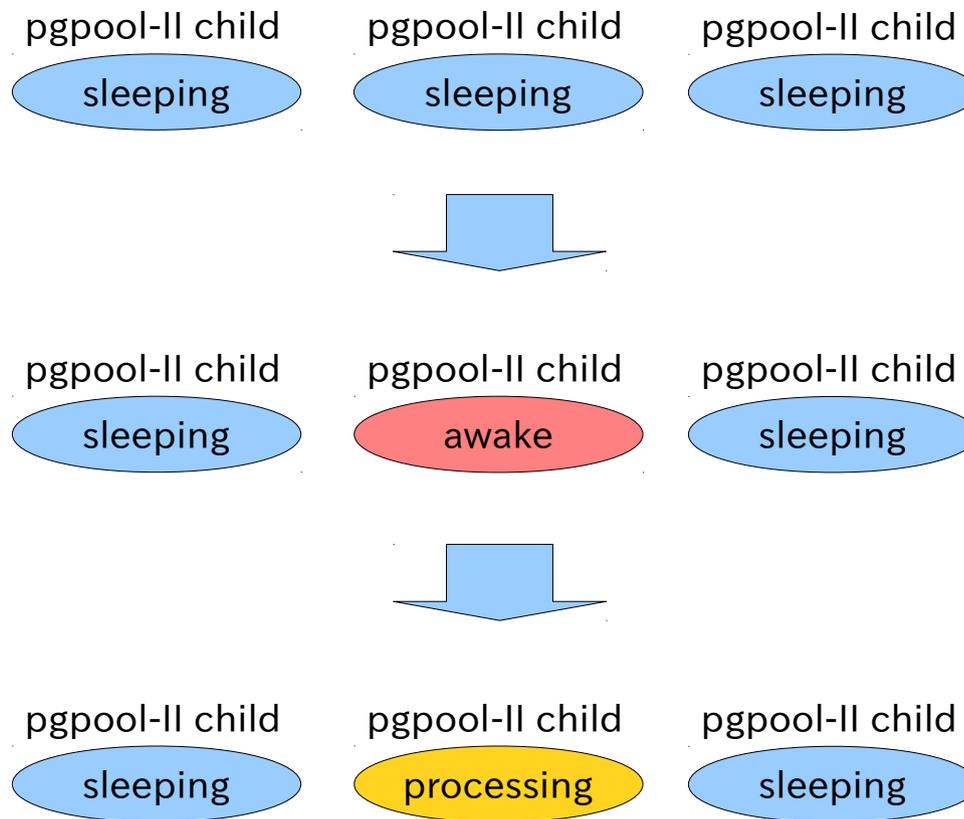
# Overcoming Thundering herd problem

pgpool-II 3.4



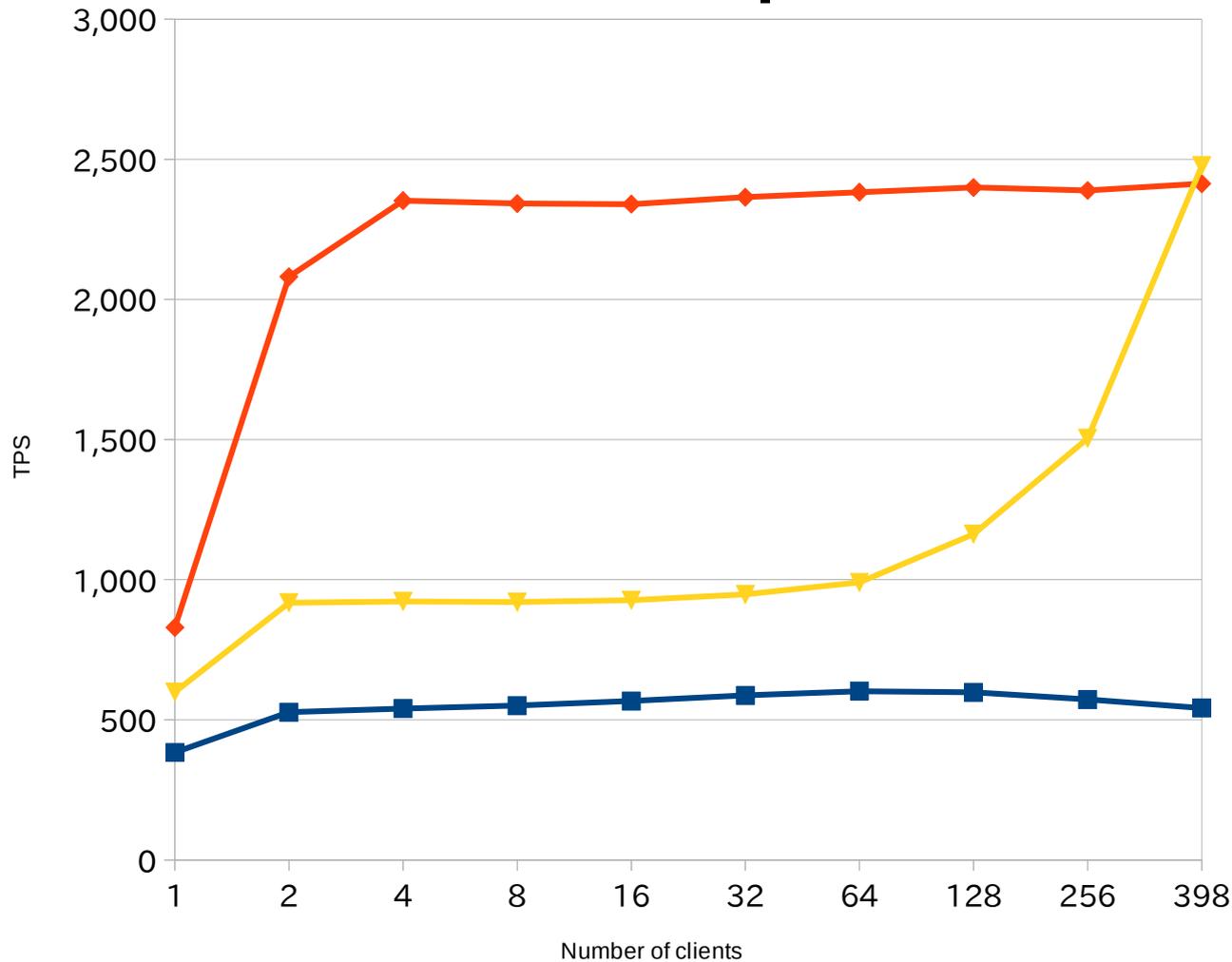
# Overcoming Thundering herd problem

pgpool-II 3.5



No thundering Herd problem

# Overcoming Thundering herd problem



If concurrent clients are fewer than number of pgpool child process, pgpool-II 3.5 is 40% to 150% faster than pgpool-II 3.4

■ PostgreSQL  
◆ pgpool-II 3.5  
▼ pgpool-II 3.4

Note PC with 16GB Mem,  
CORE i7 x2, 512GB SSD  
Ubuntu 14.04  
PostgreSQL 9.4 x2  
(streaming replication)  
pgbench -S -C -T 300

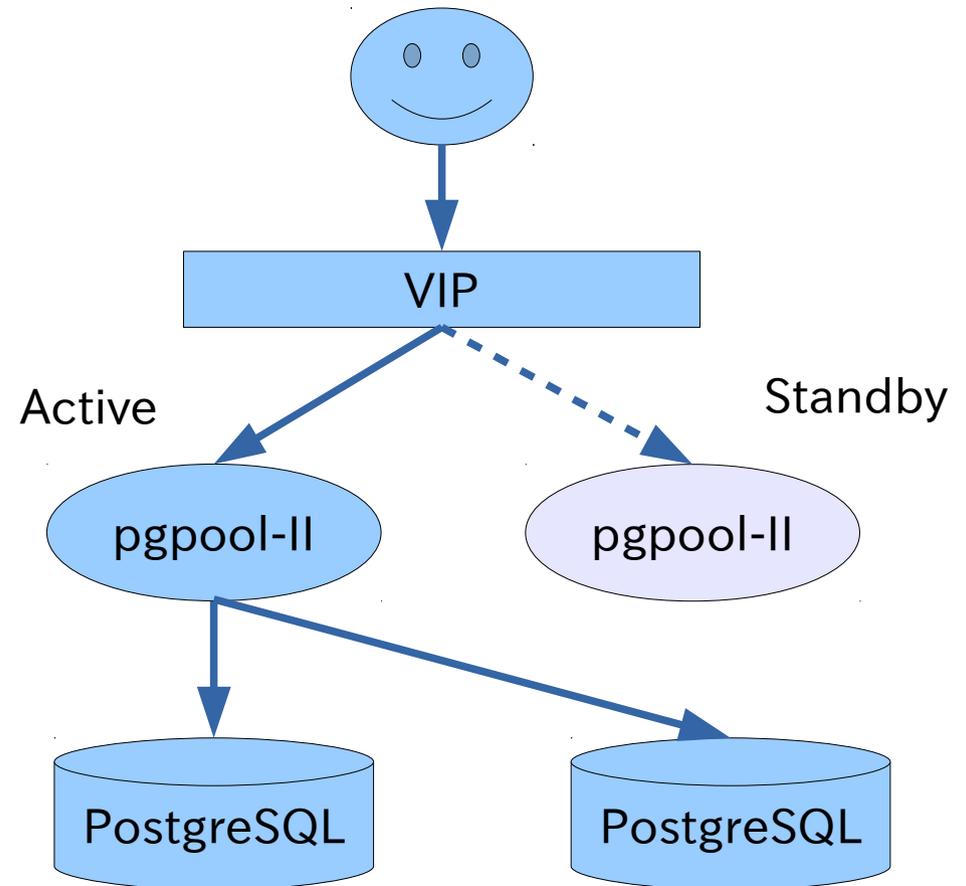
# Settings to avoid the thundering herd problem in pgpool-II 3.5

- set “serialize\_accept” to on
- set “child\_life\_time” to 0
- If concurrent connections are roughly equal to num\_init\_children, this function does not do the best (see previous slide)

# Improving watchdog

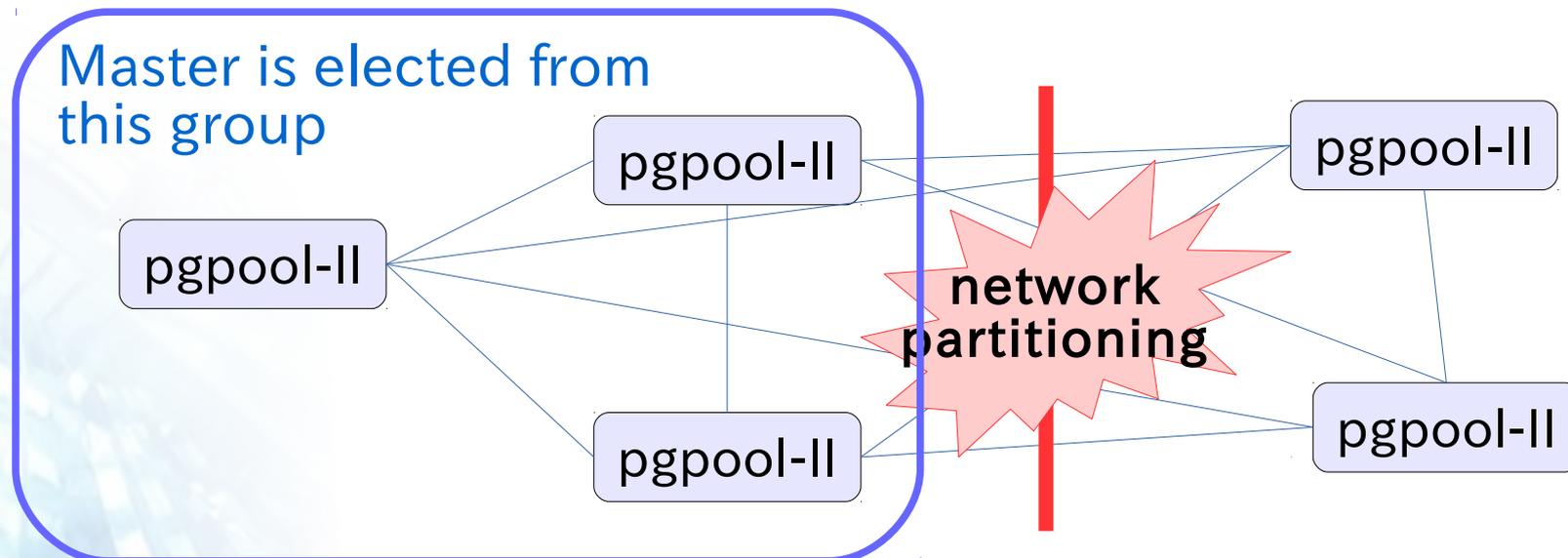
# What is “Watchdog”?

- Because pgpool-II works as a proxy, pgpool-II could be Single point of failure (SPOF)
- “Watchdog” is a built-in High Availability (HA) feature of pgpool-II
- Two or more pgpool-II instances monitors each other. If “Active” pgpool-II goes down, “Standby” pgpool-II takes over and becomes new active pgpool-II
- Active pgpool-II holds Virtual IP (VIP). Client connects to the VIP and are not worried about which is pgpool-II is alive



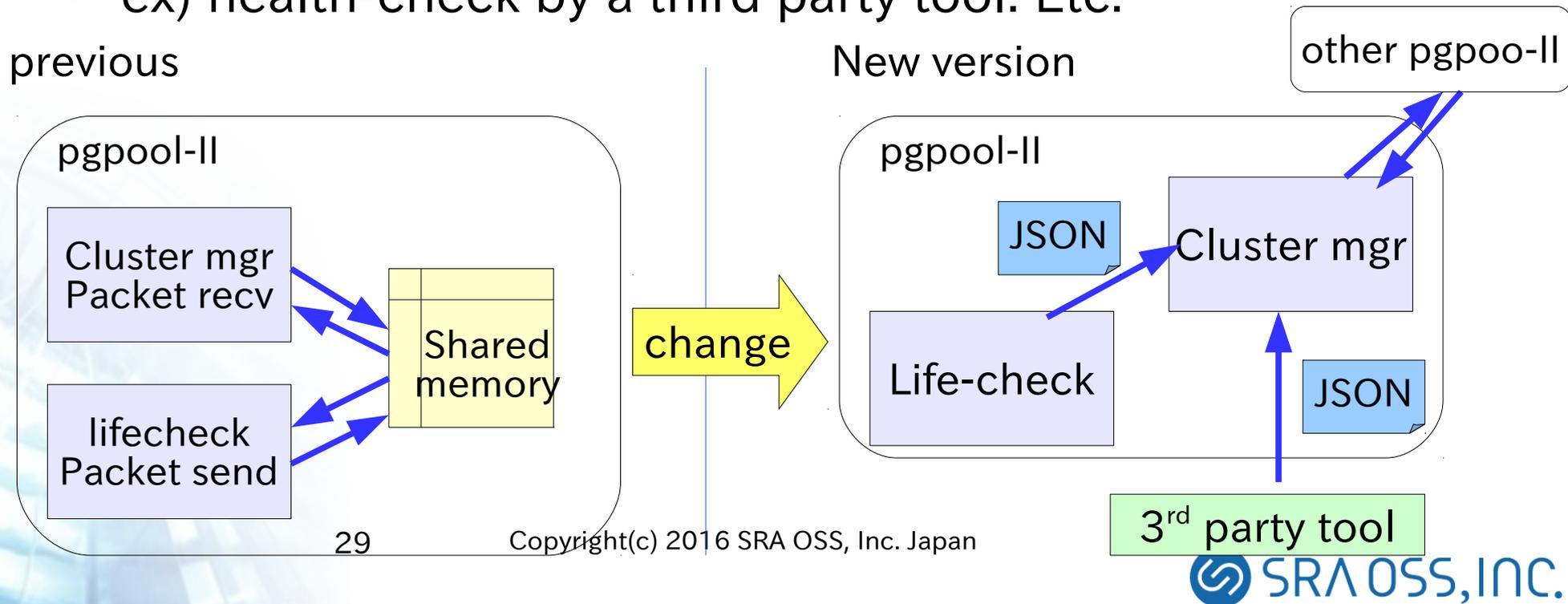
# New Watchdog Enhances Robustness against Split-brain

- Split-brain syndrome
  - It can't be decided which pgpool-II should be elected as the master when the network is participated
  - Quorum support
    - Check if more than half of nodes are belong to the group which local pgpool-II is belonging to
    - Number of pgpool-II nodes must be odd to make quorum working (in other case you can use “trusted\_servers”)



# New Watchdog enhances inter-process communication

- Change in inter-process communication method with in watchdog
  - UNIX domain socket & JSON format data
- This allows pgpool-II to work with third-party tools
  - ex) health-check by a third party tool. Etc.



# Watchdog enhancement others

- Verifies the consistency of important configuration parameters among all nodes
  - make sure they are consistent among all nodes
  - help to eliminate problems caused by ll-configured pgpool-ll nodes
- Watchdog nodes can have priorities
  - Watchdog nodes can be assigned with different priorities
  - Nodes with higher watchdog will get a preference when the cluster is electing its new leader node.

# Ready for PostgreSQL 9.5

# Importing PostgreSQL 9.5 parser

- Parser of PostgreSQL 9.5 is ported into pgpool-II 3.5
  - Previous parser was imported from PostgreSQL 9.4
- Load-balancing and query-cache supports the new select syntax.
  - GROUPING SET, CUBE, ROLLUP,
  - TABLESAMPLE
- Query-rewriting in the native replication mode supports the new insert/update syntax.
  - INSERT ... ON CONFLICT
  - UPDATE tab SET (col1,col2,...) = (SELECT ...), ...

# Improvements in pgpool-II 3.5: others

# Health check and replication delay checking target database

- In some systems (for example, Heroku) do not allow to connect to “postgres” or “template1” database
- pgpool-II issues query to those databases for health check and streaming replication delay
- pgpool-II allows to use particular database instead of the databases
  - health\_check\_database
  - sr\_check\_database

# Enhanced PCP comand

- What is “PCP” command
  - Stands for “Pgpool Control Protocol”
  - Set of commands to control pgpool-II
    - Attaching/detaching PostgreSQL to pgpool-II
    - Execute on-line recovery
    - Retrieve various information from pgpool-II
- Problem with older versions
  - Single session – no concurrent execution of commands is not allowed
  - Position arguments – hard to use
  - Needs to pass password as an argument – security issue
- All of the weakness is now eliminated

# Showing SELECT count

- “show pool\_nodes” command now shows the number of SELECTs being issued to each DB node

New!



```
test=# show pool_nodes;
```

node_id	hostname	port	status	lb_weight	role	select_cnt
0	/tmp	11002	2	0.500000	primary	338230
1	/tmp	11003	2	0.500000	standby	163939

(2 rows)

# Chinese translations!

- Pgpool-II documentation
- pgpoolAdmin message catalog

# Caution!

- “Parallel mode” is removed
  - Too many restrictions
  - Virtually 0 user
  - Complexity of code

# Future plans

- Pgpool-II 3.6 is expected out late this year
  - Will focus on stability and usability
  - Followings are new features in discussion:
    - More comprehensive test cases
    - SET command
    - Improved documentation (SGML?)
    - Minimize users session disconnections in case of fail over

# Where to get pgpool-II?

- pgpool-II official site
  - <http://www.pgpool.net>
- SRA OSS
  - <http://www.sraoss.co.jp>
  - Commercial support is available

Thank you!  
С п а с и б о !

