SRA OSS, INC.

Making master/slave systems work better with pgpool-II

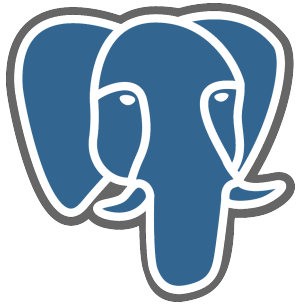
SRA OSS, Inc. Japan
Tatsuo Ishii

About me

- Running SRA OSS, Inc. Japan
- PostgreSQL developer/committer
  - I18N works
  - Several contrib tools including pgbench
- Writing books/articles about PostgreSQL

About SRA OSS, Inc. Japan

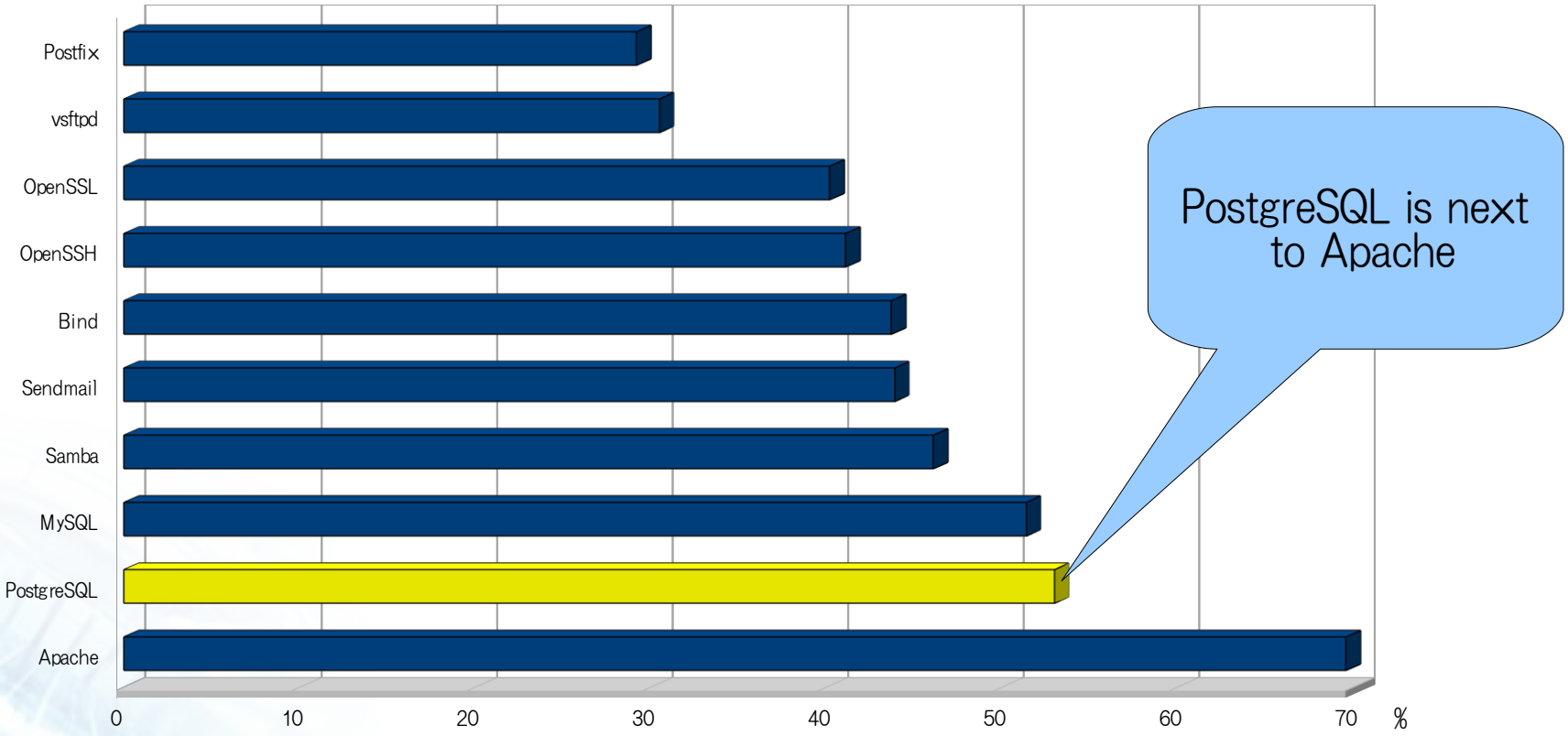
- Established in 2005
- Provids commercial support for PostgreSQL and other OSS
- No MySQL support
- Sells commercial packages based on PostgreSQL
- Trainings and certifications
- Supports PostgreSQL community
  - Funding JPUG
  - Board members



PowerGres

PostgreSQL CE

PostgreSQL is quite popular in Japan

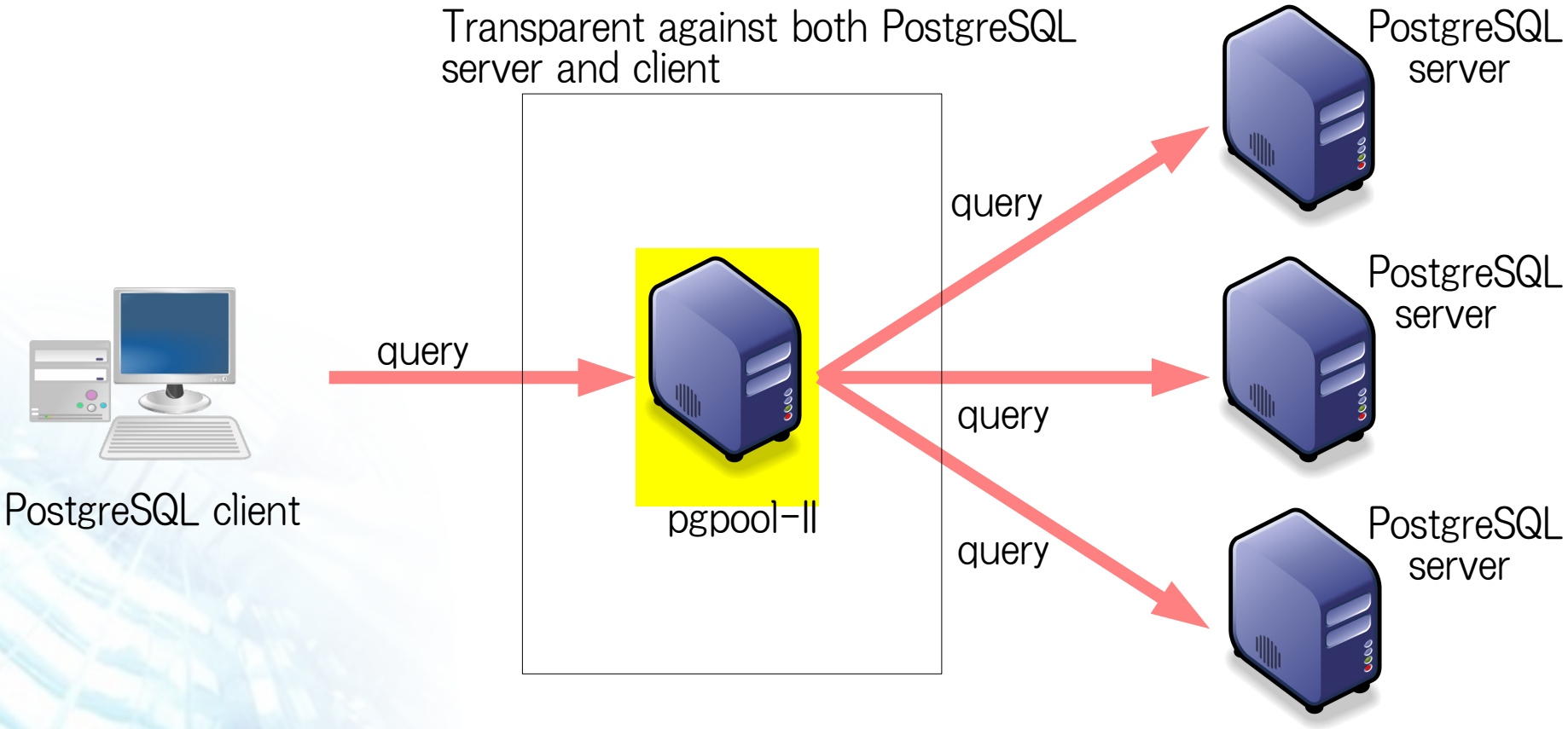


From IPA(Information technology promotion agency)'s survey in 2008

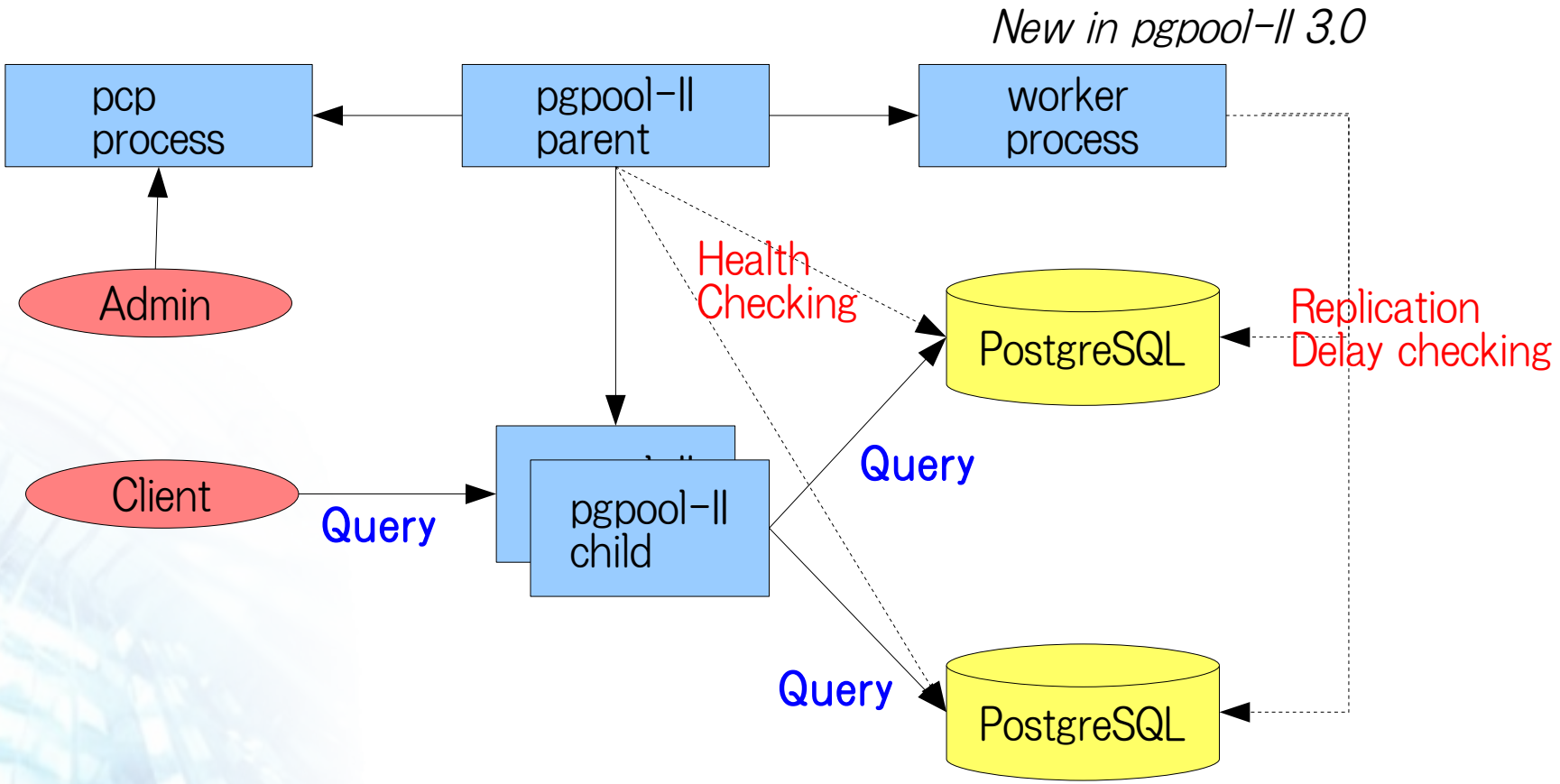
What is pgpool-II?

- Yet another cluster middle ware dedicated for PostgreSQL
- Community developed open source software project
- Rich features
  - Synchronous replication
  - Load balancing
  - Automatic failover
  - Connection pooling
  - Parallel query
  - Collaborating with other replication tools
    - Slony-I, Worm standby, Streaming replication

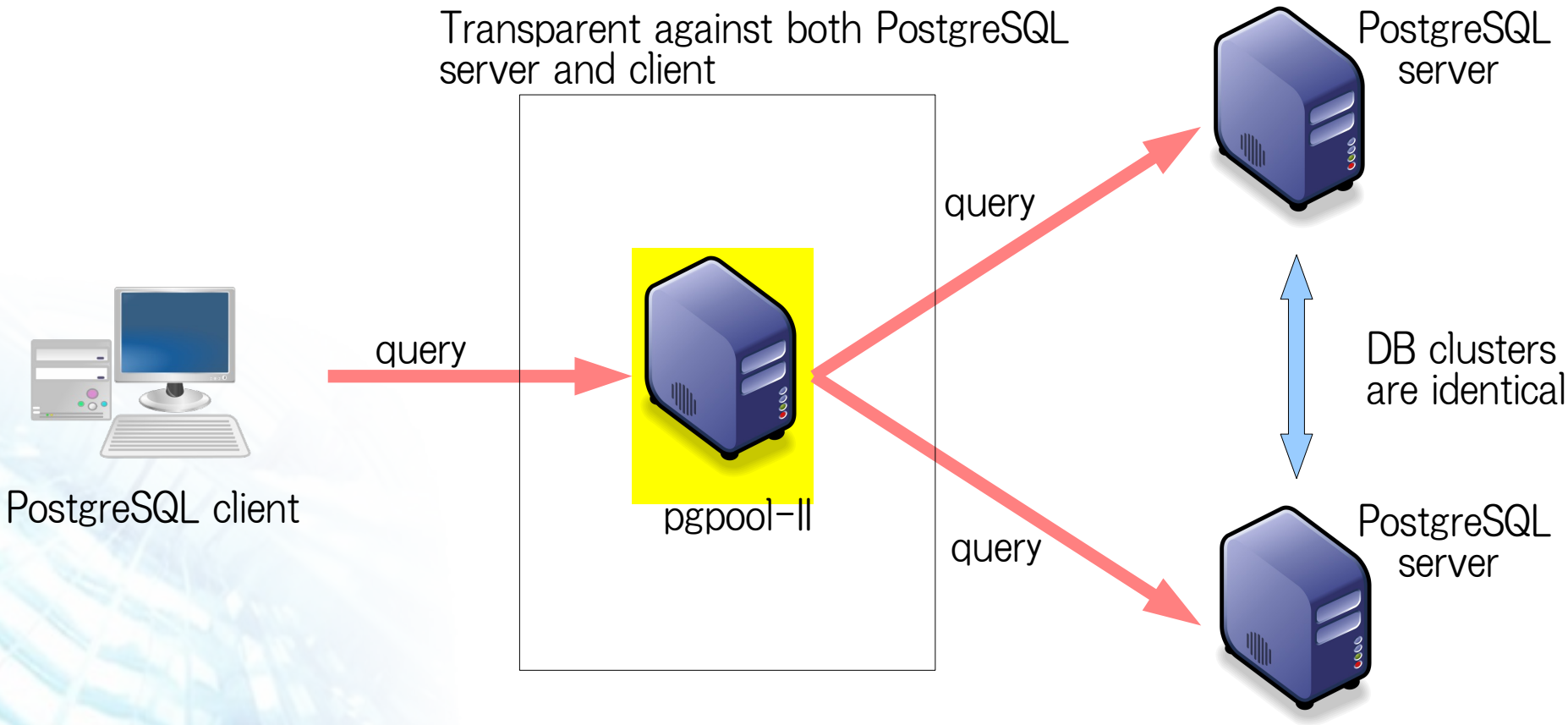
Basic idea of pgpool-II



The architecture of pgpool-II



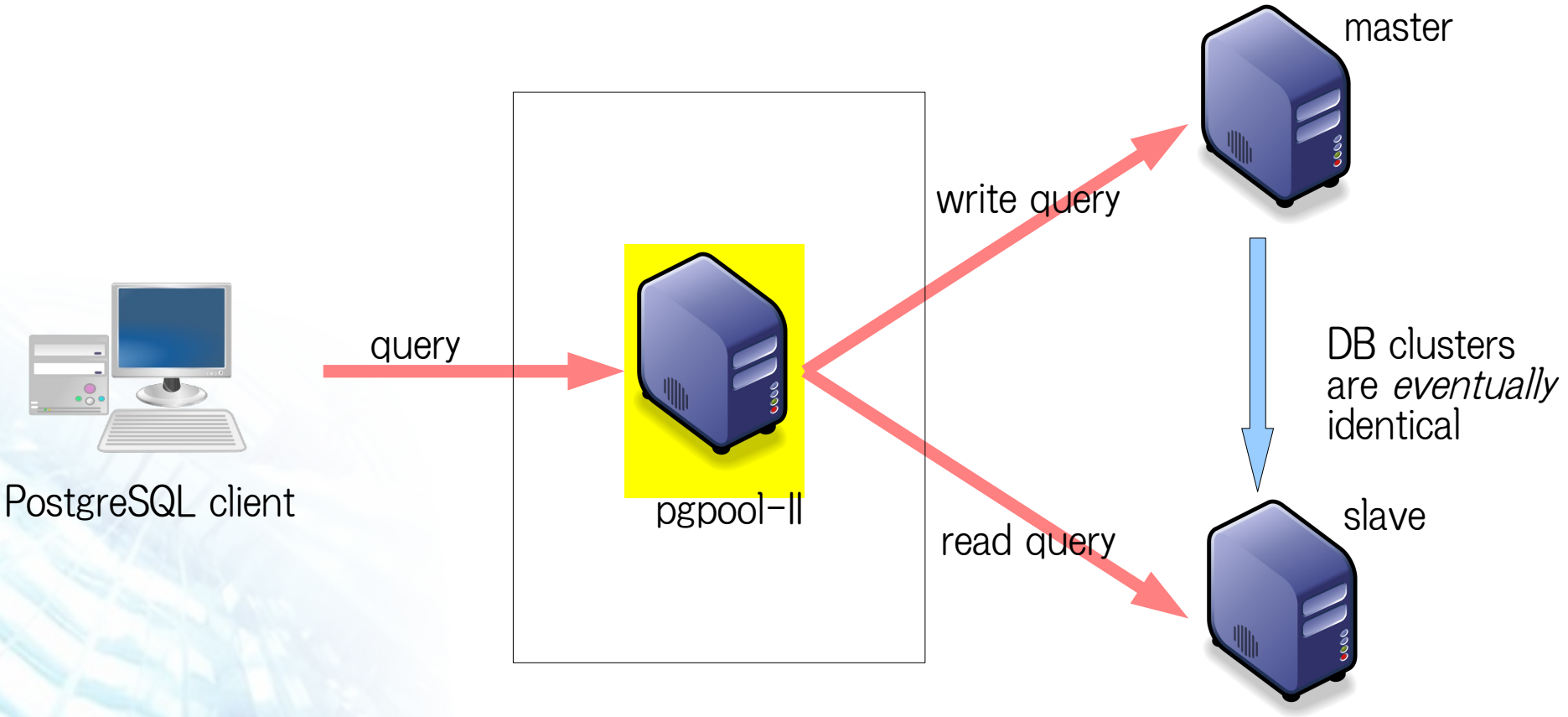
Replication mode



Pros and cons of pgpool-II replication

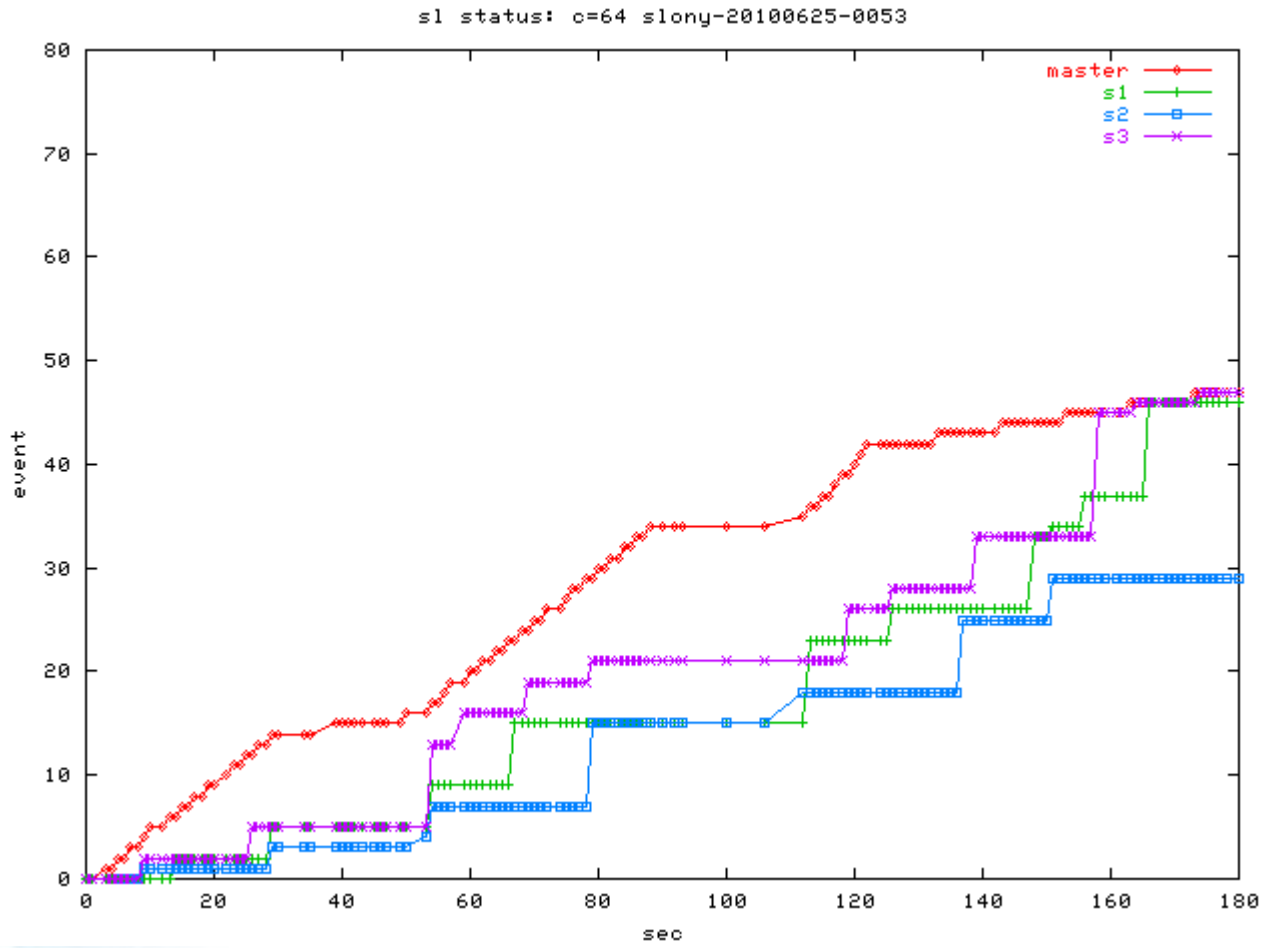
- Pros
  - Synchronous replication: no need to worry about "eventually consistent" problem. No transaction loss
  - Automatic failover: no phone calls from angry customers while you are in bed
  - Connection pooling and load balancing: boost performance
  - Online recovery. Without stopping pgpool-II, you can repair or add a DB node
  - Easy to configure
- Cons
  - Write performance is not good(about 30% overhead)
  - Some queries confuse pgpool-II: random(), sequences, functions those have a side effect(write to the database)

Master slave mode with Slony-I



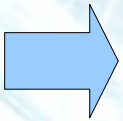
Pros and cons of master/slave mode

- Pros
  - Write performance is good (10-20% overhead)
  - Automatic failover of slaves
  - Connection pooling and load balancing: boost performance
- Cons
  - Asynchronous replication
    - Replication delay is relatively large
  - No DDL replication
    - Administrator's headache
  - No large objects replication
  - Hard to configure


s1 status: c=64 slony-20100625-0053
master
s1
s2
s3
event
sec
0
10
20
30
40
50
60
70
80
0
20
40
60
80
100
120
140
160
180

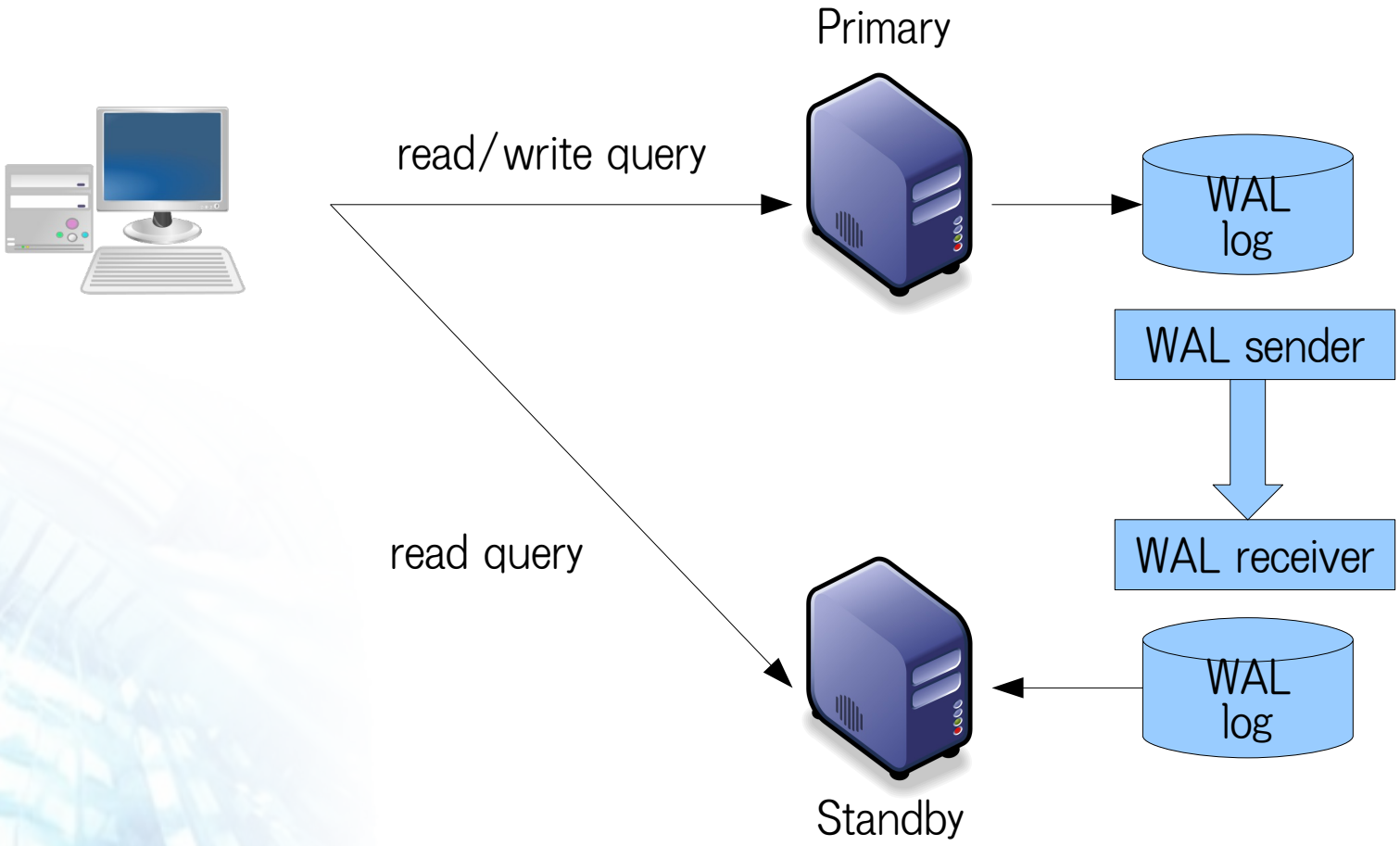
What we need is:

- Low write overhead
- Low replication delay
- Transparent replication
  - DDL replication
  - Large object replication
- Easy to manage

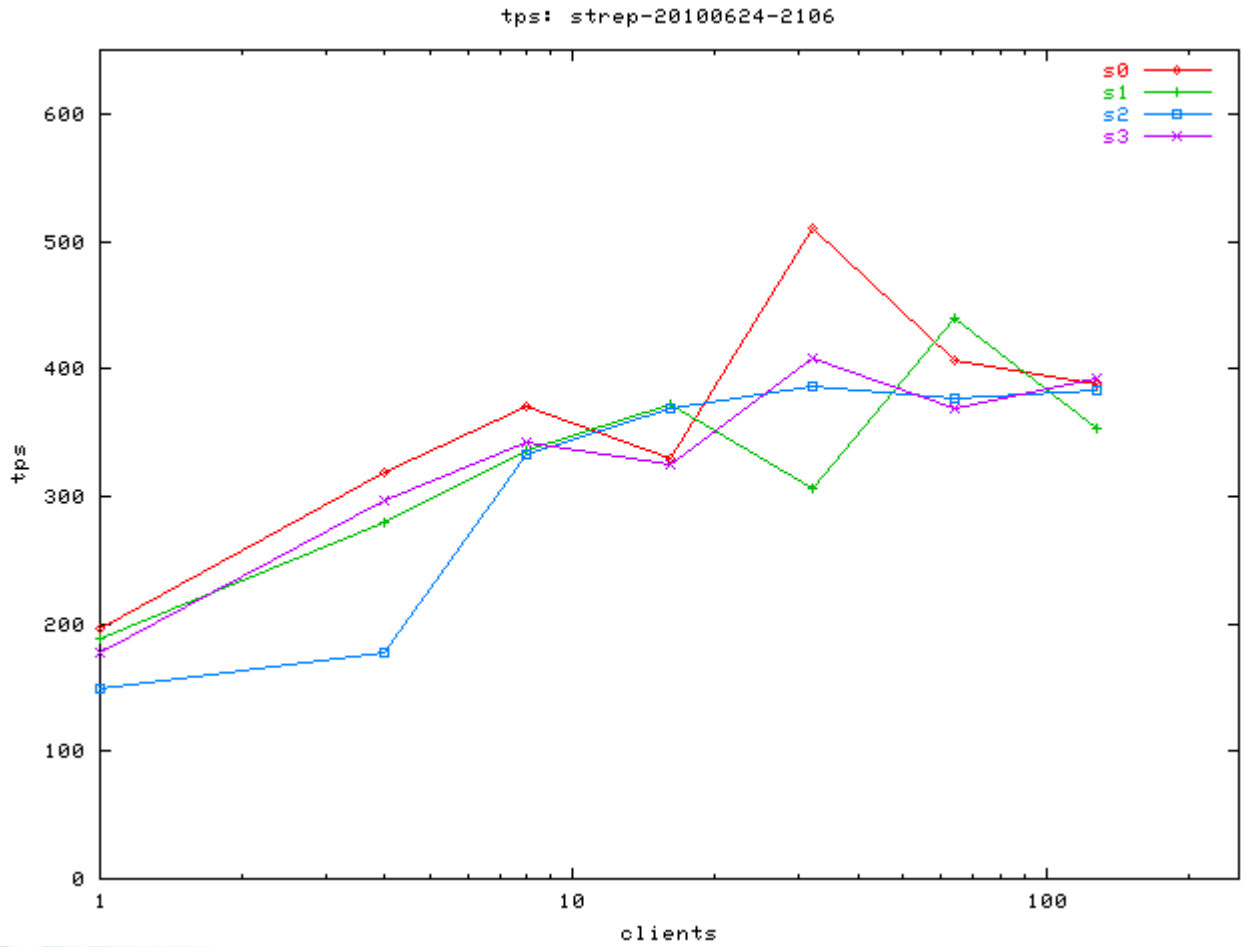


Sreaming Replication+Hot Standby

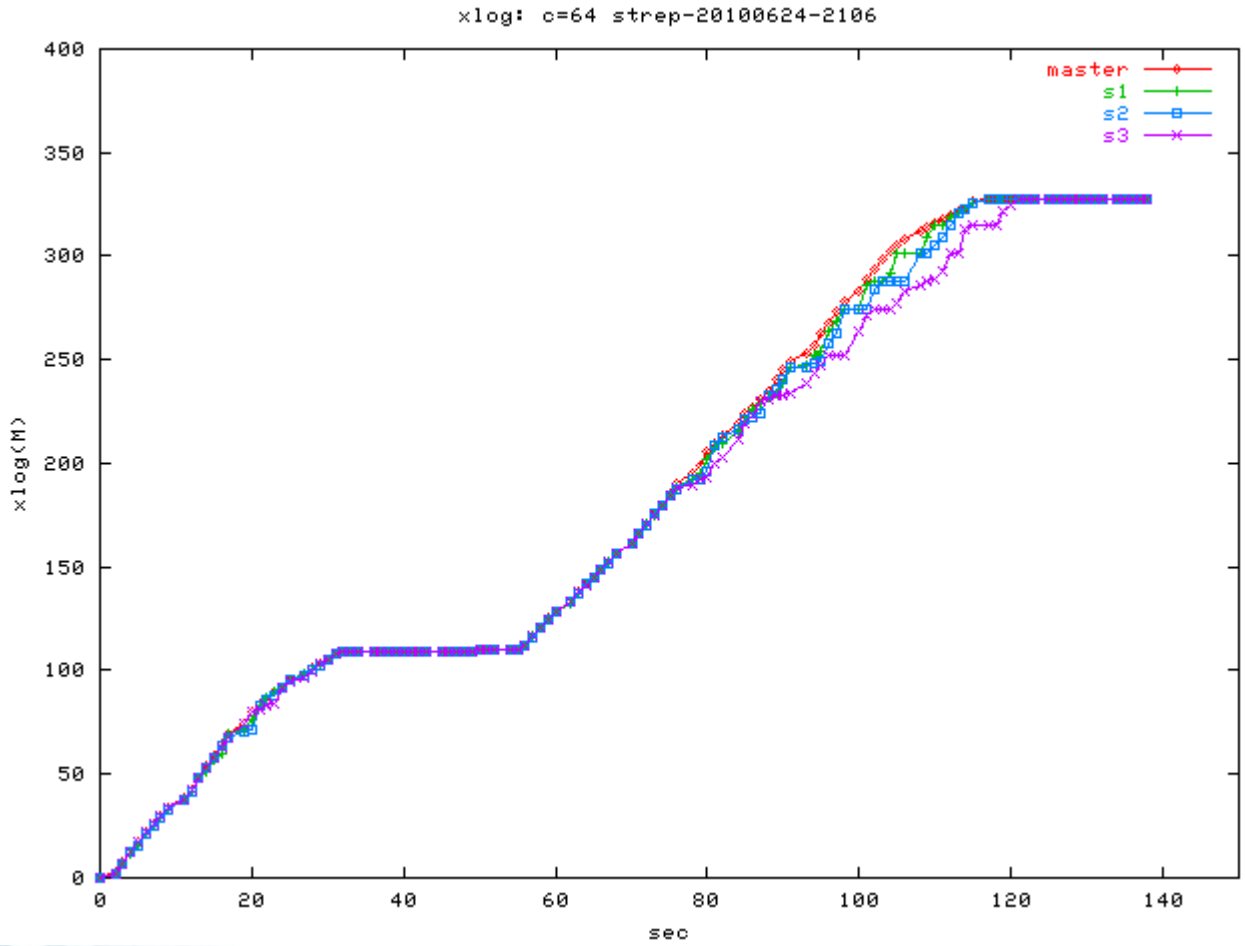
Streaming replication+Hot Standby



Low write overhead



Low replication delay



However...

- No automatic failover
- No connection pooling, load balancing
- You need to know what the standby dislike:

Data Manipulation Language (DML) - INSERT, UPDATE, DELETE, COPY FROM, TRUNCATE. Note that there are no allowed actions that result in a trigger being executed during recovery.

Data Definition Language (DDL) - CREATE, DROP, ALTER, COMMENT. This also applies to temporary tables also because currently their definition causes writes to catalog tables.

SELECT ... FOR SHARE | UPDATE which cause row locks to be written

Rules on SELECT statements that generate DML commands.

LOCK that explicitly requests a mode higher than ROW EXCLUSIVE MODE.

LOCK in short default form, since it requests ACCESS EXCLUSIVE MODE.

Transaction management commands that explicitly set non-read-only state:

BEGIN READ WRITE, START TRANSACTION READ WRITE

SET TRANSACTION READ WRITE, SET SESSION CHARACTERISTICS AS TRANSACTION READ WRITE

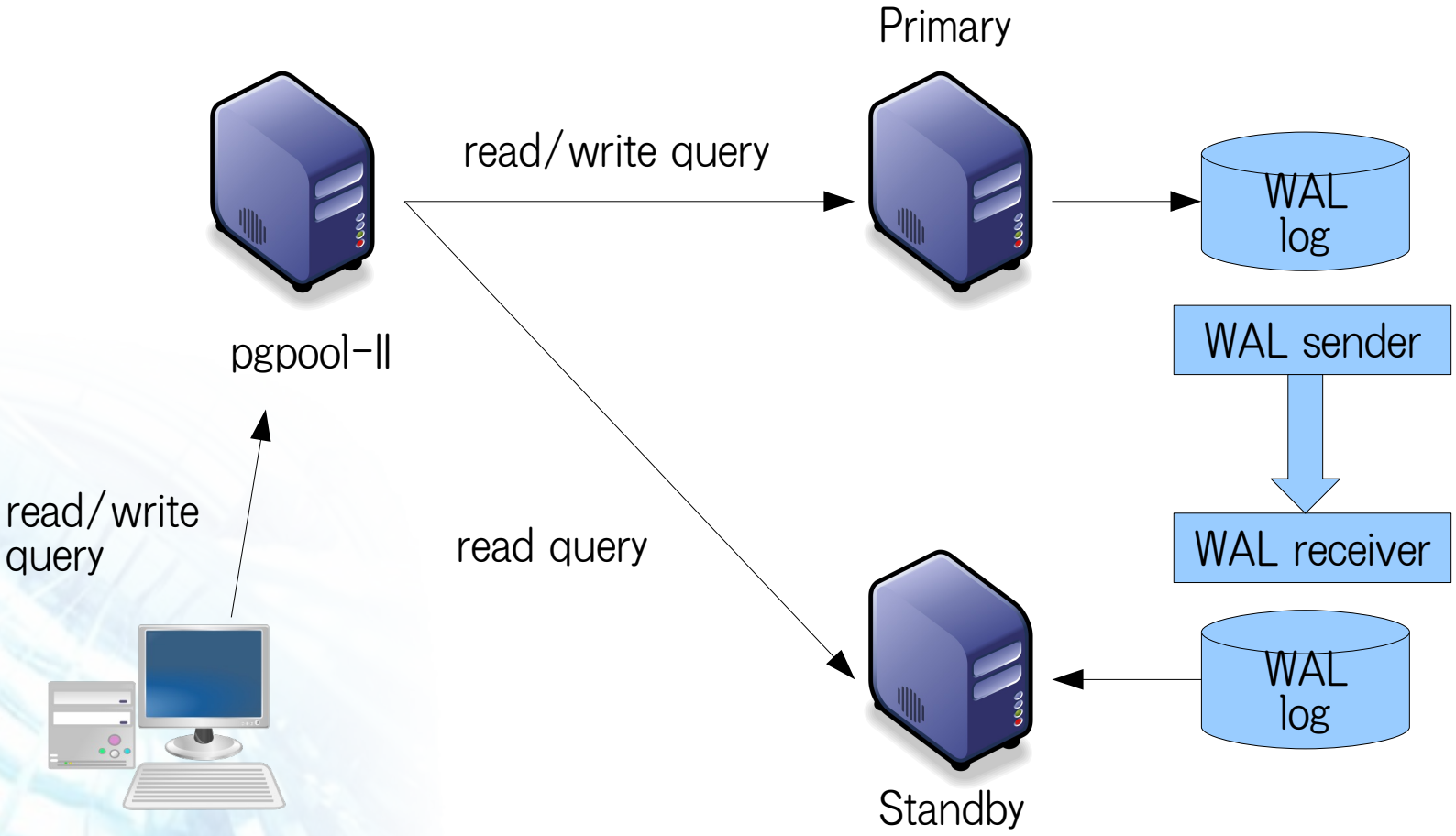
SET transaction_read_only = off

Two-phase commit commands - PREPARE TRANSACTION, COMMIT PREPARED, ROLLBACK PREPARED because even read-only transactions need to write WAL in the prepare phase (the first phase of two phase commit).

Sequence updates - nextval(), setval()

LISTEN, UNLISTEN, NOTIFY

Streaming replication+Hot Standby+ pgpool-II configuration



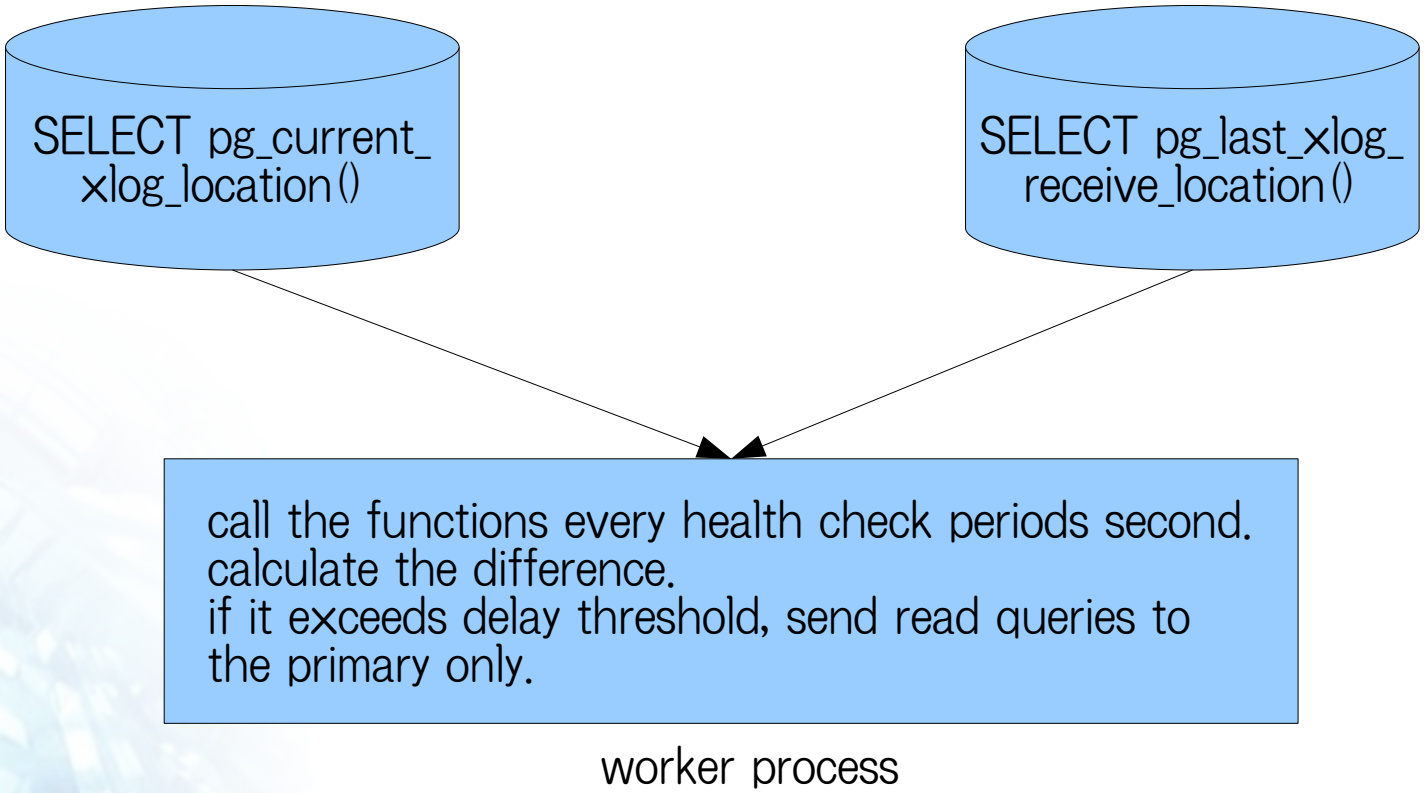
Pros and cons of Streaming replication+ Hot Standby+pgpool-II

- Pros
  - Write performance is good (10-20% overhead)
  - Automatic failover of slaves
  - Connection pooling and load balancing: boost performance
  - DDL replication
  - Large object replication
- Cons
  - Asynchronous replication
    - However replication delay is relatively low

New features in pgpool-II 3.0

- New master/slave "sub_mode" dedicated for Streaming replication and Hot standby
- Automatic query dispatches
  - Send write queries to the primary
- Intelligent load balancing
  - Prior releases: send read queries to the standby whenever possible
  - 3.0: Check the replication delay between the primary and the standby. If it's exceeds "delay_threshold", then send to the primary only.
- Adding standby servers without stopping pgpool-II

Replication delay detection: how it works



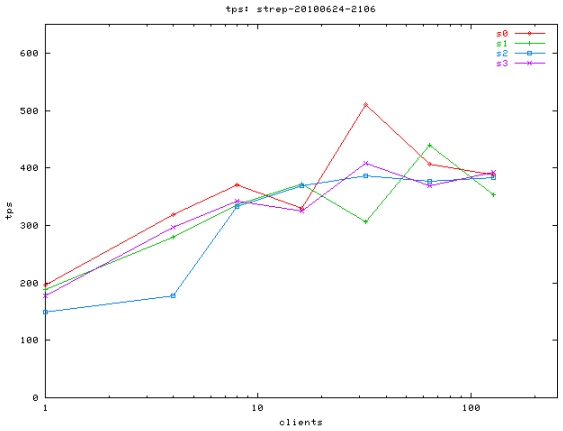
Logging replication delay

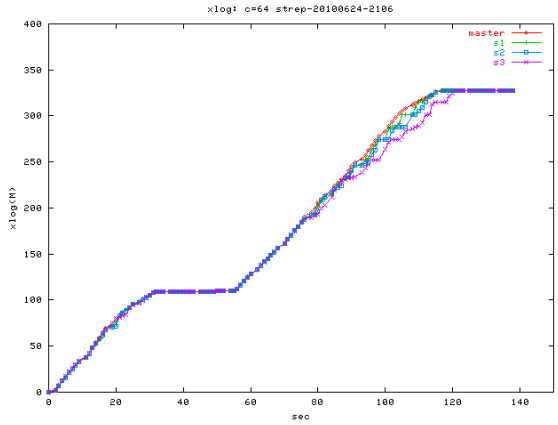
- log_standby_delay
  - 'none': no logging delay
  - 'if_over_threshold': log only when delay exceeds delay_threshold
  - 'always': always log delay

```
2010-06-28 15:51:32 LOG:   pid 13223: Replication of node:1 is behind 1228800 bytes
                from the primary server (node:0)
2010-06-28 15:51:42 LOG:   pid 13223: Replication of node:1 is behind 3325952 bytes
                from the primary server (node:0)
2010-06-28 15:51:52 LOG:   pid 13223: Replication of node:1 is behind 974848 bytes
                from the primary server (node:0)
2010-06-28 15:52:02 LOG:   pid 13223: Replication of node:1 is behind 2990080 bytes
                from the primary server (node:0)
2010-06-28 15:52:12 LOG:   pid 13223: Replication of node:1 is behind 901120 bytes
                from the primary server (node:0)
2010-06-28 15:52:22 LOG:   pid 13223: Replication of node:1 is behind 2433024 bytes
                from the primary server (node:0)
```

Scaling up with pgpool-II+Streaming replication+Hot standby

- Streaming replication scales well
  - We can add standby servers without sacrificing performance
  - More standby servers means better read performance with load balancing





Summary

- pgpool-II overview
- Various configurations of pgpool-II and their pros and cons
  - Replication mode
  - Master/Slave mode (Slony-I)
  - Master/Slave mode (Streaming replication + Hot standby)
- pgpool-II 3.0 + Streaming replication + Hot standby will offer one of the ideal replication solutions
  - Easy to use/manage
  - Low replication delay
  - Low write overhead+good scalability

URLS

- pgpool-II can be downloaded here:
  - http://pgfoundry.org/projects/pgpool/
- Visit SRA OSS's website if you need commercial support for pgpool-II and/or PostgreSQL:
  - http://www.srasoss.co.jp/index_en.php