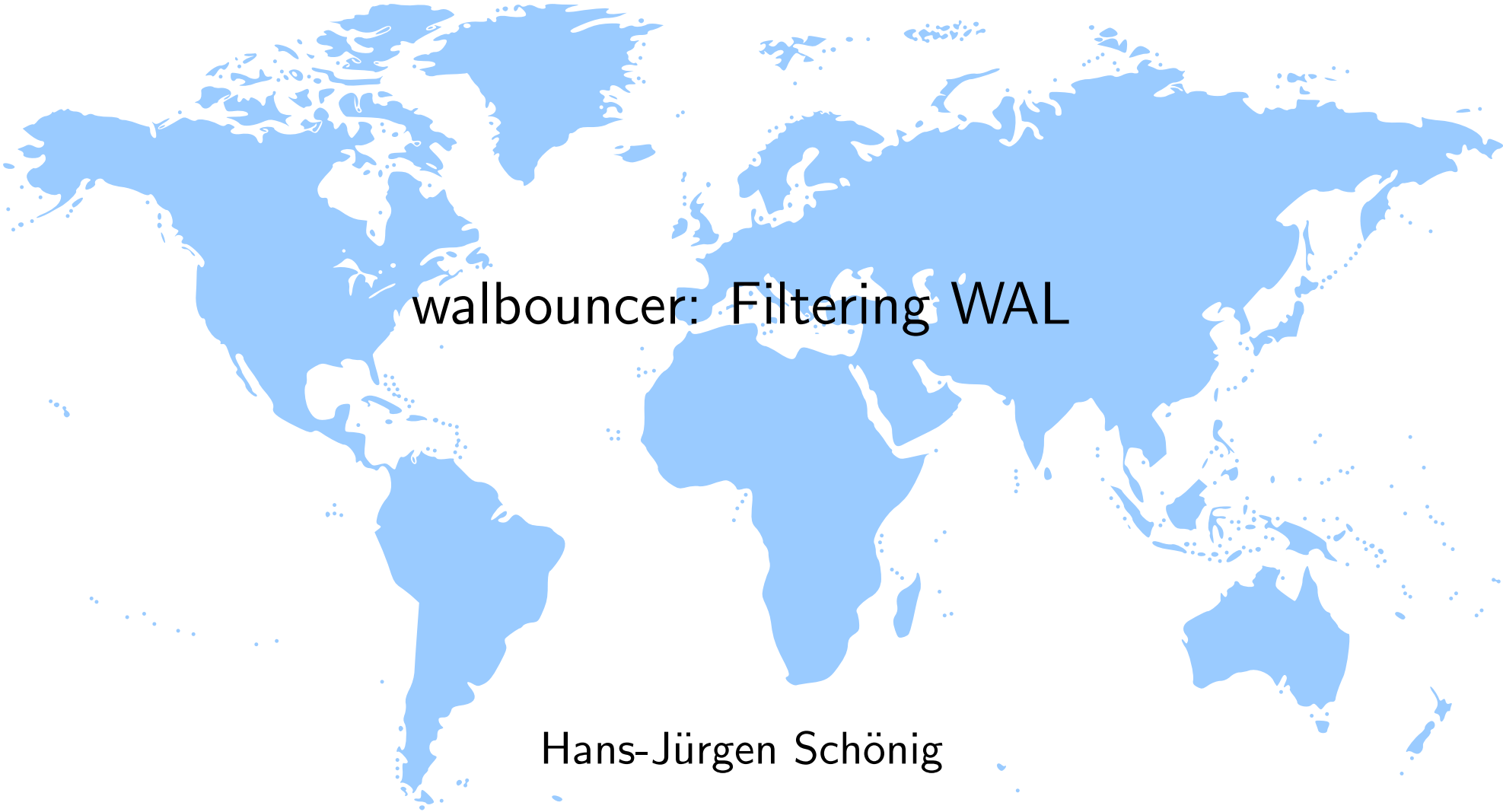

walbouncer: Filtering WAL

Hans-Jürgen Schönig

www.postgresql-support.de

walbouncer: Filtering WAL

Current status

- WAL streaming is the basis for replication

Limitations:

- Currently an entire database instance has to be replicated
- There is no way to replicate single databases
- WAL used to be hard to read

The goal

- Create a "WAL-server" to filter the transaction log
- Put walbouncer between the PostgreSQL master and the "partial" slave

How is it done?

CYBERTEC
The PostgreSQL Database Company

- The basic structure of the WAL is very actually fairly nice to filter
- Each WAL record that accesses a database has a RelFileNode:
  - database OID
  - tablespace OID
  - data file OID
- What more do we need?

WAL format

WAL format

- WAL logically is a stream of records.
- Each record is identified by position in this stream.
- WAL is stored in 16MB files called segments.
- Each segment is composed of 8KB WAL pages.

WAL pages

- Wal pages have a header containing:
  - 16bit “magic” value
  - Flag bits
  - Timeline ID
  - Xlog position of this page
  - Length of data remaining from last record on previous page
- Additionally first page of each segment has the following information for correctness validation:
  - System identifier
  - WAL segment and block sizes

Xlog record structre

- Total length of record
- Transaction ID that produced the record
- Length of record specific data excluding header and backup blocks
- Flags
- Record type (e.g. Xlog checkpoint, transaction commit, btree insert)
- Start position of previous record
- Checksum of this record
- Record specific data
- Full page images

Handling WAL positions

- WAL positions are highly critical
- WAL addresses must not be changed
  - addresses are stored in data page headers to decide if replay is necessary.
- The solution:
  - inject dummy records into the WAL

Dummy records

- PostgreSQL has infrastructure for dummy WAL entries (basically “zero” values)
- Valid WAL records can therefore be replaced with dummy ones quite easily.
- The slave will consume and ignore them

Question: What to filter?

- What about the shared catalog?
- We got to replicate the shared catalog
- This has some consequences:
  - The catalog might think that a database called X is around but in fact files are missing.
- This is totally desirable

Catalog issues

- There is no reasonably simple way to filter the content of the shared catalog (skip rows or so).
- It is hardly possible to add semantics to the filtering
- But, this should be fine for most users
- If you want to access a missing element, you will simply get an error (missing file, etc.).

Replication protocol

Streaming replication

- Replication connections use the same wire protocol as regular client connections.
- However they are serviced by special backend processes called walsenders
- Walsenders are started by including “replication=true” in the startup packet.
- Replication connections support different commands from a regular backend.

Starting up replication 1/2

- When a slave connects it first executes `IDENTIFY_SYSTEM`

```
postgres=# IDENTIFY_SYSTEM;
      systemid       | timeline |  xlogpos   | dbname
---------------------+----------+------------+--------
 6069865756165247251 |        2 | 0/3B7E910 |
```

- Then any necessary timeline history files are fetched:

```
postgres=# TIMELINE_HISTORY 2;
     filename     |         content
------------------+-------------------------------
 00000002.history | 1  0/3000090  no recovery
                  target specified+
                    |
```

Starting up replication 2/2

- Streaming of writeahead log is started by executing:

```
START_REPLICATION [SLOT slot_name] [PHYSICAL] XXX/XXX
    [TIMELINE tli]
```

- `START_REPLICATION` switches the connection to a bidirectional COPY mode.

Replication messages

- Replication messages are embedded in protocol level copy messages. 4 types of messages:
  - XLogData (server -> client)
  - Keepalive message (server -> client)
  - Standby status update (client -> server)
  - Hot Standby feedback (client -> server)
- To end replication either end can close the copying with a CopyDone protocol message.
- If the WAL stream was from an old timeline the server sends a result set with the next timeline ID and start position upon completion.

Other replication commands

- Other replication commands:
  - `START_REPLICATION ... LOGICAL ...`
  - `CREATE_REPLICATION_SLOT`
  - `DROP_REPLICATION_SLOT`
  - `BASE_BACKUP`
- Not supported by walbouncer yet.

Software design

- Client connects to the WAL bouncer instead of the master.
- WAL bouncer forks, connects to the master and streams xlog to the client.
- A this point the WAL proxy does not buffer stuff.
- One connection to the master per client.

Software design - filtering

- WAL stream is split into records.
- Splitting works as a state machine consuming input and immediately forwarding it. We only buffer when we need to wait for a RelFileNode to decide whether we need to filter.
- Based on record type we extract the RelFileNode from the WAL record and decide if we want to filter it out.
- If we want to filter we replace the record with a `XLOG_NOOP` that has the same total length.

Synchronizing record decoding

- Starting streaming is hard. Client may request streaming from the middle of a record.
- We have a state machine for synchronization.
  - Determine if we are in a continuation record from WAL page header.
  - If we are, stream data until we have buffered the next record header.
  - From next record header we read the previous record link, then restart decoding from that position.
  - Once we hit the requested position stream the filtered data out to client.

Using WAL bouncer

Streaming

- At this point we did not want to take the complexity of implementing buffering.
- Getting right what we got is already an important step

How to set things up

- First f all an initial base backup is needed
- The easiest thing here is rsync
  - Skip all directories in "base", which do not belong to your desired setup
  - pg_database is needed to figure out, what to skip.

Start streaming

- Once you got the initial copy, setup streaming replication just as if you had a "normal" master.
- Use the address of the walbouncer in your `primary_conninfo`
- You will not notice any difference during replication

Important aspects

- If you use walbouncer use the usual streaming replication precautions
  - enough `wal_keep_segments` or use
  - take care of conflicts (`hot_standby_feedback`)
  - etc.
- You cannot promote a slave that has filtered data to master.

Adding databases

- List of database OIDs to filter out is only fetched at walbouncer backend startup.
  - Disconnect any streaming slaves and reconfigure filtering while you are creating the database.
  - If you try to do it online you the slaves will not know to filter the new database.

Dropping databases

- Have all slaves that want to filter out the dropped database actively streaming before you execute the drop.
- Otherwise the slaves will not know to skip the drop record and xlog replay will fail with an error.

Can we filter on individual tables

- For filtering individual tables you can use tablespace filtering functionality.
- Same caveats apply for adding-removing tablespaces as for databases.
- You can safely add/remove tables in filtered tablespaces and even move tables between filtered/non-filtered tablespaces.

Bonus feature

- You can use walbouncer to switch the master server without restarting the slave.

Limitations

- Currently PostgreSQL 9.4 only.
- No SSL support yet.

Simple configuration

A sample config file (1)


```
listen_port: 5433
master:
    host: localhost
    port: 5432
configurations:
    - slave1:
        match:
            application_name: slave1
        filter:
            include_tablespaces: [spc_slave1]
            exclude_databases: [test]
```

A sample config file (2)


```
- slave2:
    match:
        application_name: slave2
    filter:
        include_tablespaces: [spc_slave2]
```

application_name

- The application name is needed to support synchronous replication as well as better monitoring

include_tablespaces / exclude_tablespaces

- A good option to exclude entire groups of databases
- In a perverted way this can be used to filter on tables
- No need to mention that you should not

Finally

Where can we download the stuff?

- For download and more information visit:: www.postgresql-support.de/walbouncer/

Thank you for your attention

Any questions?

Contact

Cybertec Schönig & Schönig GmbH
Gröhrmühlgasse 26
A-2700 Wiener Neustadt

www.postgresql-support.de