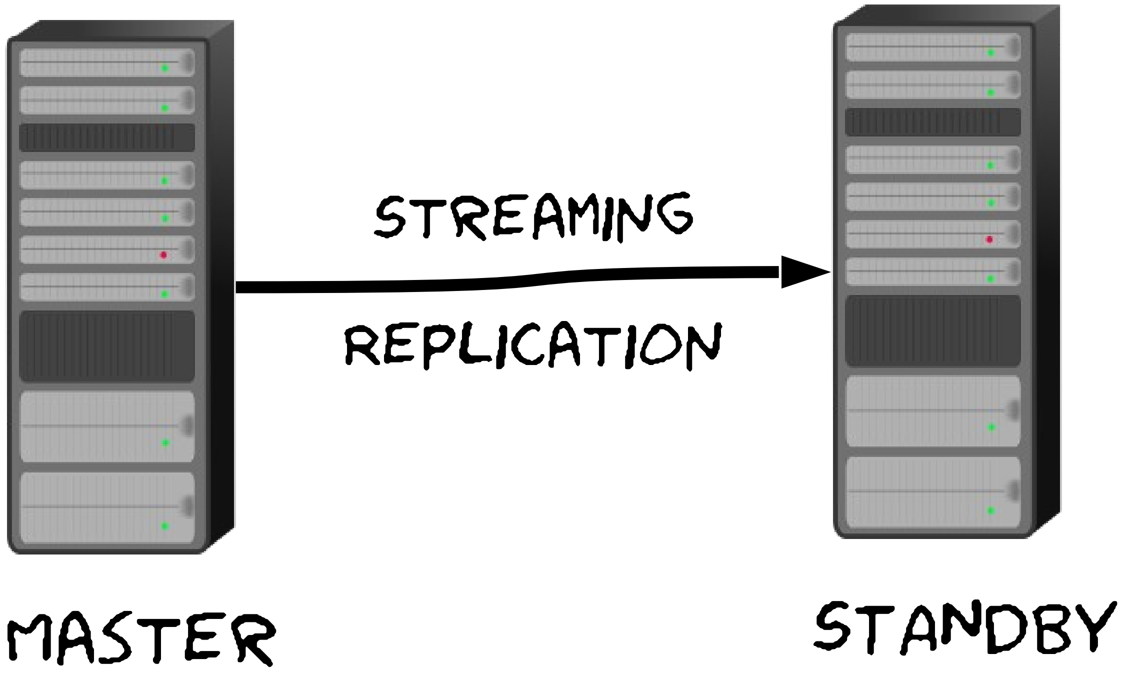
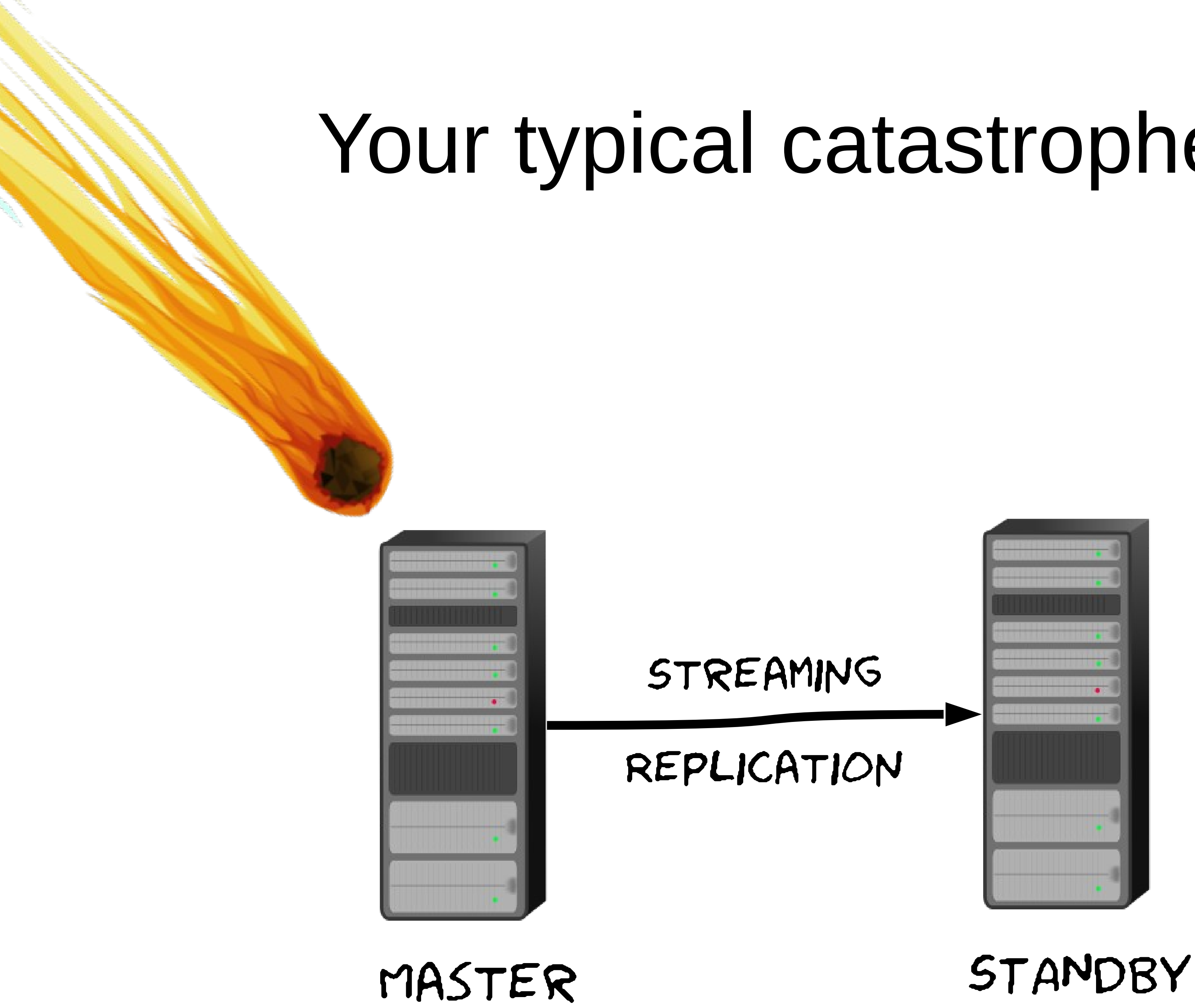
`pg_rewind`

Heikki Linnakangas

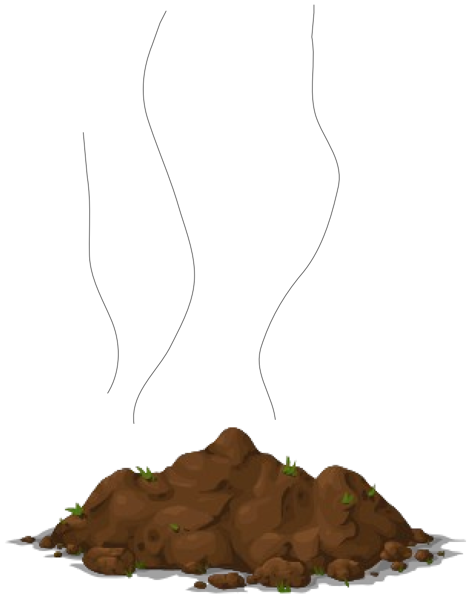
Your typical setup



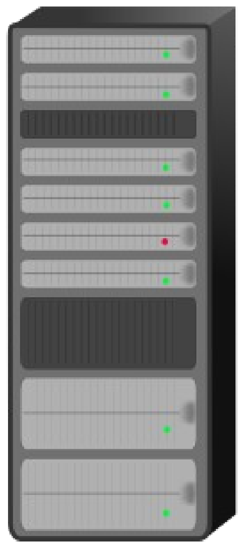
Your typical catastrophe



Standby takes over



~~MASTER~~

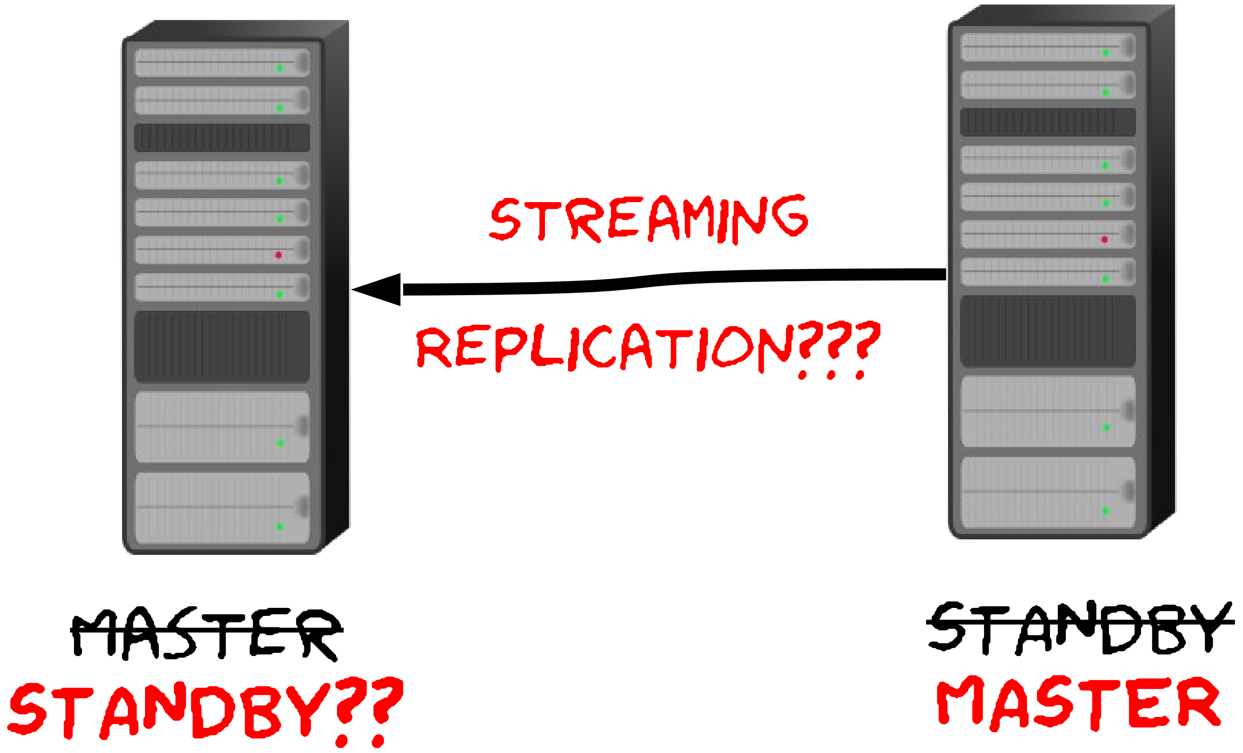


~~STANDBY~~
MASTER

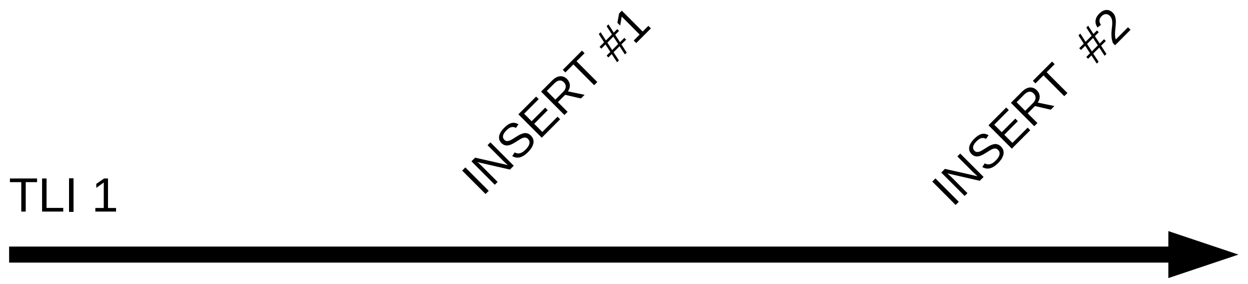
Wait, the old master survived after all!

~~MASTER~~
~~STANDBY~~
MASTER

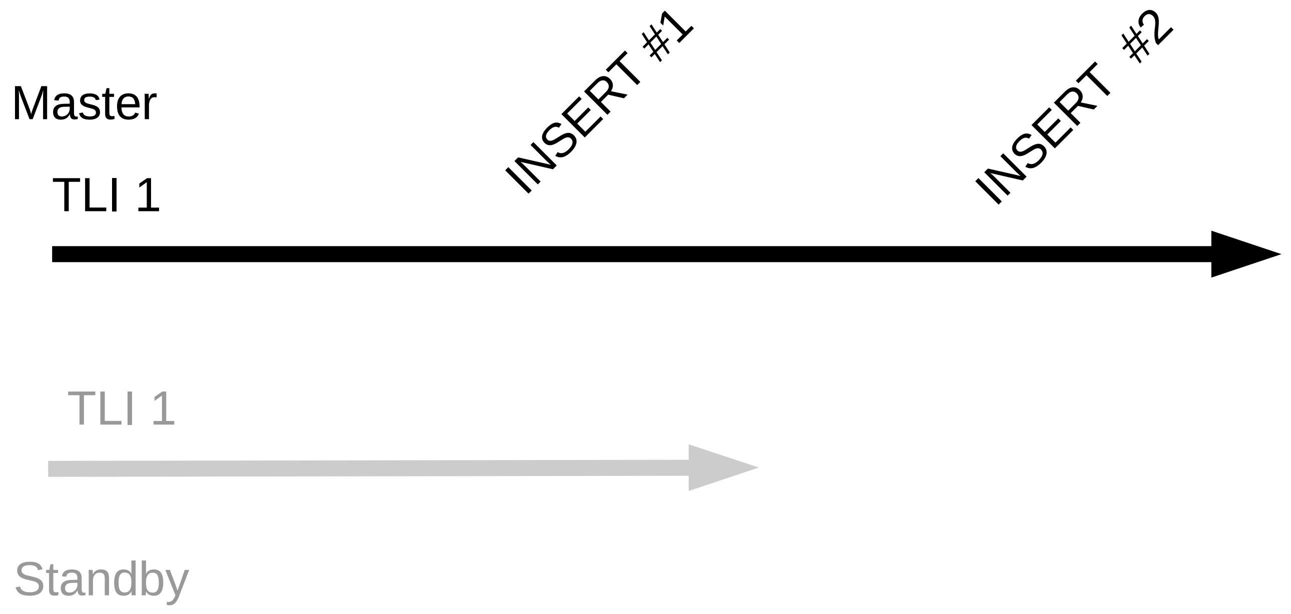
How do you turn the old master into standby?



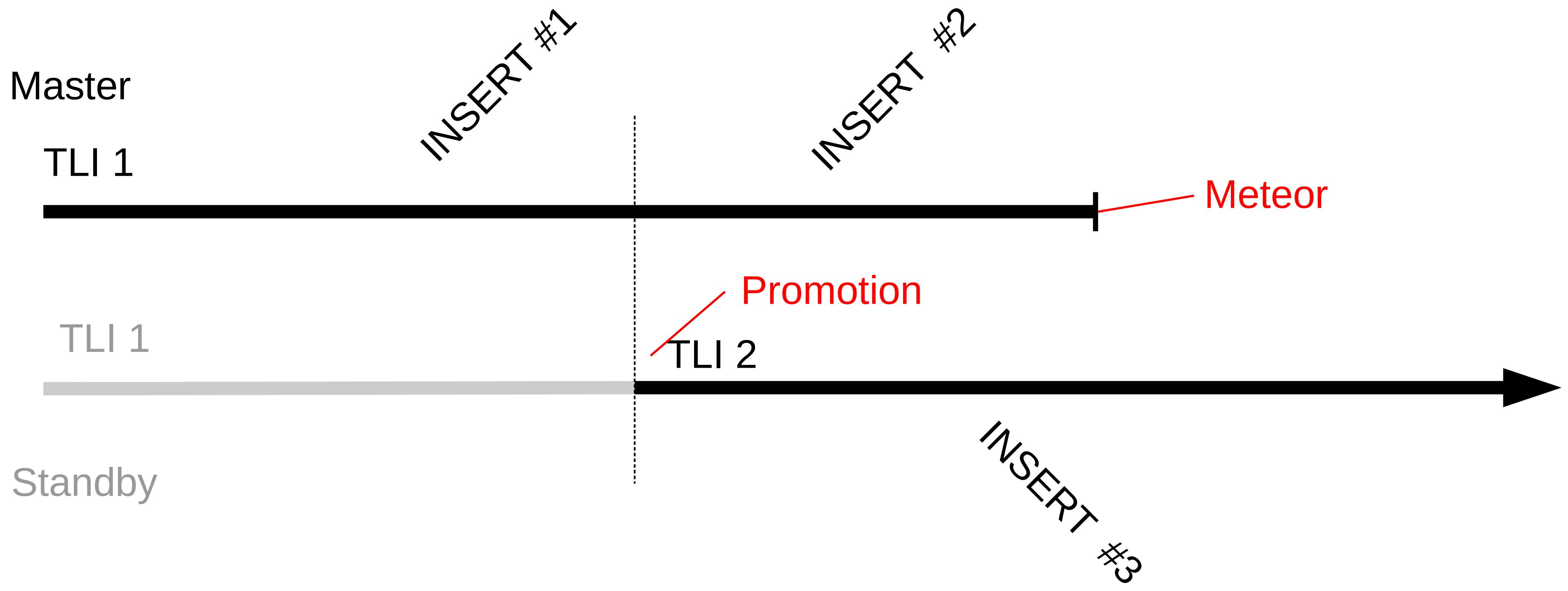
WAL Timelines



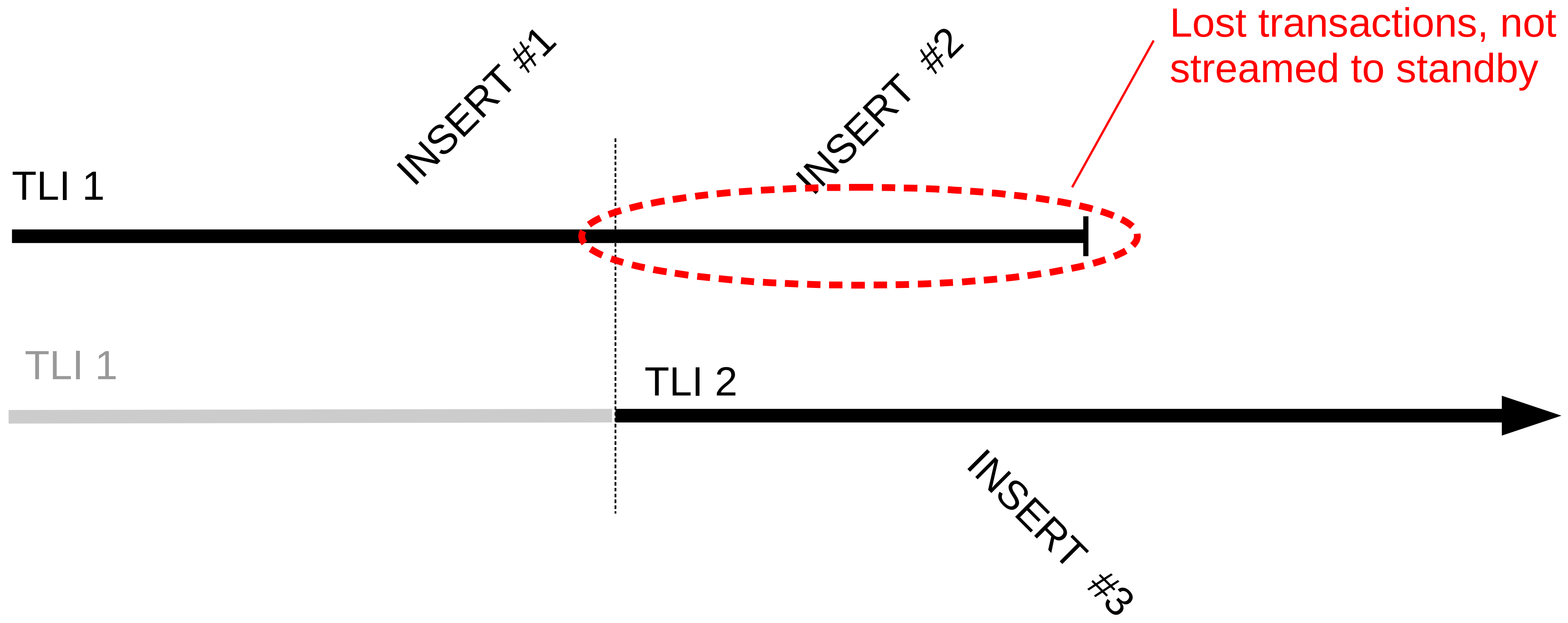
WAL Timelines



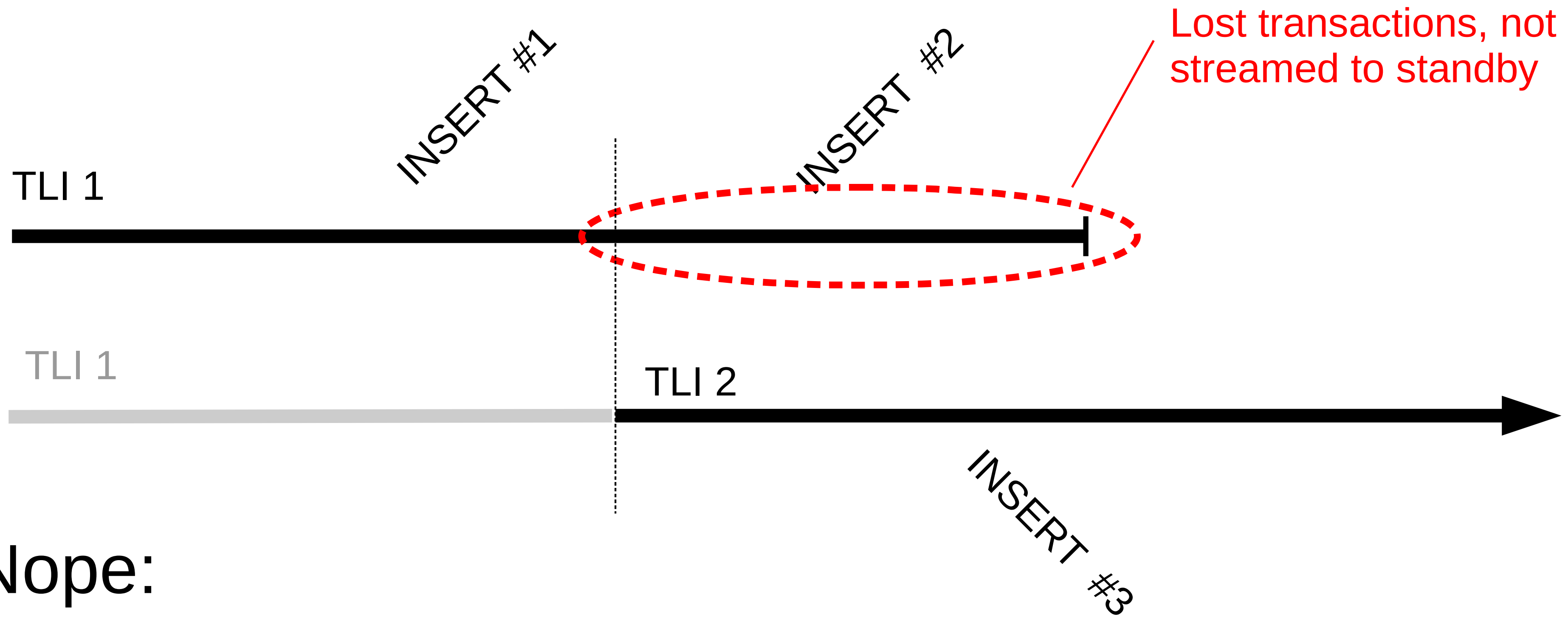
Promotion


Master
TLI 1
INSERT #1
INSERT #2
Meteor
Promotion
TLI 1
TLI 2
INSERT #3
Standby

Lost transactions



What about synchronous replication?



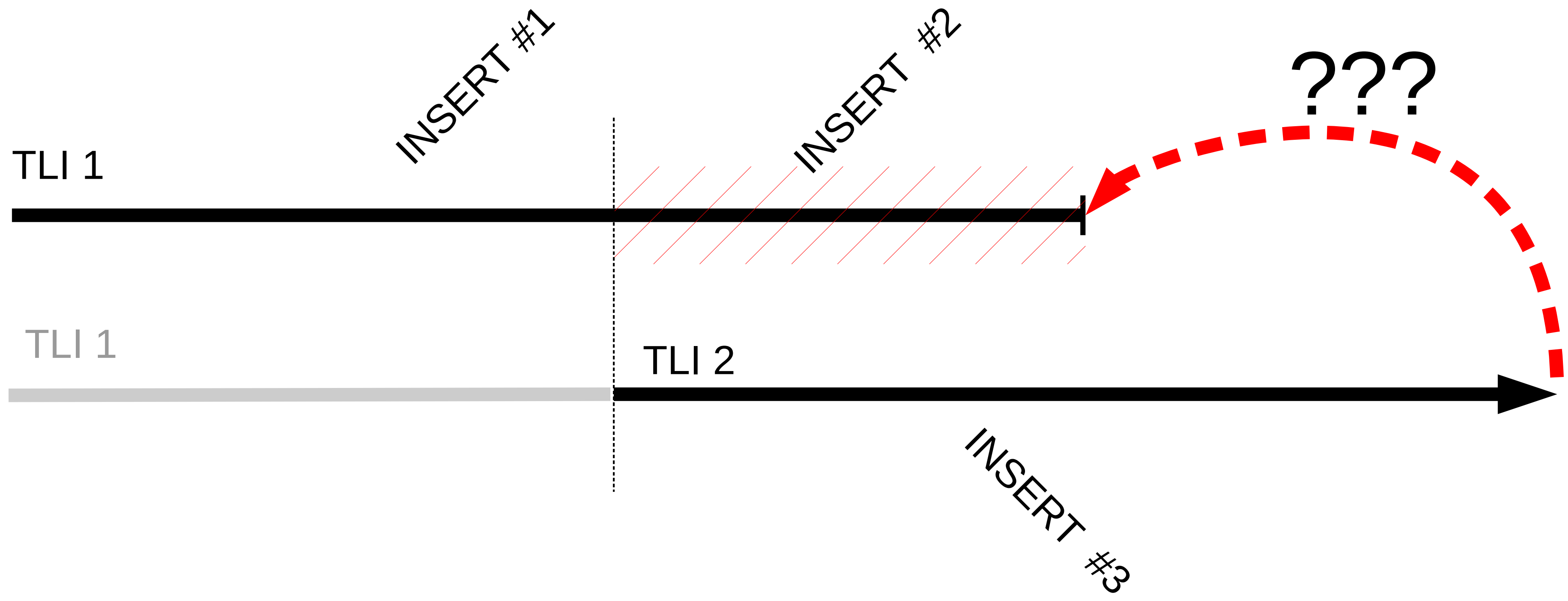
Nope:

- only commits are synchronized
- records may hit the disk in master before they're replicated anyway

Even controlled failover is tricky

- How do you verify that the standby got all the WAL?

How to resynchronize?



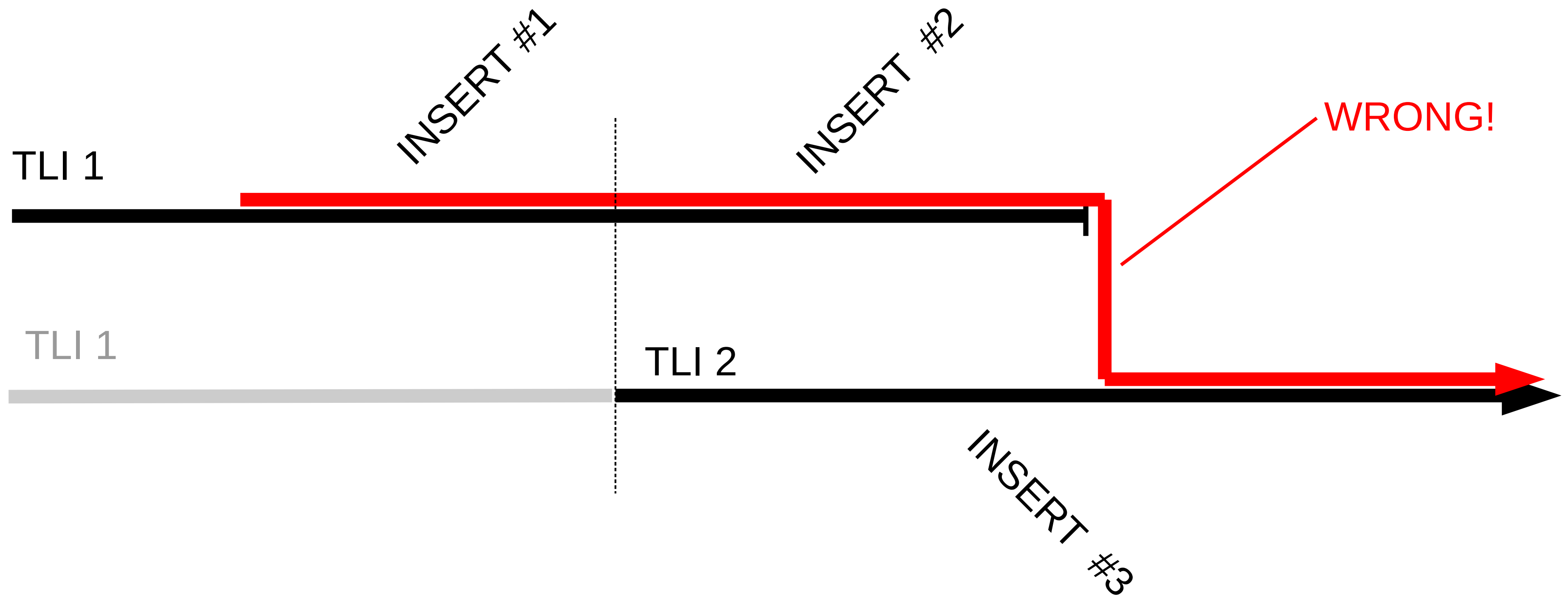
Naive approach

- Just create a `recovery.conf` file on old master to point to new master
- Will not work:

```
LOG:  database system was shut down at 2015-03-05 15:26:37 EET
LOG:  entering standby mode
LOG:  consistent recovery state reached at 0/4000098
LOG:  invalid record length at 0/4000098
LOG:  fetching timeline history file for timeline 2 from primary server
FATAL:  could not start WAL streaming: ERROR:  requested starting point
0/4000000 on timeline 1 is not in this server's history
DETAIL:  This server's history forked from timeline 1 at 0/3010758.
```

- Might appear to work, but may silently corrupt your database!

Wrong approach



Solution 1: Rebuild from scratch

- Erase old master, take new base backup from new master, and copy it over.
- Is slow
  - Reads all data from disk
  - Sends all data through the network
  - Writes all data to disk

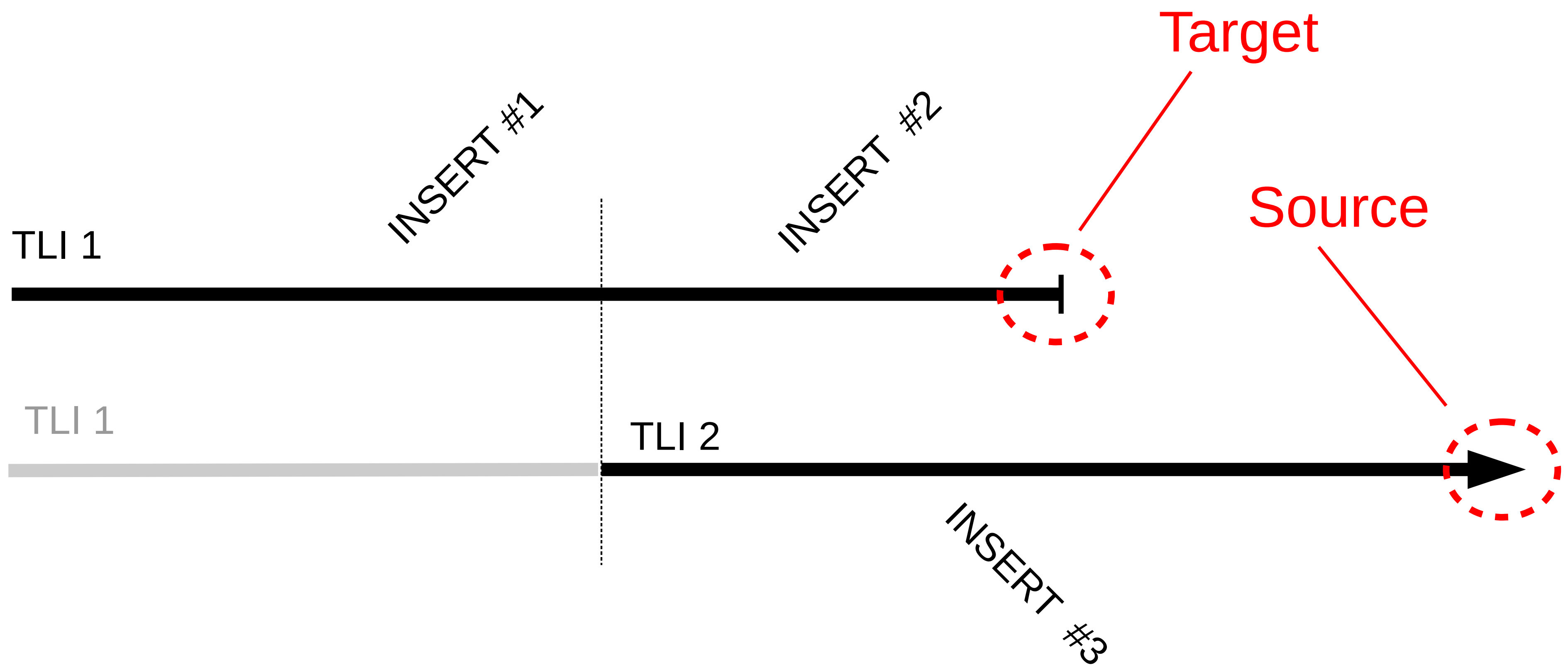
Solution 2: rsync

- Call `pg_start_backup()` in new master
- Use `rsync` to resynchronize the data dir
- Be careful which options you use
- Still slow
  - Reads all data from disk

Solution 3: pg_rewind

- Fast
  - Only reads and copies data that was changed

Terminology



Source: New master. Not modified.

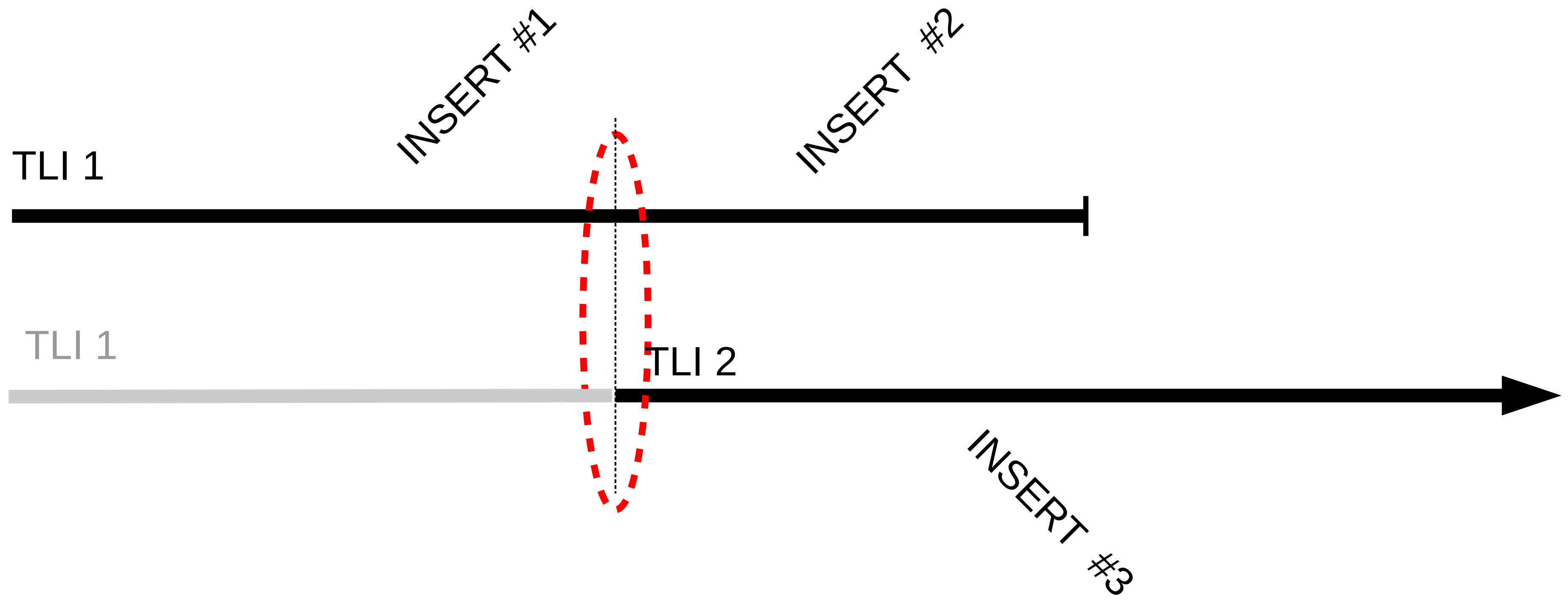
Target: Old master. Overwritten with data from source.

How it works

- Find out what blocks the lost transactions modified
- Copy those blocks from source to target

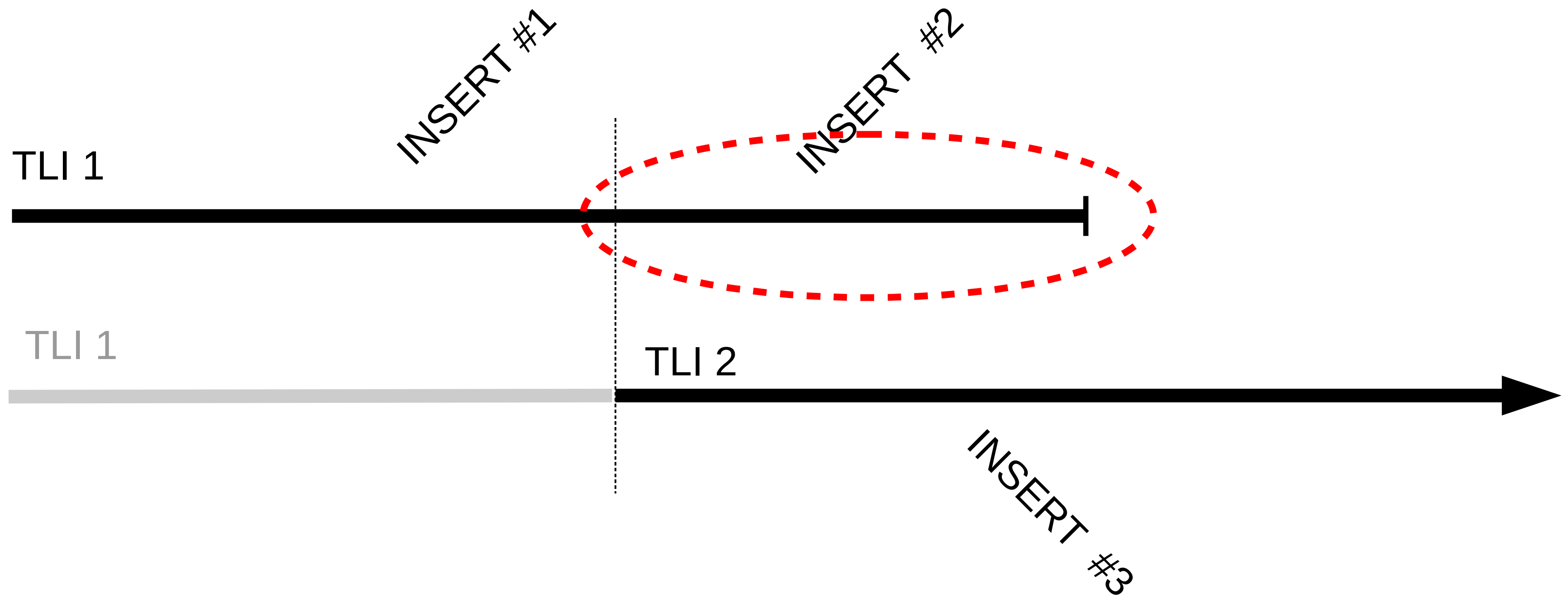
~ rsync on steroids

How it works?
1. Determine point of divergence



- Looks at the `pg_control` file on both systems

How it works?
2. Scan the old WAL



- Build a list of blocks that were changed on TLI 1
  - lost transactions

How it works?
3. Copy over all changed blocks

- Copies everything **except** those blocks of relation files that were not modified
  - pg_clog, etc.
  - Configuration files
  - FSM and VM files

File map

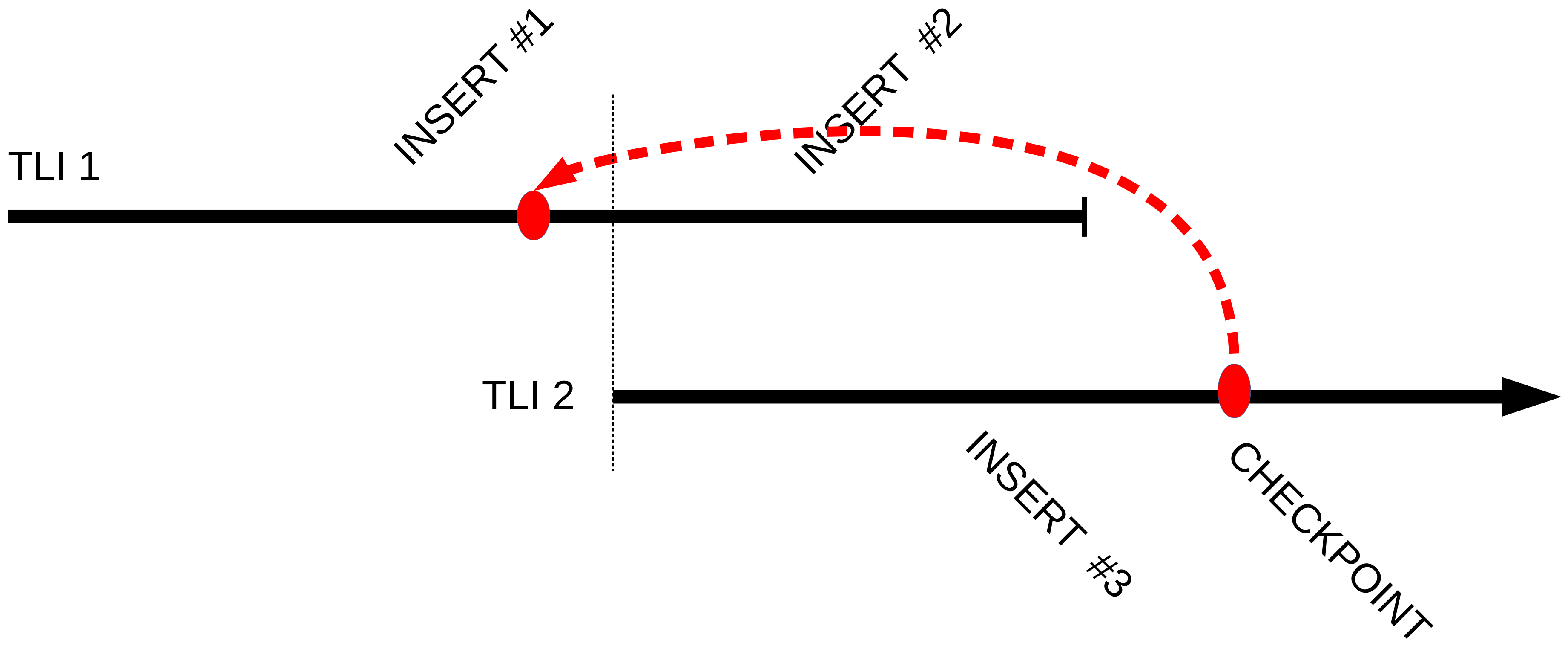
```
backup_label.old (COPY)
base/1/12454_fsm (COPY)
base/1/12454_vm (COPY)
base/1/12456_fsm (COPY)

...

pg_xlog/archive_status/000000010000000000000003.done (COPY)
pg_xlog/archive_status/00000002.history.done (COPY)
postgresql.auto.conf (COPY)
postgresql.conf (COPY)
recovery.done (COPY)
base/12726/12475 (COPY_TAIL)
pg_xlog/archive_status/000000010000000000000003.ready (REMOVE)
pg_xlog/archive_status/000000010000000000000002.00000028.backup.done
(REMOVE)
pg_xlog/archive_status/000000010000000000000001.done (REMOVE)
pg_xlog/000000010000000000000004 (REMOVE)
pg_xlog/000000010000000000000002.00000028.backup (REMOVE)
pg_xlog/000000010000000000000001 (REMOVE)
pg_stat/global.stat (REMOVE)
pg_stat/db_12726.stat (REMOVE)
pg_stat/db_0.stat (REMOVE)
```

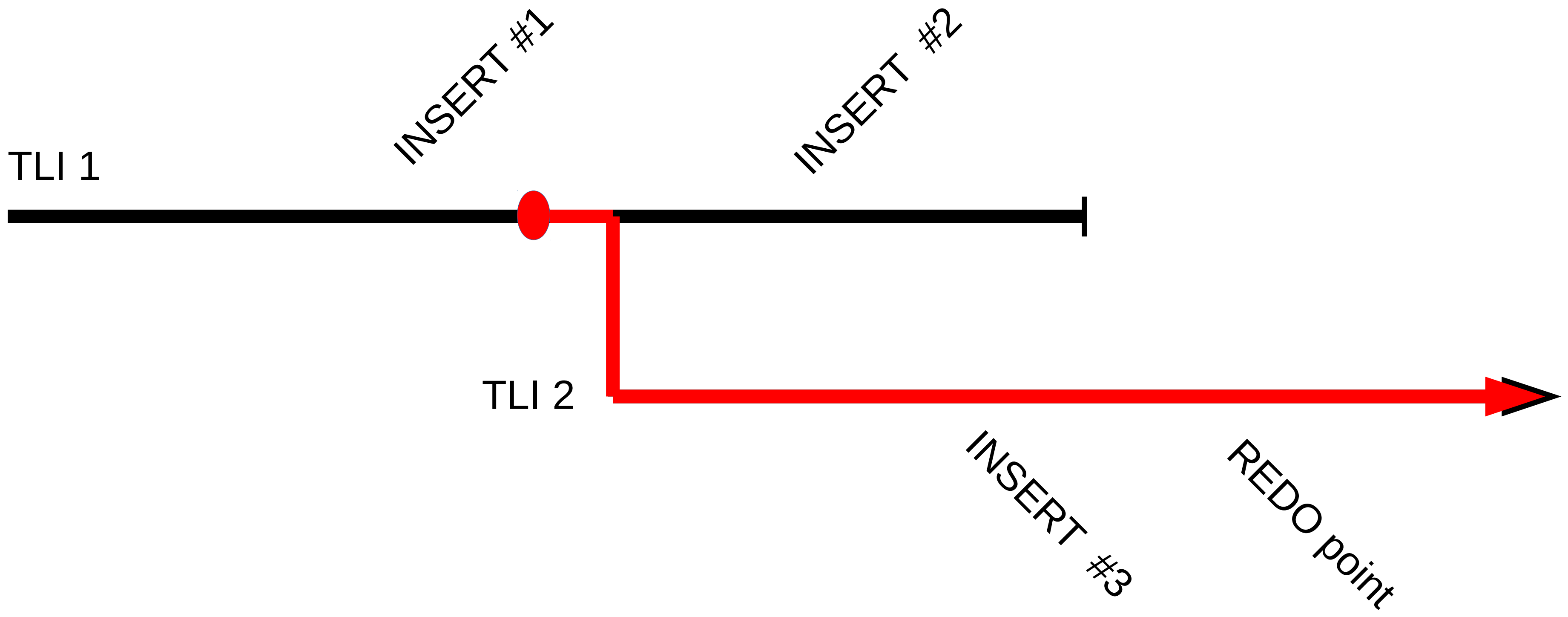
How it works?
4. Reset the control file

- Start recovery from the point of divergence, not some later checkpoint.



How it works?
5. Replay new WAL

- On first startup (not by pg_rewind)



Usage

```
Usage:
  pg_rewind [OPTION]...

Options:
  -D, --target-pgdata=DIRECTORY
                 existing data directory to modify
  --source-pgdata=DIRECTORY
                 source data directory to sync with
  --source-server=CONNSTR
                 source server to sync with

  -P, --progress write progress messages
  -n, --dry-run  stop before modifying anything
  --debug        write a lot of debug messages
  -V, --version  output version information, then
exit
  -?, --help     show this help, then exit
```

Example

```
$ pg_rewind --source-server="host=localhost port=5433
dbname=postgres" --target-pgdata=data-master

The servers diverged at WAL position 0/3000060 on timeline 1.
Rewinding from last common checkpoint at 0/2000060 on timeline 1
Done!
```

Example: --progress

```
$ pg_rewind --progress --source-server="host=localhost
port=5433 dbname=postgres" -target-pgdata=data-master
```

```
connected to remote server
The servers diverged at WAL position 0/3000060 on
timeline 1.
Rewinding from last common checkpoint at 0/2000060 on
timeline 1
reading source file list
reading target file list
reading WAL in target
Need to copy 51 MB (total source directory size is 67
MB)
53071/53071 kB (100%) copied
creating backup label and updating control file
Done!
```

Example: clean failover

```
$ pg_rewind --source-server="host=localhost port=5433
dbname=postgres" --target-pgdata=data-master

The servers diverged at WAL position 0/4000098 on timeline 1.
No rewind required.
```

Caveats

- Must set `wal_log_hints=on` in postgresql.conf
  - before the meteor strikes
  - or use checksums (initdb -k)
- All WAL needs to be available in the `pg_xlog` directories

More use cases

- Synchronize new master to old master, instead of the other way 'round
- Synchronize a second standby after failing over
- Rewind back to an earlier base backup

(haven't tested those, might not work currently)

Design goals

- Safety
  - exit gracefully without modifying target if rewind is not possible
  - dry-run mode
  - unrecognized files are copied in toto
- Ease of use
- Speed
  - Faster than reading through all data

In PostgreSQL 9.5

- Included in PostgreSQL 9.5
- In `src/bin/pg_rewind`
- Changed WAL record format in 9.5
  - to support `pg_rewind` among other things

pg_rewind – for 9.3 and 9.4

Stand-alone versions available for 9.3 and 9.4

- https://github.com/vmware/pg_rewind
- PostgreSQL-licensed

Future development

- Be smarter about what to copy
  - Free Space Maps, Visibility Maps
  - `pg_clog`, `pg_subtrans`, etc.
- When copying a whole file, use checksums to skip unchanged parts
  - like `rsync` does
- Allow using pg_rewind when there have been timeline switches in the target
  - http://www.postgresql.org/message-id/CAPpHfdtaqYGz6JKvx4AdySA_ceqPH7Lki=F1HxUeNNaBRC7Mtw@mail.gmail.com

Thank you!

- Thanks to Michael Paquier and everyone else involved!
- Questions?