Engineering better user experience in
Web applications & Internet architectures

Getting Hot and Streamy with Postgres

Using Postgres' built in replication facilities

May 16th, 2012

Who am I?

Phil Sorber

DBA @ OmniTI

- We do full stack development including databases
- We build large sites
- Our website - http://omniti.com
- Surge - http://omniti.com/surge/2012
- We're hiring - http://omniti.com/is/hiring

Contact me

- phil@omniti.com
- @PhilSorber
- Blog - http://philsorber.blogspot.com

It's just another block in the WAL

Write Ahead Log

- roll-forward recovery aka REDO
- flushed to disk to guarantee commit durability
- sequential writes
- lower cost than flushing page cache

Allows us to do cool things

- Crash recovery
- Binary backups
- Replication!

What's Logged?

wal_level

- minimal (crash recovery)
- archive (disaster recovery + replication)
- hot_standby

almost everything, except...

- unlogged tables (hence the name)
- temporary tables
- hash indexes?! (generally don't use these)

More info

- Attend Amit's talk in MRT 219 tomorrow @ 4:30

PITR (Point In Time Recovery)

Introduced in 8.0

Used for

- On-line Backup - pg_start/stop_backup

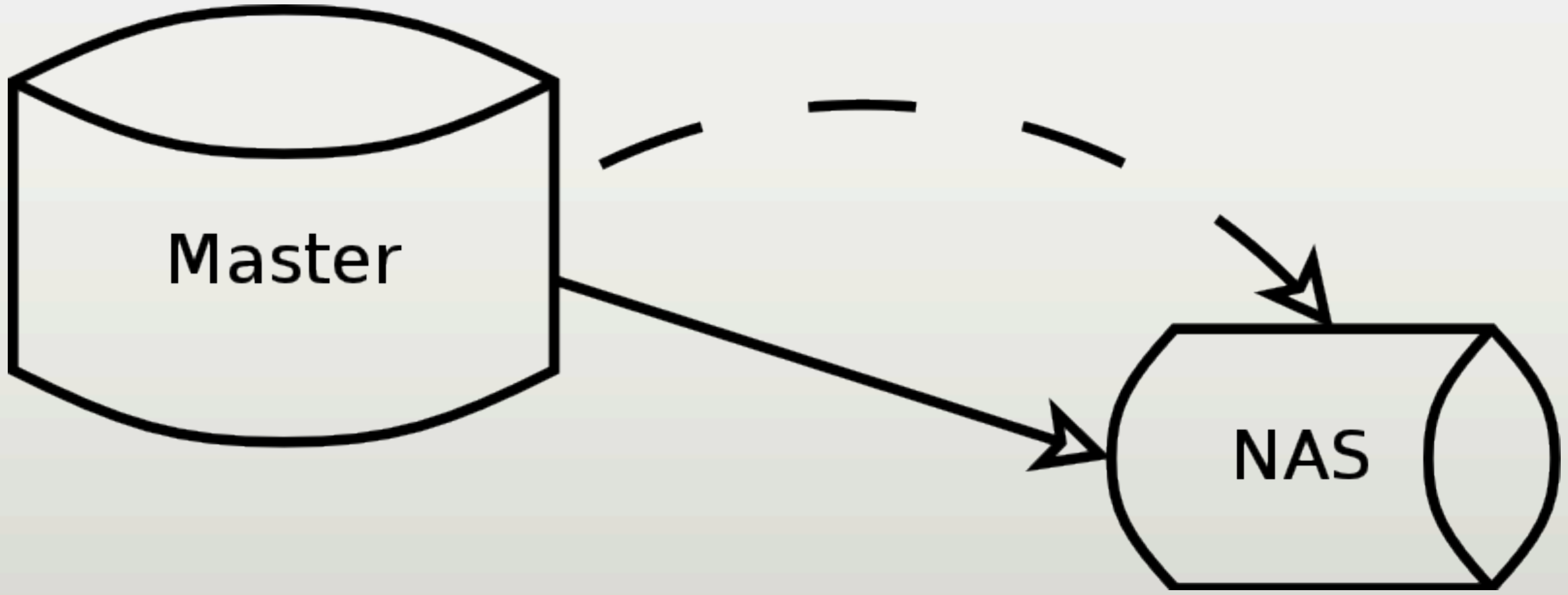
Config Options (Master)

- archive_mode=on (8.3) (RR)
- archive_command
- archive_timeout (8.2)
- wal_level=archive (9.0) (RR)

Tools

- omnipitr-archive
- omnipitr-backup-master

PITR (Point In Time Recovery) *cont.*



PITR

Warm Standby (Log Shipping)

Introduced in 8.2 (technically possible since 8.0)

Used for

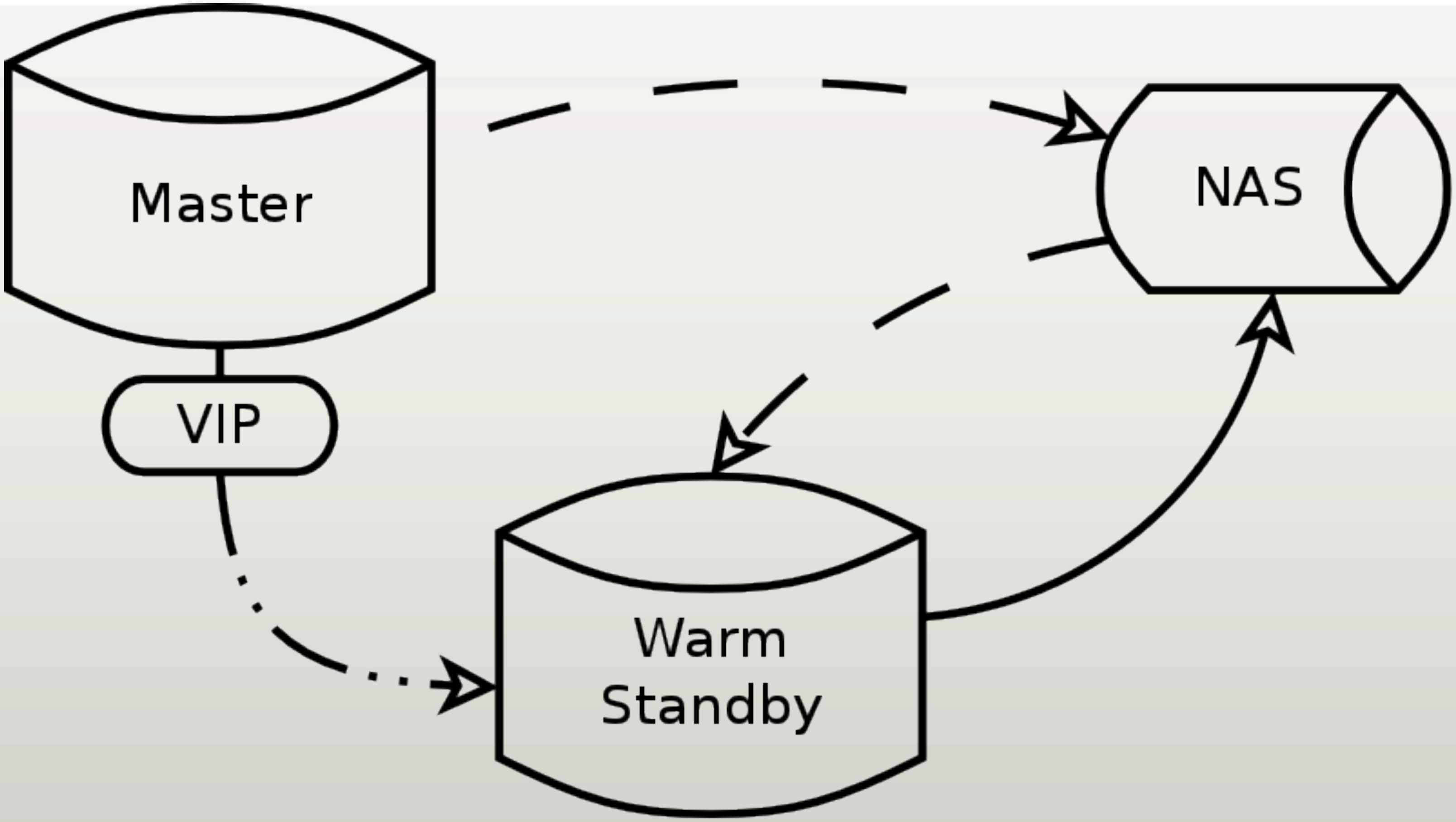
- Failover
- On-line Backup (not officially supported)

Config Option (Slave) - restore_command (RR)

Tools

- omnipitr-synch
- pg_basebackup (9.1)
- omnipitr-restore
- pg_standby (8.3)
- omnipitr-backup-slave

Warm Standby (Log Shipping) *cont.*



Warm Standby

Streaming Replication

Introduced in 9.0

Used for

- Near real-time replication

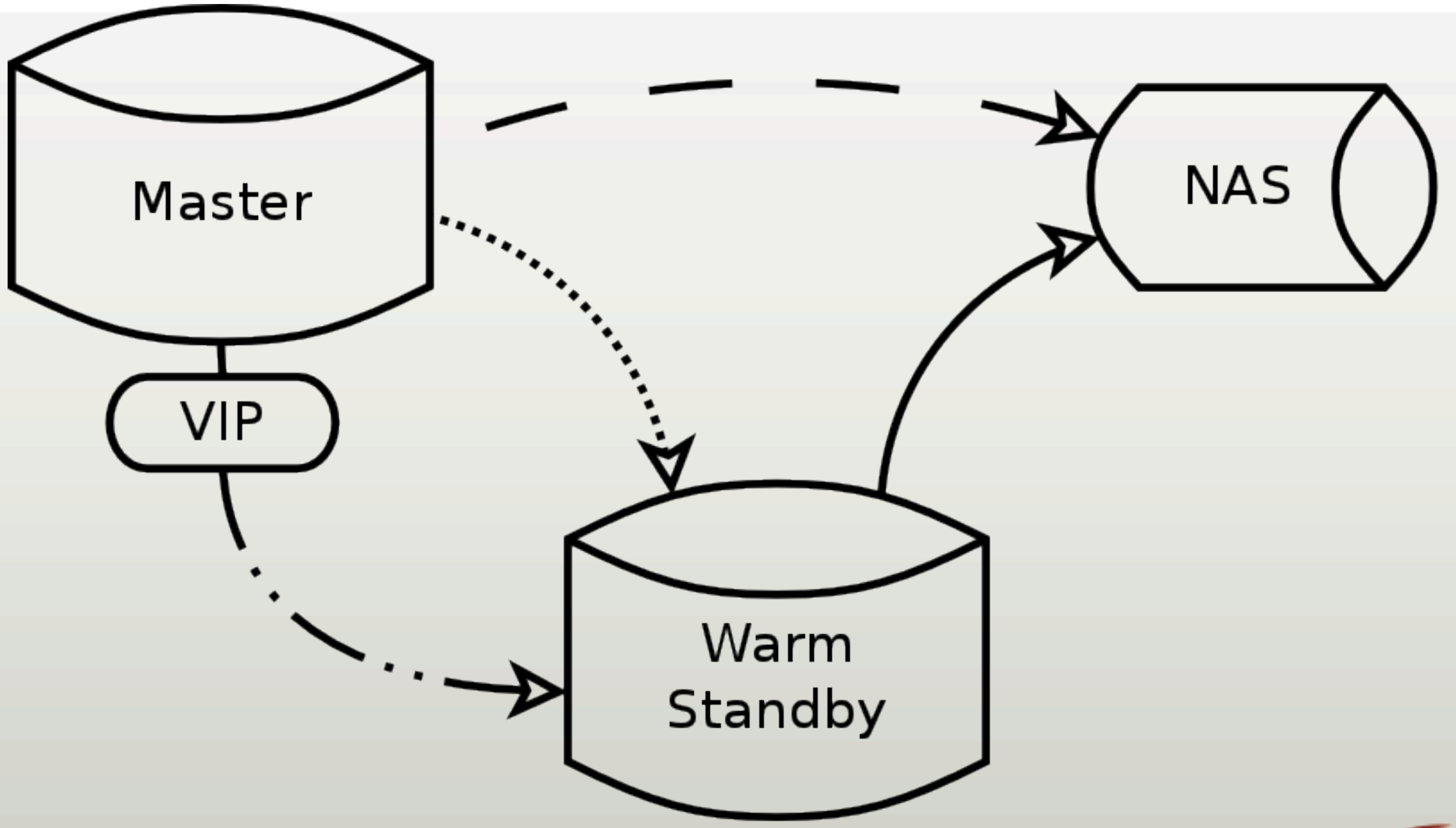
Benefits over Log Shipping

- Less hassle in setting up archive and restore commands
- Less lag due to low activity (archive_timeout)

Disadvantages

- Master does not keep WAL segments indefinitely
- More potential load on the master

Streaming Replication *cont.*



Streaming Warm Standby

Streaming Replication II

WAL kept in sync

- wal_sender reads from WAL on disk (usually cached)
- wal_receiver writes to WAL on disk

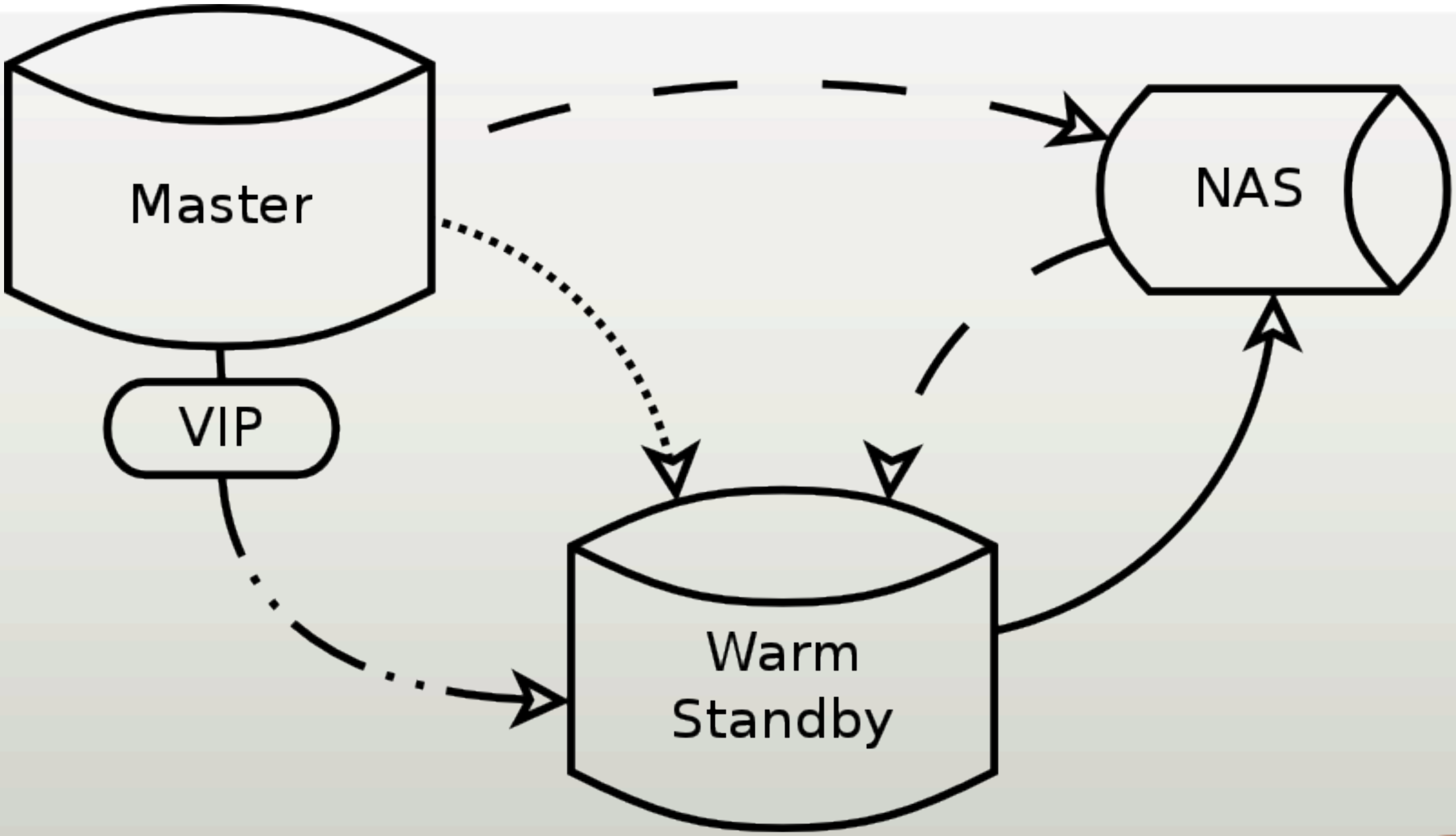
Hybrid Mode

- Streaming Replication & Log Shipping
- Lets the slaves sustain more down time
- Less IO on master used for finding "old" WAL
- Still have to deal with the hassle of log shipping

pg_hba.conf

- replication pseudo database
- replication privilege (9.1)

Streaming Replication II *cont.*



Hybrid Streaming Warm Standby

Streaming Replication III

Config Options

Master

- max_wal_senders (RR)
- wal_keep_segments
- wal_level=archive (RR)
- archive_timeout=0

Streaming Replication IV

Config Options

Slave (recovery.conf) (RR)

- standby_mode=on
- primary_conninfo
  - 'host=192.168.1.50 port=5432 user=foo password=foopass'
- trigger_file
- archive_cleanup_command
- recovery_target_timeline (8.0,9.1)
- restore_command

Streaming Replication V

Tools

- pg_basebackup
- omnipitr-synch
- pg_archivecleanup
- omnipitr-cleanup

Hot Standby

Introduced in 9.0

Independent of Streaming Replication

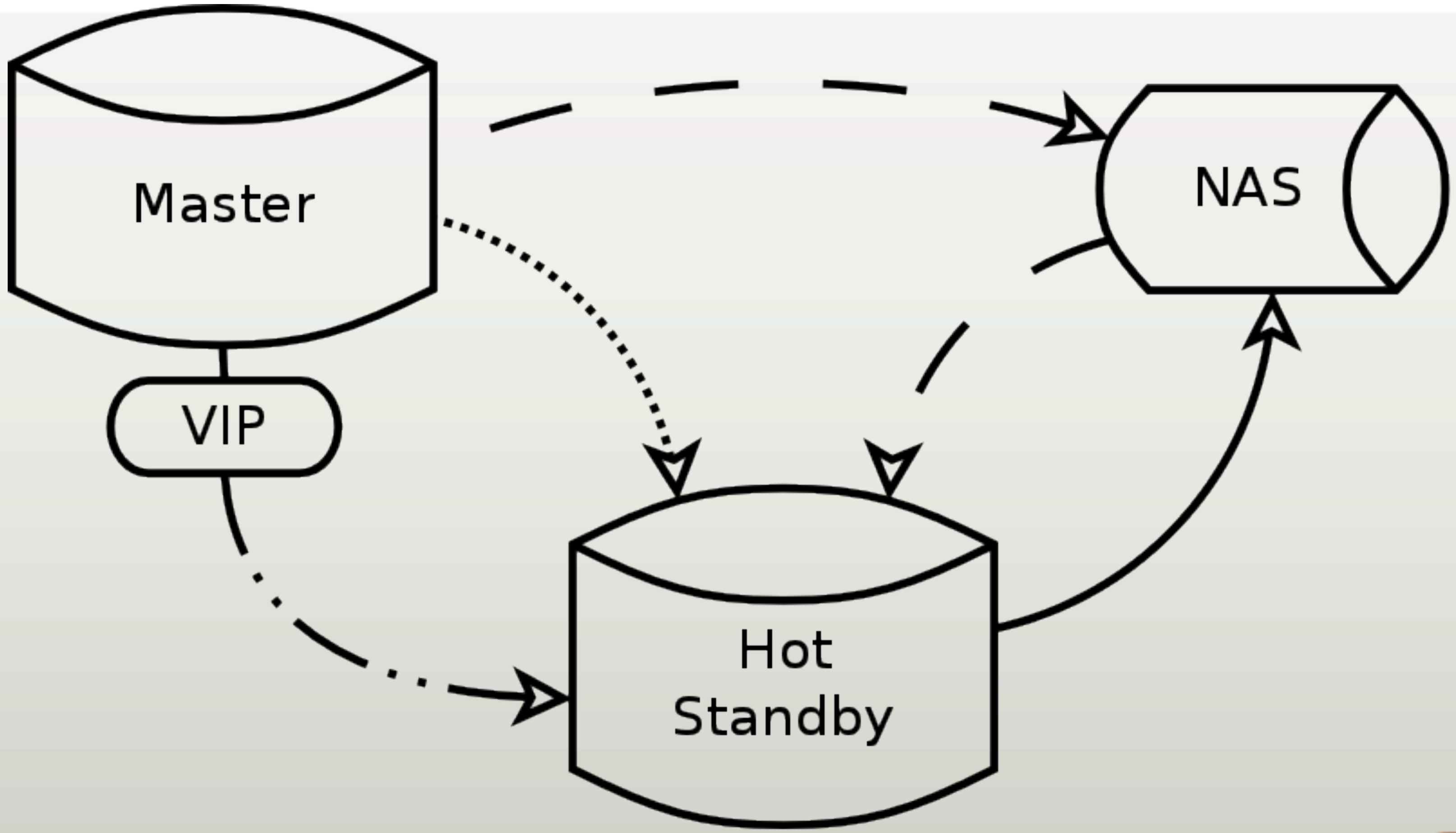
Read only queries

- SELECT (no functions that alter)
- pg_dump

Conflicts

- Soft (I/O)
- Hard (Updates)

Hot Standby *cont.*



Hybrid Streaming Hot Standby

Hot Standby II

Config Options

Master

- wal_level=hot_standby (RR)
- vacuum_defer_cleanup_age

Slave

- hot_standby=on (RR)
- hot_standby_feedback
- max_standby_archive_delay
- max_standby_streaming_delay

Synchronous Streaming Replication

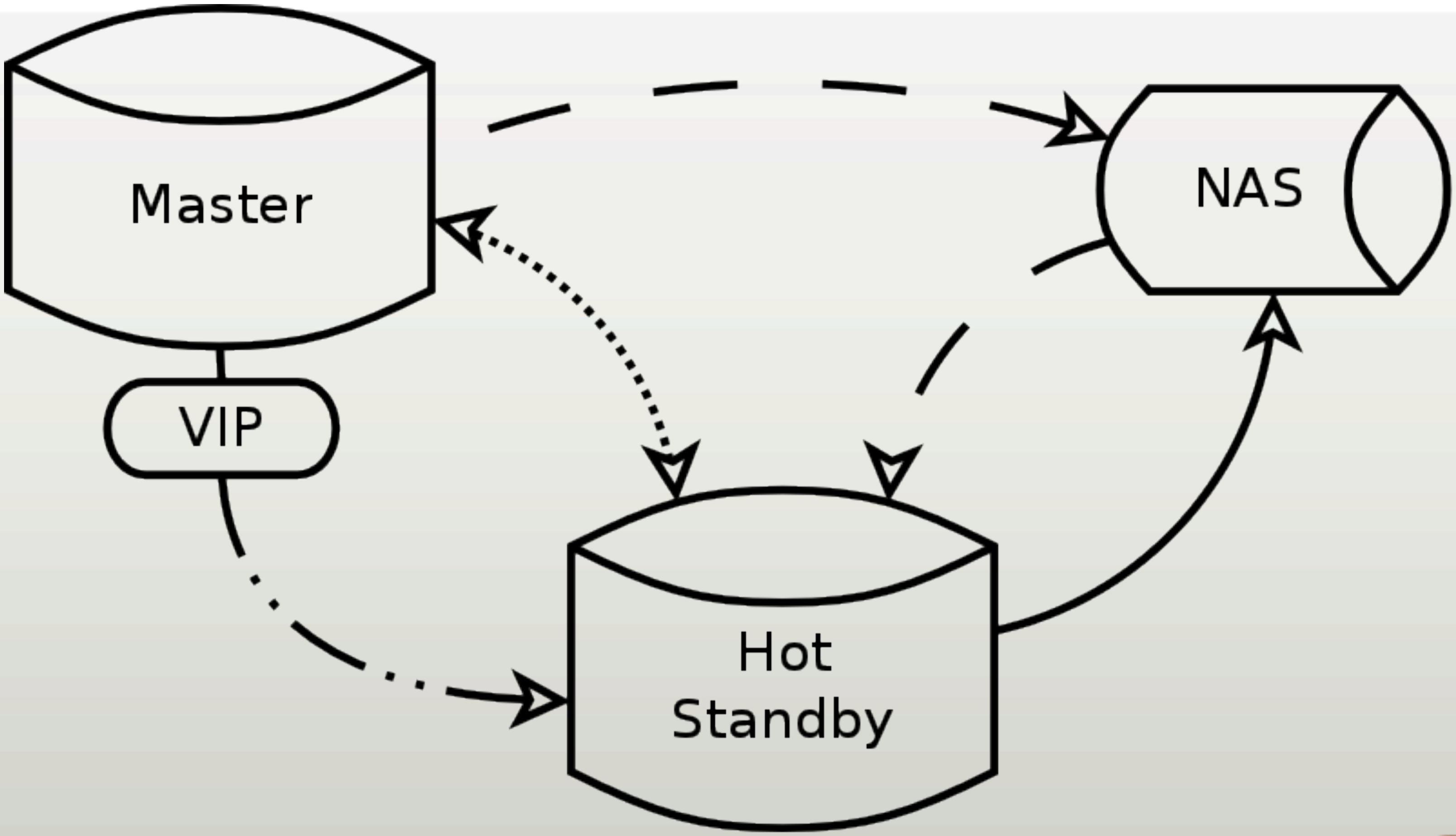
Introduced in 9.1

Also known as 2-safe replication

Commit does not return until data written to WAL and flushed to disk of both primary and standby

Read only transactions and rollbacks do not wait

Synchronous Streaming Replication *cont.*



Hybrid Synchronous Streaming Hot Standby

Synchronous Replication II

Config Options

synchronous_commit = on (default)

- Can be set per transaction

synchronous_standby_names

- Only first entry must respond
- Failover list
- Set to empty for asynchronous behavior

Synchronous Replication III

Pro's

- Greater durability

Con's

- Latency
- Contention
- Points of failure

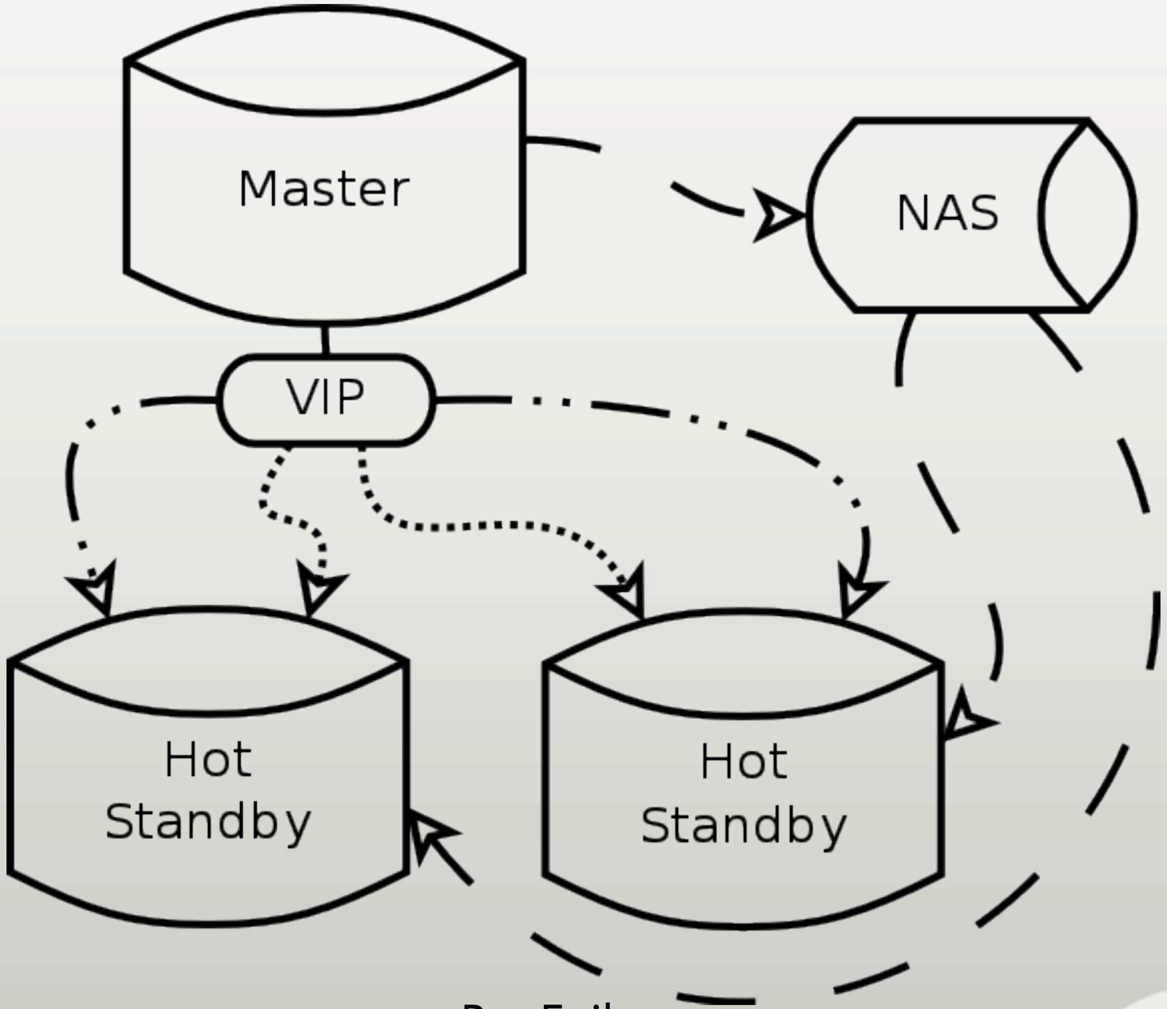
Multiple Standby Failover

Timeline Changes

Missing and extra transactions

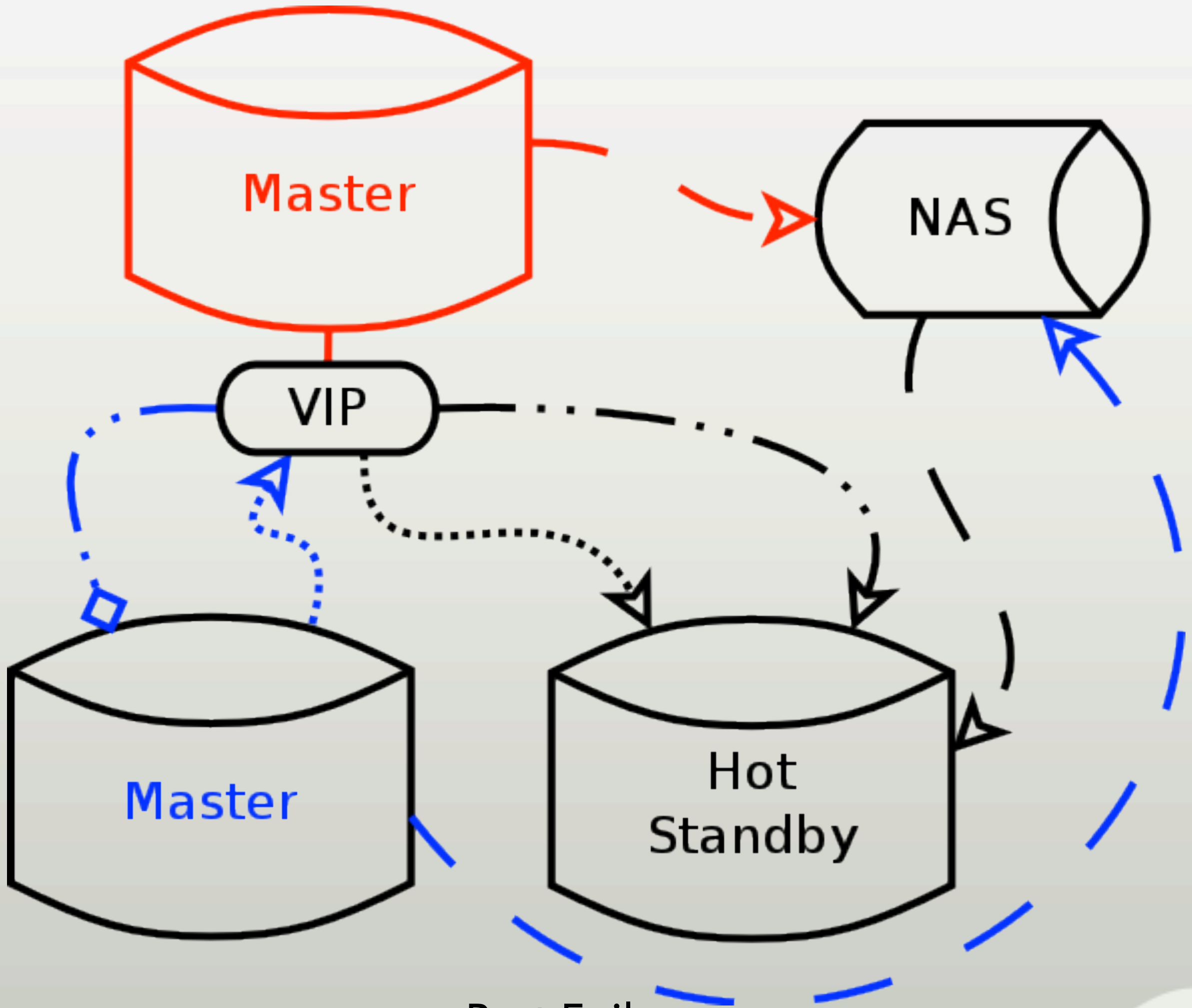
History Files

Multiple Standby Failover *cont.*



Pre Failover

Multiple Standby Failover *cont.*



Post Failover

Cascading Replication

Coming in 9.2

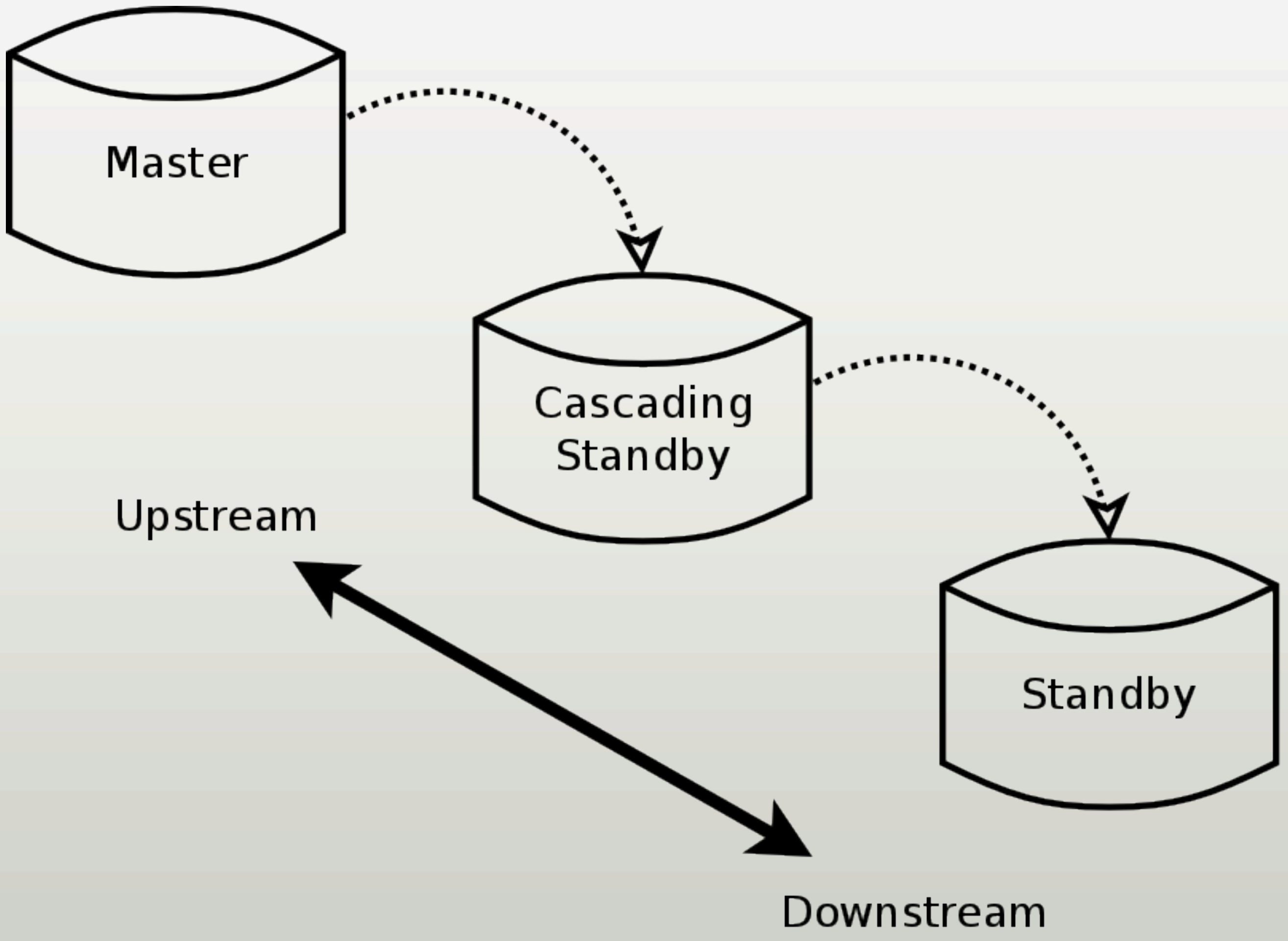
Same options as regular streaming

Asynchronous only

Hot standby feedback propagates upstream

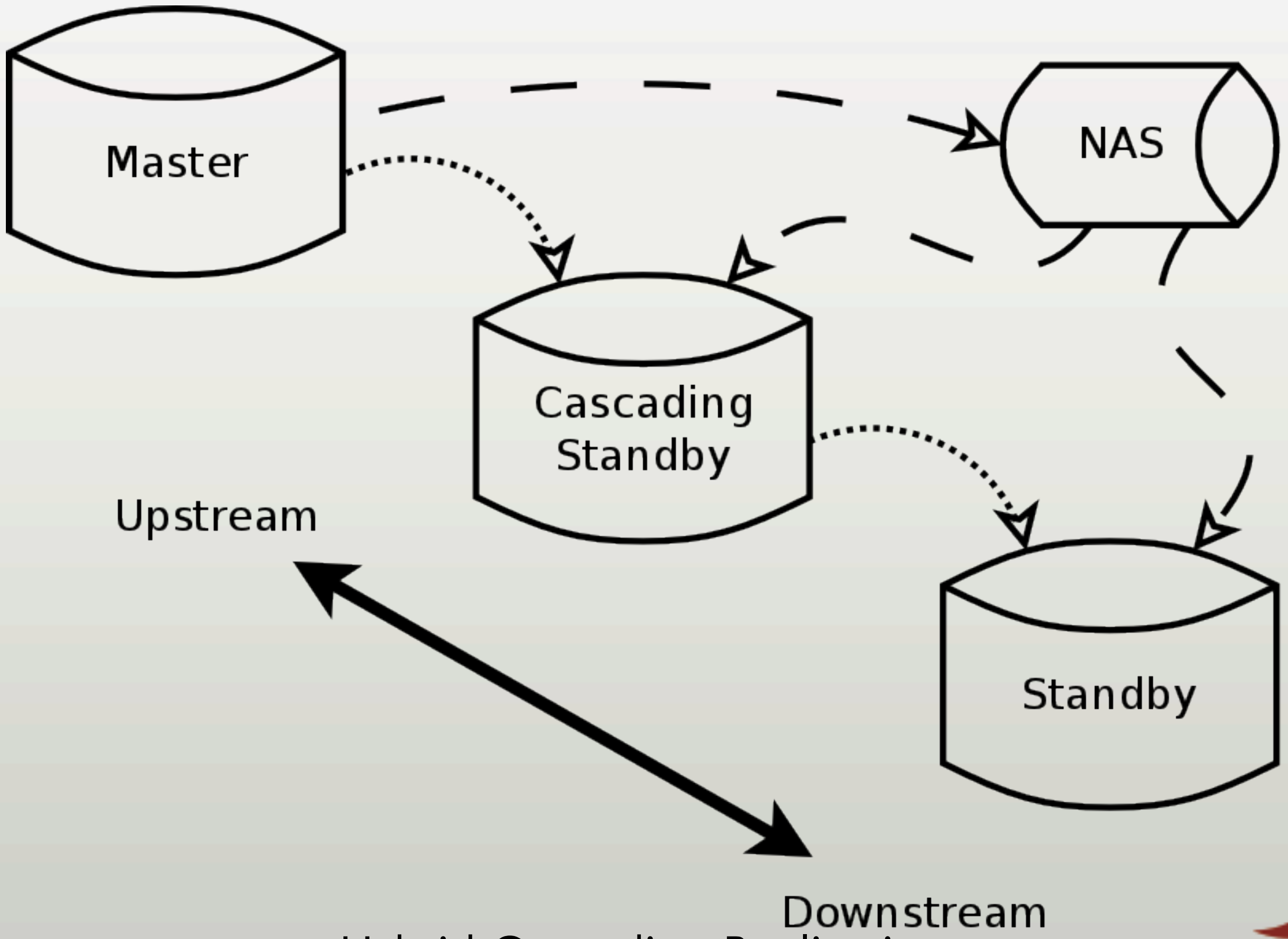
Promoting cascading standby terminates downstream replication

Cascading Replication *cont.*



Cascading Replication

Cascading Replication *cont.*



Hybrid Cascading Replication

Questions?

Thank You!

http://omniti.com

+1 443 325 1357