

# HAST

Highly Available Storage  
for FreeBSD

Paweł Jakub Dawidek  
<[pjd@FreeBSD.org](mailto:pjd@FreeBSD.org)>

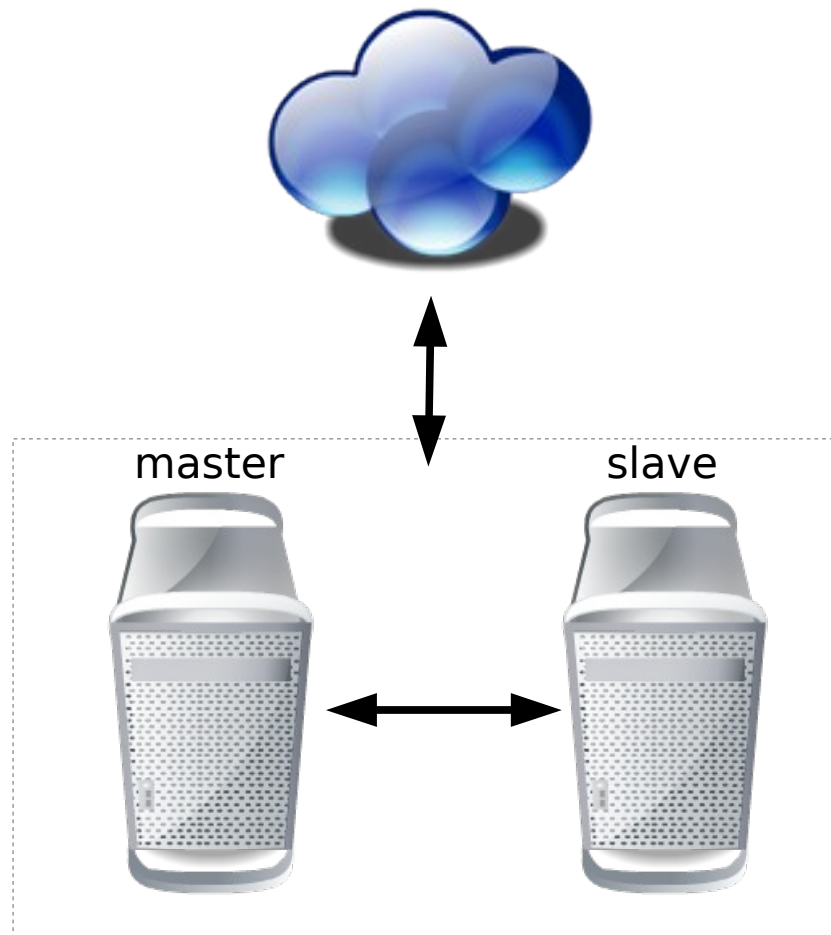
# High Availability

- 100% availability (well, almost)
- no single point of failure (redundancy)
- active-passive mode of operation for components that can't work in active-active mode



# High Availability

Big picture



# FreeBSD and High Availability

## Role selection

- CARP (carp(4))
- heartbeat (ports/sysutils/heartbeat/)
- UCARP (ports/net/ucarp/)



# FreeBSD and High Availability

## Keeping data in sync

- pfsync (pfsync(4) keeps PF states in sync)
- rsync (ports/net/rsync)
- zfs send/recv
- application-specific methods (eg. mysql replication)



# Highly Available Data

## File systems

- network file systems (eg. NFS, SMB)
- shared disk file systems (eg. Red Hat's GFS)
- distributed file systems (eg. Isilon's OneFS)



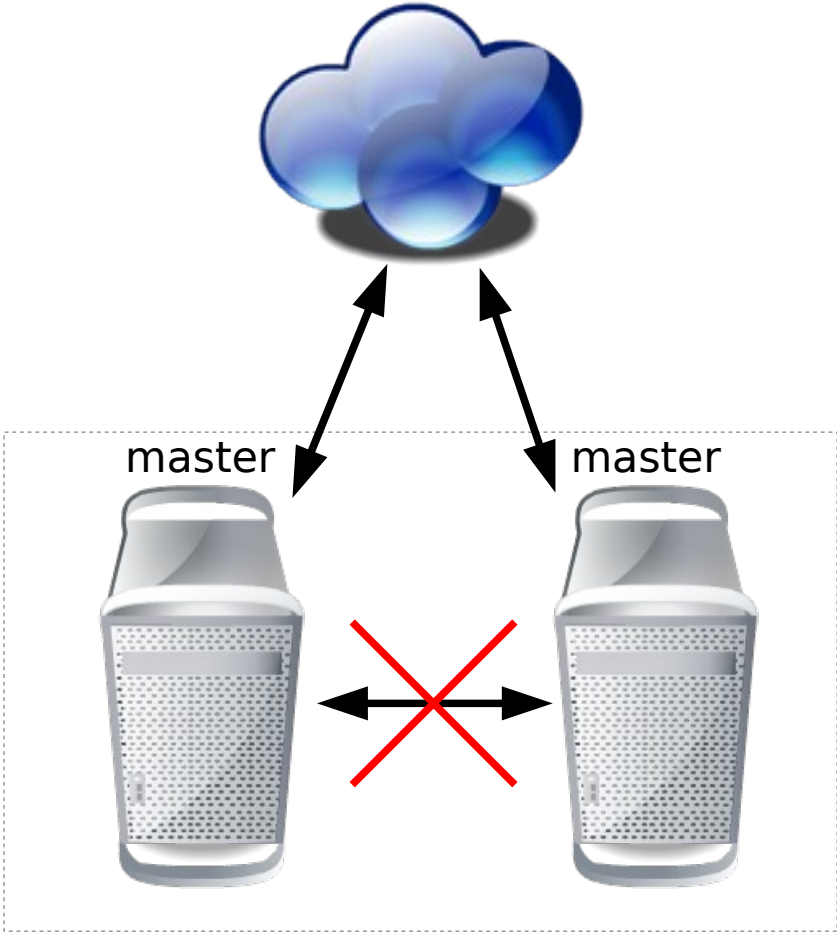
# Highly Available Data

## Block level devices

- SAN
- DRBD for Linux
- HAST for FreeBSD

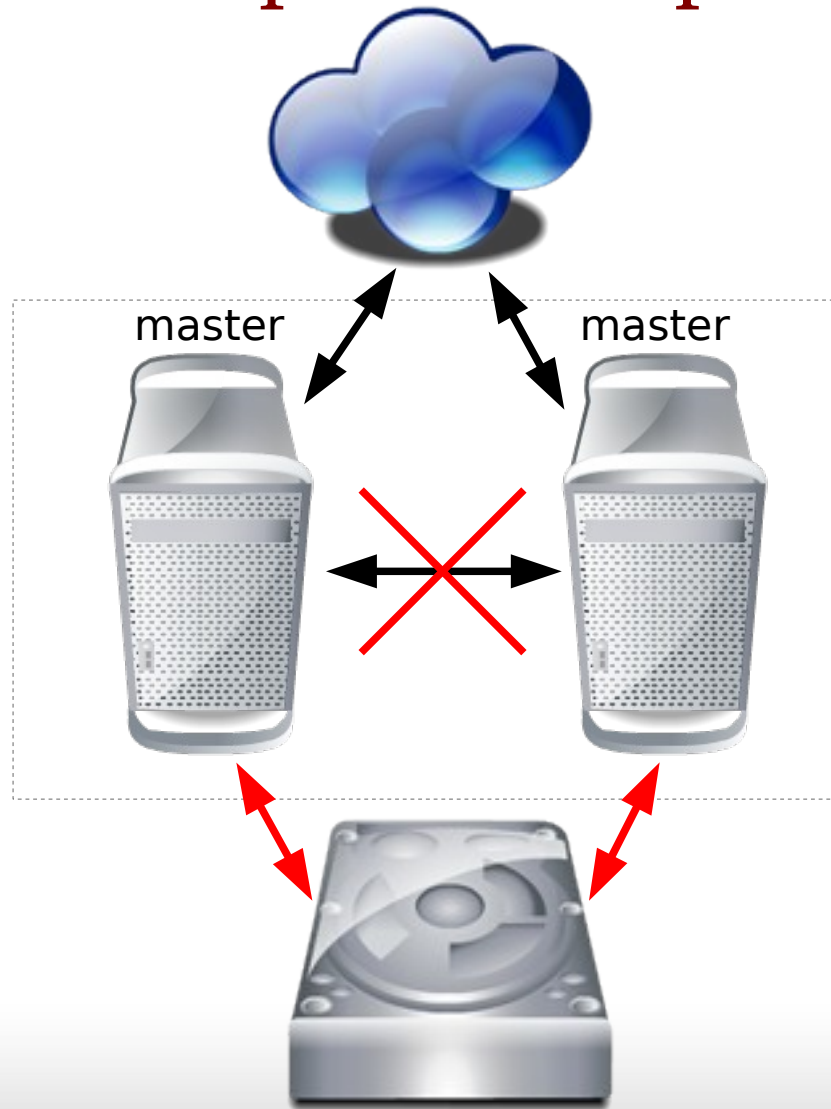


# Split-brain



# Problem with SAN

## Data corruption on split-brain



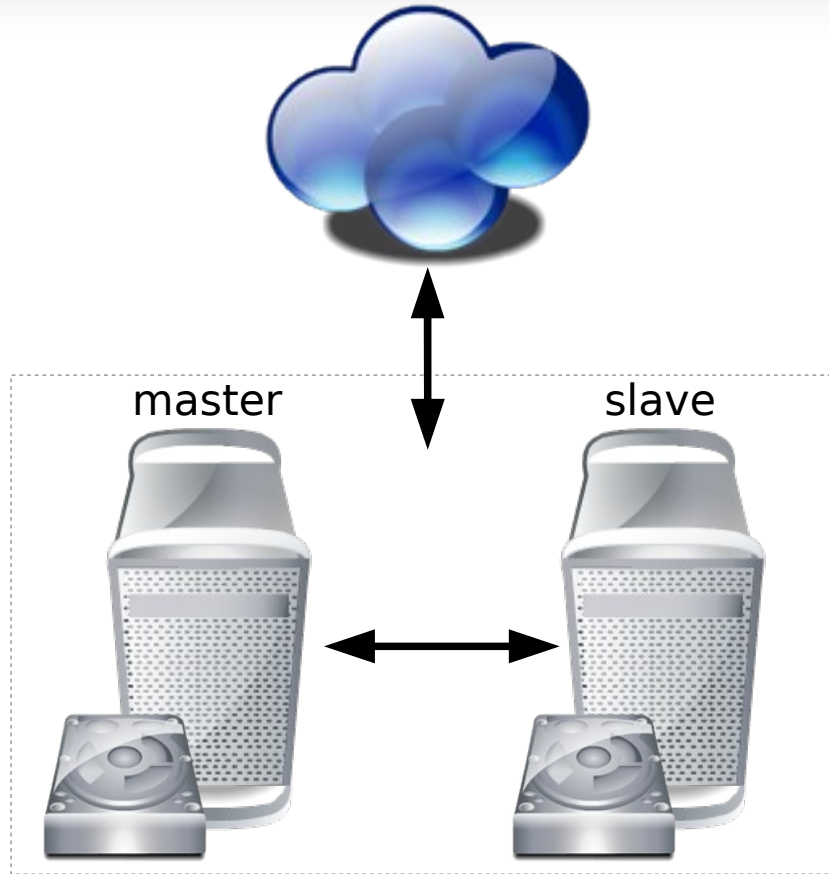
# HAST

## The challenge

The most common reason for HA clusters outage are problems in HA implementations



# HAST model



# HAST

- synchronous data replication
- operates on block-level (file system independent)
- reuses GEOM Gate class
- `hastd(8)` daemon doing the work
- `hastctl(8)` control utility



# HAST

- doesn't decide about its role
- limited to two nodes: one master and one slave
- fast recovery (synchronize only data modified during outage)
- split-brain detection



# HAST: How to write?

1. write data locally and send data to slave
2. report success



# HAST: How to write?

1. write data locally and send data to slave
2. report success

**WRONG!**



# HAST: How to write?

1. mark extent as dirty
2. write data locally and send data to slave
3. slave ack on data write
4. mark extent as clean
5. report success



# HAST: How to write?

1. mark extent as dirty
2. write data locally and send data to slave
3. slave ack on data write
4. mark extent as clean
5. report success

Correct, but slooow...

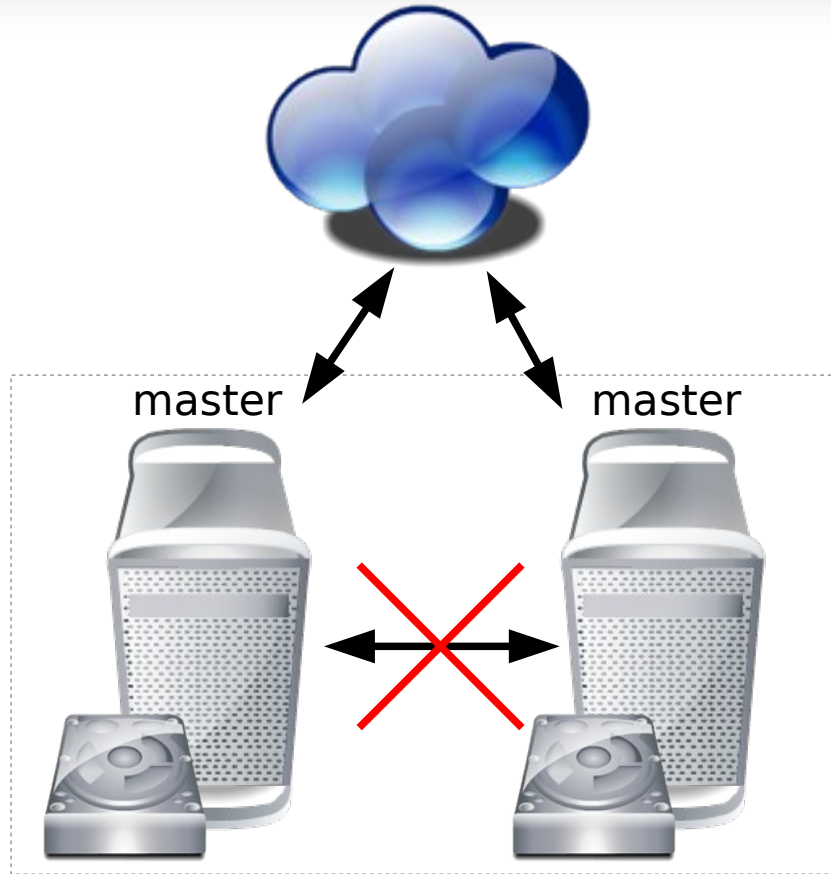


# HAST: How to write?

1. mark extent as dirty
2. write data locally and send data to slave
3. slave ack on data receive, not on write
4. do **not** mark extent as clean
5. report success



# HAST and split-brain



Questions?