PostgreSQL on ZFS

- Replication, Backup, Human-disaster Recovery, and More...

- Keith Paskett
- Keith@Paskett.org
- @klpaskett

Agenda

- Introduction
- Defining the Problem (Opportunity)
- ZFS Advantages for your Postgres instance
- What ZFS is and where you can get it
- ZFS snapshots—where the magic happens
- Cloning for Validation, Testing and Recovery
- Adding ZFS to Log/Streaming Replication
- Beyond the database
- Summary and Demo


Trouble lurks
Where you least expect it

Backup & Recovery Disconnect

- **Less likely disaster scenarios**
  - Server failure
  - Multiple drive failures in a raid array
  - Data center flattened by (choose your disaster)
- **Common 'human' disaster scenarios**
  - Dropped table
  - Deleted data
  - Altered data
- **Many backup solutions focus on the least likely disaster**
- **ZFS helps protect against more common disasters**

I Wish I Could...

- Test an upgrade script on a multi-terabyte database without having to set up a separate server with terabytes of storage
- Quickly roll back from an upgrade if things go badly
- Have point-in-time access to a large database without having to do a restore then replay a week's worth of transaction logs

ZFS Advantages for Databases

- Fast efficient replication (one-way periodic update)
- Low/no-impact snapshots
- Read/write access to snapshots via clones
- Pool physical devices
- Send/receive snapshots
- Bidirectional incremental send/receive
- Solid state cache drives
- Upgrade to larger physical drives with zero downtime
- Continuous integrity checking and automatic repair
- Built in compression—can actually improve performance

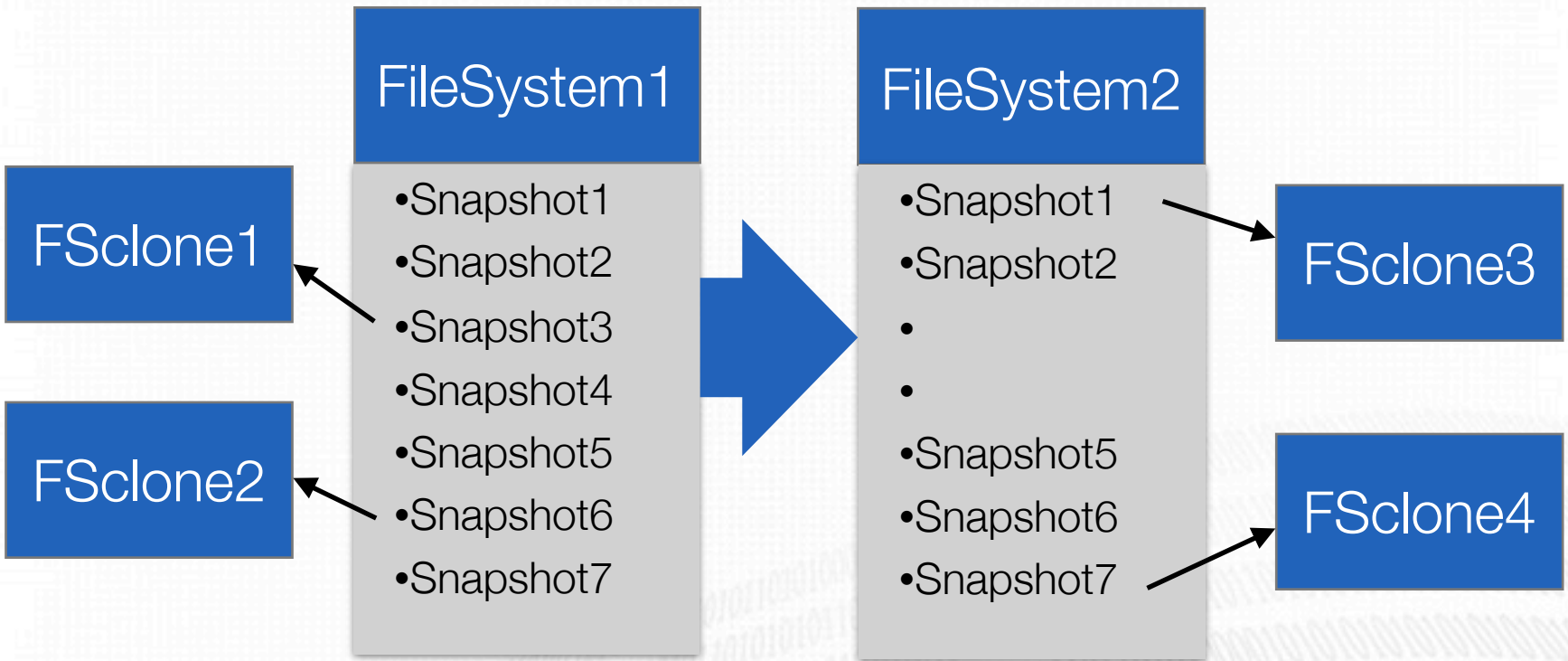
What

- ZFS ⇒ Zettabyte File System.
- Not just a file system. It's a storage infrastructure.
- Transactional, Copy-on-write
- Always consistent on disk.
- Fully checksummed.
- Loves memory and SSD, and it knows how to use them.
- It's easy! There are only two commands.

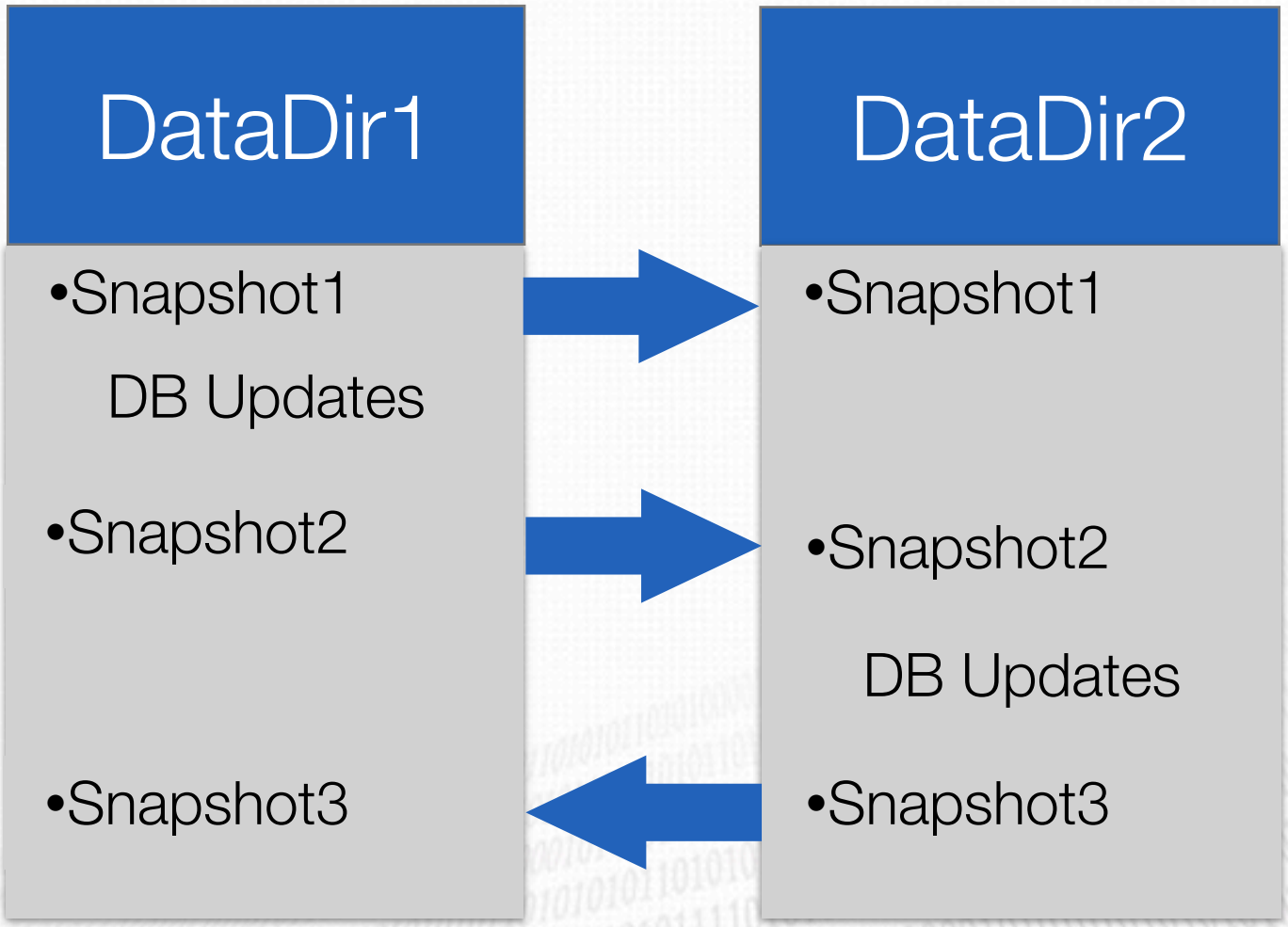
Where

- **IllumOS**
  - OmniOS
  - OpenIndiana
  - SmartOS (No OS drive. Really.)
- **FreeBSD**
- **Linux - Yes it is production ready**
- **OS X**
- **That other OS that helps fund the purchase of tropical islands**

ZFS - It's all about the snapshots



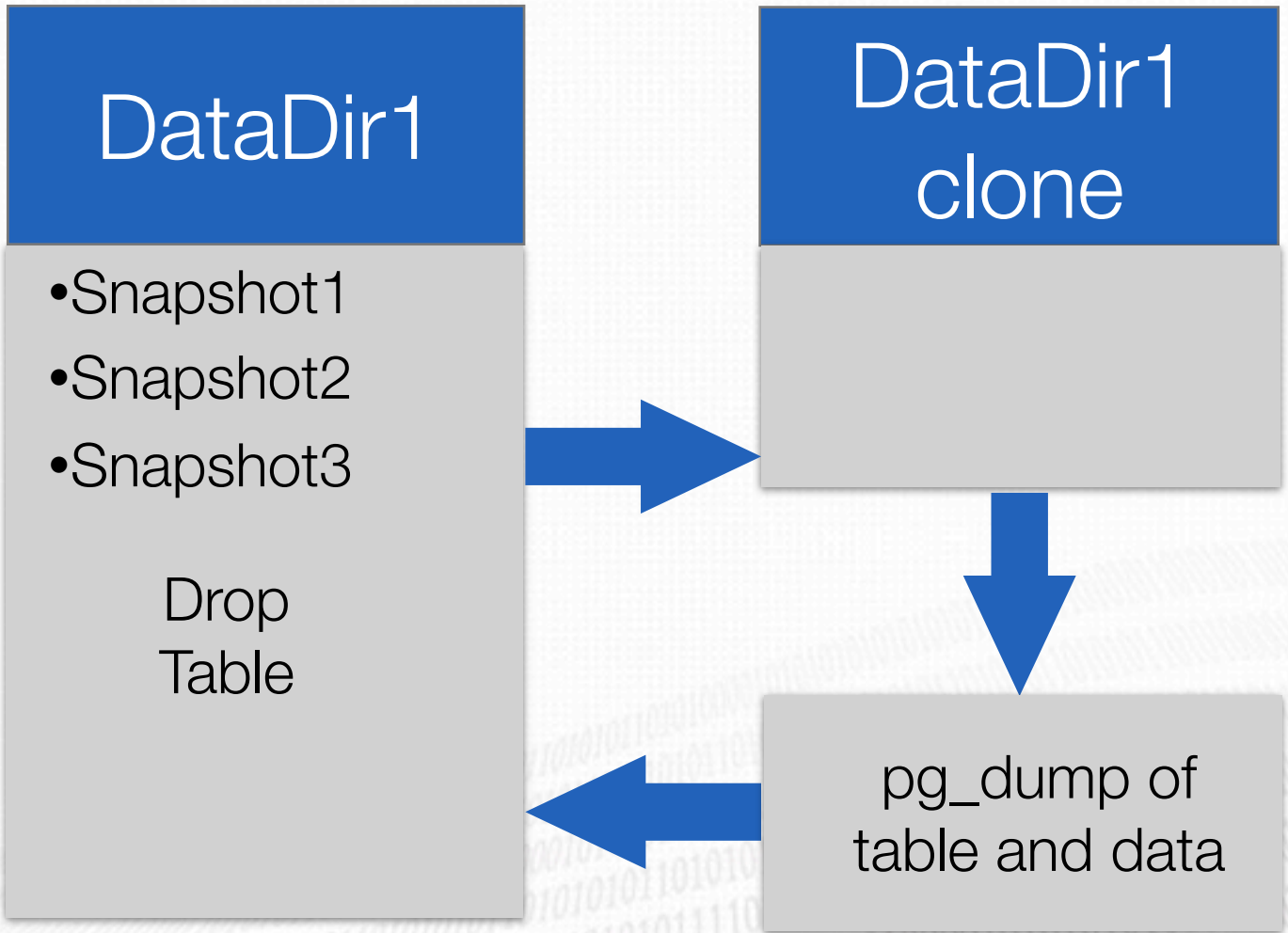
Send & Receive



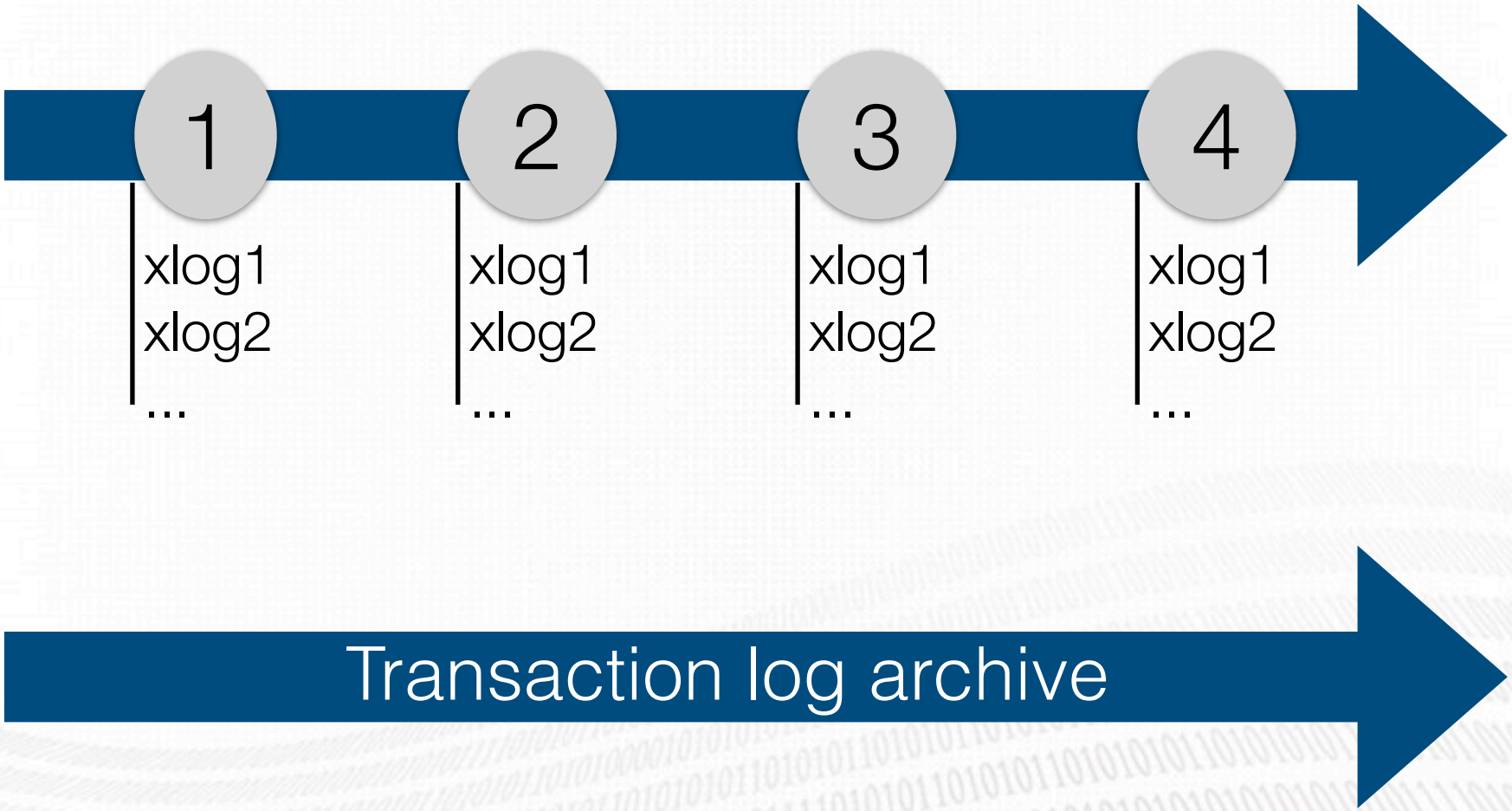
Accessing Snapshot Contents

- .zfs directory
  - 'cd .zfs/snapshot/snapname' from base ZFS directory
  - 'ls -a' in base directory won't show the .zfs directory
  - Snapshot directories are read only
  - Tape backup of snapshot directory will be consistent
- Create a clone from a snapshot
  - zfs clone fsname@snapshotname clonename
  - Clones are writable
  - Only take additional disk space as data is changed
  - Simply awesome for replicating a TB database in seconds

Clone & Recover



Snapshots & Transaction Logs



Snapshot Frequency

- Tune to your needs
- My preferences for transactional databases
  - Every 10-20 minutes, keep for 9 hours
  - Daily, keep for 10 days
  - Weekly, keep for 8 weeks
- cronable scripts (examples in the VM)
  - snapshots.sh creates snapshots with hour, day, or week name embeded
  - snapshot-cleanup.sh removes snapshots older than a specified time

ZFS Compression

- Can improve performance especially for streaming data sets
- Turning on only compresses subsequent writes
- Check out test results from citusdata
  - http://citusdata.com/blog/64-zfs-compression
- Commands
  - zfs set compression=on fs/name (LZJB)
  - zfs set compression=gzip fs/name
  - zfs set compression=gzip-9 fs/name

Beyond the Database

- For Solaris and its derivatives
  - Replicate the OS drive
  - Simple bare-metal restore
- OS files with metadata in the DB
  - Snapshot both for a full consistent test/recovery environment
- Using Postgres FDW with non-DB files
  - Same story—snapshots, replication, and clones

Summary & Caveats

- Great for quick duplication of very large databases
- Very efficient for periodic updates of replicas
- Combine snapshots with transaction log archives for faster access to point-in-time data
- Disk space is only freed when files and snapshots are deleted
- Have to shut down the 'receiving' DB during the update
- If your backup software finds the .zfs directory you will experience anti-deduplication
- Recursive snapshots vs snapshots on separate file systems

PostgreSQL and ZFS

They may not have been designed to work together, but they do nonetheless

VM Setup

- **Before you import the VM into VirtualBox**
  - VirtualBox 4.2.12 with VirtualBox Extension Pack
  - VT-x/AMD-V acceleration (may need to enable in BIOS)
  - Enough memory to allocate 2GB to the VM
  - PuTTY or other terminal to access the VM's zones via ssh
  - Add a Host-Only Ethernet adapter
    with an IPv4 address of 192.168.68.1 and a netmask of 255.255.255.0
- **Import the OmniOS VM and start it up**
  - More detailed instructions and the VM are at:
    http://static4.usurf.usu.edu/vms
  - You may get a message about modifying the network settings on import
- **From your host's shell or PuTTY connect to 192.168.68.10**
  - log in as user: admin password: nimda

VM Exploration

- **Root privileges**
  - pfexec executes the following command with root privileges
  - pfbash assumes root status
- **List, start, access and stop zones**
  - zoneadm list -cv
  - pfexec bin/start-zones.sh
  - ssh to admin@192.168.68.(11,12, 13, or 14) password nimda
  - pfexec bin/stop-zones.sh
- **List and explore zfs filesystems**
  - zfs list
  - zfs list -t all
  - zfs list -t all -r rpool/space
  - zfs get all rpool/space

ZFS Practice

- **Create a filesystem and add some content**
  - zfs create rpool/myfs; cp -r Downloads /rpool/myfs/
- **Create a snapshot and replicate from it**
  - zfs snapshot rpool/myfs@rep1
  - zfs send rpool/myfs@rep1 | zfs recv rpool/myfs-copy@rep1
- **Add more content**
  - cp PGrep* /rpool/myfs/
- **Explore .zfs directories**
  - cd /rpool/myfs/.zfs/snapshot/rep1
- **Destroy the file systems**
  - zfs destroy -r rpool/myfs
  - zfs destroy -r rpool/myfs-copy

The OmniOS Virtual Machine

Global zone

- The /rpool/space/shared1 and /rpool/space/shared2 directories in the global zone are mounted as /shared1 and /shared2 in zones 1-4
- Each zone is its own isolated OS environment which can be rebooted independently. There is a postgres instance in each zone configured for replication with another zone.
- Zones are connected via a host-only network using the 192.168.68.x subnet

Zone1
File-based
Master

Zone2
File-based
Slave

Zone3
Streaming
Master

Zone4
Streaming
Slave

PostgreSQL Built-in Replication Demos

- **Replication based on log-file shipping**
  - bin/log-shipping-launch.sh
    starts replication from zone1 to zone2
  - Add tables to PG in zone1 and veiw in zone2's PG
  - Monitor server log on zone2 to see log file ingest
    less +F /var/postgres/server.log
  - bin/log-shipping-reset.sh to stop and reset
- **Streaming replication**
  - bin/stream-shipping-launch.sh
    starts replication from zone3 to zone4
  - Add tables to PG in zone3 and veiw in zone4's PG
  - bin/log-shipping-reset.sh to stop and reset