CRUNCHY DATA SOLUTIONS, INC.

Efficiently Backing up Terabytes of Data with PgBackRest

David Steele

New York City PostgreSQL User Group
April 21, 2015

About the Speaker

- Senior Data Architect at Crunchy Data Solutions, the PostgreSQL company for secure enterprises.
- Actively developing with PostgreSQL since 1999.

Agenda

- Why Backup?
- How to Backup?
- PgBackRest Design
- Performance
- Living Backups
- Demo

Why Backup?

- Hardware Failure
  - No amount of redundancy can prevent it
- Replication
  - WAL archive for when async streaming gets behind
  - Sync replica from backup instead of master
- Corruption
  - Can be caused by hardware or software
  - Detection is of course a challenge
- Accidents
  - So you dropped a table?
  - Deleted your most important account?

Why Backup? - Continued

- Development
  - No more realistic data than production!
  - May not be practical due to size / privacy issues
- Reporting
  - Use backups to standup an independent reporting server
- Forensics
  - Recover important data that was removed on purpose

How to Backup?

- pg_dump
- Base Backup
- Manual
- ThirdParty
    - OmniPITR
    - Barman
    - Etc.

- PgBackRest?

PgBackRest Design - Say No to Rsync

- Rsync powers many database backup solutions but it has some serious limitations:
  - Single-threaded
  - One second timestamp resolution
  - No destination compression
  - Incremental backups require previous backup to be uncompressed.
- PgBackRest does not use rsync, tar or any other tools of that type:
  - Protocol supports local/remote operation
  - Solves timestamp resolution issue

PgBackRest Design - Features

- Compression is performed and checksums are calculated in-stream
- Asynchronous compression and transfer for WAL archiving
- Remote or local operation
- Threading for parallel compression and transfer
- Full, differential, and incremental support
- Backup and archive expiration policies
- Resumable backups
- Optional hard-linking of diff and incr backups
- Works with PostgreSQL >= 8.3

PgBackRest Design - Backup Structure

- Clear simple structure
- Plaintext manifest
- Valid Postgres data directory
- Postgres can be started in the backup directory if no compression is used
- Archive logs needed to make the backup consistent can optionally be copied to pg_xlog (no need to used recovery.conf or have access to the archive logs)

PgBackRest Performance vs Rsync

Parameters	PgBackRest	Rsync
threads: 1 network compression: l3 destination compression: none	141.0 seconds	124.5 seconds **.13X Faster**
threads: 2 network compression: l3 destination compression: none	84.1 seconds **1.48X Faster** (than 1 rsync thread)	**N/A**
threads: 1 network compression: l6 destination compression: l6	334.4 seconds **1.52X Faster**	510.3 seconds
threads: 2 network compression: l6 destination compression: l6	174.4 seconds **2.93X Faster** (than 1 rsync thread)	**N/A**

Living Backups

- Find a way to use your backups:
  - Syncing / New Replicas
  - Offline reporting
  - Offline data archiving
  - Development
- Unused code paths will not work when you need them unless they are tested:
  - Regularly scheduled automated failover using backups to restore the old primary
  - Regularly scheduled disaster recovery (during a main window if possible) to test restore techniques

Do you think they backup?

Demo Time!

- Live Demo, this will be fun…

Thank You! Questions?

email: david@pgbackrest.net

email: david.steele@crunchydata.com

release page: https://github.com/pgmasters/backrest/releases

slides & demo: https://github.com/dwsteele/conference/releases/tag/release/HeavyDutyPgBackRest-NYC-PUG-2015

Crunchy Enterprise PostgreSQL