

# High Performance pgBackRest

David Steele  
Crunchy Data

PGCon 2019  
May 30, 2019



# Agenda

- 1 Introduction
- 2 Core Commands
- 3 Archive Push
- 4 Backup
- 5 Archive Get
- 6 Restore
- 7 Other Considerations
- 8 Questions?

# About the Speaker

- Principal Architect at Crunchy Data, the Trusted Open Source Enterprise PostgreSQL Leader.
- Actively developing with PostgreSQL since 1999.
- Primary author of pgBackRest and co-author of pgAudit.
- PostgreSQL Contributor.

# What is pgBackRest?

pgBackRest aims to be a simple, reliable backup and restore system that can seamlessly scale up to the largest databases and workloads.

pgBackRest has a strong emphasis on performance, including:

- Parallel/asynchronous operation for all core commands
- Backup from Standby
- Advanced configuration for tuning specific commands

# What is pgBackRest?

pgBackRest aims to be a simple, reliable backup and restore system that can seamlessly scale up to the largest databases and workloads.

pgBackRest has a strong emphasis on performance, including:

- Parallel/asynchronous operation for all core commands
- Backup from Standby
- Advanced configuration for tuning specific commands

# What is pgBackRest?

pgBackRest aims to be a simple, reliable backup and restore system that can seamlessly scale up to the largest databases and workloads.

pgBackRest has a strong emphasis on performance, including:

- Parallel/asynchronous operation for all core commands
- Backup from Standby
- Advanced configuration for tuning specific commands

# What is pgBackRest?

pgBackRest aims to be a simple, reliable backup and restore system that can seamlessly scale up to the largest databases and workloads.

pgBackRest has a strong emphasis on performance, including:

- Parallel/asynchronous operation for all core commands
- Backup from Standby
- Advanced configuration for tuning specific commands

# Core Commands

- Archive Push

Allows PostgreSQL to push a completed WAL segment to the repository.

- Backup

Backup a PostgreSQL cluster.

- Archive Get

Allows PostgreSQL to get a completed WAL segment from the repository.

- Restore

Restore a PostgreSQL cluster.



# Core Commands

- Archive Push

Allows PostgreSQL to push a completed WAL segment to the repository.

- Backup

Backup a PostgreSQL cluster.

- Archive Get

Allows PostgreSQL to get a completed WAL segment from the repository.

- Restore

Restore a PostgreSQL cluster.

# Core Commands

- Archive Push

Allows PostgreSQL to push a completed WAL segment to the repository.

- Backup

Backup a PostgreSQL cluster.

- Archive Get

Allows PostgreSQL to get a completed WAL segment from the repository.

- Restore

Restore a PostgreSQL cluster.

# Core Commands

- Archive Push

Allows PostgreSQL to push a completed WAL segment to the repository.

- Backup

Backup a PostgreSQL cluster.

- Archive Get

Allows PostgreSQL to get a completed WAL segment from the repository.

- Restore

Restore a PostgreSQL cluster.

# Archive Push Features

- Asynchronous operation
  - Asynchronously scan the `archive_status` directory for WAL segments that are ready to be archived.
  - Store status of each WAL segment locally so PostgreSQL can be notified via the `archive_command` of success or failure.
  - Written in pure C for performance.
- Parallelism
  - Checksum, compress, encrypt, and transfer in parallel to improve throughput.

# Archive Push Features

- Asynchronous operation
  - Asynchronously scan the `archive_status` directory for WAL segments that are ready to be archived.
  - Store status of each WAL segment locally so PostgreSQL can be notified via the `archive_command` of success or failure.
  - Written in pure C for performance.
- Parallelism
  - Checksum, compress, encrypt, and transfer in parallel to improve throughput.

# Archive Push Features

- Asynchronous operation
  - Asynchronously scan the `archive_status` directory for WAL segments that are ready to be archived.
  - Store status of each WAL segment locally so PostgreSQL can be notified via the `archive_command` of success or failure.
  - Written in pure C for performance.
- Parallelism
  - Checksum, compress, encrypt, and transfer in parallel to improve throughput.

# Archive Push Features

- Asynchronous operation
  - Asynchronously scan the `archive_status` directory for WAL segments that are ready to be archived.
  - Store status of each WAL segment locally so PostgreSQL can be notified via the `archive_command` of success or failure.
  - Written in pure C for performance.
- Parallelism
  - Checksum, compress, encrypt, and transfer in parallel to improve throughput.

# Archive Push Configuration

## pgbackrest.conf

---

```
[global:archive-push]
archive-async=y
process-max=4
spool-path=/path/to/spool
```

---

- The `spool-path` parameter is optional (defaults to `/var/spool/pgbackrest`).
- The spool directory must exist for asynchronous operation.
- Note that configuration may be done with environment variables, e.g. `PGBACKREST_ARCHIVE_ASYNC`, or the command-line, e.g. `--archive-async`.



# Archive Push Configuration

## pgbackrest.conf

---

```
[global:archive-push]
archive-async=y
process-max=4
spool-path=/path/to/spool
```

---

- The `spool-path` parameter is optional (defaults to `/var/spool/pgbackrest`).
- The spool directory must exist for asynchronous operation.
- Note that configuration may be done with environment variables, e.g. `PGBACKREST_ARCHIVE_ASYNC`, or the command-line, e.g. `--archive-async`.

# Archive Push Configuration

## pgbackrest.conf

```
[global:archive-push]
archive-async=y
process-max=4
spool-path=/path/to/spool
```

- The `spool-path` parameter is optional (defaults to `/var/spool/pgbackrest`).
- The spool directory must exist for asynchronous operation.
- Note that configuration may be done with environment variables, e.g. `PGBACKREST_ARCHIVE_ASYNC`, or the command-line, e.g. `--archive-async`.

# Archive Push Configuration

## pgbackrest.conf

```
[global:archive-push]
archive-async=y
process-max=4
spool-path=/path/to/spool
```

- The `spool-path` parameter is optional (defaults to `/var/spool/pgbackrest`).
- The spool directory must exist for asynchronous operation.
- Note that configuration may be done with environment variables, e.g. `PGBACKREST_ARCHIVE_ASYNC`, or the command-line, e.g. `--archive-async`.

# Backup Features

- Backup from Standby
  - Perform most of the backup from a standby to reduce load on the primary.
  - Primary and standby are automatically selected from a list of clusters.
- Parallelism
  - Checksum, compress, encrypt, and transfer in parallel to improve throughput.

# Backup Features

- Backup from Standby
  - Perform most of the backup from a standby to reduce load on the primary.
  - Primary and standby are automatically selected from a list of clusters.
- Parallelism
  - Checksum, compress, encrypt, and transfer in parallel to improve throughput.

# Backup Features

- Backup from Standby
  - Perform most of the backup from a standby to reduce load on the primary.
  - Primary and standby are automatically selected from a list of clusters.
- Parallelism
  - Checksum, compress, encrypt, and transfer in parallel to improve throughput.

# Backup Configuration

## pgbackrest.conf

---

```
[global:backup]
backup-standby=y
process-max=8
```

```
[demo]
pg1-host=pg1
pg1-path=/var/lib/postgresql/10
pg2-host=pg2
pg2-path=/var/lib/postgresql/10
pg3-host=pg3
pg3-path=/var/lib/postgresql/10
```

---

- The current primary can be in any position in the list of PostgreSQL servers.
- The first live standby found will be used to perform the backup.

# Backup Configuration

## pgbackrest.conf

```
[global:backup]
backup-standby=y
process-max=8

[demo]
pg1-host=pg1
pg1-path=/var/lib/postgresql/10
pg2-host=pg2
pg2-path=/var/lib/postgresql/10
pg3-host=pg3
pg3-path=/var/lib/postgresql/10
```

- The current primary can be in any position in the list of PostgreSQL servers.
- The first live standby found will be used to perform the backup.



# Backup Configuration

## pgbackrest.conf

---

```
[global:backup]
backup-standby=y
process-max=8

[demo]
pg1-host=pg1
pg1-path=/var/lib/postgresql/10
pg2-host=pg2
pg2-path=/var/lib/postgresql/10
pg3-host=pg3
pg3-path=/var/lib/postgresql/10
```

---

- The current primary can be in any position in the list of PostgreSQL servers.
- The first live standby found will be used to perform the backup.

# Archive Get Features

- Asynchronous operation
  - Asynchronously build a queue of WAL segments that PostgreSQL will need.
  - Move or copy segments from the queue when requested by `restore_command`.
  - The spool directory should be located on the same device as `pg_xlog/pg_wal` for best performance.
  - Written in pure C for performance.
- Parallelism
  - Transfer, decrypt, decompress, and checksum in parallel to improve throughput.

# Archive Get Features

- Asynchronous operation
  - Asynchronously build a queue of WAL segments that PostgreSQL will need.
  - Move or copy segments from the queue when requested by `restore_command`.
  - The spool directory should be located on the same device as `pg_xlog/pg_wal` for best performance.
  - Written in pure C for performance.
- Parallelism
  - Transfer, decrypt, decompress, and checksum in parallel to improve throughput.

# Archive Get Features

- Asynchronous operation
  - Asynchronously build a queue of WAL segments that PostgreSQL will need.
  - Move or copy segments from the queue when requested by `restore_command`.
  - The spool directory should be located on the same device as `pg_xlog/pg_wal` for best performance.
    - Written in pure C for performance.
- Parallelism
  - Transfer, decrypt, decompress, and checksum in parallel to improve throughput.

# Archive Get Features

- Asynchronous operation
  - Asynchronously build a queue of WAL segments that PostgreSQL will need.
  - Move or copy segments from the queue when requested by `restore_command`.
  - The spool directory should be located on the same device as `pg_xlog/pg_wal` for best performance.
  - Written in pure C for performance.
- Parallelism
  - Transfer, decrypt, decompress, and checksum in parallel to improve throughput.

# Archive Get Features

- Asynchronous operation
  - Asynchronously build a queue of WAL segments that PostgreSQL will need.
  - Move or copy segments from the queue when requested by `restore_command`.
  - The spool directory should be located on the same device as `pg_xlog/pg_wal` for best performance.
  - Written in pure C for performance.
- Parallelism
  - Transfer, decrypt, decompress, and checksum in parallel to improve throughput.

# Archive Get Configuration

## pgbackrest.conf

---

```
[global:archive-get]
archive-async=y
archive-get-queue-max=1GB
process-max=2
```

---

- Archive Get generally requires fewer processes than Archive Push because decompression is less CPU-intensive than compression.
- On the other hand, clusters in recovery have more CPU resources to spare.
- The idea is to keep PostgreSQL supplied with WAL so that it doesn't need to wait.

# Archive Get Configuration

## pgbackrest.conf

---

```
[global:archive-get]
archive-async=y
archive-get-queue-max=1GB
process-max=2
```

---

- Archive Get generally requires fewer processes than Archive Push because decompression is less CPU-intensive than compression.
- On the other hand, clusters in recovery have more CPU resources to spare.
- The idea is to keep PostgreSQL supplied with WAL so that it doesn't need to wait.



# Archive Get Configuration

## pgbackrest.conf

---

```
[global:archive-get]
archive-async=y
archive-get-queue-max=1GB
process-max=2
```

---

- Archive Get generally requires fewer processes than Archive Push because decompression is less CPU-intensive than compression.
- On the other hand, clusters in recovery have more CPU resources to spare.
- The idea is to keep PostgreSQL supplied with WAL so that it doesn't need to wait.

# Archive Get Configuration

## pgbackrest.conf

---

```
[global:archive-get]
archive-async=y
archive-get-queue-max=1GB
process-max=2
```

---

- Archive Get generally requires fewer processes than Archive Push because decompression is less CPU-intensive than compression.
- On the other hand, clusters in recovery have more CPU resources to spare.
- The idea is to keep PostgreSQL supplied with WAL so that it doesn't need to wait.

# Restore Features

Restore performance is far more important than backup performance!

- Delta operation
  - Checksum local cluster files to determine what can be preserved.
  - Transfer only files that have changed since the last backup from the repository.
- Parallelism
  - Transfer, decrypt, decompress, and checksum in parallel to improve throughput.

# Restore Features

Restore performance is far more important than backup performance!

- Delta operation
  - Checksum local cluster files to determine what can be preserved.
  - Transfer only files that have changed since the last backup from the repository.
- Parallelism
  - Transfer, decrypt, decompress, and checksum in parallel to improve throughput.

# Restore Features

Restore performance is far more important than backup performance!

- Delta operation
  - Checksum local cluster files to determine what can be preserved.
  - Transfer only files that have changed since the last backup from the repository.
- Parallelism
  - Transfer, decrypt, decompress, and checksum in parallel to improve throughput.

# Restore Features

Restore performance is far more important than backup performance!

- Delta operation
  - Checksum local cluster files to determine what can be preserved.
  - Transfer only files that have changed since the last backup from the repository.
- Parallelism
  - Transfer, decrypt, decompress, and checksum in parallel to improve throughput.

# Restore Configuration

pgbackrest.conf

```
[global:restore]  
process-max=16  
delta=y
```

# High Latency

The `process-max` option can be used to speed transfers on high latency storage such as S3.



# Compression

The `compress-level` option can be lowered (e.g. 6 to 3) to reduce the CPU cost of compression. This also reduces the compression ratio, but the time savings are often worth it.

We are introducing `lz4` support soon for a faster alternative to `gzip`.

# Compression

The `compress-level` option can be lowered (e.g. 6 to 3) to reduce the CPU cost of compression. This also reduces the compression ratio, but the time savings are often worth it.

We are introducing `lz4` support soon for a faster alternative to `gzip`.

# The Future

The entire project will be migrated to C by the end of 2019. We are adding in many performance enhancements as we go.

# Questions?

website: <http://www.pgbackrest.org>

email: [david@pgbackrest.org](mailto:david@pgbackrest.org)

email: [david@crunchydata.com](mailto:david@crunchydata.com)

releases: <https://github.com/pgbackrest/pgbackrest/releases>

slides: <https://github.com/dwsteele/conference/releases>