

Rapid Upgrades With Pg_Upgrade

BRUCE MOMJIAN



Pg_Upgrade allows migration between major releases of Postgres without a data dump/reload. This presentation explains how pg_upgrade works.

Creative Commons Attribution License

<http://momjian.us/presentations>

Last updated: February, 2018

Traditional Postgres Major Upgrade Options

- ▶ Minor upgrades are simple
- ▶ `pg_dump` (logical dump)/restore
- ▶ Slony

Why Major Upgrades of Postgres Are Complex

- ▶ New features often require system table changes
- ▶ However, the internal data format rarely changes

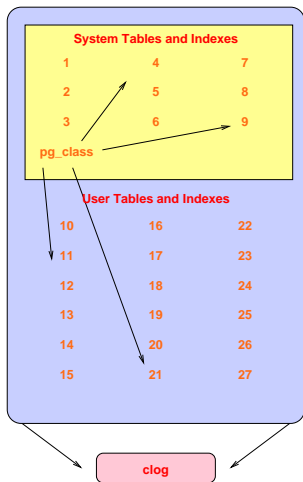
Why Pg_Upgrade

- ▶ Very fast upgrades
- ▶ Optionally no additional disk space

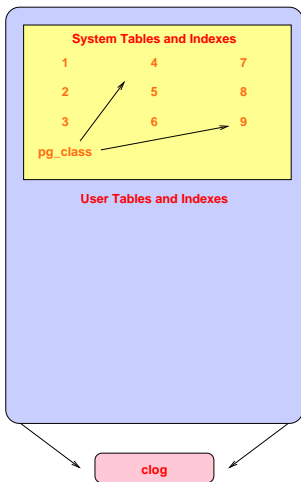
pg_upgrade installs new system tables while using data files from the previous Postgres version.

How It Works: Initial Setup

Old Cluster

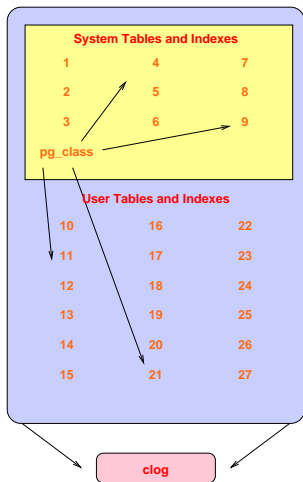


New Cluster

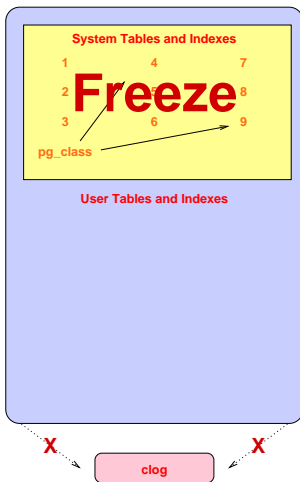


Decouple New Clog Via Freezing

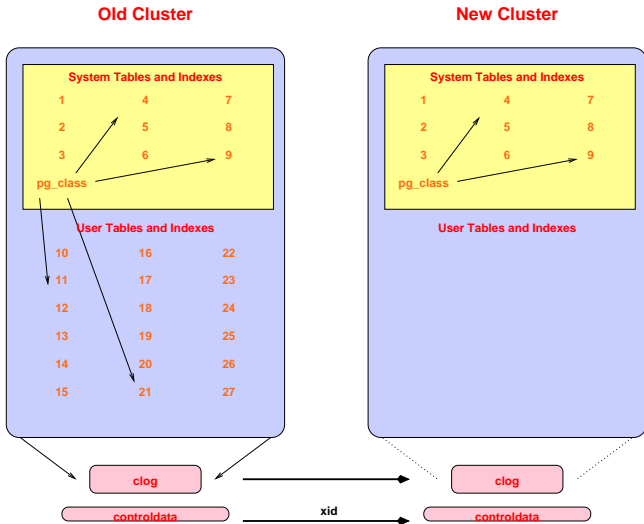
Old Cluster



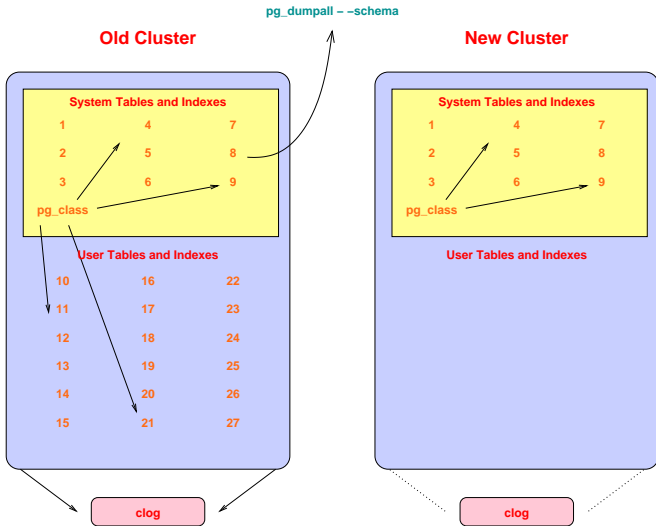
New Cluster



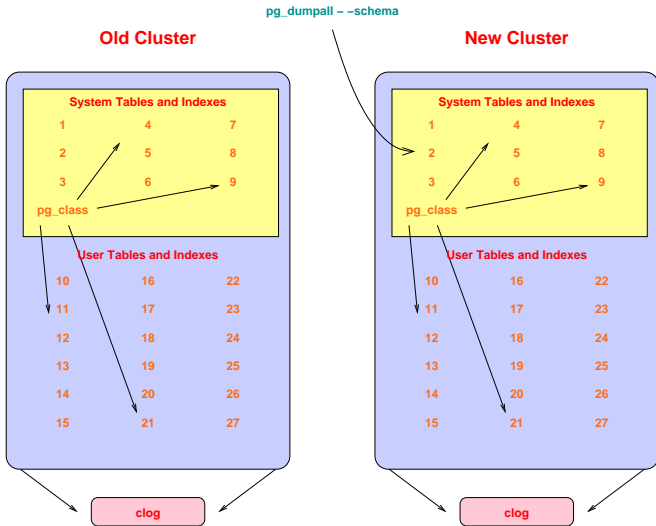
Transfer Clog and XID



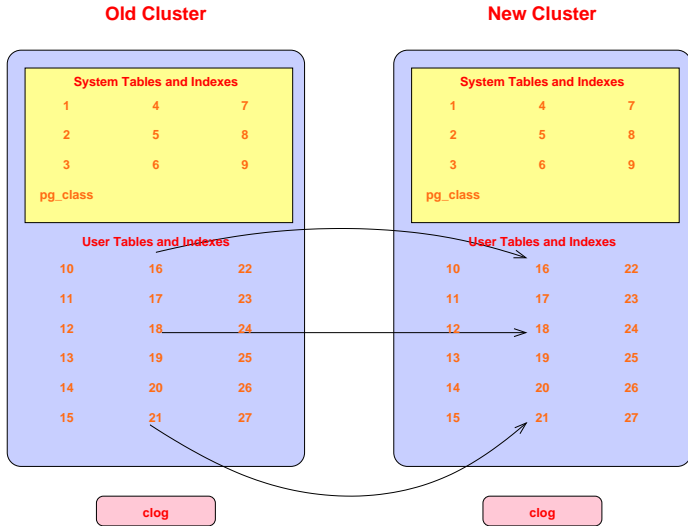
Get Schema Dump



Restore Schema In New Cluster

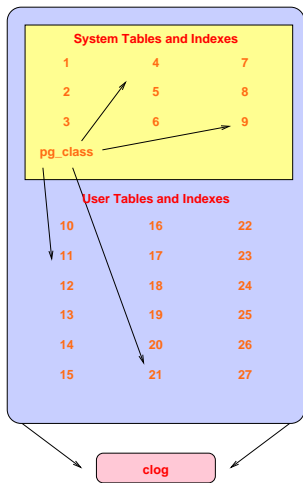


Copy User Heap/Index Files

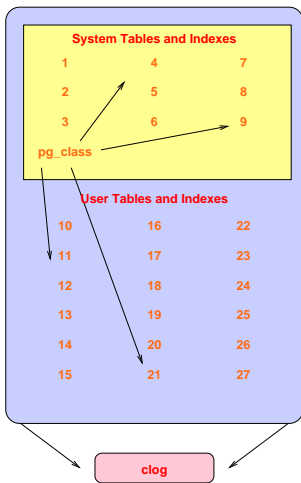


Complete

Old Cluster



New Cluster



How It Works: In Detail

- ▶ Check for cluster compatibility
 - ▶ locale
 - ▶ encoding
- ▶ Use `pg_dumpall` to dump old cluster schema (no data)
- ▶ Freeze all new cluster rows (remove reference to clog entries)
- ▶ New cluster uses old xid counter value (see freeze above)
 - ▶ Set system table frozen xids to match the current xid
- ▶ Create new users/databases
- ▶ Collect cluster information
- ▶ Install support functions that call internal backend functions
- ▶ Create schema in new cluster
- ▶ Copy or link files from old cluster to new cluster
- ▶ Warn about any remaining issues, like REINDEX requirements

Sample Run: Performing Consistency Checks

Performing Consistency Checks

```
-----  
Checking current, bin, and data directories      ok  
Checking cluster versions                       ok  
Checking database user is a superuser          ok  
Checking for prepared transactions              ok  
Checking for reg* system OID user data types   ok  
Checking for invalid indexes from concurrent  ok  
Checking for contrib/isn with bigint-passing  ok  
Creating catalog dump                          ok  
Checking for presence of required libraries    ok  
Checking database user is a superuser          ok  
Checking for prepared transactions              ok
```

If `pg_upgrade` fails after this point, you must re-initdb the new cluster before continuing.

Sample Run: Performing Migration

Performing Upgrade

| | |
|---|----|
| Analyzing all rows in the new cluster | ok |
| Freezing all rows on the new cluster | ok |
| Deleting files from new pg_clog | ok |
| Copying old pg_clog to new server | ok |
| Setting next transaction ID for new cluster | ok |
| Resetting WAL archives | ok |
| Setting frozenxid counters in new cluster | ok |
| Creating databases in the new cluster | ok |
| Adding support functions to new cluster | ok |
| Restoring database schema to new cluster | ok |
| Removing support functions from new cluster | ok |
| Adding ".old" suffix to old global/pg_control | ok |

If you want to start the old cluster, you will need to remove the ".old" suffix from /u/pgsql.old/data/global/pg_control.old. Because "link" mode was used, the old cluster cannot be safely started once the new cluster has been started.

Sample Run: Completion

```
Linking user relation files                                ok
Setting next OID for new cluster                          ok
Creating script to analyze new cluster                    ok
Creating script to delete old cluster                     ok
```

Upgrade Complete

Optimizer statistics are not transferred by pg_upgrade so,
once you start the new server, consider running:

```
analyze_new_cluster.sh
```

Running this script will delete the old cluster's data files:

```
delete_old_cluster.sh
```

Possible Data Format Changes

| Change | Conversion Method |
|-------------------------------------|-------------------------------------|
| clog | none |
| heap page header, including bitmask | convert to new page format on read |
| tuple header, including bitmask | convert to new page format on read |
| data value format | create old data type in new cluster |
| index page format | reindex, or recreate index methods |
| TOAST page format | convert to new page format on read |

Speed Comparison

| Migration Method | Minutes |
|----------------------------|---------|
| dump/restore | 300.0 |
| dump with parallel restore | 180.0 |
| pg_upgrade in copy mode | 44.0 |
| pg_upgrade in link mode | 0.7 |

Database size: 150GB, 850 tables

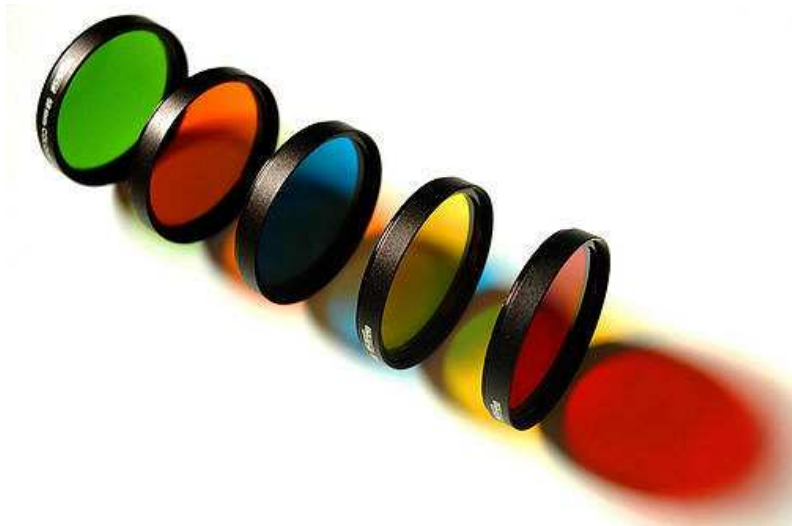
The last duration is 44 *seconds*.

*Timings courtesy of
Stefan Kaltenbrunner
(mastermind on IRC)*

Release History

- ▶ 9.0 focused on stability
- ▶ 9.1 focused on performance for databases with many relations
- ▶ 9.2 focused on improved debugging and reliability for non-standard configurations
- ▶ 9.3 focused on performance via parallelism and reduced fsync activity
- ▶ 9.4 dramatically reduced memory usage

Conclusion



<http://momjian.us/presentations>

https://www.flickr.com/photos/ram_balmur/