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.

https://momjian.us/presentations

Creative Commons Attribution License

Last updated: June 2024
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.

https://www.crunchydata.com/blog/examining-postgres-upgrades-with-pg_upgrade
How It Works: Initial Setup

Old Cluster

User Tables and Indexes
10  16  22
11  17  23
12  18  24
13  19  25
14  20  26
15  21  27

System Tables and Indexes
1   4   7
2   5   8
3   6   9
pg_class

New Cluster

User Tables and Indexes

System Tables and Indexes
1   4   7
2   5   8
3   6   9
pg_class
Decouple New Clog Via Freezing

Old Cluster

System Tables and Indexes
1 4 7
2 5 8
3 6 9

User Tables and Indexes
10 16 22
11 17 23
12 18 24
13 19 25
14 20 26
15 21 27

pg_class

clog

New Cluster

System Tables and Indexes
1 4 7
2 5 8
3 6 9

User Tables and Indexes

pg_class

Freeze

clog

X

X
Transfer Clog and XID

Old Cluster

System Tables and Indexes
1  4  7
2  5  8
3  6  9

User Tables and Indexes
10 16 22
11 17 23
12 18 24
13 19 25
14 20 26
15 21 27

pg_class

clog
controldata

New Cluster

System Tables and Indexes
1  4  7
2  5  8
3  6  9

User Tables and Indexes

pg_class

clog
controldata

xid
Get Schema Dump

Old Cluster

System Tables and Indexes
1 4 7
2 5 8
3 6 9

User Tables and Indexes
10 16 22
11 17 23
12 18 24
13 19 25
14 20 26
15 21 27

pg_class

clog

pg_dumpall -s

New Cluster

System Tables and Indexes
1 4 7
2 5 8
3 6 9

User Tables and Indexes

pg_class

clog
Restore Schema In New Cluster

Old Cluster

System Tables and Indexes
1  4  7
2  5  8
3  6  9
pg_class

User Tables and Indexes
10  16  22
11  17  23
12  18  24
13  19  25
14  20  26
15  21  27

clog

New Cluster

System Tables and Indexes
1  4  7
2  5  8
3  6  9
pg_class

User Tables and Indexes
10  16  22
11  17  23
12  18  24
13  19  25
14  20  26
15  21  27

clog

pg_dumpall --schema
Copy User Heap/Index Files

Old Cluster

System Tables and Indexes
1  4  7
2  5  8
3  6  9

User Tables and Indexes
10 16 22
11 17 23
12 18 24
13 19 25
14 20 26
15 21 27

pg_class

New Cluster

System Tables and Indexes
1  4  7
2  5  8
3  6  9

User Tables and Indexes
10 16 22
11 17 23
12 18 24
13 19 25
14 20 26
15 21 27

pg_class

clog

clog
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
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 index builds ok
Checking for contrib/isn with bigint-passing mismatch 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_xact ok
Copying old pg_xact 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  
Setting next OID for new cluster  
Creating script to analyze new cluster  
Creating script to delete old cluster  

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

<table>
<thead>
<tr>
<th>Change</th>
<th>Conversion Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>clog</td>
<td>none</td>
</tr>
<tr>
<td>heap page header, including bitmask</td>
<td>convert to new page format on read</td>
</tr>
<tr>
<td>tuple header, including bitmask</td>
<td>convert to new page format on read</td>
</tr>
<tr>
<td>data value format</td>
<td>create old data type in new cluster</td>
</tr>
<tr>
<td>index page format</td>
<td>reindex, or recreate index methods</td>
</tr>
<tr>
<td>TOAST page format</td>
<td>convert to new page format on read</td>
</tr>
</tbody>
</table>
### Speed Comparison

<table>
<thead>
<tr>
<th>Migration Method</th>
<th>Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump/restore</td>
<td>300.0</td>
</tr>
<tr>
<td>dump with parallel restore</td>
<td>180.0</td>
</tr>
<tr>
<td>pg_upgrade in copy mode</td>
<td>44.0</td>
</tr>
<tr>
<td>pg_upgrade in link mode</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Database size: 150GB, 850 tables

The last duration is 44 seconds.

Timings courtesy of
Stefan Kaltenbrunner
(mastermind on IRC)
• 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