PostgreSQL under Windows

Pavlo Golub

Senior Database Consultant

pavlo.golub@cybertec.at

@PavloGolub

CYBERTEC
The PostgreSQL Database Company

About CYBERTEC

- Inhouse Development
- International team of developers
- Specialized in Data services
- Owner-managed since 2000

CYBERTEC Worldwide

Wiener Neustadt
AUSTRIA

Tallinn
ESTONIA

Zurich
SWITZERLAND

Montevideo
URUGUAY

Client sectors

- University
- Automotive
- Government
- Industry
- Administration
- Finance
- Trade
- etc.

PostgreSQL Database Services

DATA Services

- Artificial Intelligence
- Machine learning
- BIG DATA
- Business Intelligence
- Data Mining

PostgreSQL Conference Europe 2019

October 15–18, 2019

Milan, Italy

https://2019.pgconf.eu/

Intro

Is it faster? Safer? Better?

PostgreSQL will definitely run faster on Linux than on Windows (and I say this as one of the guys who wrote the windows port of it..) It is designed for a Unix style architecture, and implements this same architecture on Windows, which means it does a number of things that Windows isn't designed to do well. It works fine, but it doesn't perform as well.

Magnus Hagander
2011-01-14

Is it faster? Safer? Better?

When PostgreSQL was ported, the old (Windows) IO stack was probably taken into account... if any :)

Ilya Kosmodemiansky
2019-09-10

PostgreSQL vs 64-bit Windows

First for the simple answer: No, there is no 64-bit version of PostgreSQL for Windows. PostgreSQL has supported 64-bit environments on Unix for many years (long before we had x64 to make it available for wintel machines), but there is no Win64 port. Yet. And given the way that PostgreSQL is developed, there is no firm date for when this will be available.

Magnus Hagander
2008-02-19

History

Changelog

- https://www.postgresql.org/docs/release/
- https://bucardo.org/postgres_all_versions.html

Changelog 6.1

- removed Windows-specific code (Bruce Momjian)

1997-06-08

Changelog 6.4

- libpq can now be compiled on Windows (Magnus Hagander)
- psql and libpq now compile under Windows using win32.mak (Magnus Hagander)

1998-10-30

Changelog 6.5

Add Windows NT backend port and enable dynamic loading

(Magnus Hagander and Daniel Horak)

1999-06-09

Changelog 7.0

Enable Windows compilation of libpq (Magnus Hagander)

2000-05-08

Changelog 7.2

- Fixes in **Cygwin** and Windows ports (Jason Tishler, Gerhard Haring, Dmitry Yurtaev, Darko Prenosil, Mikhail Terekhov)
- Fix for Windows socket communication failures (Magnus, Mikhail Terekhov)

2002-02-04

Changelog 7.3

- Add libpq PQescapeString() and PQescapeBytea() to Windows (Bruce)
- Fix for link() usage by WAL code on Windows, BeOS (Jason Tishler)
- Rename some internal identifiers to simplify Windows compile (Jan, Katherine Ward)

2002-11-27

Changelog 7.4

- Add function PQfreemem for freeing memory on Windows, suggested for NOTIFY (Bruce)
- Add Windows compatibility functions (Bruce)
- Allow client interfaces to compile under MinGW (Bruce)

2003-11-17

Changelog 8.0

- Microsoft Windows Native Server
  - This is the first PostgreSQL release to run natively on Microsoft Windows® as a server. It can run as a Windows service. This release supports NT-based Windows releases like Windows 2000 SP4, Windows XP, and Windows 2003.
- A separate installer project has been created to ease installation
- Allow the database server to run natively on Windows (Claudio, Magnus, Andrew)

2005-01-19

Changelog 8.1

- Allow the UTF8 encoding to work on Windows (Magnus)
- Add Kerberos 5 support for Windows (Magnus)
- Allow libpq to be built thread-safe on Windows (Dave Page)
- Allow IPv6 connections to be used on Windows (Andrew)

2005-11-08

Changelog 8.2

- Add native LDAP authentication (Magnus Hagander)
- Allow MSVC to compile the PostgreSQL server (Magnus, Hiroshi Saito)
- Add MSVC support for utility commands and pg_dump (Hiroshi Saito)
- Add support for code pages 1253, 1254, 1255, and 1257 (Kris Jurka)
- Drop privileges on startup, so that the server can be started from an administrative account (Magnus)
- Stability fixes (Qingqing Zhou, Magnus)
- Add native semaphore implementation (Qingqing Zhou)

2006-12-05

Changelog 8.3

- Support SSPI for authentication on Windows (Magnus)
- Use native threads in ecpg, instead of pthreads, on Windows (Magnus)
- Allow the whole PostgreSQL distribution to be compiled with Microsoft Visual C++ (Magnus and others)
- Drastically reduce postmaster's memory usage when it has many child processes (Magnus)
- Allow regression tests to be started by an administrative user (Magnus)
- Add native shared memory implementation (Magnus)

2008-02-04

Changelog 8.4

- Deprecate use of platform's `time_t` data type (Tom)
  - Some platforms have migrated to 64-bit `time_t`, some have not, and Windows can't make up its mind what it's doing. Define `pg_time_t` to have the same meaning as `time_t`, but always be 64 bits (unless the platform has no 64-bit integer type), and use that type in all module APIs and on-disk data formats.

2009-07-01

Changelog 9.0

- Write to the Windows event log in UTF16 encoding (Itagaki Takahiro)
- Support compiling on 64-bit Windows and running in 64-bit mode (Tsutomu Yamada, Magnus Hagander)
- Support server builds using Visual Studio 2008 (Magnus Hagander)

2010-09-20

Changelog 9.1

- Allow GSSAPI to be used to authenticate to servers via SSPI (Christian Ullrich)
- On Windows, allow `pg_ctl register` the service as auto-start or start-on-demand (Quan Zongliang)
- Enable building with the **MinGW64** compiler (Andrew Dunstan)
  - This allows building 64-bit Windows binaries even on non-Windows platforms via cross-compiling.

2011-09-12

Changelog 9.2

- Pass the safe number of file descriptors to child processes on Windows (Heikki Linnakangas)
- Support configurable event log application names on Windows (MauMau, Magnus Hagander)

2012-09-10

Changelog 9.4

- On Windows, automatically preserve quotes in command strings supplied by the user (Heikki Linnakangas)
- On Windows, ensure that a non-absolute `-D path` specification is interpreted relative to `pg_ctl`'s current directory (Kumar Rajeev Rastogi)
- Support client-only installs in MSVC (Windows) builds (MauMau)

2014-12-18

Changelog 9.5

- Allow control of `pg_ctl`'s event source logging on Windows (MauMau)
- If the server's listen address is set to a wildcard value (0.0.0.0 in IPv4 or :: in IPv6), connect via the loopback address (Kondo Yuta)
  - This fix primarily affects Windows, since on other platforms `pg_ctl` will prefer to use a Unix-domain socket
- Make `pg_basebackup` use a tablespace mapping file when using tar format, to support symbolic links and file paths of 100+ characters in length on MS Windows (Amit Kapila)
- Allow higher-precision time stamp resolution on Windows (Craig Ringer)

2016-01-07

Changelog 10

- Automatically mark all `PG_FUNCTION_INFO_V1` functions as `DLLEXPORT`-ed on Windows (Laurenz Albe)
  - If third-party code is using extern function declarations, they should also add `DLLEXPORT` markers to those declarations
- Allow `WaitLatchOrSocket()` to wait for socket connection on Windows (Andres Freund)

2017-10-05

Changelog 11

- Automatically mark all `PG_FUNCTION_INFO_V1` functions as `DLLEXPORT`-ed on Windows (Laurenz Albe)
  - If third-party code is using extern function declarations, they should also add `DLLEXPORT` markers to those declarations
- Allow `WaitLatchOrSocket()` to wait for socket connection on Windows (Andres Freund)

2017-10-05

Changelog 12

- Fix pg_test_fsync to report accurate open_datasync durations on Windows (Laurenz Albe)
- Require a C99-compliant compiler, and MSVC 2013 or later on Windows (Andres Freund)

2019-??-??

- First mentioned in 6.1 changelog
- Official client support in 7.0 version
- Full official support in 8.0 version
- x64 support in 9.0 version

What was the path on Windows

CYBERTEC
The PostgreSQL Database Company

Windows packages

Publishers

- EnterpriseDB
  - The only official installer listed on postgresql.org
- BigSQL
  - Not listed as official anymore
- Non-vanilla PostgreSQL
  - EnterpriseDB Advanced Server
  - PostgresPro
  - PowerGres

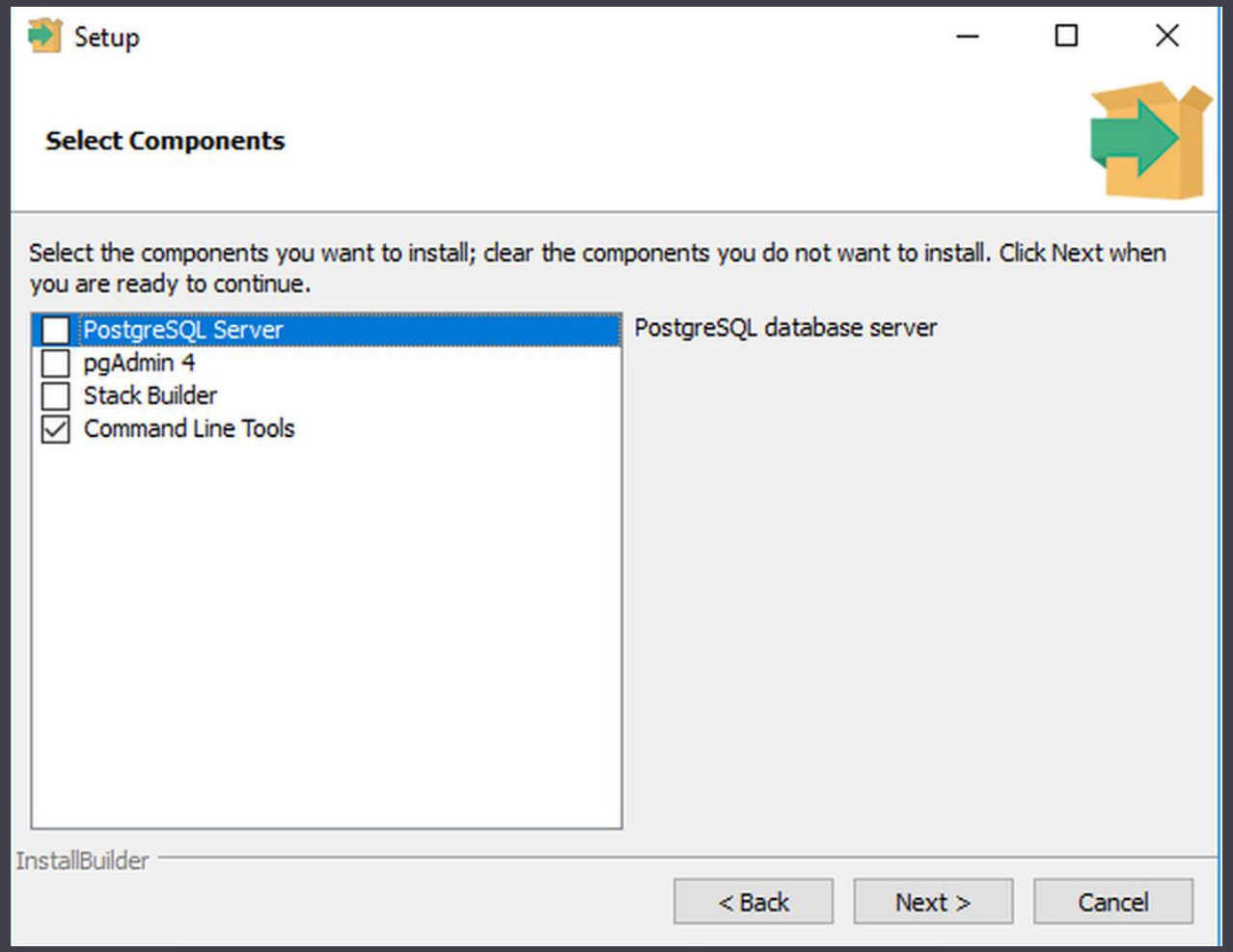
Server vs Client?

In Linux world we have different packages:

postgresql-client	libraries and client binaries
postgresql-server	core database server
postgresql-contrib	additional supplied modules
postgresql-devel	libraries and headers for development

Server vs Client?

In Windows world we have one installer to rule them all:



Binaries vs Installers

- You may download binaries only
- `initdb` - initialize a new data dir
- `pg_ctl` - start a cluster
- `pg_ctrl register` - to register PostgreSQL as a Windows service

Running on Windows

Service

- Windows service is a computer program that operates in the background and is implemented with the services API
- Handles low-level tasks that require little or no user interaction
- It is similar in concept to a Unix daemon
- `services.exe` process launches all the services and manages their actions, such as start, end, etc
- Windows services usually start when the operating system is started and run in the background as long as Windows is running
- Alternatively, they can be started manually or by an event

Service

- `pg_ctl register`
  - registers the PostgreSQL server as a system service on Windows
  - `-S auto | demand`
  - `-e source`
  - `-N servicename`
- `pg_ctl unregister`
  - unregisters a system service on Windows
- `pg_ctl kill`
  - sends a signal to a specified process

Signals

Windows has three signals, and none of them works!

Magnus Hagander

2019-09-12, PostgresOpen'19, Breakfast

Signals

- Windows does not have a built-in `kill` command
- Windows has three signals:
  - `CTRL_C_EVENT`,
  - `CTRL_BREAK_EVENT`…
- `pg_ctl kill` *`signal_name process_id`*
  - sends a signal to a specified process
  - Allowed signal names for kill:
    - `ABRT, HUP, INT, KILL, QUIT, TERM, USR1, USR2`

Path delimiters

- postgresql.conf:

```
log_directory = 'c:\temp\boom'
log_filename = pg-%a.log'
```

- pg_ctl start -D *data_dir*:

```
LOG:  unrecognized win32 error code: 123
FATAL:  could not open log file "c:   emoom/pg-Wed.log":
Invalid argument
LOG:  database system is shut down
```

shared_buffers

- On Windows, large values for `shared_buffers` aren't as effective
- You may find better results keeping the setting relatively low and using the operating system cache more instead
- The useful range for `shared_buffers` on Windows systems is generally from `64MB` to `512MB`
- That was stated in the v9.6 official manual
- Starting from v10 this statement is absent in the offiial manual
- But still present in the "Tuning Your PostgreSQL Server" wiki page and EnterpriseDB Advanced Server User Guide

huge_pages

- Huge pages are known as large pages on Windows
- User running PostreSQL needs "Lock Pages in Memory" right assigned
- Windows Group Policy tool (`gpedit.msc`) can be used to assign that right
- To run from the command line (not as a Windows service):
  - command prompt must be run as an administrator
  - User Access Control (UAC) must be disabled
- The performance improvement of using is about 2%
- Performance need to be measured for high `shared_buffers` values

shared_memory_type

- Specifies shared memory implementation
- On Windows default value is `"windows"`
- You don't want to try the other options
- The same rule applies to `dynamic_shared_memory_type` parameter

effective_cache_size

- Used only for estimation purposes
- 1/2 of total memory would be a normal conservative setting
- 3/4 of memory is a more aggressive
- Better estimate may be calculated using Task Manager
  - Performance -> Memory tab
  - Add "Cached" and "Available"

unix_socket_directories

- Specifies the directory of the Unix-domain socket(s) on which the server is to listen for connections from client applications
- Unix-domain socket is a standard component of POSIX
- This parameter is irrelevant on Windows, which is not POSIX
- This parameter is ignored on Windows
- Peter Eisentraut provided WIP patches for UDS support on hackers list

Authentication Methods

- GSSAPI
  - requires the MIT Kerberos for Windows package to build
- SSPI
  - SSPI authentication only works when both server and client are running Windows
  - or on non-Windows platforms when GSSAPI is available

Windows Defender

- Also known as Antimalware Service Executable
- May take a lot of CPU during PosstgreSQL run
- Possible workarounds:
  - "Windows Security" -> "Virus & Threat protection settings" and disable "Real-time protection"
  - "Windows Security" -> "Virus & Threat protection settings" -> "Exclusions" and "Add Exclusion":
    - Folder
    - Process
- The same applies for other Antivirus Software

Building on Windows

Visual Studio

- It is recommended to use Visual Studio Express 2019
- It is possible to build with the full Microsoft VC++ 2005 to 2019
- Binaries are dependent on Redistributable Package, e.g.

  `vcredist_x86.exe`
- The official binaries are built using Visual Studio

MinGW & MSYS

- "Minimalist GNU for Windows", is a minimalist development environment
- **MinGW** doesn't have a Unix emulation layer (no DLL for that required)
- Do not depend on any 3rd-party C-Runtime DLLs, e.g. `MSVCRT.DLL`
- Not all of the Windows API is supported, but enough for PostgreSQL
- Builds only for 32bit programs, no support for 64bit
- **MinGW** doesn't provide a POSIX-like environment
- **MSYS** is a collection of GNU utilities such as **bash, make, gawk, grep**, etc.
- **MSYS** provides shell script interpreter which is not available on Windows
- **MSYS** by itself does not contain a compiler or a C library

MinGW-w64 & MSYS2

- **MSYS2** is an independent rewrite of **MSYS**, based on modern **Cygwin** (POSIX compatibility layer) and **MinGW-w64**
- **MinGW-w64** is a improved version with 64bit support and some more of the WinAPI (still not all, but more than MinGW)
- **MinGW-w64** supports cross-compiling
- **MSYS2** features a **Pacman** package management system (Arch Linux)
- Some effort is made to work well with native Windows programs

MinGW-w64 cross compiling

```
$ sudo apt-get install mingw-w64

# download the source, cd into it

$ ./configure --host=i686-w64-mingw32 --prefix=... # 32 bit
$ ./configure --host=x86_64-w64-mingw32 --prefix=... # 64 bit

$ make
```

Cygwin

- Tries to bring a **POSIX**-compatible environment to Windows
- Provides a runtime library called `cygwin1.dll` with the POSIX compatibility layer
- Should only be used for running on older versions of Windows where the native build does not work, such as Windows 98

MSYS2 workflow

Preparation

```
pasha@PG480> c:\msys64\msys2_shell.cmd -mingw64

pasha@PG480 MINGW64 ~
$ mkdir /src && cd /src

pasha@PG480 MINGW64 /src
$ git clone https://github.com/openssl/openssl.git --branch
OpenSSL_1_1_1-stable --depth 1
...
Checking out files: 100% (18166/18166), done.

pasha@PG480 MINGW64 /src
$ git clone https://github.com/postgres/postgres.git --branch
REL_11_STABLE --depth 1
...
Checking out files: 100% (5467/5467), done.
```

Building OpenSSL

```
pasha@PG480 MINGW64 /src
$ cd /src/openssl

pasha@PG480 MINGW64 /src/openssl
$ rm -rf test/*

pasha@PG480 MINGW64 /src/openssl
$ ./Configure --prefix=/usr/local/openssl64 no-idea no-mdc2 no-rc5
shared mingw64
Configuring OpenSSL version 1.1.1d-dev (0x10101040L) for mingw64
...
OpenSSL has been successfully configured

pasha@PG480 MINGW64 /src/openssl
$ make clean && make -j4 install_dev
rm -f libcrypto-1_1-x64.dll
rm -f libssl-1_1-x64.dll
...
```

Building PostgreSQL client libraries only

```
pasha@PG480 MINGW64 /src/openssl
$ cd /src/postgres

pasha@PG480 MINGW64 /src/postgres
$ ./configure --prefix=/usr/local/postgres64 --with-openssl
--with-includes=/usr/local/openssl64/include
--with-libraries=/usr/local/openssl64/lib
...
checking for library containing CRYPTO_new_ex_data... -lcrypto
checking for library containing SSL_new... -lssl
checking for SSL_clear_options... yes
checking for SSL_get_current_compression... yes
checking for X509_get_signature_nid... yes
checking for OPENSSL_init_ssl... yes
...

pasha@PG480 MINGW64 /src/postgres
$ make clean && make -j4 -C src/common && make -j4 -C
src/interfaces install
```

Building PostgreSQL tools & libraries

```
pasha@PG480 MINGW64 /src/postgres
$ make -j4 -C src/common

pasha@PG480 MINGW64 /src/postgres
$ make -j4 -C src/interfaces install

pasha@PG480 MINGW64 /src/postgres
$ make -j4 -C src/bin/scripts install

pasha@PG480 MINGW64 /src/postgres
$ make -j4 -C src/bin/pg_dump install

pasha@PG480 MINGW64 /src/postgres
$ make -j4 -C src/bin/pg_basebackup install

pasha@PG480 MINGW64 /src/postgres
$ make -j4 -C src/bin/psql install
```

Switch toolchain to produce x86 binaries

```
pasha@PG480> c:\msys64\msys2_shell.cmd -mingw32

pasha@PG480 MINGW32 ~
$ cd /src/openssl

pasha@PG480 MINGW32 ~
$ ./Configure --prefix=/usr/local/openssl32 no-idea no-mdc2 no-rc5
shared mingw

...

pasha@PG480 MINGW64 /src/openssl
$ cd /src/postgres

pasha@PG480 MINGW64 /src/postgres
$ ./configure --prefix=/usr/local/postgres32 --with-openssl
--with-includes=/usr/local/openssl32/include
--with-libraries=/usr/local/openssl32/lib
```

Building extensions

```
pasha@PG480 MINGW64 /src
$ git clone https://github.com/anayrat/pg_sampletolog.git

pasha@PG480 MINGW64 /src
$ cd pg_sampletolog/

pasha@PG480 MINGW64 /src/pg_sampletolog
$ make install

pasha@PG480 MINGW64 /src/pg_sampletolog
$ ls /usr/local/postgres64/lib/pg_sampletolog*
/usr/local/postgres64/lib/pg_sampletolog.dll
```

High Availability (HA)

Patroni

- Patroni is a template to create customized, high-availability solution
- Uses distributed store like ZooKeeper, etcd, Consul or Kubernetes
- Written in Python for Linux
- Became extremely popular in the PostgreSQL world
- And now with Windows support
- Cybertec is in process of creating installer

SUMMARY

- There are still a lot of Windows installations
- And PostgreSQL is working just fine with it
- But how can we improve ecosystem?
- Microsoft is moving towards open source
- The question is not whether we like it or not
- The question is: How can we cope with it?

LINKS

?

I have some

- https://www.postgresql.org/docs/release/
- https://bucardo.org/postgres_all_versions.html
- https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server
- https://www.msys2.org/
- http://mingw-w64.org/
- https://github.com/pashagolub/patroni
- https://www.cybertec-postgresql.com/en/bulding-postgresql-x86-x64-and-openssl-using-msys2-and-mingw-under-windows/

QUESTIONS

PostgreSQL under Windows

Pavlo Golub

Senior Database Consultant

pavlo.golub@cybertec.at

@PavloGolub

CYBERTEC
The PostgreSQL Database Company