How to get a feature committed?

Michael Paquier – VMware

2018/12/11, PGConf.Asia 2018

About the man

- Michael Paquier.
- French, based in Tokyo.
- PostgreSQL contributor since 2009
  - Some patches, some reviews and some bug fixes.
  - Blogging.
  - Committer since June 2018.
- Working at VMware on PostgreSQL
  - Packaging.
  - Integration.
  - Support.

Community

- Core database engine
- Steady, well-designed progress
- World-wide investment
- Code of conduct (since 2018) https://www.postgresql.org/about/policies/coc/

DB engine Ranking (Dec 2018)

- https://db-engines.com/en/ranking

#	Engine	Score	Nov 2018	Dec 2017
1	Oracle	1283.22	-17.89	-58.32
2	MySQL	1161.25	+1.36	-156.82
2	MSSQL	1040.34	-11.21	-123.14
4	Postgres	460.64	+20.39	+75.21
5	MongoDB	378.62	+9.14	+47.85

PaaP

- PostgreSQL As A Product.
- (F)Orcs must die!
  - If you like it, still you can fork it.
  - Long-term prospect gets better by investing in the core engine.
  - Some folks are still able to live with this model with customer base (EDB Postgres, Amazon, Greenplum).
- Hard to contribute, initial time investment worth it long-term.
- Hundreds of man years into the code from core developers.

Database engine

- ACID
- Free to use – BSD like
- Open engineering.
- Open to new and good ideas.
- Highly pluggable: data types, plugins.
- High code quality.

Committers

- World-wide, shared-something distribution
- 28 in total: https://wiki.postgresql.org/wiki/Committers
- North America (10) - US
- Europe (12) - Czech, France, Finland, Ireland, Russia, Sweden, UK
- South America (1) - Chile
- Asia (4) - India, Japan
- Oceania (1) - NZ

Development cycle

- Schedule decided at PGCon, developer meeting: https://wiki.postgresql.org/wiki/PgCon_2018_Developer_Meeting
- 1 year, September to September, until GA
- **Roughly**
- Divided into two periods
  - Feature submission and new developments
  - Focus on stability

Release maintenance

- One branch on git per major version

  REL_11_STABLE => v11
  REL_10_STABLE => v10
  REL9_6_STABLE => v9.6

- Only bug fixes, no new features
- No change in system catalogs or WAL
- EOL’d after 5 years

Commit fests

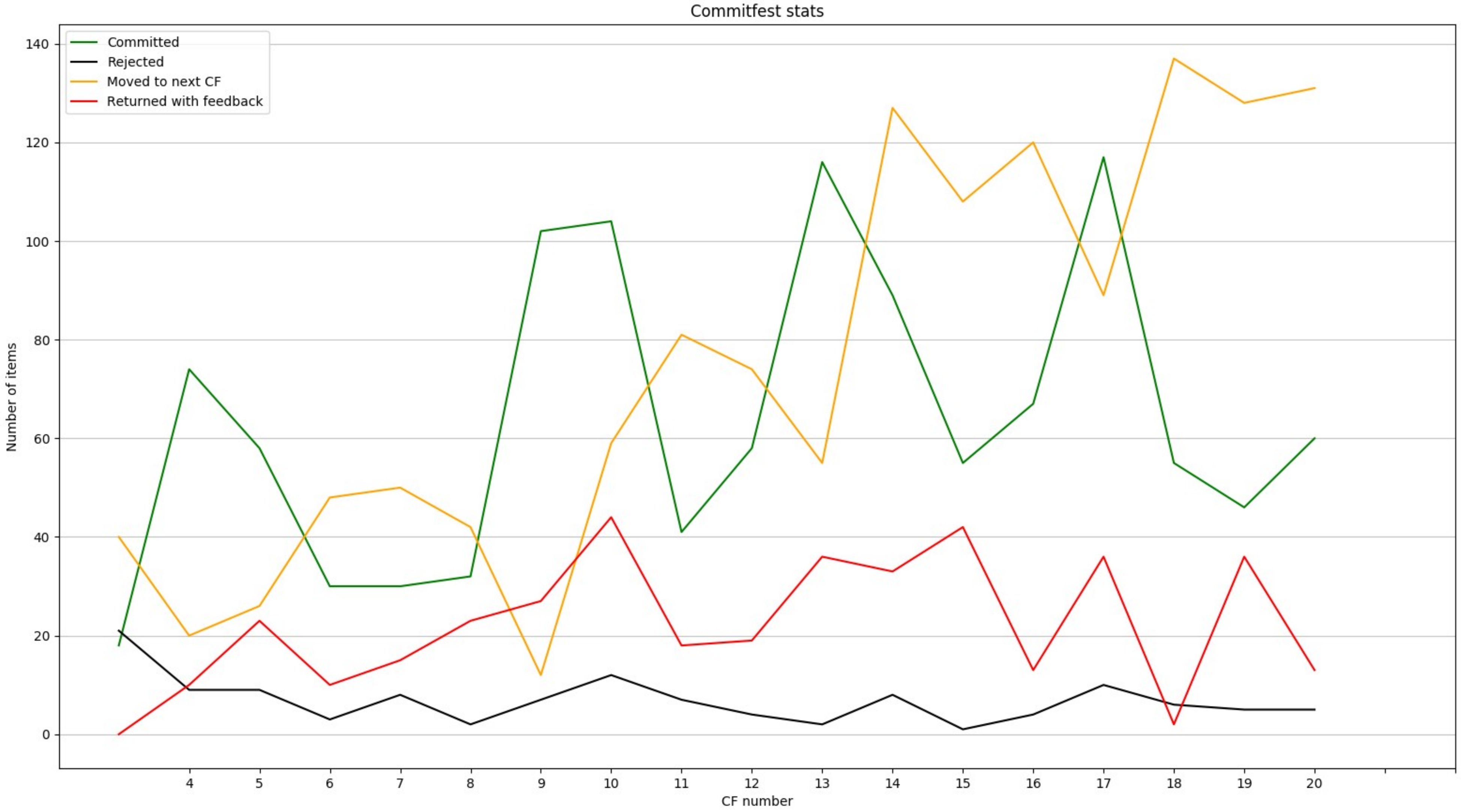
- 4~5 commit fests per development cycle.
- September to March.
- One month of break between each.
- Up to 250 patches.

Stability period

- First beta after last commit fest, in April
- Stability work:
  https://wiki.postgresql.org/wiki/PostgreSQL_11_Open_Items
- Help in stabilizing things
  - Testing, actual fixes
  - Show involvement with community
  - As important as writing cool features

Commit fest management

- Up to 250 patches.
- Few rejections.
- A lot returned with feedback.
- More patches bumped to next CF.
- Authors forget!
- Reviewers forget!
- See follow-up graph (Credits: Dmitry Dolgov).


Commitfest stats
Committed
Rejected
Moved to next CF
Returned with feedback
Number of items
0
20
40
60
80
100
120
140
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
CF number

Patch submission

- Read guidelines
  https://wiki.postgresql.org/wiki/Submitting_a_Patch
- Register it in commit fest (need community account)
- If WIP, begin discussion first, draft patch fine.
- Email to pgsql-hackers.
- Bug fixes also to pgsql-bugs.

Patch contents

- The code itself – Comments.
- Documentation!
  - Explanation in email may not be enough.
  - Self-contained docs are for the user at the end.
- Tests:
  - isolation, regression, TAP?
  - Should be designed to not be too costly, still hold value.
  - Helps in reviewing feature.
- Avoid non-ASCII characters.

Coding convention

- Signal handling, macros, error format, etc.
- Configuration in src/tools/editors/: vi, emacs.
- Documentation available https://www.postgresql.org/docs/devel/source.html

Regression tests

- Main regression suite, src/test/regress/
- Isolation test (concurrency), src/test/isolation/
- Extension tests, src/test/modules/ and contrib/
  https://www.postgresql.org/docs/11/extend-pgxs.html
- TAP tests
  - src/test/perl/ for base modules.
  - src/test/recovery/, authentication/, ssl/, etc.
- PG_TEST_EXTRA='ssl ldap kerberos'

Patch format

- Acceptable formats:
  - git diff
  - git format-patch
- Add version in the file name
- Sometimes make sense to split into multiple commits
  - Refactoring first
  - Actual feature
- Personal viewpoint: as long as it can be applied cleanly I am fine.
  - patch -p1 < your-cool-stuff-v1.patch

CF manager

- Deputy handling patch workflow
- 1 or more contributors.
- Actions
  - Tracking
  - poking.
  - mostly poking and vacuuming.
- Done by seasoned hackers.

CF bot

- Automatic check of submitted patches
  - Linux
  - Windows
- URL: http://commitfest.cputube.org/
- Divided by author: http://commitfest.cputube.org/michael-paquier.html
- Credits: Thomas Munro

Patch review (1)

- Flow
  - Exchange between author(s) and reviewer(s).
  - Patch marked as ready for committer.
  - Committer looks at it, committing it or falling back.
  - Committer has last word.
- 1 patch written = 1 review of equivalent difficulty.
- Reviews are as important as writing cool code.

Patch review (2)

- Expect one or more rewrites of the patch.
- Make sure to agree on the shape before consuming time writing it.
- **Consensus is key.**
- A complicated patch will never finish in the shape it was designed initially.

Conclusion

- Be patient
- Help others, take new challenges.
- Fixing bugs and doing maintenance helps as well.
- Remain polite, respect others.

Thanks!
Questions?