Vector **O**ptimized **L**ibrary of **K**ernels

2020 in review

What's VOLK again?

Vector Optimized Library of Kernels (VOLK)

- libvolk is written in C!
- Currently available under GPLv3
- Collection of hand-optimized SIMD kernels
- Optimized for several architectures and platforms

Focus: Software Radio applications

- Let us know about your use case!

Idea

- Common function interface
- Architecture specific SIMD implementations
- Load best implementation at runtime

Website: libvolk.org Repository: github.com/gnuradio/volk

History

Around 2010: Introduced as part of GNU Radio

Around 2015: Split off of GNU Radio as an independent library

Currently VOLK is a library under the GNU Radio organization umbrella

Maintainers

Before ‘15: GNU Radio maintainers

‘15 to ‘19: Nathan West

‘19 to current: Johannes Demel and Michael Dickens

Johannes Demel

Michael Dickens

Basics - Installation

Your favorite package manager

sudo apt install libvolk2-dev

sudo port install volk

conda install -c conda-forge volk

```
git clone --recursive \
    https://github.com/gnuradio/volk.git
cd volk
mkdir build
cd build
cmake ..
make -j8 # your choice
sudo make install
sudo ldconfig
```

First thing to do

volk_profile

Reason:

- Creates `.volk/volk_config`
- Benchmarks fastest kernel on YOUR system
- Saves this info to make future VOLK kernel calls fast!

Entries look like:

volk_32fc_x2_multiply_32fc	a_avx2_fma	u_sse3
kernel_name	aligned	unaligned

Basics - usage

Aligned memory

```
// C
// 32byte AVX aligned memory with 64 float complex elements.
const size_t avx_alignment = 32; // byte alignment
float complex* values =
    (float complex*) volk_malloc(512, avx_alignment);
volk_free(values); // don't forget this! It's your memory!
// C++
auto values = volk::vector<complex<float> >(64);
```

- Use the new C++ helpers for memory management!
- Aligned memory access often faster

Function calls

```
// C
volk_32fc_x2_multiply_32fc(result, values,
                           more_values,
                           num_values);
// C++
volk_32fc_x2_multiply_32fc(result.data(),
                           values.data(),
                           more_values.data(),
                           num_values);
```

- Common interface
- VOLK takes care of specific kernel calls

Basics - language

Runtime

VOLK library

- Written in C11

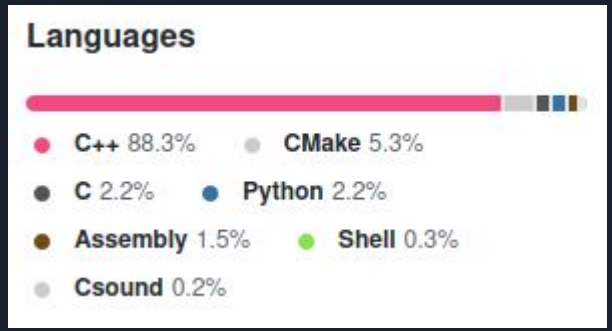
Support code

- C++
  - `volk_profile`
  - `volk-config-info`
  - Aligned vector `volk::vector`

Build time

- CMake
- Python

Github might have gotten that one wrong...



Releases

v2.2.0	v2.2.1	v2.3.0	v2.4.0	v2.4.1
2020 Feb 16	2020 Feb 24	2020 May 9	2020 Nov 22	2020 Dec 17

We aim for regular releases, e.g., every 2 months

BUT: Only if there are changes to release

Releases are **GPG** and **signify** signed!

Suggestions for improvements?

→ Please let us know!

Statistics - basics

Pull Requests 2020

- Opened: 66
- Closed: 64

Issues 2020

- Opened: 45
- Closed: 37

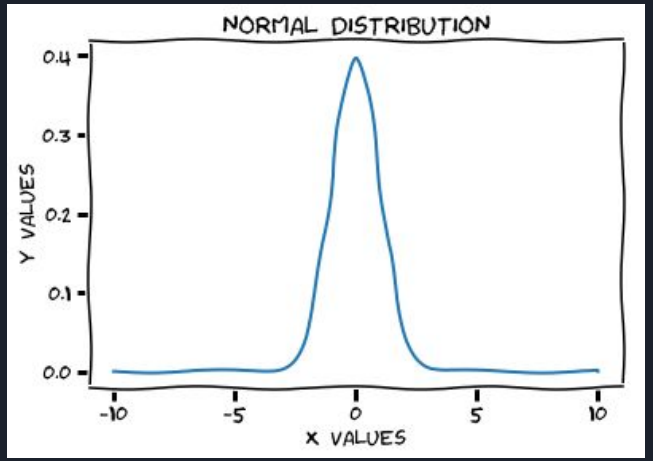
Contributors

- All time: 61
- 2020: 21

- ANSI C: 34,448 (90.35%)
- C++: 1,671 (4.38%)
- Asm: 802 (2.10%)
- Python: 735 (1.93%)
- XML: 357 (0.94%)
- sh: 114 (0.30%)

total: 38,127

generated using David A. Wheeler's 'SLOCCount'



We need more numbers to get here!

Statistics - users

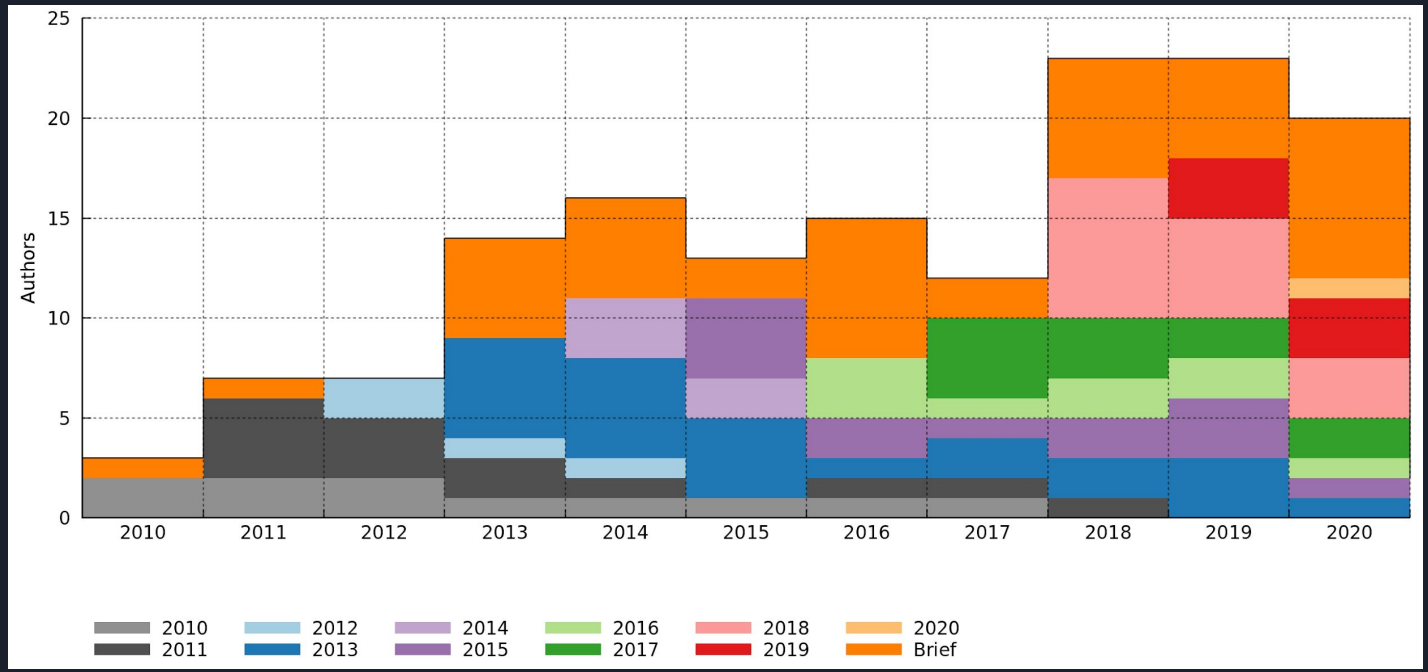
How many users do we have?

- We don’t know… BUT

Insights into our user base would be great

- Improve usability
- Identify future directions...

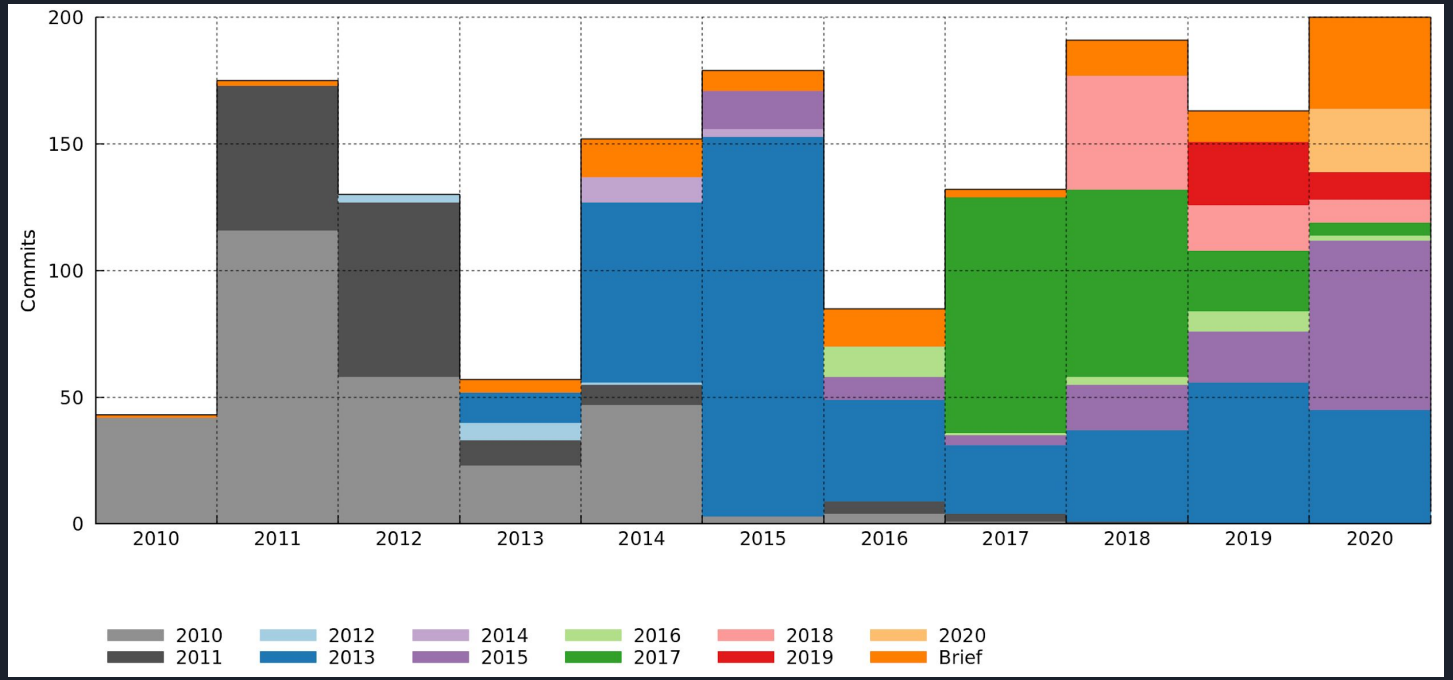
Statistics - authors



Contributors per year - and first appearance

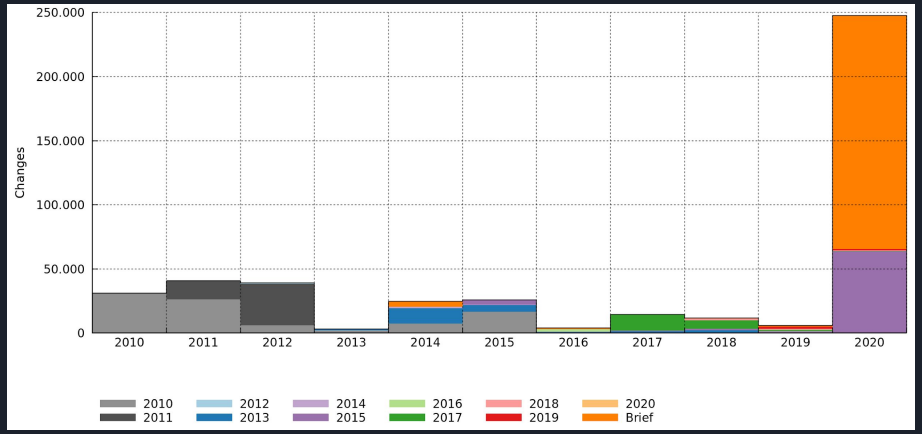
Plot tool: github.com/hpjansson/fornalder

Statistics - commits



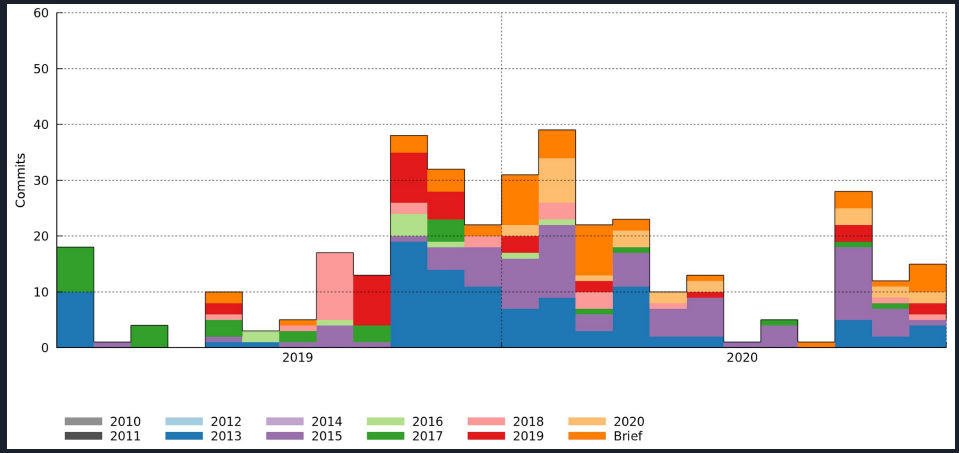
Commits over years

Statistics - more graphs



Changes over years

- Auto format ruins statistics...



Commits in 2019 - 2020

- Is VOLK a winter project? ...

Major changes - code

Prerequisites

- Require **C11**
  - Use `aligned_alloc` whenever possible
- CMake 3.8+
- Build time
  - Require Python3

Support infrastructure (volk_profile, etc.)

- Mostly remove Boost!
  - New GCC/Clang use C++17 features

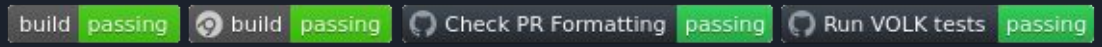
Many new NEON kernels available

Enforce code formatting

- Inherited from GNU Radio
- We use **clang-format**

Major changes - infrastructure

Continuous Integration



- Extended tests
- TravisCI
  - GCC 4.8, 6, 7, 8, 9
  - Clang 6, 8, 9, 10
  - ARM aarch64
  - QEMU cross build
  - Intel SDE
  - And broken… cf. TravisCI policy
- AppVeyor
- Github Actions
  - macOS Intel
  - Windows
  - Linux
  - Linux static library

Major changes - features

cpu_features

- Detect available CPU features
  - SSE, AVX, NEON, etc.
- Benefit from shared expertise
- Platform independent (Linux, but also macOS, etc.)

macOS and Windows

Optimized kernels available

- Fix cpu feature detection
- Have reliable CI
- Better cpu feature detection
- Currently for Intel
- macOS ARM64 coming

Move to LGPLv3

Current license

- GPLv3+

History

- VOLK was part of GNU Radio
- GNU Radio is licensed under GPLv3+

Idea: Move to LGPLv3

Why?

- Community suggestion
- VOLK should see greater adoption

Why not BSD/MIT/Apache?

- Fairness
  - You get a library
  - Give back

Who wants to participate in this effort?

Future directions

More ARM

- SVE?

More AVX

- Have an AVX version for all kernels
- Currently only few AVX512 kernels.
  - e.g., I'm stuck with AVX2 at the moment.

More architectures?

- RiscV seems appealing
- PowerPC ?

Switch to DCO instead of CLA

- Lost a potential contributor presumably because of CLA recently

VOLK v3.0? → Breaking API

- Sanitize interface
  - C library that exports C functions
- Switch to LGPLv3

Get to know YOU

We'd like to know more about our users!

Are you a VOLK package maintainer for a system? Let us know!