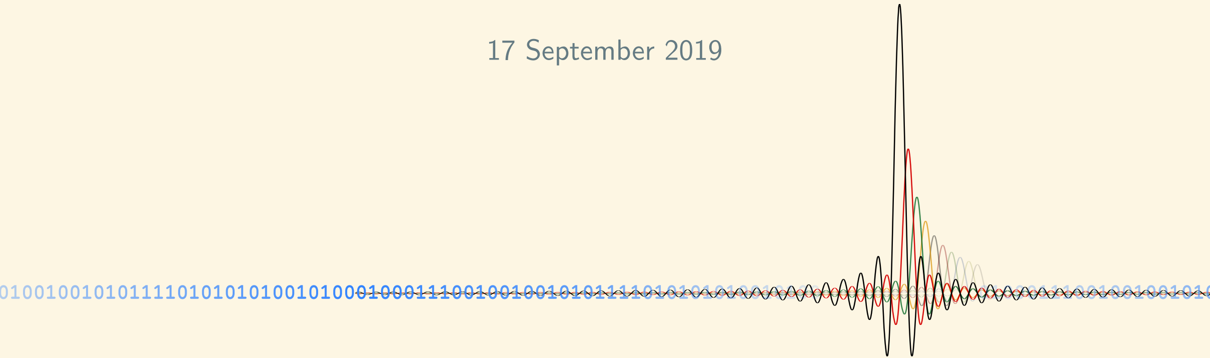
GNU Radio beyond 3.8

A technical outlook

Marcus Müller

17 September 2019



Structure

Introduction

Looking back at the releasing 3.8

GNU Radio 3.8 and on

The Next Big Thing

Questions & Answers

Marcus Müller

Bearer of a couple of roles

- Research assistant at CEL / KIT Karlsruher Institut für Technologie
  - Exercise classes for KIT EEs' *Probability Theory* and *Communications Theory I* courses ($>$ 300 students) and *Applied Information Theory*, *Advanced Radio Communications II*, *Communications Theory II* (ca 13 dB fewer students), also computer lab and a couple of advised B.Sc./M.Sc. theses
  - PhD on LDPC on non-stationary P_e / short packet channels
- Freelancing Engineer
  - Technical Consulting
  - Contract Development
  - Customer-Specific Training Courses
- Ettus Support Grumpiness supplier
- Maintainer of the GNU Radio project

Marcus Müller

Bearer of a couple of roles

- Research assistant at CEL / KIT Karlsruher Institut für Technologie
  - Exercise classes for KIT EEs' *Probability Theory* and *Communications Theory I* courses ($>$ 300 students) and *Applied Information Theory*, *Advanced Radio Communications II*, *Communications Theory II* (ca 13 dB fewer students), also computer lab and a couple of advised B.Sc./M.Sc. theses
  - PhD on LDPC on non-stationary P_e / short packet channels
- Freelancing Engineer
  - Technical Consulting
  - Contract Development
  - Customer-Specific Training Courses
- Ettus Support Grumpiness supplier
- **Maintainer of the GNU Radio project**

Marcus Müller

Contact

Depending on what you want to talk to me about, contact me using

- University Research & Teaching: `mueller@kit.edu`
- GNU Radio aspects: Preferably, `discuss-gnuradio@gnu.org`, for confident matters `mmueller@gnuradio.org`
- Ettus support: `support@ettus.com` (ask for Marcus The Younger)
- Freelancing & Private: `mueller@hostalia.de`

Looking back at the releasing 3.8

Superficially (I've been doing this way too often):

- Python2 → Python2 ˆ 3
- C++03 → C++11
- All-around source formatting
- Qt4 → Qt5
- XML → YAML
- Vintage CMake → Modern CMake
- Pixelized Canvas → Vector GRC
- Boring straight connectors → curves

Things learned from this release

- Maintenance branch mergeback model without definite dates for the future leads to stalling
- Not doing a larger release for six years hurts...
    - a lot.
- Actually working towards a release unleashes a lot of energy
    - Super happy that we didn't lose a lot of developers, it seems
    - See Ben's slide on Year-to-Year comparison of participation metrics
- Release publicity is a lot based on copy & paste

GNU Radio 3.8: What now?

Git branches now consistently reflect new development:

- `master` will become 3.9
- `maint-3.8` is from where 3.8.x.x releases are made from
- `maint-3.7` is from where 3.7.x.x releases are made from

Use this correctly: Submit patches / Pull Requests against the right branch!

- You've got a feature to merge? → `master`
- You've got a bug to fix that applies to both `master` and 3.8? → `master` (and tell us you want us to backport/cherry-pick to `maint-3.8`)
- You've got a bug that's specific to a specific release series? → `maint-`Release

GNU Radio 3.8++

GNU Radio 3.9: confirmed features

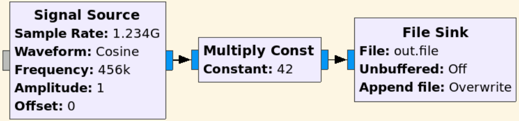
- Upstream `gr-iio`: libiio – Standard Linux sampling device inteface (e.g. Pluto)
- Upstream `gr-soapy`: hardware-abstracting universal SDR driver interface
- Python 3 only

But **when**?

- Regular release cadence
- Tentatively:
  - Release shortly before GRCon (late August)
  - Release a month after FOSDEM (mid-March)

- As Ben said: GNU Radio can now legally order booze (in Germany)
- Scheduling is actually pretty primitive
- Let's look at current scheduling to learn what needs to improve

Current Signal Flow Architecture



Signal Source
Sample Rate: 1.234G
Waveform: Cosine
Frequency: 456k
Amplitude: 1
Offset: 0

Multiply Const
Constant: 42

File Sink
File: out.file
Unbuffered: Off
Append file: Overwrite

- GNU Radio is a backpressure-driven parallel signal processing architecture
- Blocks produce as much output as they can at once, given
  - available input data ready at the start of processing
  - available output data memory
- Every block runs in its own thread
- asked to produce min(buffer size / 2, available output buffer)
- Block can start working again while downstream block is still consuming
- → high parallelism

Current Scheduling Mechanism – Abstracted

- *Scheduler* might be too strong a word
- *back pressure* limits processing speed
- great for throughput
- not so great for latency
- high parallelism stems from the ability to concurrently execute
- actual scheduling of threads done by OS
- no workload knowledge flows into OS → suboptimal . . .
- . . . but works surprisingly well.

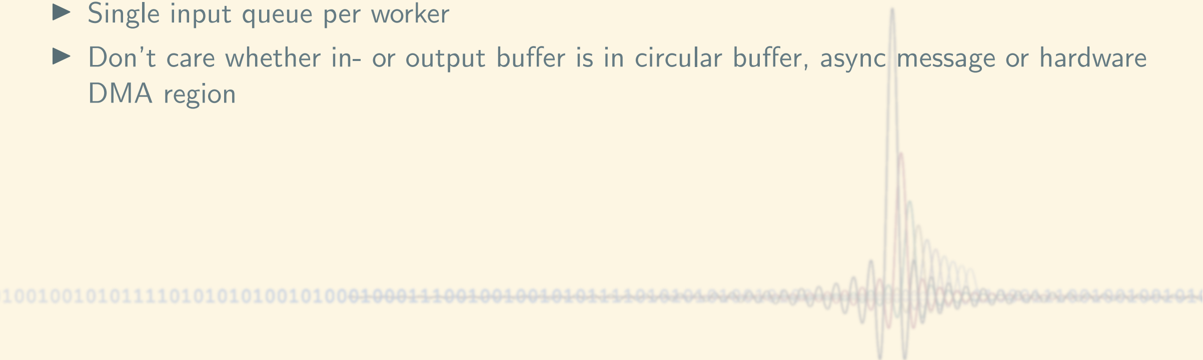
- Lots of thread-safety concerns

Introducing: The Next Big Thing (NBT)

- “Scheduling” is actually pretty suboptimal
  - One thread per block: What if number of blocks $\neq$ cores?
  - Scheduling is actually by the OS
  - no feedback of data flow into the scheduling at all
  - CPU core utilization $\gg$ not thrashing caches
- Streams and Message are not equal
  - It’s hard to impossible to do no-latency stream-produce-on-async-message blocks (ask Matt!)
  - Way to many states “I’m done”
  - can’t just apply `work` to the content of a message (invented TSB for that, not an adequate design)

NBT – Approach

- Thread-per-Block → Worker threads
- Single input queue per worker
- Don't care whether in- or output buffer is in circular buffer, async message or hardware DMA region



NBT – Approach

- Thread-per-Block → Worker threads
  - can still be 1 worker : 1 block
  - sensible: 1 worker : 1 CPU core (N blocks)
- Single input queue per worker
- Don't care whether in- or output buffer is in circular buffer, async message or hardware DMA region

NBT – Approach

- Thread-per-Block → Worker threads
- Single input queue per worker
    - No special “blocked” states
    - Migration easy
        - `stop`-less reconfiguration
    - Queue can be clever
        - reorder outstanding items to maximize cache locality
        - signal overload of worker
        - prioritize based on latency constraints . . .
    - Receive *Workload Items* via message passing
        - ZeroMQ: low-overhead transparent, thread-safe message passing
        - Reduction of hidden state: Transparently networkable GNU Radio flowgraphs
- Don’t care whether in- or output buffer is in circular buffer, async message or hardware DMA region

NBT – Approach

- Thread-per-Block → Worker threads
- Single input queue per worker
- Don't care whether in- or output buffer is in circular buffer, async message or hardware DMA region
  - Obsoletes TSBs
  - Enables Zero-Copy Accelerator, NIC interfacing

NBT – Necessary Changes

- Testing of Scheduler Correctness
- Benchmarking
  - Not only: Throughput, but also
  - Latency constraints (we can track these reasonably with queues!)
  - Number of CPU migrations
  - Cache access failures

  can well be done with eBPF
- Refactoring of `block_executor`
  - Literally among oldest code in GNU Radio
  - Dead code, unused state

Bastian Bloessl has taken the lead on this[1]

[1] Bastian Bloessl, Müller, Hollick: *Benchmarking and Profiling the GNU Radio Scheduler*, Proceedings of the 9th GNU Radio Conference, Sept. 2019

NBT – Necessary Changes

- Testing of Scheduler Correctness
- Benchmarking
  - Not only: Throughput, but also
  - Latency constraints (we can track these reasonably with queues!)
  - Number of CPU migrations
  - Cache access failures

  can well be done with eBPF
- Refactoring of `block_executor`
  - Literally among oldest code in GNU Radio
  - Dead code, unused state

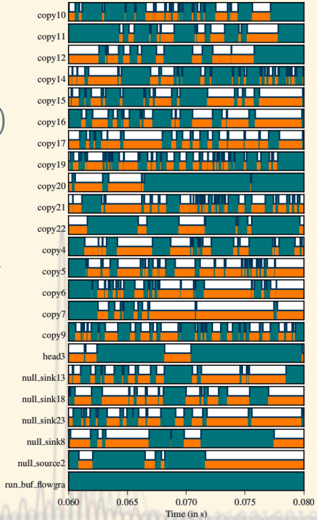
Bastian Bloessl has taken the lead on this[1]

[1] Bastian Bloessl, Müller, Hollick: *Benchmarking and Profiling the GNU Radio Scheduler*, Proceedings of the 9th GNU Radio Conference, Sept. 2019

NBT – Workload and Strategy

Immediate yields[2]:

- Refactored scheduler code to be merged into `master` (for 3.9)
- Benchmarking shows significant impact of workload size on caching we've largely ignored so far
- Benchmarking toolkit `gr-sched`[3]
  - Throughput of classical GR under different Linux schedulers / CPU pinning / "emulated" NBT scheduler
  - CPU core migrations
  - Cache hits/misses
  - Pretty specific, doesn't do automated reports incl. topology (yet)



[2] https://github.com/bastibl/gnuradio

[3] https://github.com/bastibl/gr-sched

NBT – Workload and Strategy

- Merge refactoring within 3.9 window
- NBT in GNU Radio 4

Questions & Answers

Ask away!