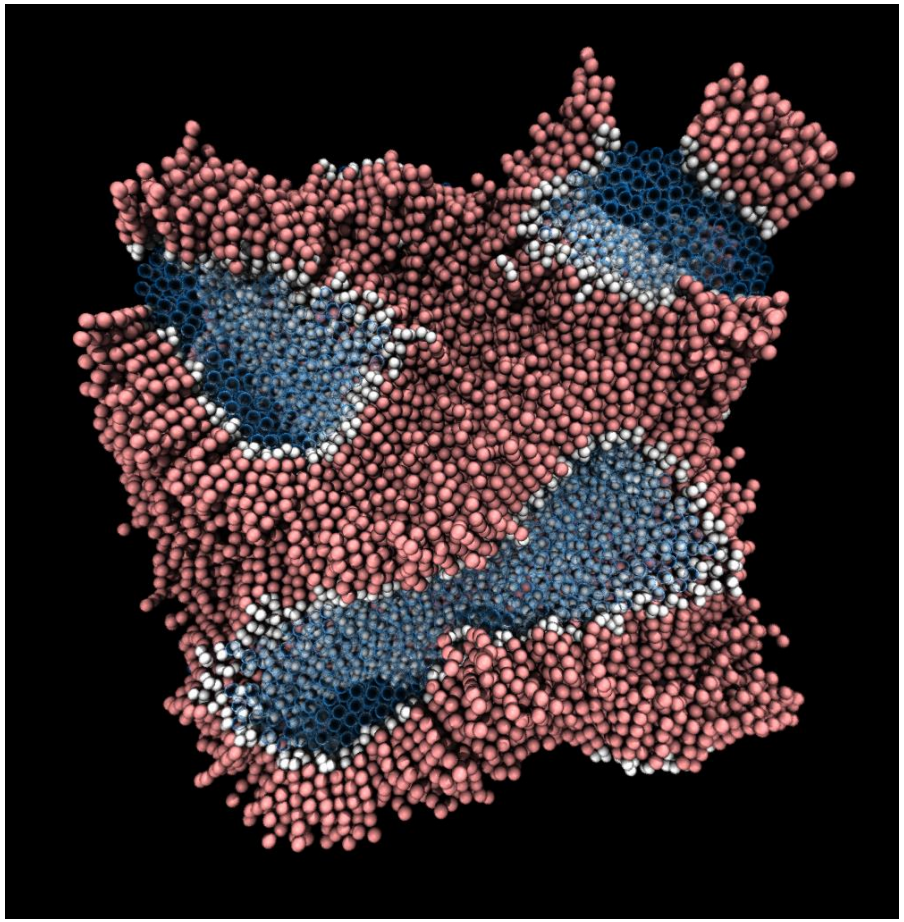
Helping build a developer community though good software practice

Dr Mark Abraham

HPC Carpentry for Chemists, University of Mauritius, August 20, 2021

ENCCS EuroHPC National Competence Centre Sweden
EURO
UPPSALA UNIVERSITET
RI. SE
Swedish Research Council
VINNOVA Sweden's Innovation Agency

Lessons from GROMACS evolution - biomolecular simulation



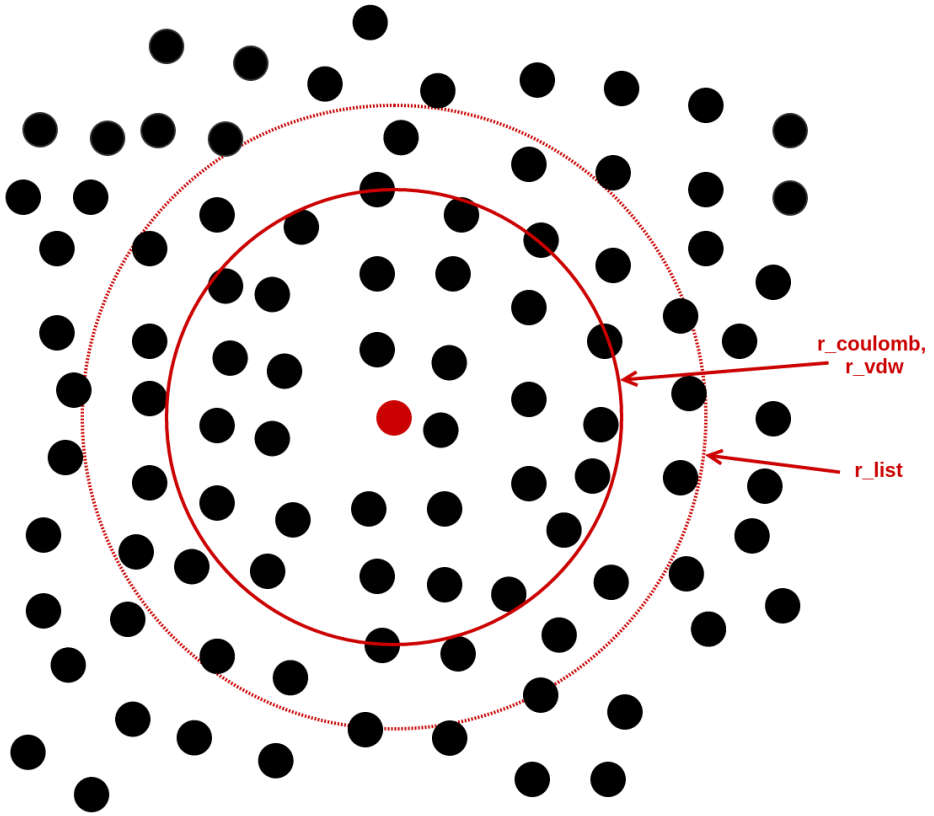
Humble origins

- Originally a **hardware** project at the University of Groningen in the early 1990s – custom-designed 32-processor ring using PVM
- Design and simulation methodology largely adopted from existing GROMOS molecular simulation package
- **Key decision**: write in a language that suits the problem you want to solve and where you want to spend your time

Early software practice

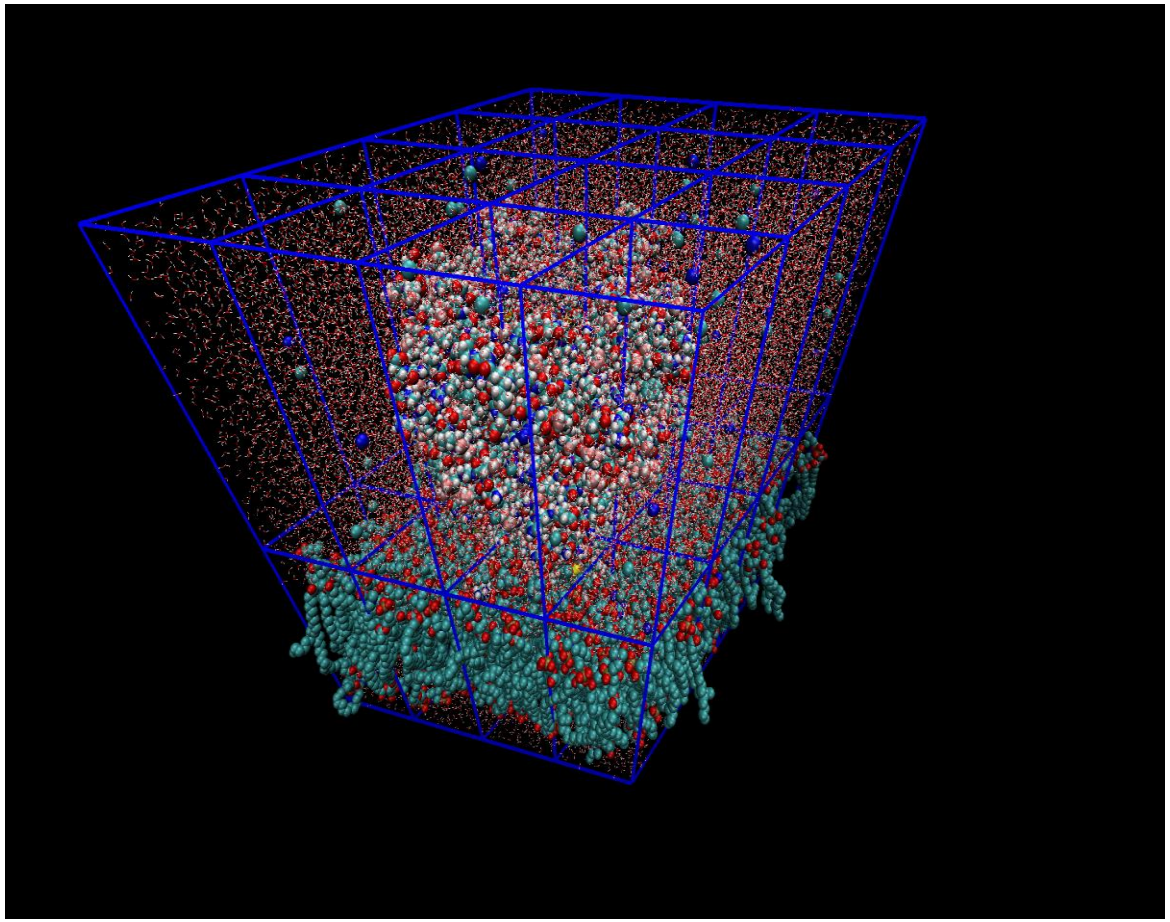
- First public release – 2.0 some time in the 1990s
- First paper – “GROMACS: a message-passing parallel molecular dynamics implementation” *Comp. Phys. Comm.* 1995
- **Key decision**: use version control – first CVS commit in 1997, already 750 files and 145k LOC
- **Key decision**: first FOSS release – 3.0 in August 2001, GPL 2.0
- **Key decision**: provide preparation and analysis tools along with the core simulation engine

N-body neighbor lists



Software practice evolves

- **Key decision**: Use issue tracking, Bugzilla introduced summer of 2005
- **Key decision**: Develop a regression test suite 2007
- **Key decision**: Switch to git for distributed version control (June 2009)
- **Key decision**: Rewrite parallelism layer, keeping the old one working

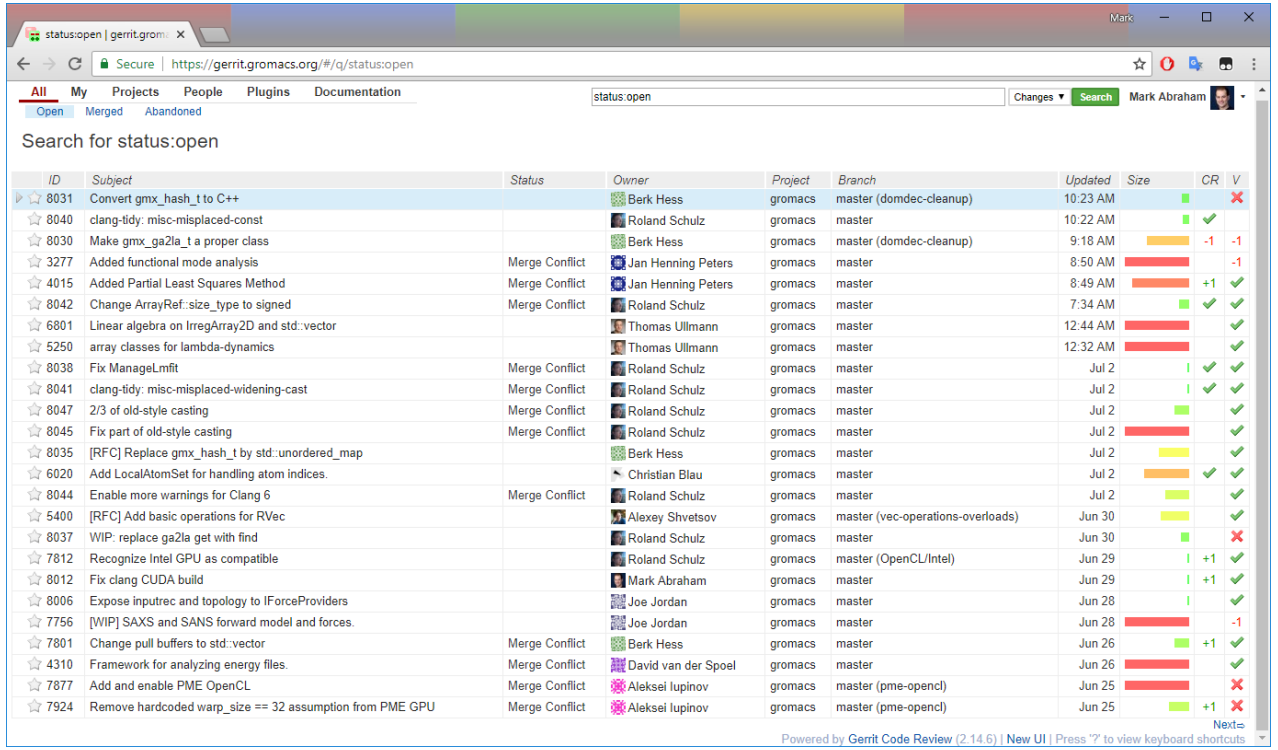


Software practice evolves…

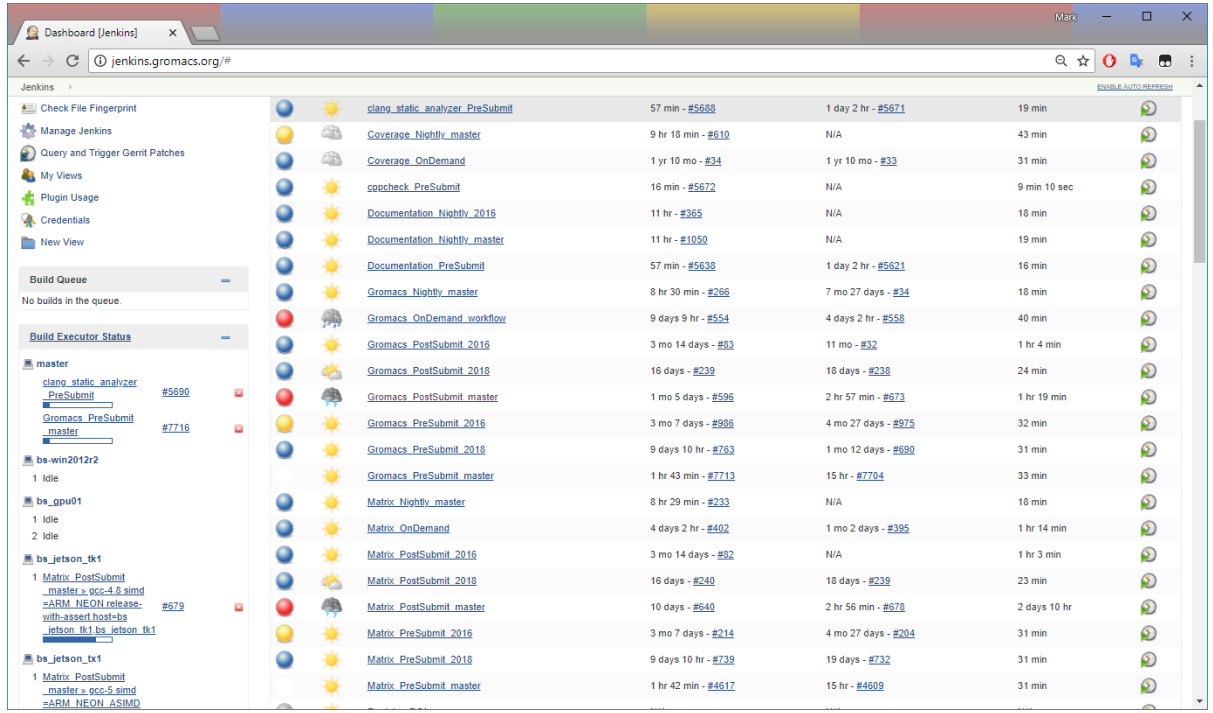
- Because of the high performance, portability, and usability, usage grew
- User community included also many with desire to modify the code and contribute back
- Many new contributors permitted to push to the central git repository over 2008-2010
- Note: not all contributions need to be new code!
- **Key decision**: unite documentation git repository with source code repository (2013)

Growth problems!

- A large community of developers who are not dog-fooding each other's code will break everything, all the time
- Many new code paths led to many bugs, some of them subtle
- **Key decision**: introduce open code review (2011) before changes are accepted – https://gerrit.gromacs.org
- **Key decision**: everybody's changes must be reviewed by others, including changes from the long-term developers (who were mostly now busy professors)



More problems!



- Portability requires ongoing testing
- Developers want to use their laptops, but HPC clusters and distributed computing environments look very different
- **Key decision**: Use continuous-integration testing, which must pass before code is considered for acceptance (2012), including static and dynamic analyzers and linting
- **Key decision**: require unit test for new functionality, add coverage also when modifying old functionality

Recent developments

- **Key decision**: Refactor to support a long-term stable API, callable from C++ and Python, easing pressure for non-core developers to want to contribute to the core, and reducing burden on core developers
- **Key decision**: switch to GitLab so everything is under one roof
- **Key decision**: use containerization to reduce CI maintenance burden

Lessons for your developer community

- Rome wasn't built in a day – pick the battles that suit the current state of your project, the needs of your users, and the resources you can afford
- Automation of processes pays off well
- Pre-commit CI testing makes bug hunting easier – you know you can go back in time, build the old code and it will work
- If you need more performance, consider your algorithms and data structures, match them to the capabilities of the target hardware.
- Plan first how you will test that your code is correct!

Thank you!

enccs.se