Scientific Software Development Infrastructure

Rossen Apostolov
KTH/PDC
rossen@kth.se

A use case:

GROMACS

GROMACS

- Project started 1995 in Groningen, most of the core developers are now in Sweden

- Highly tuned code for molecular dynamics, minimization, normal mode analysis; post-processing tools (>100)

- Open source & Free software: L-GPL

- 3000-5000 users world wide, through Folding@Home 300k active CPUs, Apr. 2012

- Strong focus on optimized algorithms and efficient code

Source code in a nutshell…

- has had 14,086 commits made by 51 contributors, representing 1,722,962 lines of code
- is mostly written in C/C++, with an average number of source code comments
- took an estimated 496 years of effort (COCOMO model), starting with its first commit in November, 1997

Taken from http://www.ohloh.net/p/gromacs

Activity

30 Day Summary

Feb 1 2014 — Mar 3 2014

83 Commits

14 Contributors

including 2 new contributors

12 Month Summary

Mar 3 2013 — Mar 3 2014

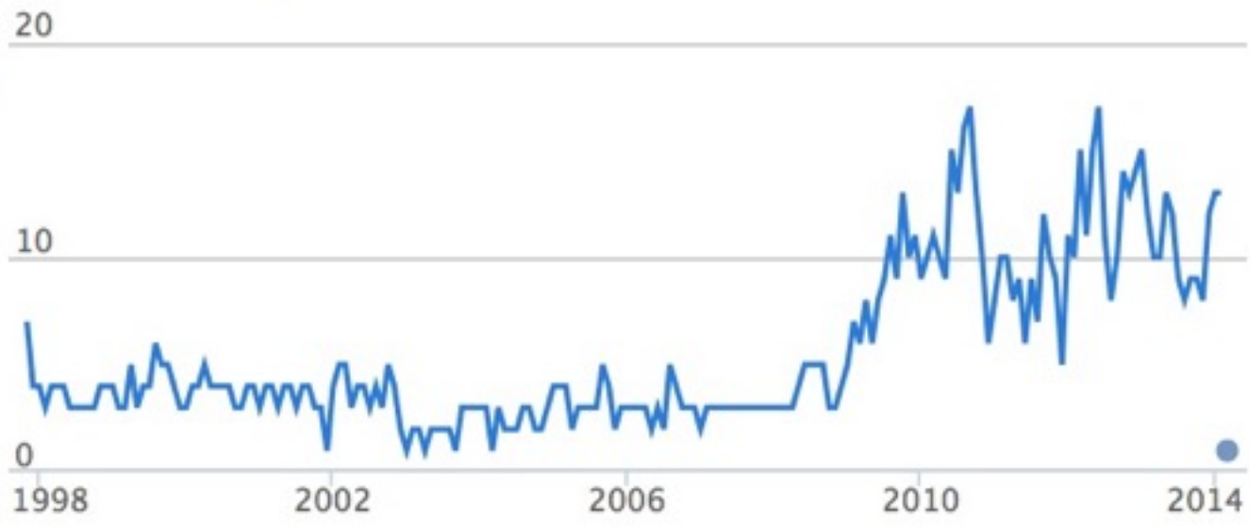
725 Commits

Down -609 (45%) from previous 12 months

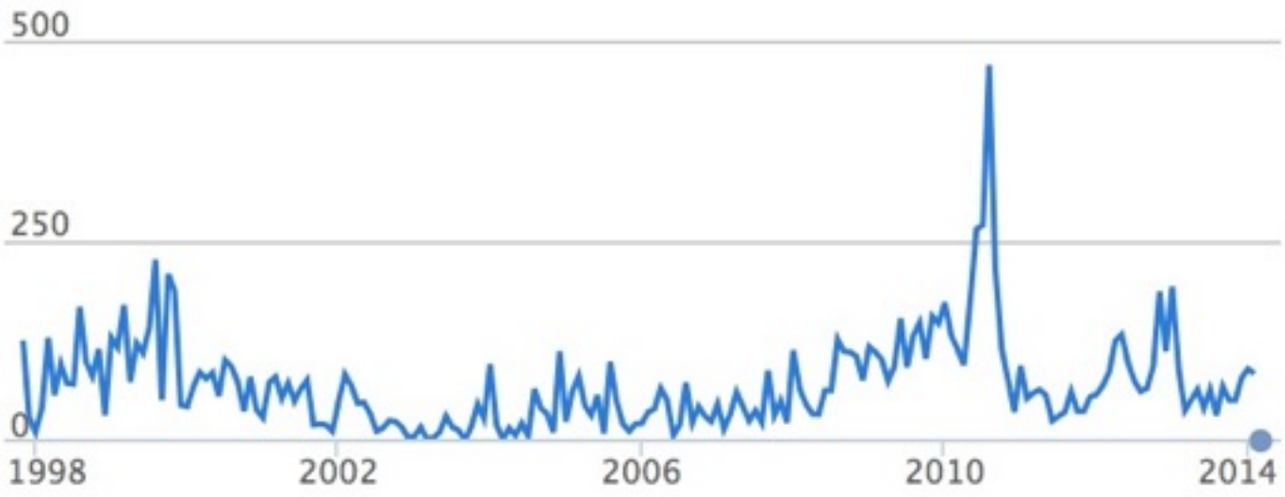
27 Contributors

Down -1 (3%) from previous 12 months

Contributors per Month



Commits per Month



Languages

C	81%	Fortran (Fixed-format)	7%
C++	7%	14 Other	5%

Lines of Code

pre- and post-2009

- CVS -> git
- Autoconf -> CMake
- No code review -> gerrit
- No auto-building -> jenkins
- Bugzilla -> Redmine

GIT

- Distributed version control system
- No real ‘central’ repository
- Very powerful
- Easy branching, easy merging, easy history changes
- Easy sharing
- **git.gromacs.org**
- other popular system is *Mercurial*

CMake

- Cross-platform!
- No dependencies (need C++ compiler)
- Generates makefile and projects
- Simple syntax
- Testing framework
- Package builders

Gerrit (code review)

- Produce better code
- Code review + verification
- Anyone can submit and review patches
- Couples great with git, jenkins
- gerrit.gromacs.org
- other systems: *Differential (Facebook), Crucible (Atlassian), pull request (Github)*

Jenkins

- Building/testing software projects continuously
- Monitoring executions of externally-run jobs
- Example uses:
  - *Unit tests; Regression tests; Static analysis; Coding style conformity; Benchmarking; Profiling; Generate documentation; etc...*
- jenkins.gromacs.org
- Another popular system is *Travis*

- Multiple projects support; Flexible role based access control
- Flexible issue tracking system; Issue creation via email
- Gantt chart; calendar; news,;documents & files management; wiki; forums; time tracking;
- SCM integration (SVN, CVS, Git, Mercurial, Bazaar and Darcs)
- Multiple LDAP authentication support
- Other issue tracking systems
  - probably >100
  - web-based ones are most popular
  - connect the systems with other tools - repository, code review etc.

GitHub

git repository

Gerrit

git repository

patch 1

patch 2

patch 3

patch 4

.....

Clone repository

Developer

local git

Push new patch

"Fixes #207"

Virtual Build Servers

Ub.10.04

Win 2008

Unix64

CentOS 6.3

Ub.12.04

Ub.13.01

Mac OS X

+2

Fetch patch for verification

Jenkins

Fetch patch for review

+1

Developer

Redmine

"Change uploaded for #207"

Now for the general use case

Status quo in e-Science

- **Software development** is the core of e-Science research

- Many groups use **outdated technologies**

- Might work for a single developer; a disaster for a team and **long-term maintenance**

- Lack of adoption due to **unfamiliarity** with existing solutions; **fear** of steep learning curve

- Course in Software development tools – success and valuable experience

Nordic status-quo

- Nordic researchers produce widely used software with big global impact
- Some groups have already adopted best practices
- Application experts with necessary knowledge
- Nordic e-infrastructure providers possess the means to improve the software development processes

Project Idea

Address the growing needs of e-Science communities by:

- Establishing a software development e-Infrastructure
- Coupling it with necessary technical expertise
- Extensive training and on-boarding activities

e-Infrastructure

- Distributed version control (DVC) repositories for scientific codes
- Issue tracking systems
- Code review systems
- Code analyzers, debuggers and profilers framework
- Build systems, unit testing frameworks and report boards
- Code Integration and benchmarking

Couple Infrastructure and Expertise

- There exists expertise among researchers, some groups already apply best practices
- Build upon experiences of e.g. SNIC application experts network to nucleate a wider communication environment
- This will be achieved naturally through jointly developing the infrastructure and running the training activities

Training and On-boarding

- Very important!
- **Extensive** training and on-boarding activities
- Organize series of training events, **as close** to users as possible
- **Quickly** bring groups up-to-speed

- Benefits for Researchers
    - Productivity
    - Quality of codes
    - Skills of developers
- Benefits for National Providers
    - Increased competence of users
    - Improve resource utilization
    - Long-term maintainability of software
- Benefits for Nordic Region
    - Strengthen the community
    - Increased interactions between groups

Questions:

- What is your experience with such systems?
- What needs do you see in your communities?