EasyBuild for Bioinformatics

A wishlist from UPPMAX

Douglas Scofield
UPPMAX / ITC / Evolutionary Biology Centre
Uppsala University, Sweden

About me, very quickly...

- BS Computer Science
- Compiler internals for 9 years in industry
- Biology much more interesting
- PhD Biology, plant biology, labwork, fieldwork (198-2004) ...
- But then bioinformatics pulled me back in
- 50% a biologist in the Evolutionary Biology Centre @ Uppsala
  - bioinformatics and other technical help in a wide variety of projects
- 50% @ UPPMAX doing multifarious bioinformatics support

Bioinformatics … is different

Traditional HPC

HPC for Bioinformatics

Measured? Yes!

- "Traditional HPC"
  - relatively few well-established tools
  - high compute expertise among users
- Bioinformatics HPC
  - biology, medicine, forestry, etc
  - many, many, many tools
  - widely varying user/developer expertise
  - many more PIs and users
  - much more scientific impact
  - especially impact / core-hour

OXFORD (GIGA)n SCIENCE

GigaScience, 7, 2018, 1–11

doi: 10.1093/gigascience/giy028
Advance Access Publication Date: 5 April 2018
Research

RESEARCH

Tracking the NGS revolution: managing life science research on shared high-performance computing clusters

Martin Dahlö [1,2,3,*,†], Douglas G. Scofield [2,4,*,†], Wesley Schaal [1,2,3] and Ola Spjuth [1,2,3]

[1]Science for Life Laboratory, Uppsala University, Uppsala, SE-750 03, Sweden, [2]Uppsala Multidisciplinary Center for Advanced Computational Science, Uppsala University, Uppsala, SE-751 05, Sweden, [3]Department of Pharmaceutical Biosciences, Uppsala University, Uppsala, SE-751 24, Sweden and [4]Department of Ecology and Genetics: Evolutionary Biology, Uppsala University, Uppsala, SE-752 36, Sweden

*Correspondence address. Martin Dahlö. E-mail: martin.dahlo@scilifelab.uu.se https://orcid.org/0000-0001-5447-9465; Douglas G. Scofield. E-mail: douglas.scofield@ebc.uu.se https://orcid.org/0000-0001-5235-6461
†These authors contributed equally to this work.

Abstract

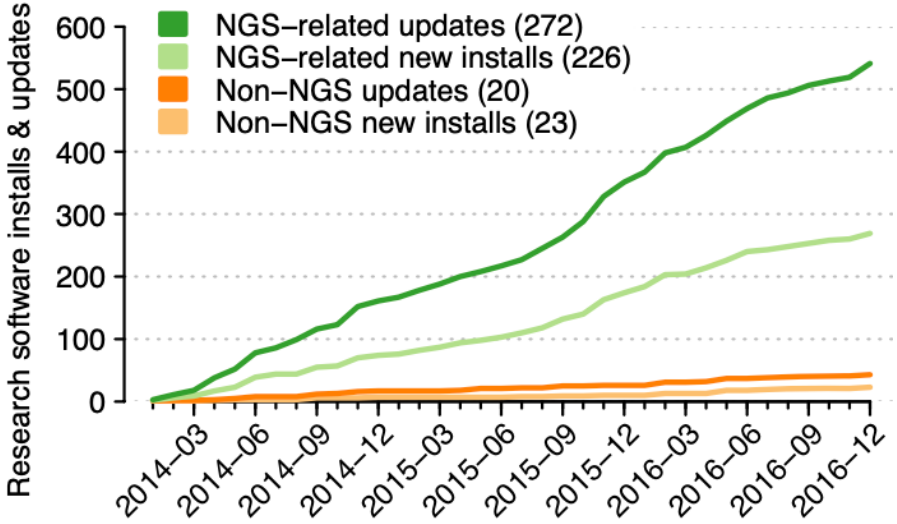
Background: Next-generation sequencing (NGS) has transformed the life sciences, and many research groups are newly dependent upon computer clusters to store and analyze large datasets. This creates challenges for e-infrastructures accustomed to hosting computationally mature research in other sciences. Using data gathered from our own clusters at UPPMAX computing center at Uppsala University, Sweden, where core hour usage of ~800 NGS and ~200 non-NGS projects is now similar, we compare and contrast the growth, administrative burden, and cluster usage of NGS projects with projects from other sciences. **Results:** The number of NGS projects has grown rapidly since 2010, with growth driven by entry of new research groups. Storage used by NGS projects has grown more rapidly since 2013 and is now limited by disk capacity. NGS users submit nearly twice as many support tickets per user, and 11 more tools are installed each month for NGS projects than for non-NGS projects. We developed usage and efficiency metrics and show that computing jobs for NGS projects use more RAM than non-NGS projects, are more variable in core usage, and rarely span multiple nodes. NGS jobs use booked resources less efficiently for a variety of reasons. Active monitoring can improve this somewhat. **Conclusions:** Hosting NGS projects imposes a large administrative burden at UPPMAX due to large numbers of inexperienced users and diverse and rapidly evolving research areas. We provide a set of recommendations for e-infrastructures that host NGS research projects. We provide anonymized versions of our storage, job, and efficiency databases.

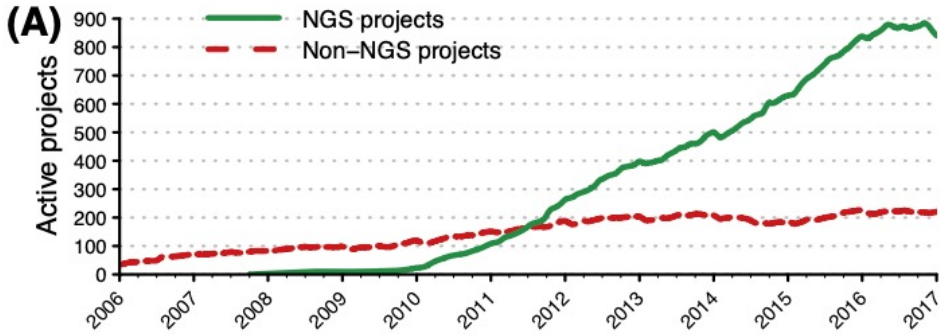
Keywords: high-performance computing; e-infrastructures; bioinformatics; resource usage efficiency; efficiency metrics; storage; next generation sequencing

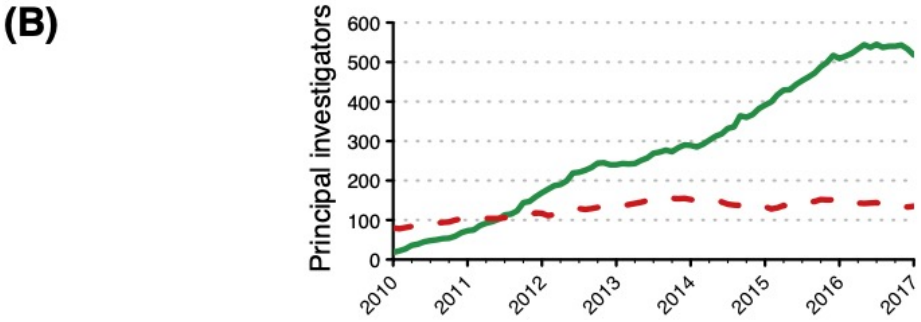
doi: 10.1093/gigascience/giy028

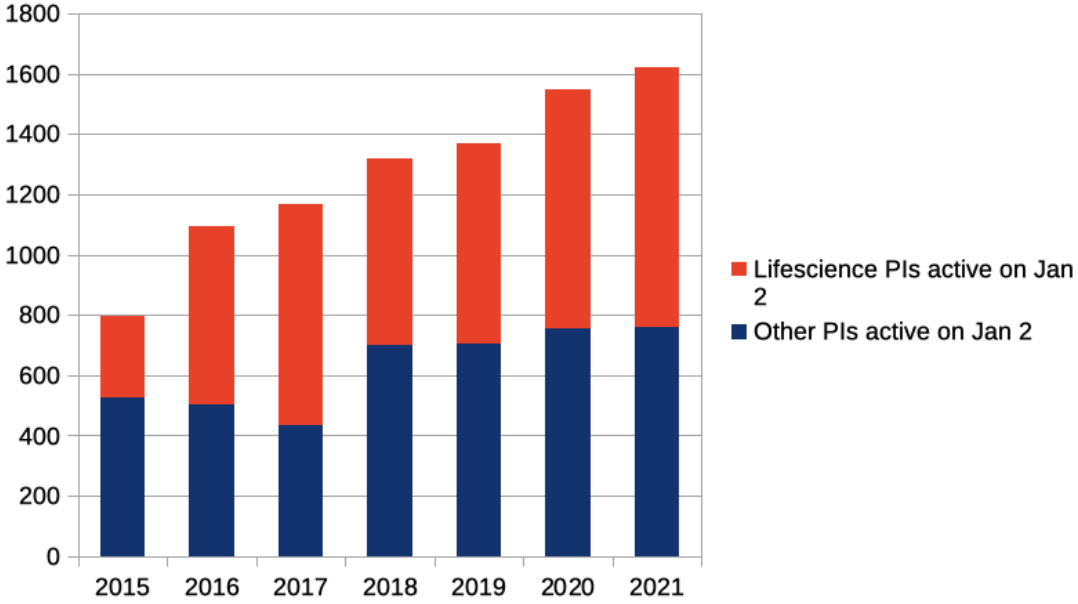
Received: 21 December 2017; **Revised:** 14 February 2018; **Accepted:** 22 March 2018

- From ~2010, growth in projects
- ... unique PIs
- ... sw installations/updates
- ... tickets and tickets/user

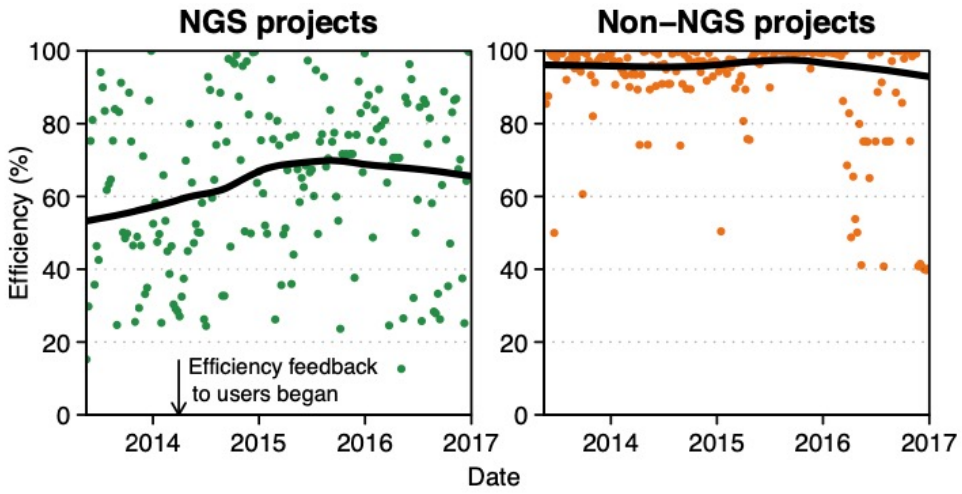


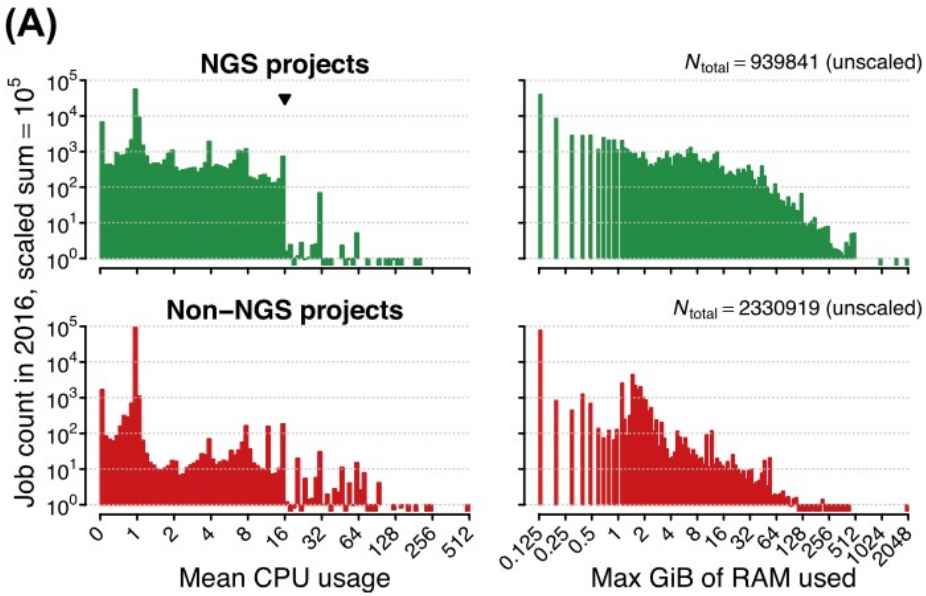


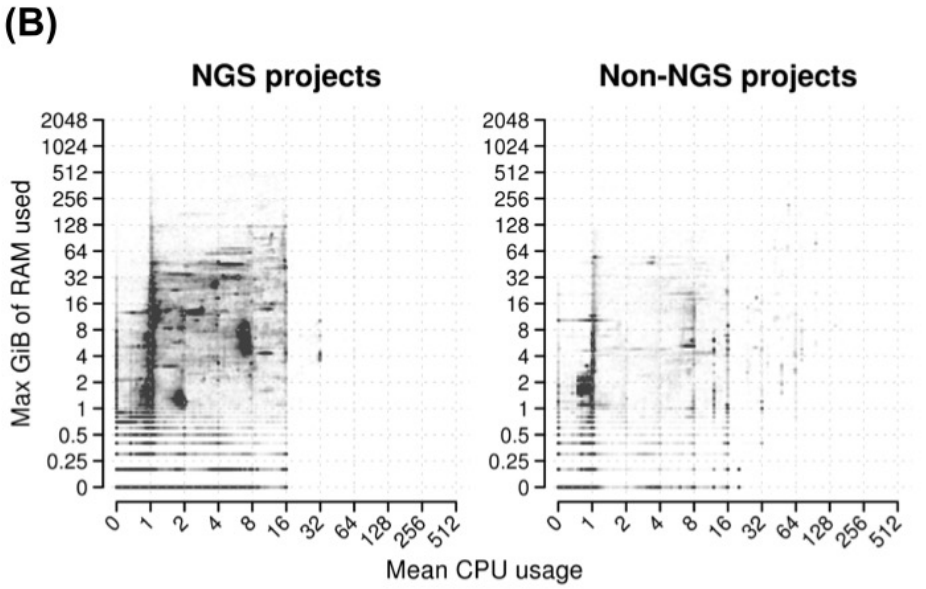




- Very few multi-node jobs
- Single-core and partial-node jobs
- Long run times (7d+)
- High RAM / core
- “Low efficiency” CPU usage
- Low core hours with high storage
  - 5k-50k h/month, but 20TB-200TB+







How do we use software?

- Pipelines typically use many heterogeneous tools
  - with a lot of user-driven swapping/experimentation
  - data flow and interchange, not CPU performance
- Extreme lack of release engineering
  - a huge problem that truly needs solving
- Nevertheless, some traditional provisioning
  - autotools, cmake, prebuilt (sometimes static) executables
- Much usage of pypi, conda
- Conda discipline is often nonexistent
- Increasing usage of containers
  - We allow Singularity (Apptainer)
- Pipelines swap environments freely

What do we want from EasyBuild?

- Being consistent with toolchains is rarely necessary
  - mostly integer work (sometimes use GMP, MPFR, others)
- Greater support for isolation, for example:
  - build samtools 1.20 (newest) with recent toolchain
  - Use simultaneously with `bwa/0.7.17` built in 2017
  - Load each only modifies PATH (plus bookkeeping), no other deps
- Is RPATH support sufficient for this?
- Can a post-hoc tool work?
  - Something like `eb isolate samtools/1.20-`*`toolchain`* to create `samtools/1.20` free of loaded dependencies
- Being tied to interpreter versions is rarely necessary
  - Python virtualenv/EBPYTHONPREFIXES so python tools using different python versions can be loaded and run at the same time

What can we give EasyBuild?

- Easyconfigs/easyblocks for all of our builds
- We have a github repository containing our encoded suffering
  - https://github.com/UPPMAX/install-methods
  - our READMEs and scripts (not always pretty)
- Data modules: what we have done
  - a data module is primarily loaded as a dependency
  - provides environment variables and sometimes extensive help
  - many data sources are versioned (`Pfam/31.0`, `Pfam/35.0`)
  - where not, we use /YYYYMMDD or `/latest` when updating is frequent
  - e.g., `blast/2.15.0+` loads `blast_databases/latest` which sets BLASTDB
- How to manage old software and old builds with reasonable life cycles