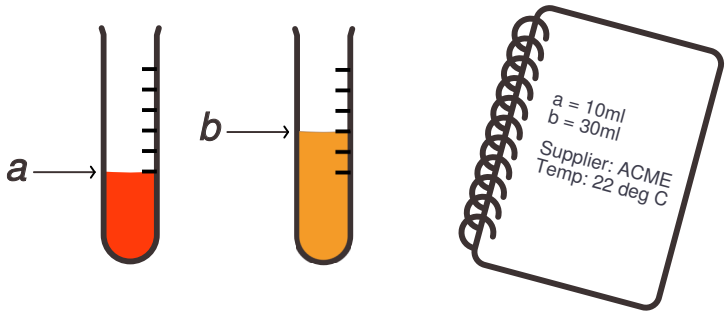
GuixHPC

Reproducible software deployment for high-performance computing.



Activity Report 2017–2018

12 February 2019
Eric Bavier, Ludovic Courtès, Paul Garlick, Pjotr Prins, Ricardo Wurmus

2.

Guix-HPC is a collaborative effort to bring reproducible software deployment to scientific workflows and high-performance computing (HPC). Guix-HPC builds upon the GNU Guix[1] software deployment tool and aims to make it a better tool for HPC practitioners and scientists concerned with reproducible research.

Guix-HPC was launched in September 2017 as a joint software development project involving three research institutes: Inria[2], the Max Delbrück Center for Molecular Medicine (MDC)[3], and the Utrecht Bioinformatics Center (UBC)[4]. GNU Guix for HPC and reproducible science has received contributions from additional individuals and organizations, including Cray, Inc.[5] and Tourbillion Technology[6].

This report highlights key achievements of Guix-HPC between its launch date in September 2017 and today, February 2019.

---

[1] *https://www.gnu.org/software/guix/*

[2] *https://www.inria.fr/en/centre/bordeaux/news/towards-reproducible-software-environments-in-hpc-with-guix*

[3] *https://www.mdc-berlin.de/*

[4] *https://ubc.uu.nl/*

[5] *https://www.cray.com*

[6] *http://tourbillion-technology.com/*

Outline

Guix-HPC started up with the following high-level objectives for the 2017–2018 period:

- *Reproducible scientific workflows.* Improve the GNU Guix tool set to better support reproducible scientific workflows and to simplify sharing and publication of software environments.
- *Cluster usage.* Streamlining Guix deployment on HPC clusters, and providing interoperability with clusters not running Guix.
- *Outreach & user support.* Reaching out to the HPC and scientific research communities and organizing training sessions.

The following sections detail work that has been carried out in each of these areas.

Reproducible Scientific Workflows

Research heavily depends on computational results, which in turn depends on the ability to reproduce software environments. As key scientific organizations such as the Association for Computer Machinery (ACM) and the Nature scientific journals begin requiring authors to publish code alongside their scientific articles, reproducing software environments remains difficult.

GNU Guix offers a way to address these issues that does not suffer from the opacity and lack of reproducibility of "container-based" solutions such as Docker or Singularity.

Software Environment Version Control

In June 2018, we developed tools to aid users who wish to have tight control over their software environments. The `guix pull` command can now be used to deploy a specific revision of Guix, and `guix describe` provides information about the currently used revision. Along with the new *channels* facility, which allows users to obtain software packages from third-party repositories, this offers a transparent way to replicate a Guix setup, as explained in the release notes of version 0.16.0[7]. Better yet, Guix allows mixing software packages coming from different Guix revisions through a new mechanism called *inferiors*.

With the help of the Software Heritage[8] engineers, we designed and implemented a back-end that allows Guix to fetch source code from Software Heritage[9]. Software Heritage is a persistent source code archive that preserves complete source code repository histories. This functionality thus allows Guix to retrieve source code even if the original source code reposi-

[7] *https://guix-hpc.bordeaux.inria.fr/blog/2018/12/hpc-reproducible-research-in-guix-0-16-0/*

[8] *https://www.softwareheritage.org*

[9] *https://issues.guix.info/issue/33432*

tory vanished or got corrupted—an obvious requirement to reproduce software environments. To our knowledge this makes Guix the first software deployment tool backed by a persistent and reliable source code archive.

GNU Guix is involved in the Reproducible Builds[10] effort. In 2018 we were again present at the Summit, along with a dozen of other projects concerned with software deployment. Together we worked to further reproducible builds and take advantage of them[11]. This is ground work that, we believe, is key to enabling reproducible scientific workflows.

Reproducible Pipelines

Jupyter Notebooks[12] have become a tool of choice for scientists willing to share, and hopefully reproduce computational experiments. Yet, nothing in a notebook specifies which software packages it relies on, which puts reproducibility at risk. For example, a notebook might rely on Python 3 and a specific version of NumPy and Scipy; if someone receives the notebook and tries to execute it with, say, Python 2 and another version of NumPy and SciPy, the result may well be different, or execution might fail altogether. To address this, during a 4-month internship at Inria, Pierre-Antoine Rouby implemented a prototype Guix "kernel" for Jupyter[13]. In a nutshell, the kernel allows notebook writers to precisely specify the software environment the notebook depends on: the Guix packages, and the Guix commit. This ensures that someone replaying the notebook will run it in the right environment as the author intended.

For less interactive computations that are to be evaluated on HPC clusters, scientists often compose applications to build so-called pipelines that express common data processing workflows. In state of the art workflow systems, it is often the users' responsibility to prepare a suitable environment in which the workflow's assumptions about software applications and

[10] *https://reproducible-builds.org*

[11] *https://www.gnu.org/software/guix/blog/2018/reproducible-builds-summit-4th-edition/*

[12] *https://jupyter.org*

[13] *https://gitlab.inria.fr/guix-hpc/guix-kernel*

libraries are satisfied. Some workflow systems allow the authors to declare a process to be dependent on software provided in a Docker application bundle, which is convenient but ignores the problem of software provenance.

As demonstrated by Pjotr Prins in a blog post[14], GNU Guix can be used to build reproducible software environments incrementally or declaratively to prepare the context in which an existing Common Workflow Language (CWL) workflow is to be executed. Compared to the use of Docker containers this unlocks software provenance and source/binary transparency while only requiring minor modifications to existing workflow definitions. While the burden of preparing the environment still lies with the user, this approach allows for a smooth transition to more reproducible workflows as Guix environments can be transparently described with a plain text manifest.

The genomics pipelines presented in the paper *PiGx: Reproducible Genomics Analysis Pipelines with GNU Guix*[15] are an example for an attempt to move the responsibility of provisioning the required software environment from the pipeline user to the package manager. By using Guix at build time the PiGx pipelines are able to benefit from reproducibile software environments and pass that benefit down to the users at runtime.

The Guix Workflow Language (GWL)[16] takes a different approach: instead of assuming that a suitable software environment is provided—by the user, by black box container images, or through a build system—it *extends* Guix itself with a workflow definition language that can make use of its rich facilities for reproducible software deployment. In the past year the GWL has gained support for a Python-like whitespace-aware workflow definitions through Wisp[17], syntax for embedding foreign language code snippets in processes, and facilities to more conveniently specify or detect data dependencies between processes.

---

[14] *https://guix-hpc.bordeaux.inria.fr/blog/2019/01/creating-a-reproducible-workflow-with-cwl/*

[15] *https://doi.org/10.1093/gigascience/giy123*

[16] *https://guixwl.org*

[17] *https://www.draketo.de/english/wisp*

The GWL and the Guix Jupyter kernel take the same approach: making reproducible software deployment a built-in feature of a larger tool. While there are other beneficial ways to integrate Guix into existing tools, as demonstrated by work on PiGx, we believe tight integration of software deployment and "workflow execution" is a novel and powerful approach that we will keep exploring.

Packaging

Since the Guix-HPC effort was started in September 2017, around 3,000 packages were added to Guix itself; of these many had to do with linear algebra, computational fluid dynamics, bioinformatics, and statistics, as reported in the HPC release notes on the Guix-HPC blog[18].

In addition, our institutes have developed their own package collections, sometimes as a staging area before packages are reviewed and integrated in Guix proper:

- The Guix-HPC repository[19] currently contains packages for HPC tools and run-time support and linear algebra libraries developed by research teams at Inria[20].
- The Guix-BIMSB repository[21] currently contains packages for bioinformatics tools and package variants used at the Berlin Institute for Medical Systems Biology[22] of the Max Delbrück Center for Molecular Medicine[23].
- This UMCU Genetics repository[24] has more bioinformatics packages in use at the Center for Molecular Medicine at UMC Utrecht[25].

---

[18] *https://guix-hpc.bordeaux.inria.fr/blog*

[19] *https://gitlab.inria.fr/guix-hpc/guix-hpc*

[20] *https://www.inria.fr/en/*

[21] *https://github.com/BIMSBbioinfo/guix-bimsb*

[22] *https://www.mdc-berlin.de/bimsb*

[23] *https://www.mdc-berlin.de*

[24] *https://github.com/UMCUGenetics/guix-additions*

[25] *http://www.umcutrecht.nl/en/Research/Research-centers/Center-for-Molecular-Medicine*

- The ACE repository[26] provides packages used by the Australian Centre for Ecogenomics[27].
- This Genenetwork repository[28] contains bioinformatics and HPC packages used by Genenetwork[29].

These package collections, along with the curated package set that comes with Guix (more than 9,000 packages), cover a wide range of HPC use cases.

One such case is numerical simulation. Development work in this area has been supported by Tourbillion Technology[30]. Within Guix a module named `(gnu packages simulation)` has been established to contain package definitions for simulation software. In 2018 the FEniCS[31] finite element framework was added to the module, complementing the OpenFOAM[32] finite volume framework that was added in 2017. Together these packages allow engineers and scientists to tackle a broad range of challenging problems within industry and academia. The ability of Guix to reproduce software environments on different systems is significantly helpful in the workflow associated with this type of project. In these projects the initial stages of model development are often undertaken on relatively small computer systems, scaling-up to HPC systems for the full computations when the models are ready. This move can sometimes be problematic, especially for models with complex dependencies on underlying libraries. By setting up consistent environments Guix ensures that even highly-complex models can be confidently transfered.

---

[26] *https://github.com/Ecogenomics/ace-guix*

[27] *http://ecogenomic.org/*

[28] *https://gitlab.com/genenetwork/guix-bioinformatics*

[29] *http://genenetwork.org/*

[30] *http://tourbillion-technology.com/*

[31] *https://fenicsproject.org/*

[32] *https://openfoam.org/*

Guix added support in its 0.13.0 release of mid-2017 for ARM's 64-bit aarch64 processors, which are becoming increasingly popular as a target for HPC clusters. Since then, several core linear algebra and maths libraries have had work done, thanks to Cray Inc[33], to better support this architecture.

[33] *http://www.cray.com/*

Cluster Usage

GNU Guix has been deployed on clusters at our research institutes and in other places. One of our first task has been to simplify the deployment and installation of Guix on clusters, providing new features for distributed setups to its build daemon and command-line tools, and documenting the installation process for system administrators[34]. This is the option we recommend because it gives cluster users a lot of flexibility: each user can install, upgrade, and remove packages at will, create software environments on the fly with `guix environment`, and so on.

However, scientists may also need to target clusters where Guix is not installed, and we wanted to offer interoperability with those. As so-called "container-based solutions" like Docker and Singularity are being deployed on clusters, we developed `guix pack`, a tool that can create "container images"[35]. In this setup, users use `guix pack` on their laptop to generate an image that contains precisely the software environment they need, and then send it over to the cluster to run their application. `guix pack` can generate images usable by both Singularity and Docker; it can also generate tarballs containing relocatable executables[36]. This interoperability tool allows users to not give up on the reproducibility, transparency, and flexibility benefits offered by Guix.

To help cluster users get started with Guix, hpcguix-web[37], initially developed at the UMC Utrecht, provides a web interface that allows users to search for software packages and to learn how to install them. A public instance is visible on the Guix-HPC web site[38]. Hpcguix-web is customizable

---

[34] *https://guix-hpc.bordeaux.inria.fr/blog/2017/11/installing-guix-on-a-cluster/*

[35] *https://guix-hpc.bordeaux.inria.fr/blog/2017/10/using-guix-without-being-root/*

[36] *https://www.gnu.org/software/guix/blog/2018/tarballs-the-ultimate-container-image-format/*

[37] *https://github.com/UMCUGenetics/hpcguix-web*

[38] *https://guix-hpc.bordeaux.inria.fr/browse*

and the instance running in Utrecht provides users with additional information such as how to use the batch scheduler.

Outreach and User Support

One aspect of our work on Guix-HPC is to "spread the word" about the importance of being able not just to replicate software environments, but also to inspect and modify those software environments. These are key to proper scientific understanding and experimentation. This section summarizes articles we published and talks we gave around these topics.

Articles

Since the inception of Guix-HPC, two scientific articles were published in peer-reviewed conferences:

- *Code Staging in GNU Guix*[39] (Ludovic Courtès, Oct. 2017) discusses programming language design issues. It was presented at the 16th International Conference on Generative Programming: Concepts & Experience (GPCE 2017)[40].
- *PiGx: Reproducible Genomics Analysis Pipelines with GNU Guix*[41] (Ricardo Wurmus et al, Dec. 2018) was published in the Open Access journal GigaScience and was presented at the International Conference on Genomics (ICG-13)[42] where it was awarded 2nd Runner Up in the GigaScience Prize Track[43].

Altuna Alkalin, research team leader at the Max Delbrück Center (MDC), wrote an article entitled *Scientific Data Analysis Pipelines and Reproducibility*[44] (Oct. 2018). The article discusses the "reproducibility spectrum"

---

[39] *https://hal.inria.fr/hal-01580582/en*

[40] *http://conf.researchr.org/home/gpce-2017*

[41] *https://doi.org/10.1093/gigascience/giy123*

[42] *http://www.icg-13.org/*

[43] *https://guix-hpc.bordeaux.inria.fr/blog/2019/01/pigx-paper-awarded-at-the-international-conference-on-genomics-icg-13/*

and compares existing tools to achieve software environment reproducibility: application bundles (also referred to as "containers"), CONDA, and Guix.

We have published 12 articles on this blog[45] touching a range of technical topics: running Guix without being root, on the performance of pre-built binaries, creating reproducible workflows with CWL or PiGx, and more.

In September 2017, Inria, the MDC, and the Utrecht Bioinformatics Center published an article[46] for the project launch. On-line magazine HPC Wire covered it[47].

Talks

Since Guix-HPC was started, we gave talks at a number of venues:

- EasyBuild User Days, Jan. 2018[48] (Ricardo Wurmus, Pjotr Prins)
- GNU Guix Day, Feb. 2018[49] (Ludovic Courtès, Roel Janssen, Pjotr Prins, Ricardo Wurmus)
- HPC track at FOSDEM, Feb. 2018[50] (Ludovic Courtès)
- CERN, May 2018[51] (Ricardo Wurmus, Ludovic Courtès)
- Software development plenary, Inria, May 2018
- Bio-IT World, Data Computing track, May 2018[52] (Ricardo Wurmus)

---

[44] *https://medium.com/@aakalin/scientific-data-analysis-pipelines-and-reproducibility-75ff9df5b4c5*

[45] *https://guix-hpc.bordeaux.inria.fr/blog*

[46] *https://www.inria.fr/en/centre/bordeaux/news/towards-reproducible-software-environments-in-hpc-with-guix*

[47] *https://www.hpcwire.com/off-the-wire/free-software-helps-tackle-reproducibility-problem/*

[48] *https://github.com/easybuilders/easybuild/wiki/3rd-EasyBuild-User-Meeting*

[49] *https://libreplanet.org/wiki/Group:Guix/FOSDEM2018*

[50] *https://archive.fosdem.org/2018/schedule/event/guix_workflows/*

[51] *https://cds.cern.ch/record/2316926*

[52] *https://www.bio-itworldexpo.com/18/data-computing*

- International Conference on Genomics (ICG-13), Oct. 2018[53] (Ricardo Wurmus)
- JCAD, Nov. 2018[54] (Ludovic Courtès)
- iHub Nairobi, May 2018[55] (Pjotr Prins)
- Biohackathon Japan, Dec. 2018[56] (Pjotr Prins)
- Minimalistic languages track at FOSDEM, Feb. 2019[57] (Ricardo Wurmus)
- Distributions track at FOSDEM, Feb. 2019[58] (Ludovic Courtès)

Training Sessions

We've organized a number of Guix training sessions for HPC, in particular at Inria (March and October 2018), at the MDC (October 2018), and UR-FIST, the French unit for training and scientific information in Bordeaux (November 2018).

---

[53] *http://www.icg-13.org/*

[54] *https://jcad2018.sciencesconf.org/resource/page/id/7*

[55] *https://ihub.co.ke/event/75/pjotr-prins-in-nairobi-on-functional-programming-hpcs-in-research-gnu-guix*

[56] *http://2018.biohackathon.org/symposium*

[57] *https://fosdem.org/2019/schedule/event/guixinfra/*

[58] *https://fosdem.org/2019/schedule/event/gnu_guix_new_approach_to_software_distribution/*

Personnel

GNU Guix is a collaborative effort, receiving contributions from more than 40 people every month. As part of Guix-HPC though, participating institutions have dedicated work hours to the project, which we summarize here.

- Cray, Inc.: 0.4 person-year (Eric Bavier)
- Inria: 2 person-years (Ludovic Courtès) + 4 person-months (Pierre-Antoine Rouby, intern)
- Max Delbrück Center for Molecular Medicine (MDC): 2 person-years (Ricardo Wurmus)
- Tourbillion Technology: 0.5 person-year (Paul Garlick)
- University of Tennessee Health Science Center (UTHSC): 0.3 person-year (Pjotr Prins)
- Utrecht Bioinformatics Center (UBC): 1 person-year (Roel Janssen)

Perspectives

In the coming years, we plan to continue our development efforts. Some of them concern directly the core of Guix. In particular, we would like to further simplify the use of Guix on clusters where Guix is not installed by refining the `guix pack` tool or by turning the build daemon into a library that would make it easy to run builds as non-root on such systems. We may also provide services built around that; `guix pack` as a service, for instance, would make it easy for users to build "container images" in a reproducible fashion.

Of the more exciting developments, work on the GWL and on the Jupyter kernel shows that integrating reproducible software deployment capabilities with existing applications is a fruitful endeavor. We believe it's an easy way to bring reproducible deployment into the hands of scientists, directly within the tools that they use daily. We will continue working in that direction, and we hope to extend to other tools as well—workflow management tools, batch schedulers, and "active paper" authoring tools come to mind.

Beyond development, one of the missions of Guix-HPC is to support reproducible science efforts. Achieving fully reproducible scientific computing pipelines is a lot of work in itself, but we believe reproducible software deployment is a prerequisite. We will continue publishing on this topic, giving talks and training sessions, and generally raise awareness of the importance of reproducible software deployment, the ways Guix helps achieve that, and the shortcomings of popular approaches such as Docker-style application bundles. We hope to work more closely with initiatives such as ReScience[59] and with scientific societies to investigate ways Guix could help improve their reviewing workflows.

We are very much open to new ideas and we'd like to hear from you[60]!

---

[59] *https://rescience.github.io*

[60] *https://guix-hpc.bordeaux.inria.fr/about*

Credits

20.