Journée SED 2022

Distribution et déploiement reproductible des logiciels de recherche

Sommaire

01.. Distribution Solveurs
02.. Déploiement logiciel
03.. Recherche

reproductible

& science ouverte
04.. Sujet de recherche

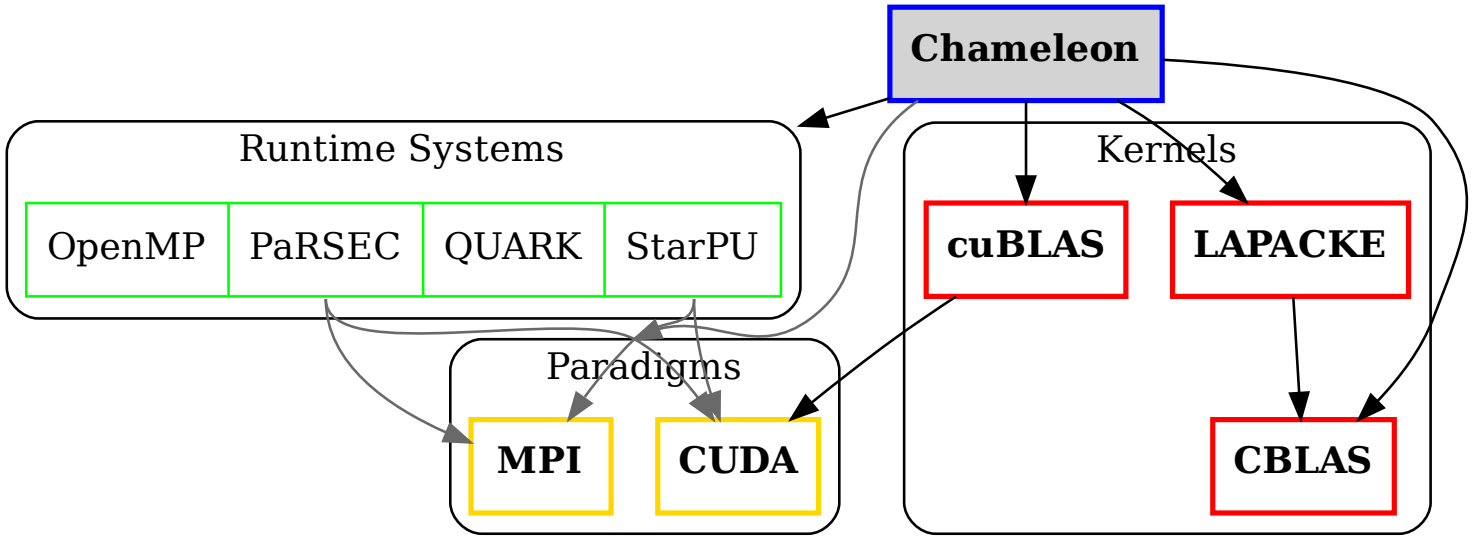
Inria

01

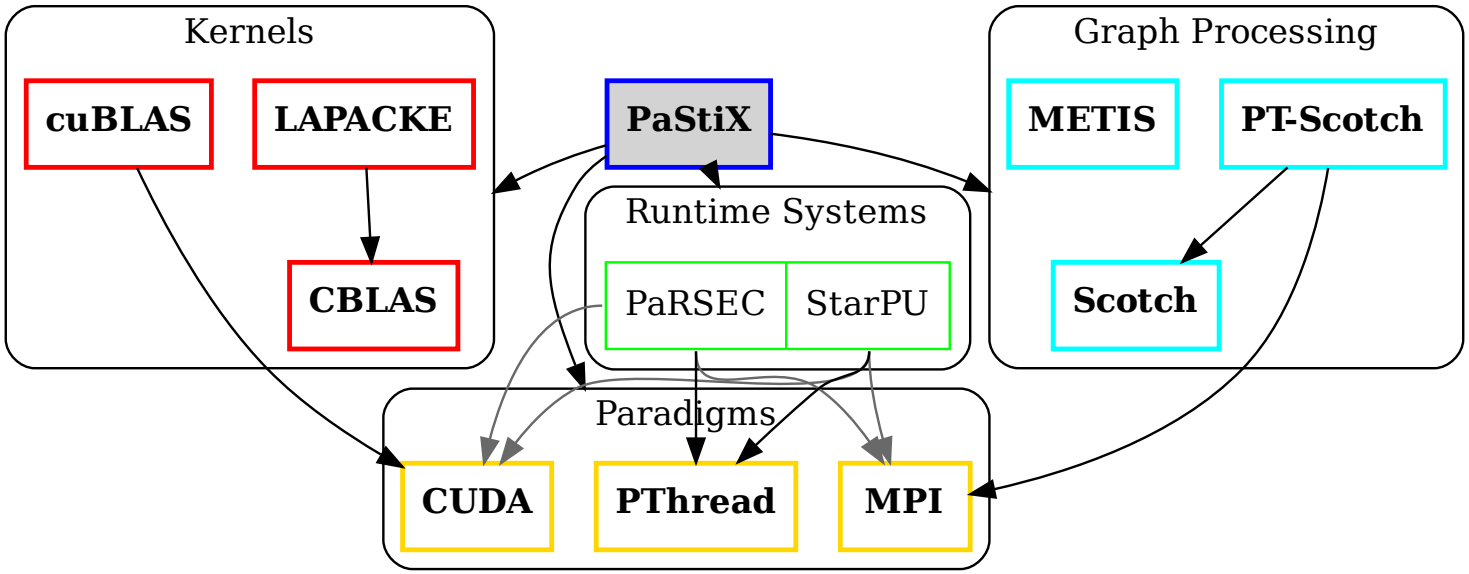
Distribution Solveurs

Inria

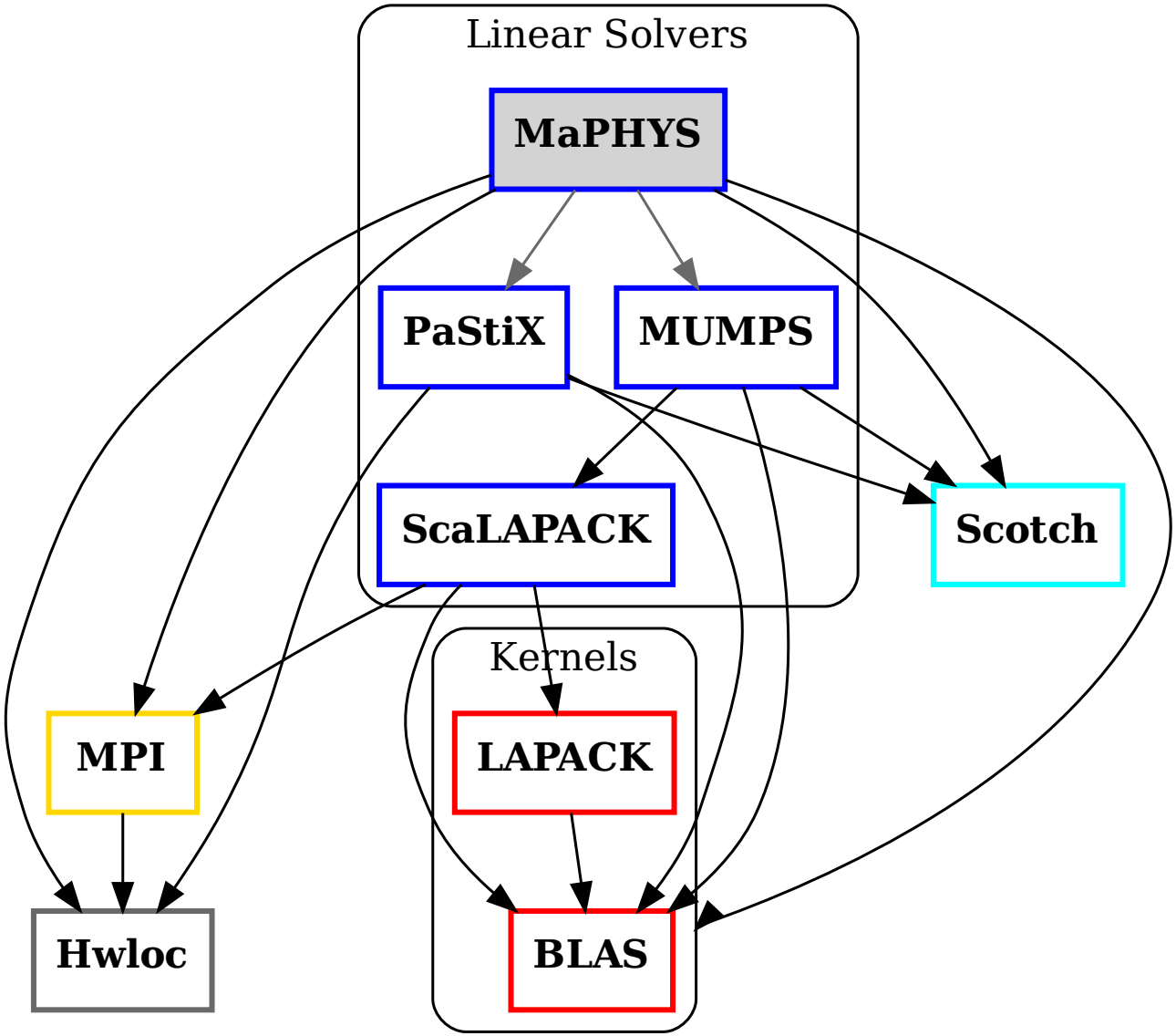
Solveurs - EP Concace, Topal, ...



Solveurs - EP Concace, Topal, ...



Solveurs - EP Concace, Topal, ...



- logiciels : chameleon, compose, pastix, qr_mumps, scalfmm
- partage/factorisation de pratiques CMake 
- fichier de configuration installé *e.g.* `SCOTCHConfig.cmake`
- système de détection des dépendances similaire

```
find_package(PASTIX) --> target PASTIX::pastix, PASTIX_LIBRARIES
```

- modules `find_package` pour les dépendances non CMake :

solvers	mumps, hypre, petsc, plasma, slepc, scalapack
runtimes	quark, parsec, starpu
kernels	cblas, lapacke, fftw
graphs	(par)metis, (pt)scotch, pampa
misc	hwloc, fxt, eztrace, simgrid, spm

https://gitlab.inria.fr/solverstack/morse_cmake

CI et environnement de tests Docker/Guix

Add slave

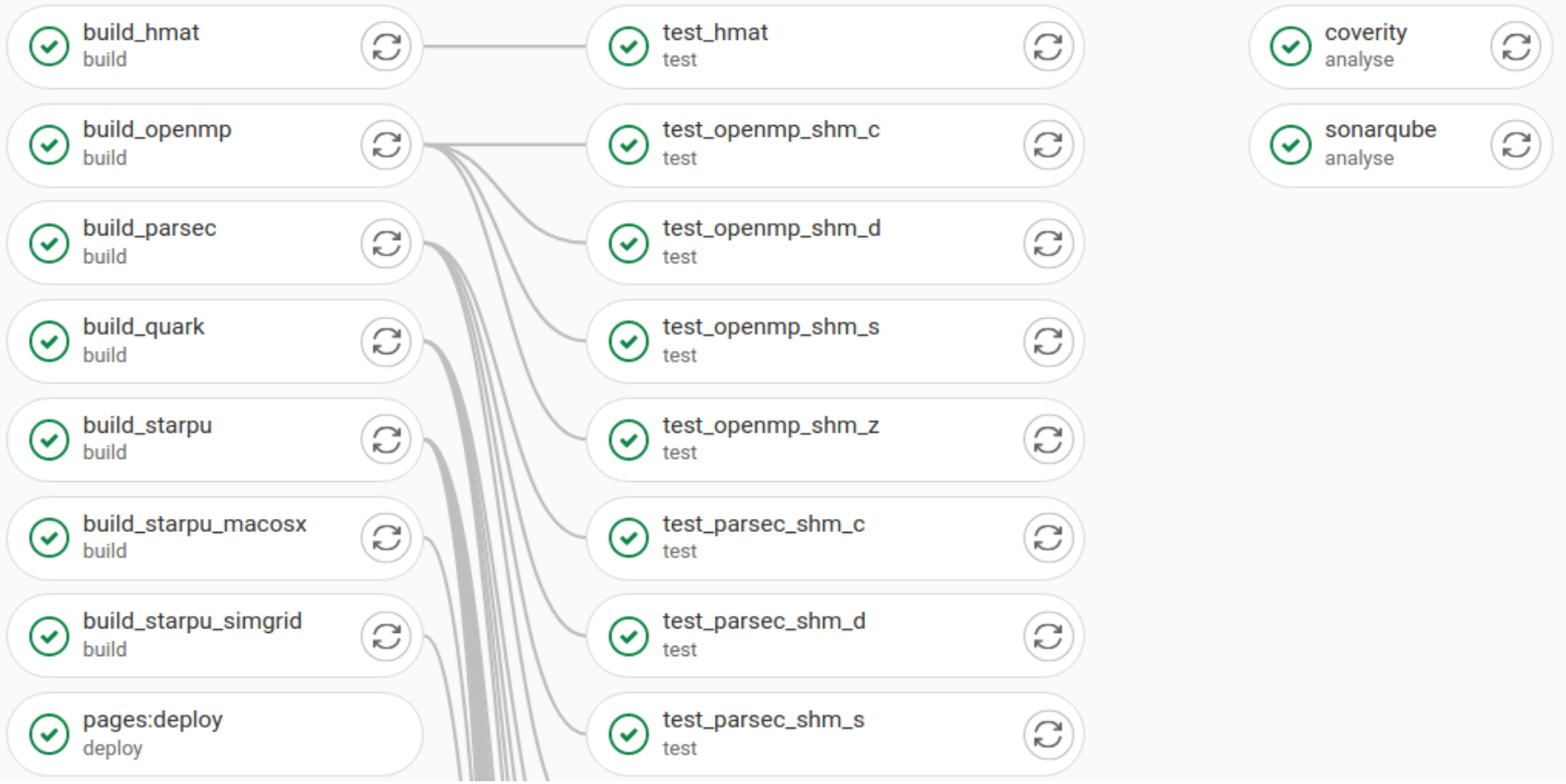
Status	Display name	Hostname/IP	OS	CPU/Memory	Created	Expire	Actions
Running	morse-ubuntu16-maphys	morse-ubuntu16-maphys / 172.21.13.118	ubuntu-16.04.3-server-amd64	4*2 Ghz / 12 Gb	15/09/2017 15:55	18/04/2023 10:00 Extend	Connect Stop Delete
Running	morse-ubuntu16-scalfmm	morse-ubuntu16-scalfmm / 172.21.14.36	ubuntu-16.04.3-server-amd64	4*2 Ghz / 12 Gb	15/09/2017 11:06	18/04/2023 10:00 Extend	Connect Stop Delete
Running	morse-ubuntu16-pastix	morse-ubuntu16-pastix / 172.21.12.246	ubuntu-16.04.3-server-amd64	4*2 Ghz / 12 Gb	15/09/2017 10:12	18/04/2023 10:00 Extend	Connect Stop Delete
Running	morse-ubuntu16-chameleon	morse-ubuntu16-chameleon / 172.21.13.6	ubuntu-16.04.3-server-amd64	4*2 Ghz / 12 Gb	14/09/2017 11:45	18/04/2023 10:00 Extend	Connect Stop Delete
Running	morse-cmake-modules	morse-cmake-modules / 172.21.11.189	ubuntu-16.04.3-server-amd64	1*1.024 Ghz / 1 Gb	14/09/2017 14:51	18/04/2023 10:00 Extend	Connect Stop Delete

CI et environnement de tests Docker/Guix

```
FROM registry.gitlab.inria.fr/solverstack/docker/ci

USER root

# Installing as root: docker images are usually set up as root.
# Since some autotools scripts might complain about this being unsafe, we set
# FORCE_UNSAFE_CONFIGURE=1 to avoid configure errors.
ENV FORCE_UNSAFE_CONFIGURE=1
ENV DEBIAN_FRONTEND noninteractive

# Install common packages
RUN apt-get update && apt-get install -y \
  gnupg2 \
  libacl1-dev \
  libboost-dev \
  libboost-context-dev \
  libfftw3-dev \
  libfxt-dev \
  libgtg-dev \
  libhypre-dev \
  liblapack-dev \
  liblapacke-dev \
  libmetis-dev \
  libmumps-dev \
  libopenmpi-dev \
  libpapi-dev \
  libparmetis-dev \
  libpfm4-dev \
  libptscotch-dev \
  libscalapack-mpi-dev \
  libsuitesparse-dev \
  nvidia-cuda-dev \
  nvidia-cuda-toolkit \
  ocl-icd-opencl-dev \
  petsc-dev \
  slepc-dev \
  zlib1g-dev \
  python3-numpy \
  python3-mpi4py \
  python3-scipy && \
  apt-get autoremove -y

RUN apt-get autoremove -y
RUN apt-get autoclean -y
RUN apt-get purge -y

# For cppdiodon
RUN apt-get install -y libhdf5-mpi-dev

# change the default shell to be bash
SHELL ["/bin/bash", "-c"]

# Switch to gitlab user to install specific libraries in its home
```

```
  export CHAMELEON_BUILD_OPTIONS="-DCHAMELEON_USE_MPI=ON -DCHAMELEON_USE_CUDA=ON -DCMAKE_BUILD_TYPE=Rel
ease"
  export STARPU_HOSTNAME="sirocco"
  export LD_PRELOAD="/usr/lib64/libcuda.so /usr/lib64/libnvidia-fatbinaryloader.so.440.33.01"
else
  echo "$0: Please set the NODE environnement variable to bora or sirocco."
  exit -1
fi

# define env var and guix rule to use depending on the mpi vendor
GUIX_ENV="chameleon"
if [ $NODE = "sirocco" ]
then
  GUIX_ENV="chameleon-cuda"
fi
export MPI_OPTIONS=""
if [ $MPI = "openmpi" ]
then
  if [ $NODE = "bora" ]
  then
    export MPI_OPTIONS="--map-by ppr:1:node:pe=36"
  fi
  GUIX_ENV_MPI=""
  GUIX_ADHOC_MPI="openssh openmpi"
elif [ $MPI = "nmad" ]
then
  export MPI_OPTIONS="-DPIOM_DEDICATED=1 -DPIOM_DEDICATED_WAIT=1"
  GUIX_ENV_MPI="--with-input=openmpi=nmad --with-branch=starpu=starpu-1.3"
  GUIX_ADHOC_MPI="which gzip zlib tar inetutils util-linux procps openssh nmad"
else
  echo "$0: Please set the MPI environnement variable to openmpi or nmad."
  exit -1
fi
GUIX_ADHOC="coreutils gawk grep jube perl python python-click python-certifi python-elasticsearch pytho
n-gitpython python-matplotlib python-pandas python-seaborn r-ggplot2 r-plyr r-reshape2 sed slurm mkl"
GUIX_RULE="-D $GUIX_ENV $GUIX_ENV_MPI $GUIX_ADHOC $GUIX_ADHOC_MPI"

# Submit jobs

# OpenMPI version
exec guix shell --pure \
                --preserve=PLATFORM \
                --preserve=NODE \
                --preserve=LD_PRELOAD \
                --preserve=^CI \
                --preserve=^SLURM \
                --preserve=^JUBE \
                --preserve=^MPI \
                --preserve=^STARPU \
                --preserve=^CHAMELEON \
                $GUIX_RULE \
                -- /bin/bash --norc ./tools/bench/plafrim/slurm.sh
```

Pipeline Gitlab-CI



Pipeline Gitlab-CI

```
stages:
  - build
  - test
  - sonar

# JOB_INT: [ 32 | 64 | 32-64 ]
# JOB_THREAD: [ n | c | k | s | m ]
# JOB_DEBUG: [ n | a | f ]
# JOB_RENAME: [ n | r ]

.job_template: &matrix_32
  parallel:
    matrix:
      - JOB_INT: [ 32, 32-64 ]
        JOB_THREAD: [ n ]
        JOB_DEBUG: [ n ]
        JOB_RENAME: [ n ]
      - JOB_INT: [ 32-64 ]
        JOB_THREAD: [ n, k ]
        JOB_DEBUG: [ a ]
        JOB_RENAME: [ n ]
      - JOB_INT: [ 32 ]
        JOB_THREAD: [ n, c, m ]
        JOB_DEBUG: [ f ]
        JOB_RENAME: [ n ]
```

```
-:---  .gitlab-ci.yml<scotch>   Top (1,0)     Git-master  (Fundamental)
```

```
--
.build_script_template: &build_script
  stage: build
  tags: ["docker", "large"]
  artifacts:
    name: pastix_build_${VERSION}
    expire_in: 180 minutes
    untracked: true
    paths:
      - install-${VERSION}
      - build
      - pastix-build-${VERSION}.log
  script:
    - cd build
    - echo $VERSION | tee ../pastix-build-${VERSION}.log
    - cmake -DPASTIX_CI_VERSION=${VERSION}
            -DPASTIX_CI_BRANCH=${BRANCH}
            -C ../.gitlab/ci-test-initial-cache.cmake  ..
    - make -j 4    | tee -a ../pastix-build-${VERSION}.log
    - make install | tee -a ../pastix-build-${VERSION}.log
  only:
    - branches
    - master@pastix/pastix
    - master@solverstack/pastix

.build_script_template_seq: &build_script_seq
```

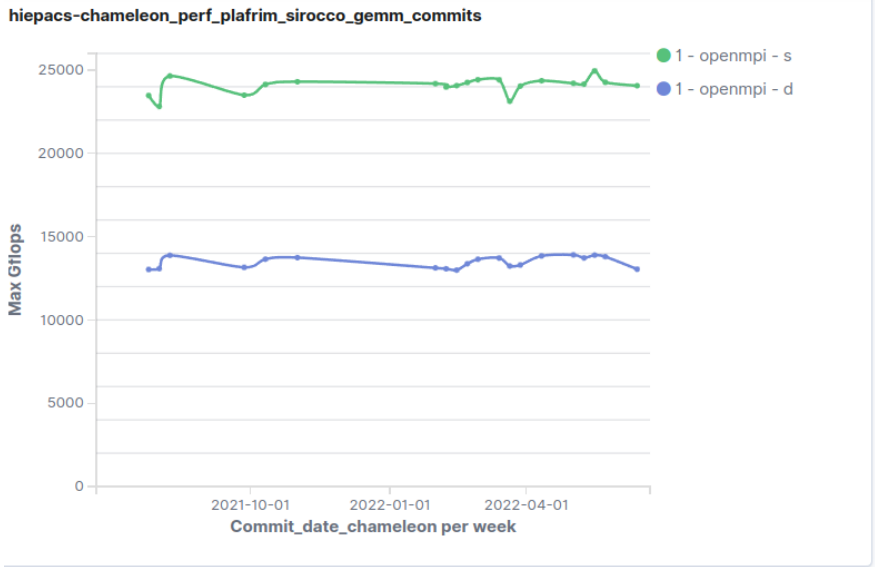
```
-:---  build.yml<pastix>   Top (1,0)     Git-master  (Fundamental)
```

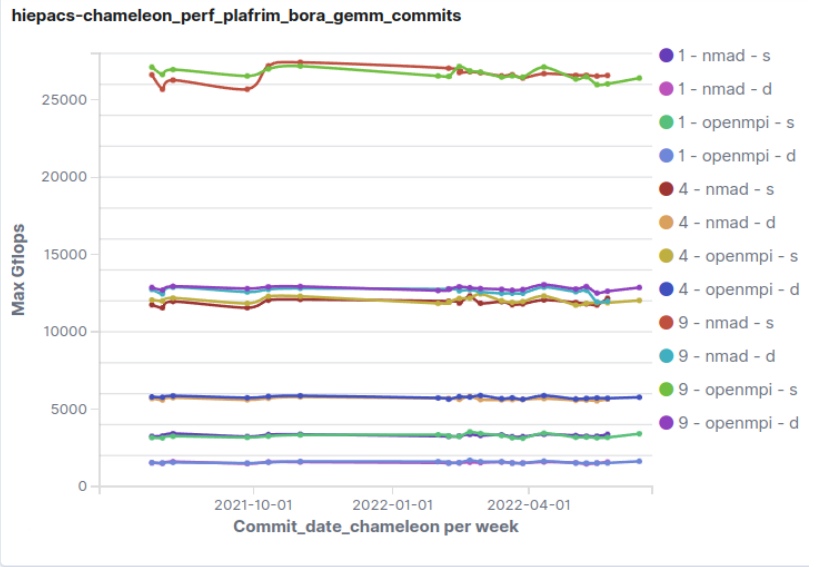
```
--
build_openmp:
  extends: .build_script_template
  artifacts:
    name: build_openmp
    expire_in: 48 hours
    paths:
      - build-openmp
      - chameleon_openmp.log
  variables:
    BUILD_OPTIONS: "-DCHAMELEON_SCHED=OPENMP"
    VERSION: openmp

build_parsec:
  extends: .build_script_template
  artifacts:
    name: build_parsec
    expire_in: 48 hours
    paths:
      - build-parsec
      - chameleon_parsec.log
  variables:
    BUILD_OPTIONS: "-DCHAMELEON_SCHED=PARSEC"
    VERSION: parsec

build_quark:
```

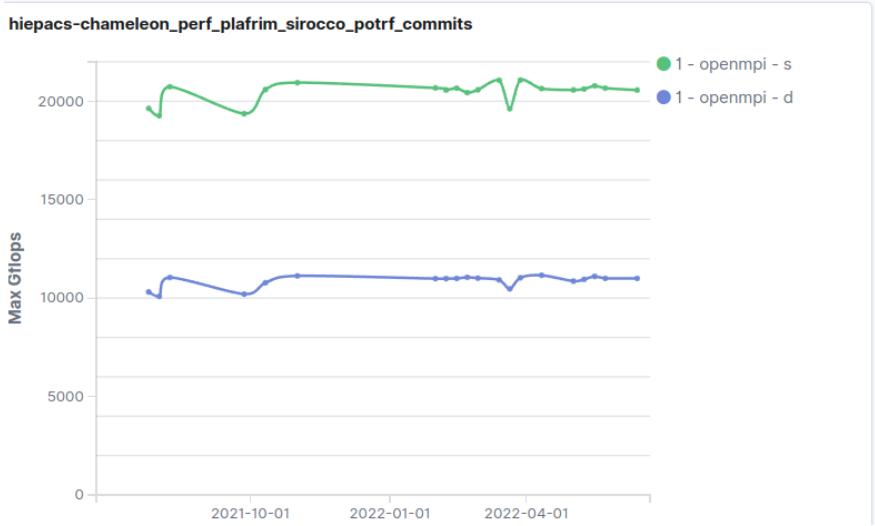
```
variables:
  GIT_SUBMODULE_STRATEGY: recursive

unit-tests:
  tags: ['guix']
  script:
    - guix time-machine -C .guix-extra/channels-fixed.scm -- describe
    - git submodule update --init --recursive
    - mkdir build install
    - cd build
    - echo 'export OMPI_MCA_rmaps_base_oversubscribe=1' > testscript.sh
    - echo 'cmake .. -DCMAKE_CXX_FLAGS="--coverage" -DMAPHYSPP_GCC_WARNINGS=ON -DMAPHYSPP_USE_FABULOUS=
ON' >> testscript.sh
    - echo 'ctest -D ExperimentalBuild -D ExperimentalTest' >> testscript.sh
    - chmod +x testscript.sh
    - guix time-machine -C ../.guix-extra/channels-fixed.scm -- shell --pure -D maphys++ arpack-ng-3.8
eigen emacs emacs-org zlib fabulous -- ./testscript.sh

  only:
    refs:
      - branches
    variables:
      - ($CI_COMMIT_BRANCH == "develop")
    changes:
      - include/**/*
      - .gitlab-ci.yml
```

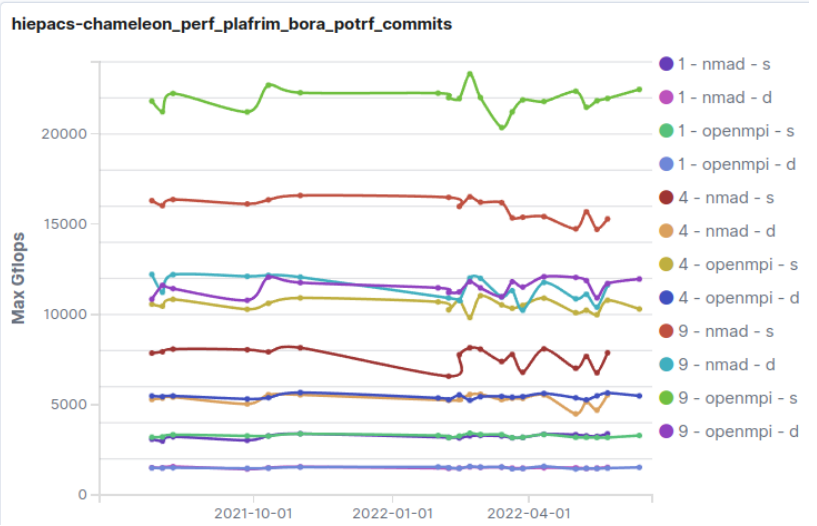

Inria

Test de non-regression sur Plafrim

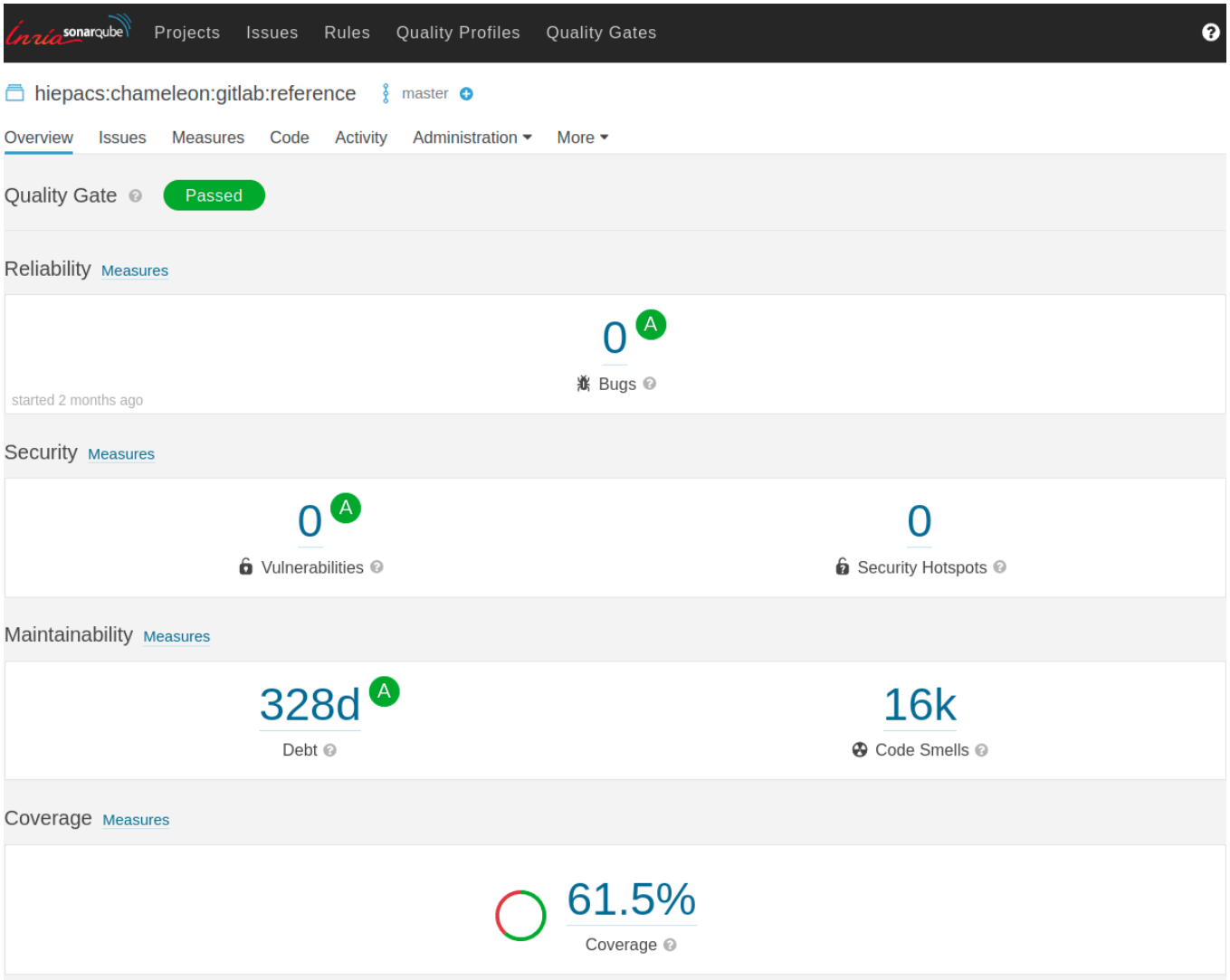




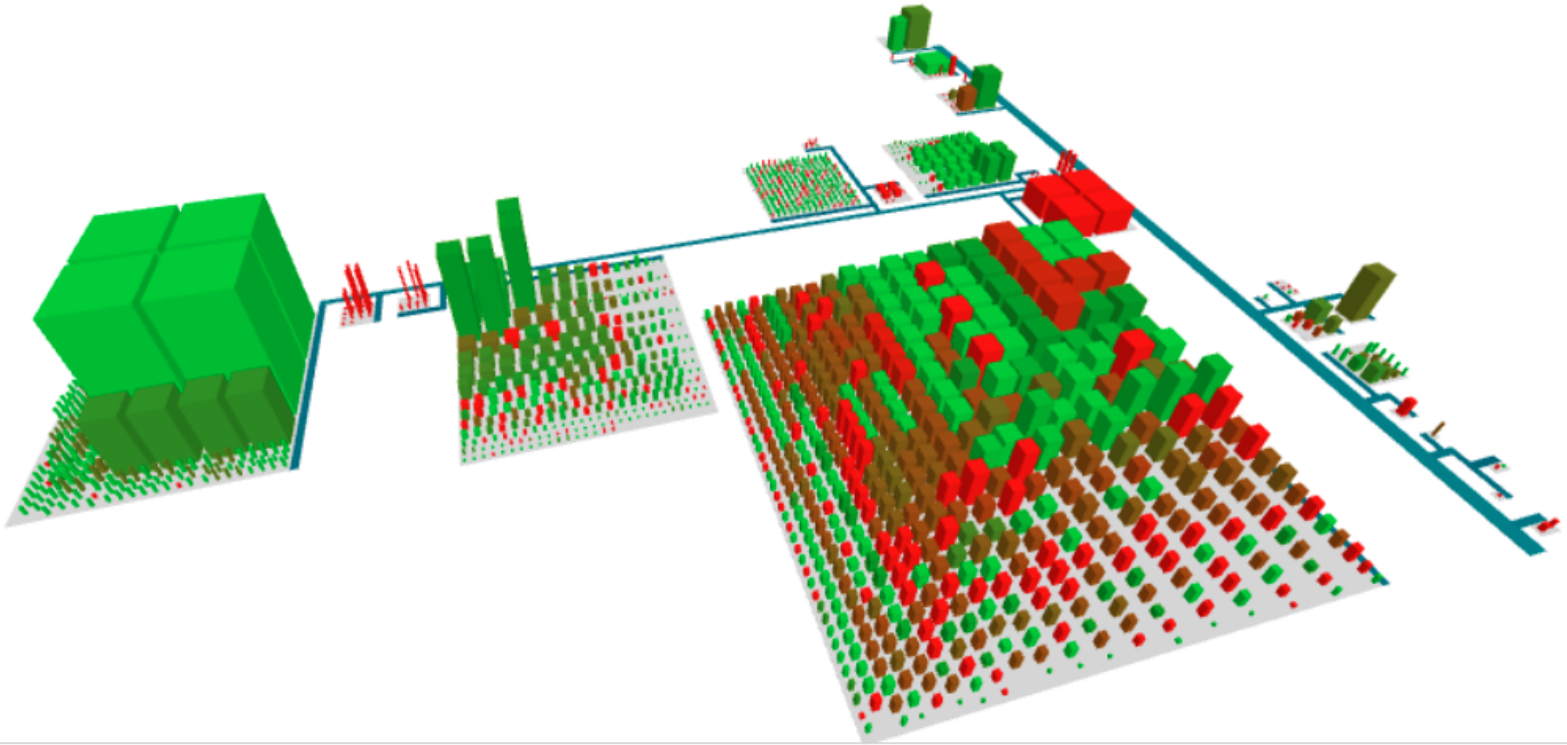




Test de qualité de code avec SonarQube



Test de qualité de code avec SonarQube



- Brew 
  https://gitlab.inria.fr/solverstack/brew-repo
- Spack 
  https://gitlab.inria.fr/solverstack/spack-repo
- Guix 
  https://gitlab.inria.fr/guix-hpc/guix-hpc

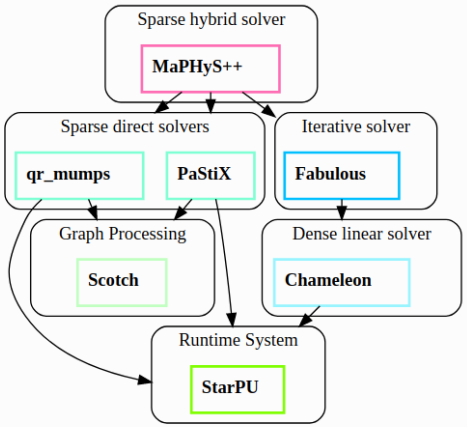
Site Web solveurs

https://solverstack.gitlabpages.inria.fr/

SOLVERSTACK@INRIA BORDEAUX SOLVERS NEWS

Solverstack@Inria Bordeaux

Solverstack@Inria Bordeaux aims at providing a high-performance (HPC) linear algebra solver stack. It provides a collection of solvers, partitioning tools and runtimes systems, which may be employed on modern supercomputers to operate on dense and sparse matrices. It includes both direct and iterative methods (together with various preconditioners), as well as hybrid direct/iterative ones.



One of the main objective is the *portability of performance*, from the laptops to the supercomputer. The software are mainly developed in C, C++ and Fortran by researchers and engineers working in -- or in strong collababoration with -- Inria joint project-teams with Bordeaux INP, Bordeaux University and CNRS (LaBRI UMR 5800) HiePACS, Storm and TADaaM located at Inria Bordeaux Sud-Ouest.

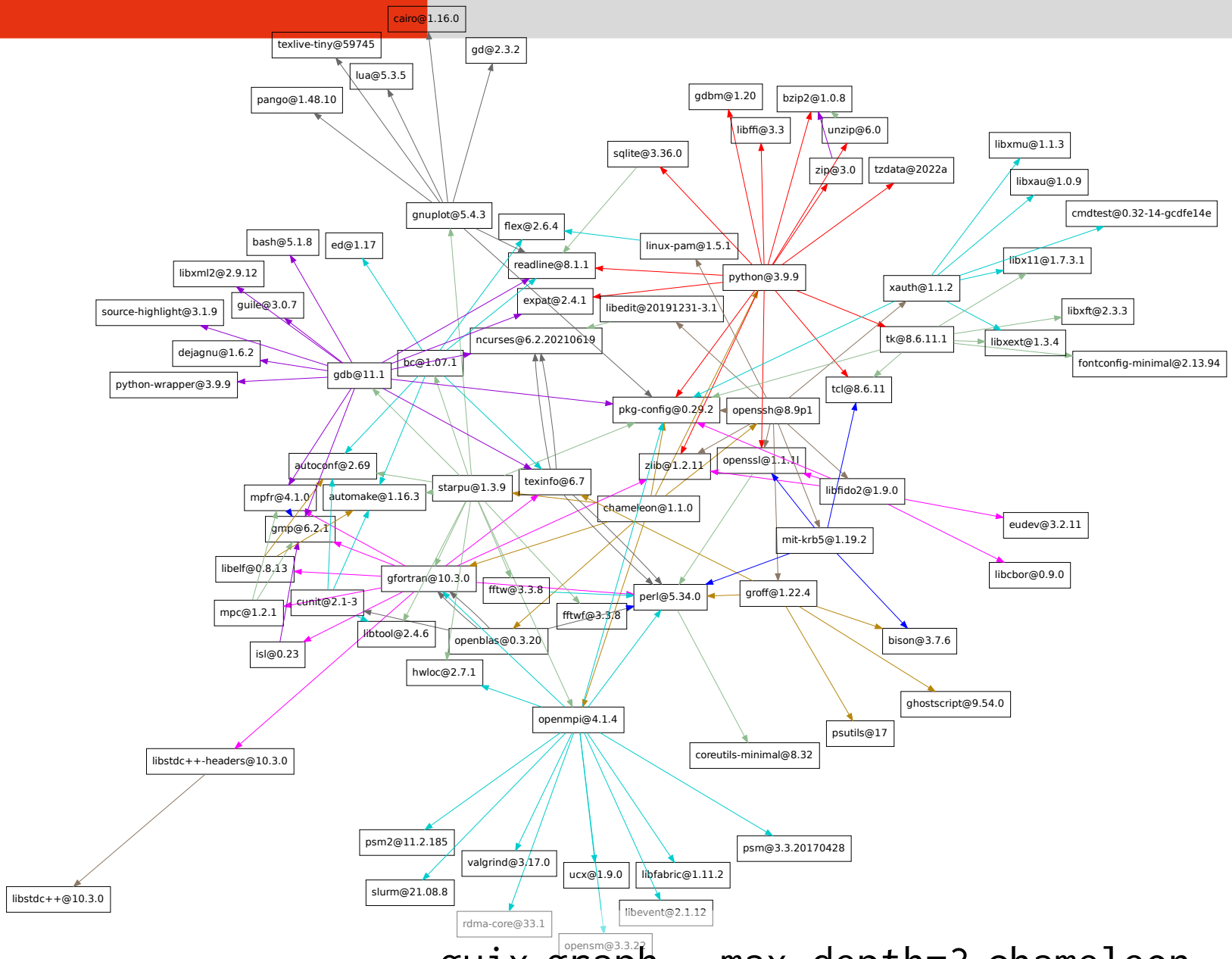
Impacts

- traitements automatiques : productivité ↗
- environnement scriptés : contrôle/reproductibilité ↗
- tests non-regression et qualité : confiance ↗
- Packages et site web : visibilité/collaboration ↗

02

Déploiement logiciel

Inria



guix graph --max-depth=3 chameleon

https://gitlab.inria.fr/guix-hpc/guix-hpc/

- StarPU
- Chameleon
- PaSTiX
- NewMadeleine
- Maphys
- Aerosol
- Simgrid
- ...

- **réécriture du graphe** de dépendances
- choix de la **chaîne de compilation**
- déploiement **depuis Git**

- **réécriture du graphe** de dépendances
- choix de la **chaîne de compilation**
- déploiement **depuis Git**
- déploiement vers **Singularity, Docker**
- aide aux **mésocentres** (GriCAD, GLICID), Grid'5000, PlaFRIM, etc.
- …

- *automn school on high-performance numerical simulation* (2019)
- action nationale de formation (ANF) *user tools for HPC* (2021)
- Midi de la bidouille, Café Guix
- MOOC recherche reproductible v2 (2022)
- …

03

Recherche reproductible & science ouverte

https://www.acm.org/publications/policies/artifact-review-badging

Inria

d. ***Open source software*** and source code that generally include software whose source code is made publicly available, in a timely and user-friendly manner, in human- and machine-readable and modifiable format, under an open license that grants others the right to use, access, modify, expand, study, create derivative works and share the software and its source code, design or blueprint. The source code must be included in the software release and made available on openly accessible repositories and the chosen license must allow modifications, derivative works and sharing under equal or compatible open terms and conditions. In the context of open science, when open source code is a component of a research process, enabling reuse and replication generally requires that it be accompanied with open data and open specifications of the environment required to compile and run it.

https://en.unesco.org/science-sustainable-future/open-science/recommendation

Les établissements publics et fondations reconnues d'utilité publique mentionnés au troisième alinéa de l'article L. 211-2 du code de la recherche :
1° Assurent la sensibilisation et la formation de leurs personnels au respect des exigences de l'intégrité scientifique, ainsi que de leurs étudiants dans le cadre de la formation à et par la recherche ;
2° Veillent à ce que l'organisation des travaux de recherche de leurs personnels soit menée dans le respect de ces exigences ;
3° Promeuvent la diffusion des publications en accès ouvert et la mise à disposition des méthodes et protocoles, des données et des codes sources associés aux résultats de la recherche afin d'en garantir la traçabilité et la reproductibilité. Ils incitent à la publication des résultats de recherche dits négatifs ;

décret relatif au respect des exigences de l'intégrité scientifique (déc. 2021)

```
bob@laptop$ guix install gcc-toolchain pt-scotch
bob@laptop$ guix describe
  guix cabba9e
   repository URL: https://git.sv.gnu.org/git/guix.git
    commit: cabba9e15900d20927c1f69c6c87d7d2a62040fe
```


Inria

```
bob@laptop$ guix install gcc-toolchain pt-scotch
bob@laptop$ guix describe
  guix cabba9e
   repository URL: https://git.sv.gnu.org/git/guix.git
    commit: cabba9e15900d20927c1f69c6c87d7d2a62040fe
```

```
alice@supcomp$ guix time-machine --commit=cabba9e -- \
                    install gcc-toolchain pt-scotch
```

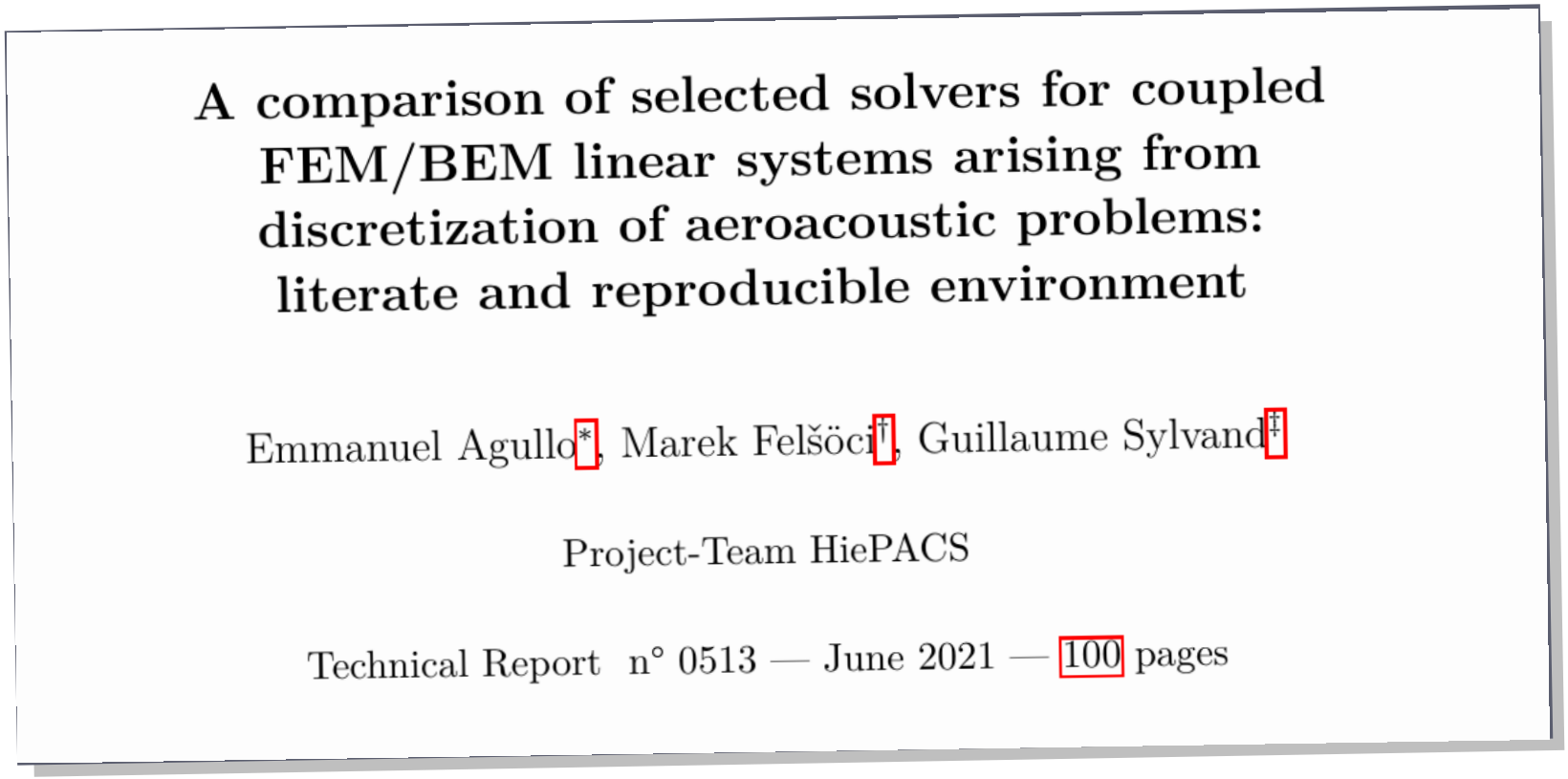

travel in space *and* time!

```
guix time-machine --commit=cabba9e -- \
  install gcc-toolchain pt-scotch
```


Inria


```
guix time-machine --commit=cabba9e -- \
  install gcc-toolchain pt-scotch
```

https://www.softwareheritage.org/2019/04/18/software-heritage-and-gnu-guix-join-forces-to-enable-long-term-reproducibility/



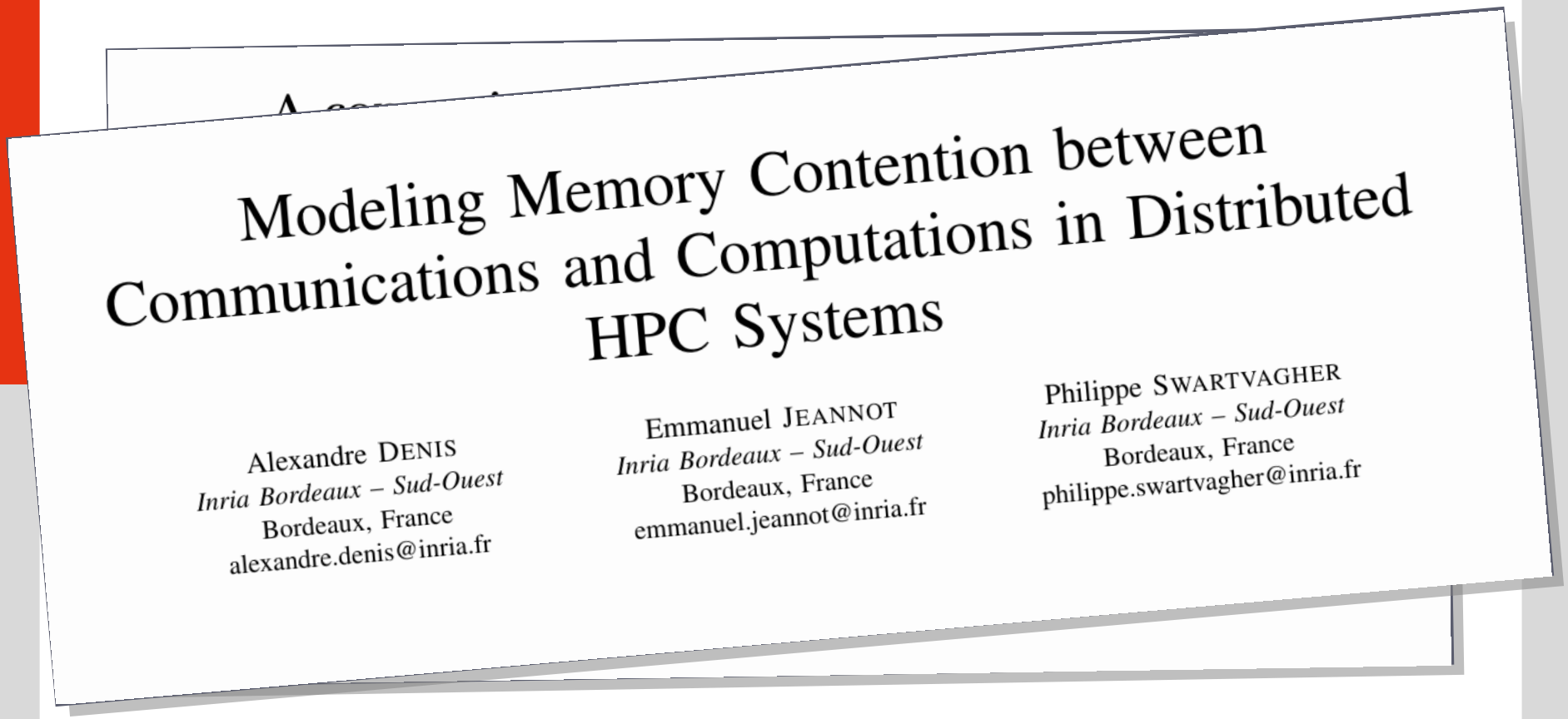
A comparison of selected solvers for coupled FEM/BEM linear systems arising from discretization of aeroacoustic problems: literate and reproducible environment

Emmanuel Agullo*, Marek Felšöci†, Guillaume Sylvand‡

Project-Team HiePACS

Technical Report n° 0513 — June 2021 — 100 pages

https://hal.inria.fr/hal-03263620



Modeling Memory Contention between Communications and Computations in Distributed HPC Systems

Alexandre DENIS
Inria Bordeaux – Sud-Ouest
Bordeaux, France
alexandre.denis@inria.fr

Emmanuel JEANNOT
Inria Bordeaux – Sud-Ouest
Bordeaux, France
emmanuel.jeannot@inria.fr

Philippe SWARTVAGHER
Inria Bordeaux – Sud-Ouest
Bordeaux, France
philippe.swartvagher@inria.fr

https://hal.inria.fr/hal-03682199

Acknowledgment

This work is supported by the Agence Nationale de la Recherche, under grant ANR-19-CE46-0009. This work is supported by the Région Nouvelle-Aquitaine, under grant 2018-1R50119 *HPC scalable ecosystem*. Experiments presented in this paper were carried out using the PlaFRIM

A public companion contains the instructions to reproduce our study: https://gitlab.inria.fr/pswartva/paper-model-memory-contention-r13y, archived on https://www.softwareheritage.org/ with the ID `swh:1:snp:306f7c10cf69a5860587e5aad62b76070b798ecd`.

https://hal.inria.fr/hal-03682199

TECHNICAL NOTE

PiGx: reproducible genomics analysis pipelines with GNU Guix

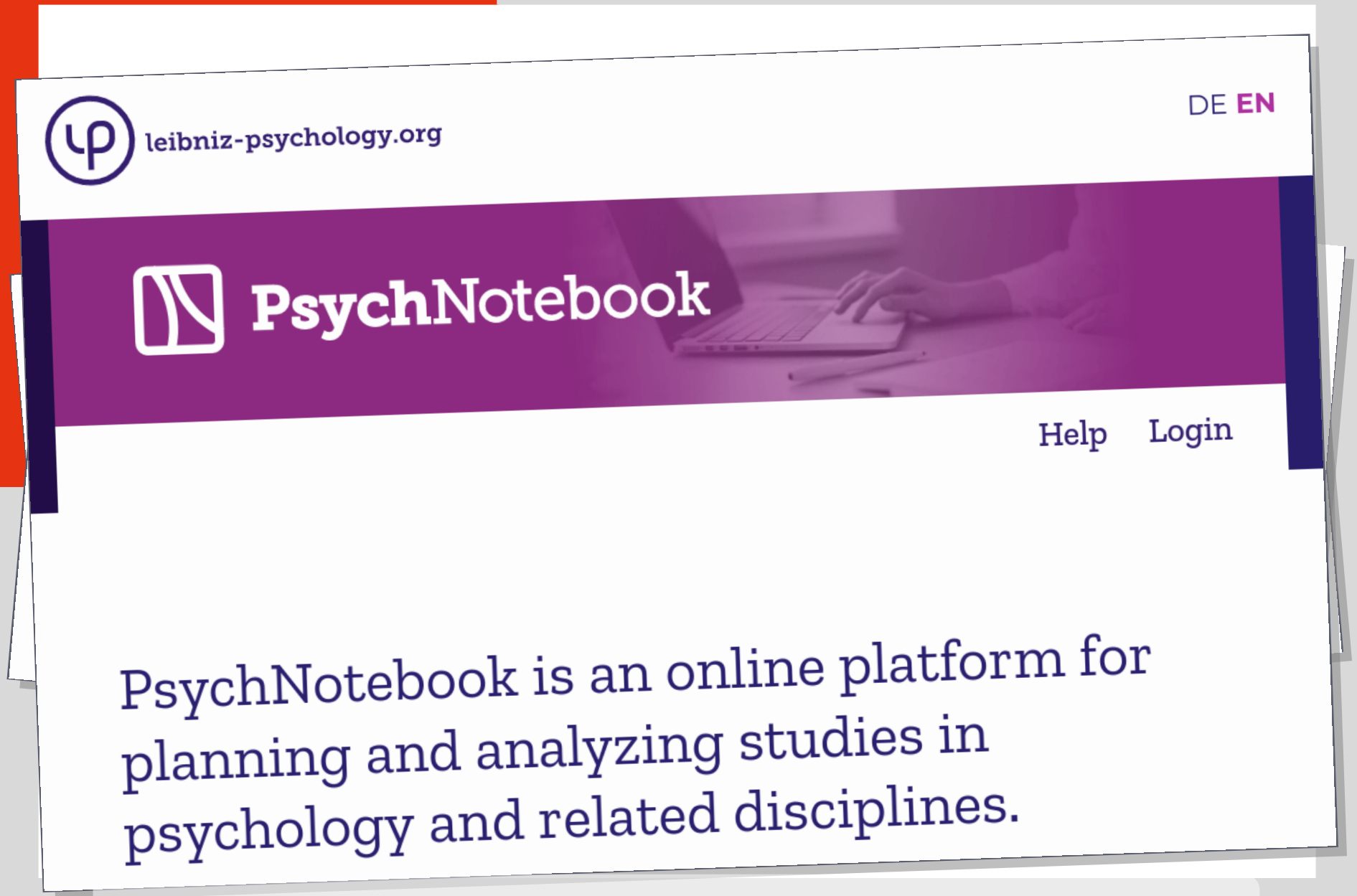
Ricardo Wurmus[†], Bora Uyar[†], Brendan Osberg[†], Vedran Franke[†], Alexander Gosdschan[†], Katarzyna Wreczycka, Jonathan Ronen and Altuna Akalin[*]

Bioinformatics Platform, The Berlin Institute for Medical Systems Biology, Max-Delbrück Center for Molecular Medicine, Robert-Rössle-Strasse 10, 13125 Berlin, Germany

[*]**Correspondence address.** Altuna Akalin, Bioinformatics Platform, The Berlin Institute for Medical Systems Biology, Max-Delbrück Center for Molecular Medicine, Robert-Rössle-Strasse 10, 13125 Berlin, Germany. E-mail: altuna.akalin@mdc-berlin.de http://orcid.org/0000-0002-0468-0117

[†]Equal contributions.

http://dx.doi.org/10.1093/gigascience/giy123



https://www.psychnotebook.org/

04

Sujet de recherche

https://rescience.github.io/ten-years/

Ten Years Reproducibility Challenge

RESCIENCE SPECIAL ISSUE

FREE TO READ - FREE TO PUBLISH

Would you dare to run the code from your past self ?

(the one that does not answer mail)

Reproducible and User-Controlled Software Environments in HPC with Guix

Ludovic Courtès[1] and Ricardo Wurmus[2]

[1] Inria, Bordeaux, France
[2] Max Delbrück Center for Molecular Medicine, Berlin, Germany

Abstract. Support teams of high-performance computing (HPC) systems often find themselves between a rock and a hard place: on one hand, they understandably administrate these large systems in a conservative way, but on the other hand, they try to satisfy their users by deploying up-to-date tool chains as well as libraries and scientific software. HPC system users often have no guarantee that they will be able to reproduce results at a later point in time, even on the same system—software may have been upgraded, removed, or recompiled under their feet, and they have little hope of being able to reproduce the same software environment elsewhere. We present GNU Guix and the functional package management paradigm and show how it can improve reproducibility and sharing among researchers with representative use cases.

https://hal.inria.fr/hal-01161771 (2015)

Reproducible and User-Controlled Software Environments in HPC with Guix

Reproducibility and Performance: Why Choose?

Ludovic Courtès
Inria

***Abstract*—Research processes often rely on high-performance computing (HPC), but HPC is often seen as antithetical to "reproducibility": one would have to choose between software that achieves high performance, and software that can be deployed in a reproducible fashion.**

vironment elsewhere. We present GNU Guix and the
management paradigm and show how it can improve reproducibility and
sharing among researchers with representative use cases.

https://hal.inria.fr/hal-03604971/ (2022)

Reproducible and User-Controlled Software Environments in HPC with Guix

Reproducibility and

Code Staging in GNU Guix

Ludovic Courtès
Inria
Bordeaux, France

***Abstract*—Research processes often rely on high-performance computing (HPC), but HPC is often seen as antithetical to "reproducibility": one would have to choose between software that achieves high performance, and software that can be deployed in a reproducible fashion.**

vironment elsewhere. We present GNU Guix and the
management paradigm and show how it can improve reproducibility and
sharing among researchers with representative use cases.

https://hal.inria.fr/hal-01580582/ (2017)

Reproducible and User-Controlled Software Environments in HPC with Guix

Building a Secure Software Supply Chain with GNU Guix

Ludovic Courtès[a]

a Inria, France

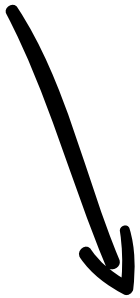
Abstract The *software supply chain* is becoming a widespread analogy to designate the series of steps taken to go from source code published by developers to executables running on the users' computers. A security vulnerability in any of these steps puts users at risk, and evidence shows that attacks on the supply chain are becoming more common. The consequences of an attack on the software supply chain can be tragic in a society that relies on many interconnected software systems, and this has led research interest as well as governmental incentives for supply chain security to rise.

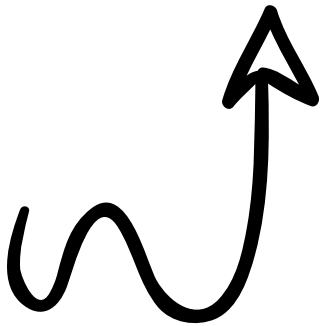
often seen as antithetical to "reproducibility": one would have to choose between software that achieves high performance, and software that can be deployed in a reproducible fashion.

vironment elsewhere. We present GNU Guix and the
management paradigm and show how it can improve reproducibility and

https://doi.org/10.22152/programming-journal.org/2023/7/1
(2022)

Software Heritage





The ReScience Journal

Inria

Faire du
déploiement reproductible
une « bonne pratique » admise.

Inria