MAX DRIVING THE EXASCALE TRANSITION

cecam Centre Européen de Calcul Atomique et Moléculaire

ICN2 Institut Català de Nanociència i Nanotecnologia

BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación

siesta

GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN

AGENCIA ESTATAL DE INVESTIGACIÓN

CSIC

ICMAB

EXCELENCIA SEVERO OCHOA

SIESTA: Building, Deployment, and Execution

José María Escartín Esteban (ICN2)

Alberto García (ICMAB-CSIC, Barcelona)

MaX "*Materials Design at the Exascale*", has received funding from the European Union's Horizon 2020 project call H2020-INFRAEDI-2018-1, grant agreement 824143

Siesta was built and deployed on MareNostrum 4 for the School

```
$ source /gpfs/projects/nct00/nct00003/siestarc.sh
  ...


$ siesta -v
Siesta Version  : 5.0.0-beta1
Architecture    : ----
Compiler version: Intel-2021.4.0.20210910
Compiler flags  : -O2 -ip -xHost -fp-model=strict -prec-div -prec-sqrt
PP flags        : ----
Libraries       : ----
Parallelisations: MPI
GEMM3M support
NetCDF support
NetCDF-4 support
Lua support
```

Steps for building and deploying Siesta on a computer

1. **Check** the hardware requirements.
2. **Prepare** the build environment.
3. **Download** the SIESTA source code.
4. **Build** SIESTA, its utilities, and any needed dependencies.
5. **Test** the binaries.
6. **Deploy** the files you built (binaries, libraries, etc.).
7. **Run** the executables in the correct environment.

Hardware requirements

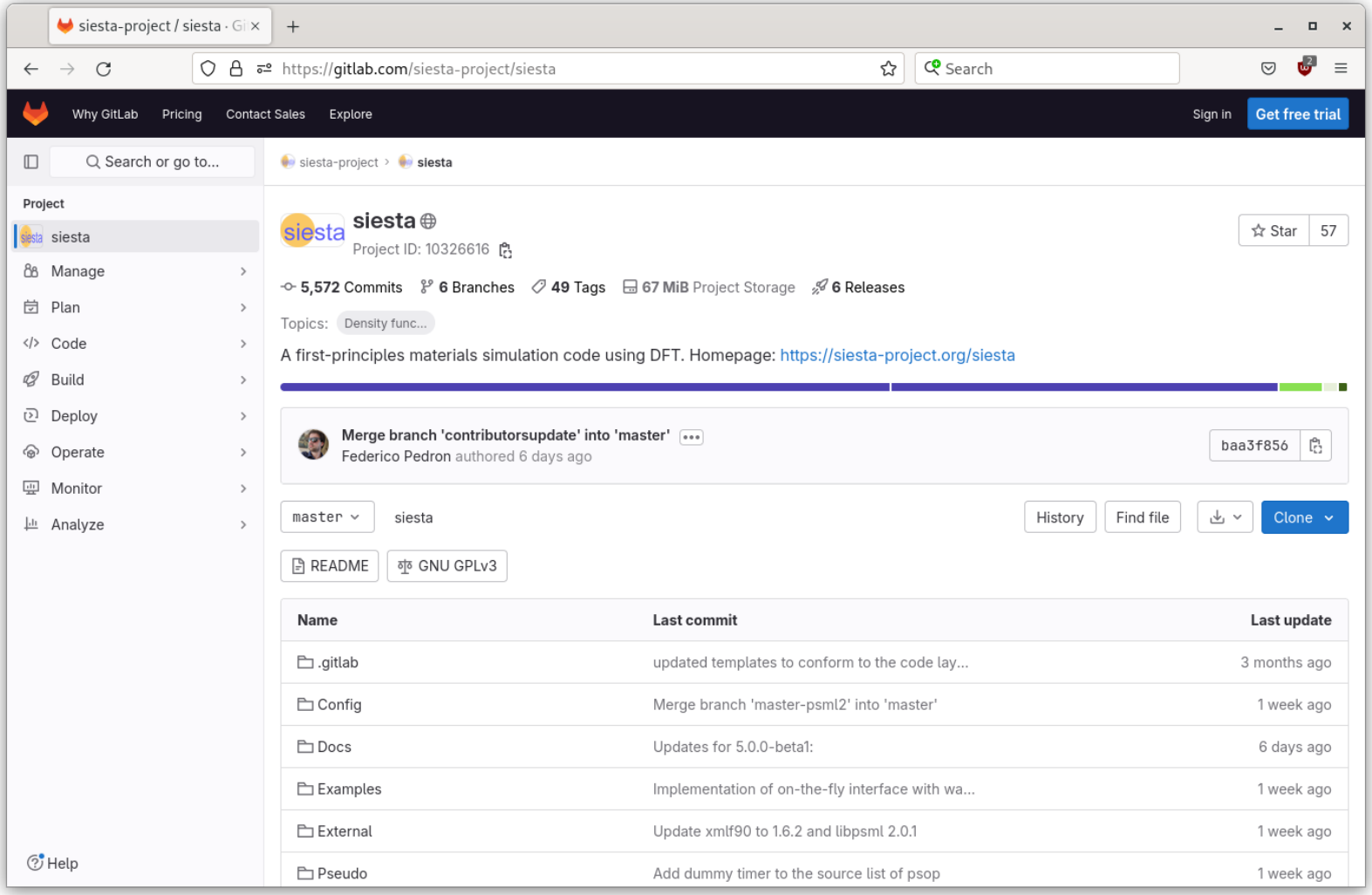
- SIESTA can work on a broad variety of computer architectures, from a Raspberry PI to massively parallel supercomputers.
- Recommended minimum requirements:
  - 1 GB RAM (although small atomic structures can be studied with less).
  - 2 GB disk storage, in particular if you want to run the SIESTA test suite.
- SIESTA will likely work on any CPU, but bear in mind that most development and testing is done on x86_64 (Intel/AMD) and ARMv8 architectures.

Building Software Requirements

- CMake version $\geq$ 3.17 (released in 2020).
- A Fortran compiler with full support of the Fortran 2003 standard and partial support of the Fortran 2008 standard (gfortran, ifort, CRAY Fortran, etc.).
  Note: mixing old compilers with new hardware is usually a bad idea.
- A C/C++ companion compiler.
- If you want to build a parallel version of SIESTA, you will also need a MPI distribution (including development files).
- If you want to offload to a Nvidia (/AMD) GPU, you will need the corresponding CUDA (/HIP) distribution.
- If you want to build the SIESTA user manual, you will also need a LaTeX distribution.

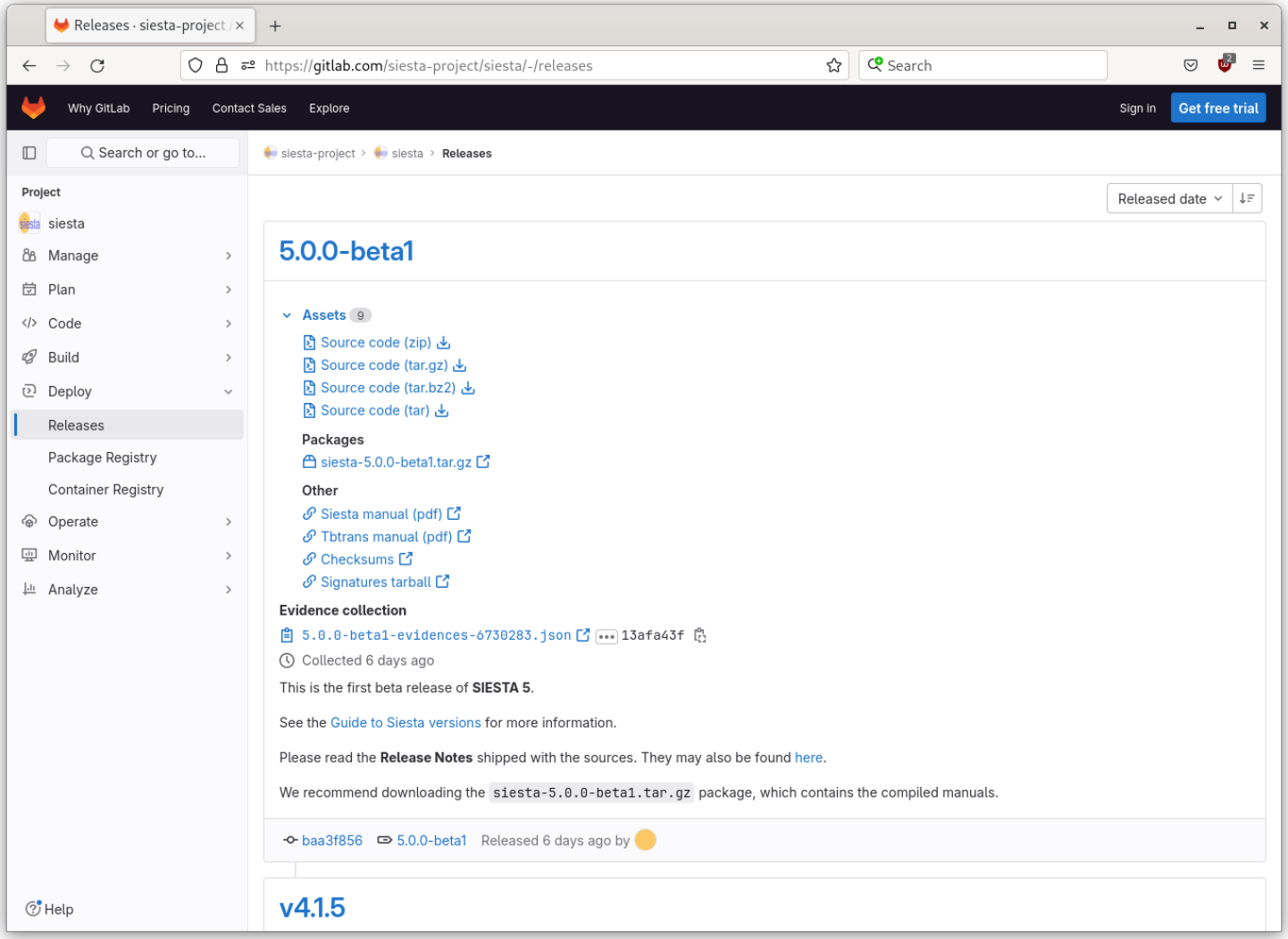
Source Code — the authoritative distribution channel

`https://gitlab.com/siesta-project/siesta/`



Siesta Releases

https://gitlab.com/siesta-project/siesta/-/releases/



Should I get Siesta-5? (now in beta)

- SIESTA 5 has lots of new functionalities compared to the current stable release, SIESTA v4.1.5.
- SIESTA 5 is easier to install and to deploy than SIESTA v4.1.5.
- We expect another SIESTA 5 beta/rc release by the end of October, and the actual 5 release by the end of 2023.
- The repository contains a few other development branches with significant improvements, see the Guide to SIESTA versions (`https://gitlab.com/siesta-project/siesta/-/wikis/Guide-to-Siesta-versions`). Most of them should be released as part of the next major version, SIESTA 6 (2024).
- Only release tarballs are supported. Users should really stick to them.
- Advanced users that want to try arbitrary branches or versions of SIESTA should really avoid the GitLab download button, and interact with the repository using git (version $\geq$ 2.13). These non-release versions are generally unsupported by the SIESTA development team.

Basic steps for building with the command line

Download tarball:

```
$ wget https://gitlab.com/.../5.0.0-beta1/downloads/siesta-5.0.0-beta1.tar.gz
```

Extract files:

```
$ tar -xvvzf siesta-5.0.0-beta1.tar.gz
```

Enter source directory:

```
$ cd siesta-5.0.0-beta1
```

Initialize build directory:

```
$ cmake -S. -B_build
```

Build:

```
$ cmake --build _build -j 4
```

Basic options (compiler and flags, MPI, OpenMP)

Check the SIESTA manual for details about all the building options.

- Specify Fortran compiler

  ```
  FC=gfortran cmake ...
  ```

- Specify Fortran compiler flags

  ```
  cmake -DFortran_FLAGS='-O3 -march=native'
  ```

- Specify toolchain file (some available in `Config/cmake/toolchains/`):

  ```
  cmake ... -DSIESTA_TOOLCHAIN=/path/to/toolchain/file
  ```

- Explicitly enable/disable MPI (default: `ON` if MPI compiler found, otherwise `OFF`):

  ```
  cmake ... -DWITH_MPI=ON|OFF
  ```

- Explicitly enable/disable OpenMP (default: `OFF`):

  ```
  cmake ... -DWITH_OPENMP=ON|OFF
  ```

Example of toolchain file (Config/cmake/toolchains/mac.cmake)

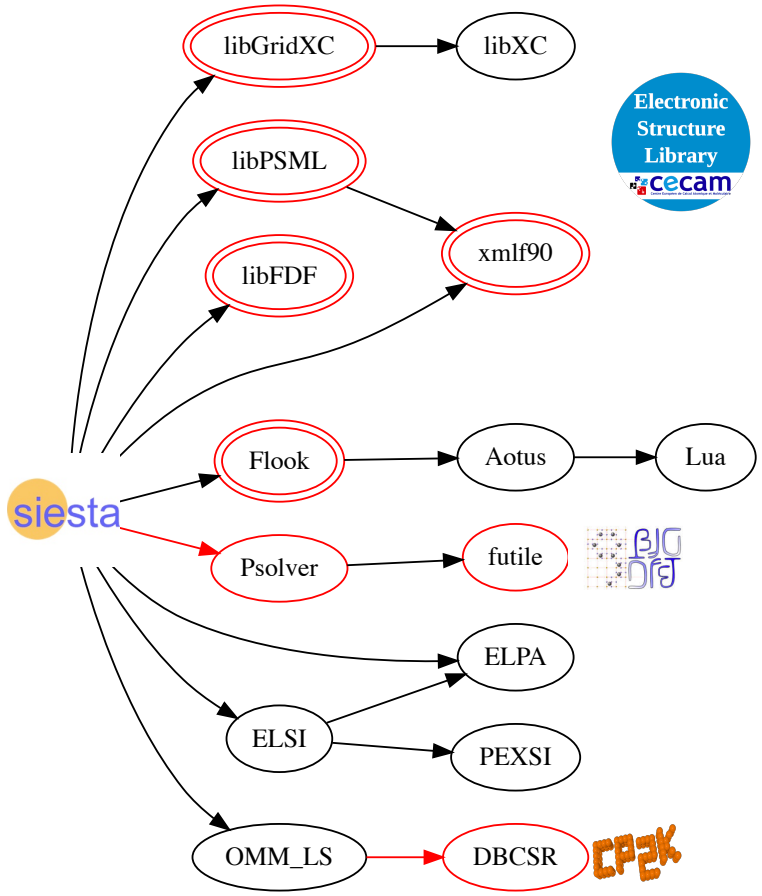
```
#
#  If you have veclibfort, and it works, you could uncomment these lines
#
set(BLAS_LIBRARY "-lveclibfort" CACHE STRING "blas library chosen")
set(LAPACK_LIBRARY "-lveclibfort" CACHE STRING "lapack library chosen")
#
#  More general settings for use with shell modules
#
# set(LAPACK_LIBRARY "$ENV{LAPACK_LIBS}" CACHE STRING "lapack library chosen")
set(SCALAPACK_LIBRARY "$ENV{SCALAPACK_LIBS}" CACHE STRING "scalapack library
chosen")
#
```

Example of toolchain file for specific supercomputer software stack

```
#
# Vega supercomputer at Maribor, Slovenia, with ELPA GPU support
#
# Using the gompic easyBuild toolchain (GCC, OpenBLAS, OpenMPI, CUDA)
#
# Set up the modules as:
#
# ml gompic/2020b
# ml OpenBLAS/0.3.12-GCC-10.2.0
# ml ScaLAPACK/2.1.0-gompic-2020b
# ml netCDF-Fortran/4.5.3-gompic-2020b
# ml CMake/3.18.4-GCCcore-10.2.0
# ml FFTW/3.3.8-gompic-2020b
#
set(WITH_OPENMP "ON" CACHE BOOL "with OpenMP")
#
set(LAPACK_LIBRARY
     "-L /cvmfs/sling.si/modules/el7/software/OpenBLAS/0.3.12-GCC-10.2.0/lib -lopenblas -lpthread -lm -ldl"
      CACHE STRING "lapack library chosen")

set(SCALAPACK_LIBRARY
    "-L/cvmfs/sling.si/modules/el7/software/ScaLAPACK/2.1.0-gompic-2020b/lib -lscalapack"
    CACHE STRING "scalapack library chosen")
```

SIESTA: Domain-specific libraries



Domain-specific libraries originating in SIESTA itself or created for new functionalities in the code:

- libGridXC: laboratory for interface design
- xmlf90: used already by other community codes
- libPSML: enables pseudopotential interoperability

Scriptability via embedded interpreter with access to data structures of the code

New Poisson solver with flexible boundary conditions and optimized for hybrid architectures

Solvers: consolidated interfaces for continuously improved and performance-portable libraries.

New class of linear-scaling algorithms with efficient sparse-matrix library DBSCR as backend

Automatic download and compilation of domain-specific libraries

```
-- Searching for xmlf90
-- | Siesta_find_package[xmlf90] METHODS | ALLOWED = cmake;pkgconf;source;fetch |
cmake;pkgconf;source;fetch
-- | CMake package lookup [xmlf90]
-- Could NOT find xmlf90 (missing: xmlf90_DIR)
-- | CMake package lookup [xmlf90] - not found
-- | pkg-config package lookup[xmlf90]
-- | Checking for module 'xmlf90'
-- |   No package 'xmlf90' found
-- | pkg-config package lookup[xmlf90] - not found
-- | source in folder: /tmp/siesta-5.0.0-beta1/External/xmlf90
-- | source in folder: /tmp/siesta-5.0.0-beta1/External/xmlf90 - not found
-- | fetching from https://gitlab.com/siesta-project/libraries/xmlf90
-- | BINARY_DIR for fetched xmlf90: /tmp/siesta-5.0.0-beta1/_test/_deps/xmlf90-build
-- | fetching from https://gitlab.com/siesta-project/libraries/xmlf90 - fetched
-- Searching for xmlf90 - found
```

Extra pre-compiled libraries

Some libraries have to be pre-installed:

- libxc
- ELPA
- netCDF (likely installed already in the system)

cmake …. -DCMAKE_PREFIX_PATH=${LIBXC_ROOT} …

Testing Siesta

```
> cd _build
> ctest -L simple -N

Test project /tmp/siesta-5.0.0-beta1/_build
  Test  #25: siesta-siesta-00.BasisSets-default_basis_mpi_np4
  Test  #26: verify-default_basis
  Test  #41: siesta-siesta-01.PseudoPotentials-psf_mpi_np4
  Test  #42: verify-psf
  Test  #43: siesta-siesta-01.PseudoPotentials-full.psml_mpi_np4
  Test  #44: verify-full.psml
  Test  #71: siesta-siesta-03.SpinOrbit-FePt-onsite_mpi_np4
  Test  #72: verify-FePt-onsite
  Test  #83: siesta-siesta-05.Bands-ge_bands_mpi_np4
  Test  #84: verify-ge_bands
  Test  #97: siesta-siesta-06.DensityOfStates-pdos_kp_mpi_np4
  Test  #98: verify-pdos_kp
  Test #105: siesta-siesta-07.ForceConstants-fc_mpi_np4
  Test #106: verify-fc
  Test #113: siesta-siesta-08.GeometryOptimization-cg_mpi_np4
  Test #114: verify-cg
  Test #133: siesta-siesta-09.MolecularDynamics-verlet_mpi_np4
  Test #134: verify-verlet
..
```

Testing Siesta

```
> ctest -L simple -E verify
Test project /tmp/siesta-5.0.0-beta1/_build
      Start  25: siesta-siesta-00.BasisSets-default_basis_mpi_np4
 1/22 Test  #25: siesta-siesta-00.BasisSets-default_basis_mpi_np4 ...........   Passed    2.40 sec
      Start  33: siesta-siesta-01.PseudoPotentials-psf_mpi_np4
 2/22 Test  #33: siesta-siesta-01.PseudoPotentials-psf_mpi_np4 ..............   Passed    1.94 sec
      Start  34: siesta-siesta-01.PseudoPotentials-full.psml_mpi_np4
 3/22 Test  #34: siesta-siesta-01.PseudoPotentials-full.psml_mpi_np4 ........   Passed    2.17 sec
      Start  48: siesta-siesta-03.SpinOrbit-FePt-onsite_mpi_np4
 4/22 Test  #48: siesta-siesta-03.SpinOrbit-FePt-onsite_mpi_np4 .............   Passed    5.47 sec
      Start  54: siesta-siesta-05.Bands-ge_bands_mpi_np4
 5/22 Test  #54: siesta-siesta-05.Bands-ge_bands_mpi_np4 ....................   Passed    2.19 sec
      Start  61: siesta-siesta-06.DensityOfStates-pdos_kp_mpi_np4
```

Installing Siesta

Tell cmake where to install SIESTA, and install it there:

```
$ cmake -S. -B_build -DCMAKE_INSTALL_PREFIX=/path/to/installation
$ cmake --build _build -j 4
$ cmake --install _build
$ ls /path/to/installation
   bin  include  lib64  share
```

Then make your environment aware of this installation:

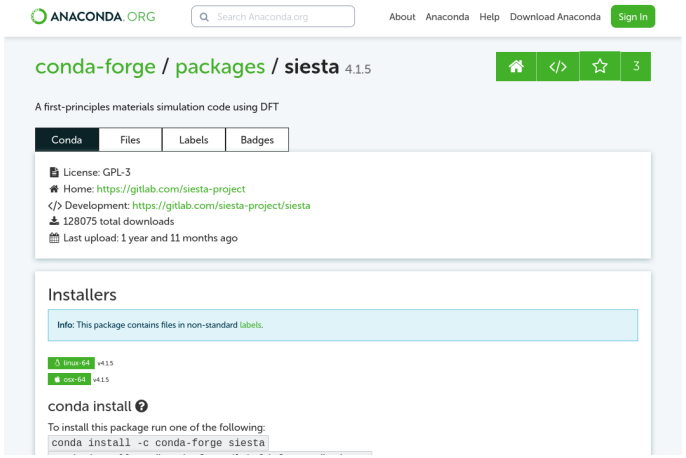
```
$ cat siestarc.sh
#!/bin/sh

LD_LIBRARY_PATH="/path/to/installation/lib:$LD_LIBRARY_PATH"
export LD_LIBRARY_PATH

PATH="/path/to/installation/bin:$PATH"
export PATH
```

Installing with conda (4.1.5 version only as of now)

- Conda is a package manager.
- Conda-Forge is community repository of recipes and packages.
- SIESTA available on Conda-forge:
  `https://anaconda.org/conda-forge/siesta`




```
$ conda install -c conda-forge siesta=4.1.5=*openmpi*
```

- Strength: easy access to SIESTA: somebody else built the package, so you can directly deploy.
- Weakness: a Conda package may run on a range of CPUs $\Longrightarrow$ package not optimized for your particular instruction set $\Longrightarrow$ SIESTA not as performant as it could be.

Installing with spack (some setup required)

```
$ spack info siesta    # (NEED the proper package files. See INSTALL.md and Config/spack_package_defs)

Safe versions:
    master    [git] https://gitlab.com/siesta-project/siesta.git on branch master

Deprecated versions:
    None

Variants:
    Name [Default]                When    Allowed values          Description
    ==========================    ====    ====================    ============================================

    build_type [RelWithDebInfo]   --      Debug, Release,         CMake build type
                                          RelWithDebInfo,
                                          MinSizeRel
    elpa [off]                    --      on, off                 Use ELPA library (native interface)
    fftw [on]                     --      on, off                 Use FFTW library (needed only for STM/ol-stm)
    ipo [off]                     --      on, off                 CMake interprocedural optimization
    libxc [off]                   --      on, off                 Use libxc
    mpi [off]                     --      on, off                 Use MPI
    netcdf [off]                  --      on, off                 Use NetCDF

Build Dependencies:
    cmake  elpa  fftw  lapack  libgridxc  libpsml  libxc  mpi  netcdf-fortran  scalapack  xmlf90
```

Other building options

- EasyBuild (similar to spack)
- Spack containers
- Singularity containers

The recipes are being updated and will be released soon


MAX
DRIVING
THE EXASCALE
TRANSITION


cecam
Centre Européen de Calcul Atomique et Moléculaire


ICN2
Institut Català
de Nanociència
i Nanotecnologia


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación


siesta


GOBIERNO
DE ESPAÑA
MINISTERIO
DE CIENCIA
E INNOVACIÓN


CSIC


ICMAB


EXCELENCIA
SEVERO
OCHOA

THANKS