Max-Planck-Institute for Solar System Research

magIC workshop

Göttingen 27.10. – 29.10.2025

Use ***magIC*** at the
MPS UNiX and HPC environment

overview

- thincLinc – client, login
- development and HPC environment
- set-up training environment, build, run example(s), falsification
- visualization utilizing magic’s python module
- save results

thinLinc – client, login

- Each user or group receive account name:
  - `guest1 – guest9` + password (see handouts)
  - Each UNiX – account can use one node accessible via SLURM reservation name
    `guest1 – guest9`
  - Each UNiX account owns two directory
    - `/home/guest<ID>` for small files (netapp)
    - `/scratch/guest<ID>` for larger files (GPFS instance)
- helper files
  - `/scratch/guest<ID>/ws-files/{conda,docs,refOutshtns,slurm}`
    - build : install-shtns.sh.patch compiler for shtns tools patch to align to MPS CPU hardware
      CMakeLists.txt.patch patch to enable PROJDIR detection
      src-CMakeLists.txt.patch patch fixes wrong IMKL lapack library name
    - conda : pkgs4magic.txt list of python modules resolving magic python module dependencies
    - docs : magICworkshop.pdf **this document (pdf) and slideset of all other talks**
    - refOut : cmake.log, make-j.log reference output of successful cmake and make runs
    - RTE : module-gcc-13.2.0.sh set predefined compile and runtime environment
    - slurm : magic-gcc-sbatch.sh SLURM sbatch template file to run simulation(s)

our wishlist / hints

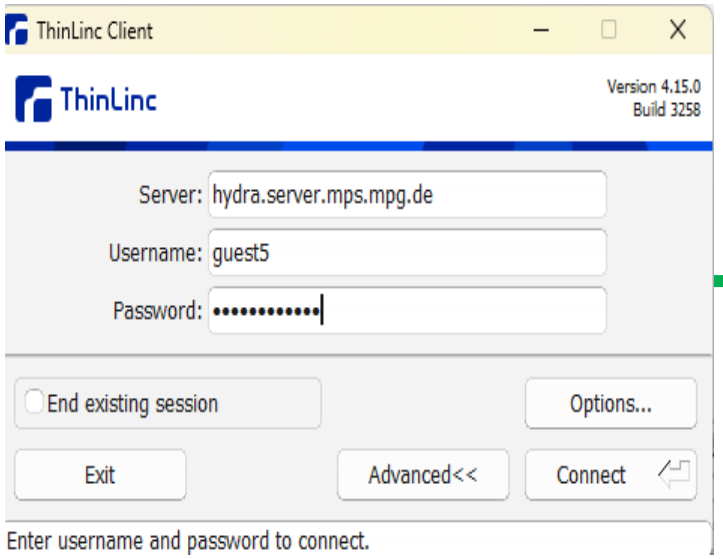
- please don’t use the USB to copy data
- please don’t send e-mail(s) utilizing your guest<ID>
- the accounts will be deleted by Oct, 31st 2025
- access to the accounts from external nodes isn’t possible
- please don’t use the login node (swanl[1,2]) to run your application(s)

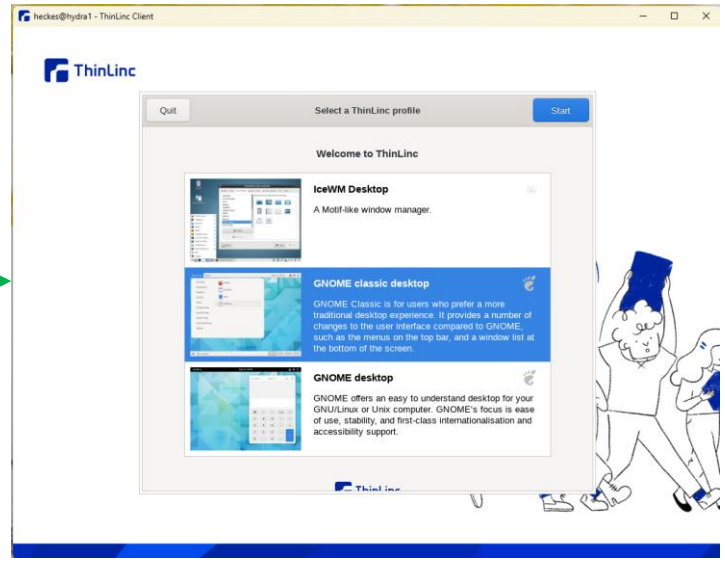
login session (1/2)

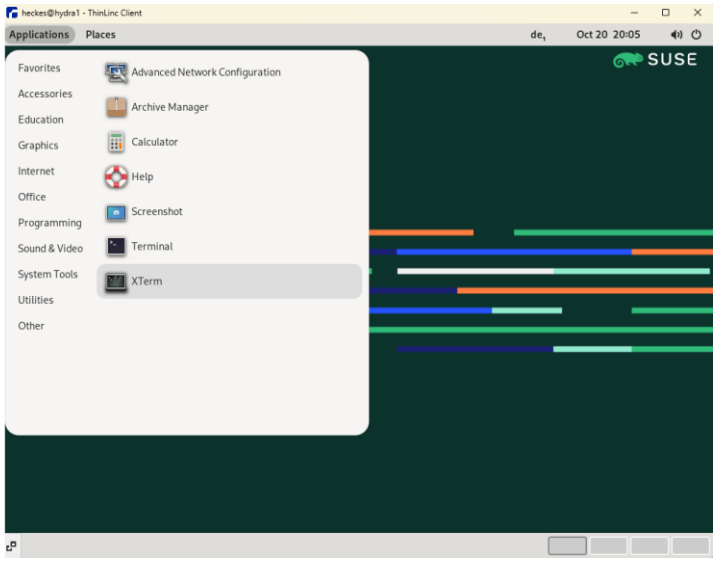
Node: `hydra.server.mps.mpg.de`
User : `guest<ID> + your passwd`

[we recommend gnome classic desktop]

open first `xterm`

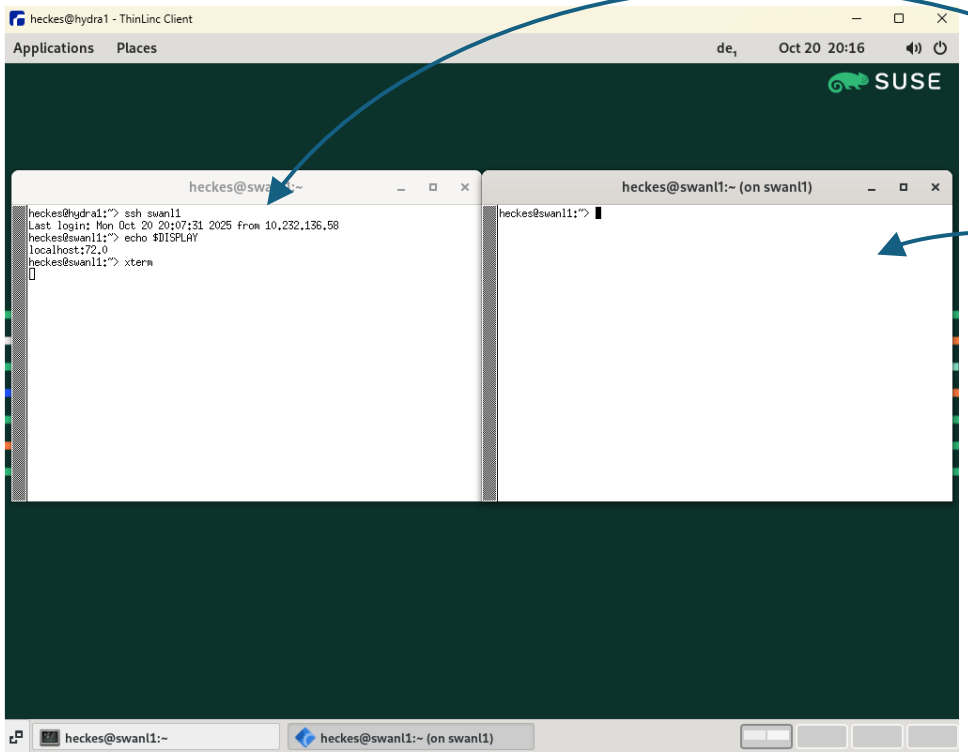






‘load’-balancing to
`hydra1` or **`hydra2`**

login session (2/2)



connect via ssh to login **`swanl1`**

X-forwarding is verified if `xterm` starts.
Important for graphical evaluation/visualization

→ session init completed (basic set-up)

press the **F8-key**, and 'disconnect' to disconnect from running session only

development and HPC environment (1/2)

- Use login nodes **`swan1[1,2]`** to compile the binaries
  or
  run compilation on the node associated with your account

```
srun --account=planets --partition=swan --Qos=swan_default
     --reservation=guest<ID> --mem=1g --time=01:00:00 bash
→ run compilation (see below)
```

- login/compute hardware (scope workshop)
  - CPU: *AMD 7713*, Memory 1TB
  - Display CPU, cache, memory topology: → `likwid-topology`
    `(module load gcc/12.3 likwid)`

development and HPC environment (2/2)

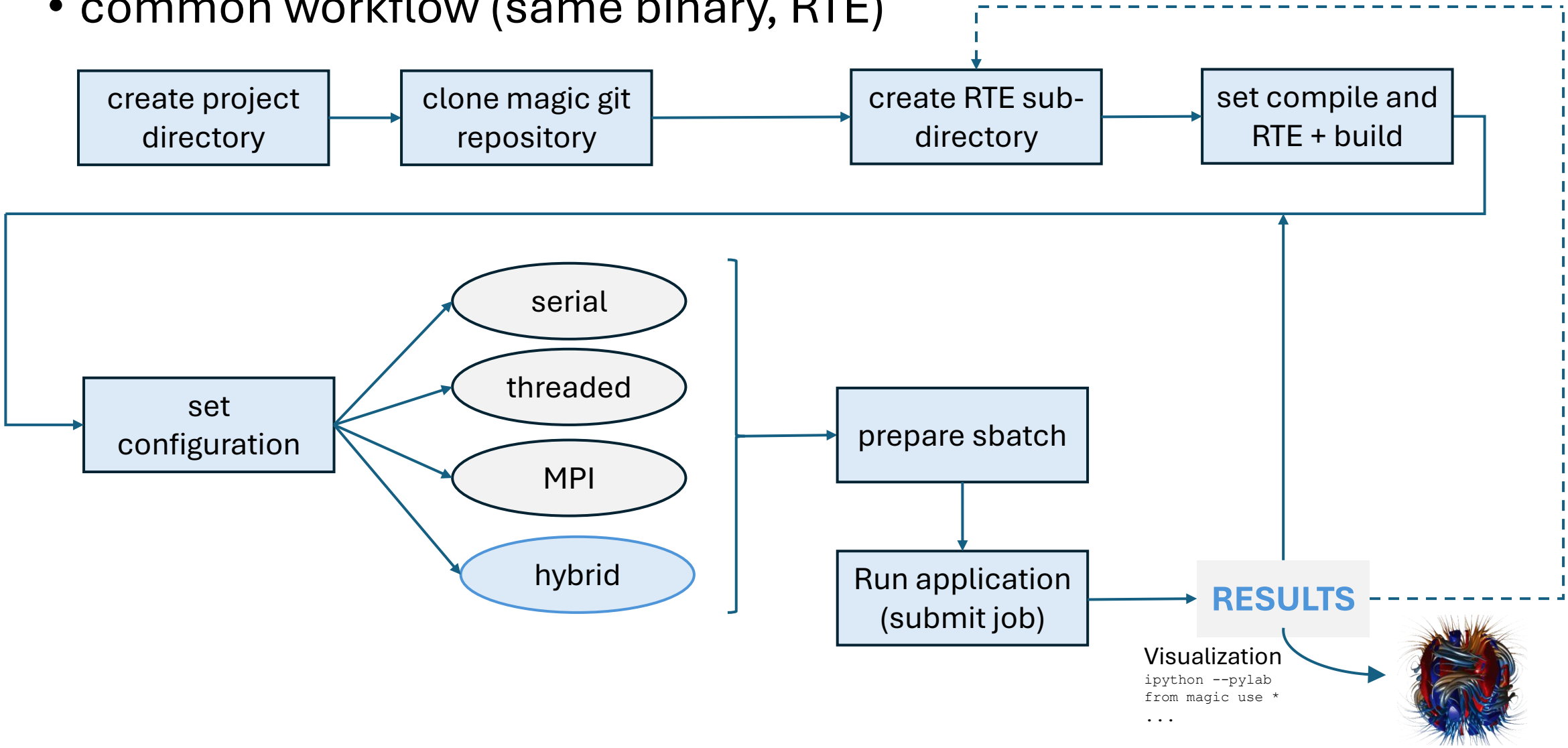
- software
  - available via modules (maintained via EasyBuild)
  - toolchains, hierarchical module name space (module visibility depends on load path)
  - most important commands
    - `module -w $COLUMNS -redirect <command>` use options for nice output
    - `module list` → show loaded modules
    - `module purge` → unload all module
    - `module load <mod-1> . . . <mod-K>` → load modules specified
    - `module spider <modName>` → search for module load path (example)
    - `module avail` → list all directly loadable
  - for the workshop a single compile and runtime environment or toolchain is used, based on:

    `gcc-13.2`

    (Note: magic needs a C- and Fortran compiler)

work flow

- common workflow (same binary, RTE)



set-up training environment

- create project directory:

```
mkdir /scratch/guest<ID>/magic-WS
```

- clone magic git repo (latest change)

```
cd /scratch/guest<ID>/magic-WS
git clone https://github.com/magic-sph/magic.git
cd magic ; source sourceme.sh
```

- create RTE directory + build

```
mkdir ${MAGIC_HOME}/proj-<compiler-Version>
e.g.:
mkdir ${MAGIC_HOME}/proj-gcc-13.2
```

- set compile and runtime environment

```
module load imkl/2025.1.0 GCC/13.2.0 OpenMPI/4.1.6
→ source module-gcc-13.2.0.sh  (alternative)
export PROJDIR=${MAGIC_HOME}/proj-gcc-13.2
```

build / configuration

- build shtns

```
export CC=gcc
cd ${MAGIC_HOME}/bin
```

  - patch enables compiler optimization for AMD and add linking of missing mkl lib.

```
patch install-shtns.sh /scratch/guest<ID>/ws-files/build/install-shtns.sh.patch
```

  - configure, compile, install SHTHNS

```
./install-shtns.sh > build.log 2>&1 &
```

Upon failure, before next try run: `rm -rf shtns`

→ include files and libraries will installed
→ check for errors, warning and whether compiler flags make sense for your CPU architecture

- build magic

```
export FC=mpif90
cd ${MAGIC_HOME}
patch CMakeLists.txt  /scratch/guest<ID>/ws-files/build/CMakeLists.txt.patch
patch src/CMakeLists.txt /scratch/guest<ID>/ws-files/build/src-CMakeLists.txt.patch
mkdir build
cd build
```

  - create cmake and make files

```
cmake .. -DUSE_SHTNS=yes -DUSE_OMP=yes -DUSE_MPI=yes
```

→ online screen session and `ws-files/refout/cmake.log`

Upon failure: `cd .. ; rm -rf build`

  - run compilation of the magIC binary

```
make -j
```

→ online screen session and `ws-files/refout/make-j.log`

  - Upon success: `magic.exe` created in `build`

build / configuration

- set configuration for model (input files)
  - Copy magIC binary to $PROJDIR:

    ```
    cp ${MAGIC_HOME}/build/magic.exe ${PROJDIR}
    ```

  - create subdirectory for system modeled (dynamo_bench):

    ```
    cd ${PROJDIR}
    rsync -av ${MAGIC_HOME}/samples/dynamo_benchmark .
    ```

    no editing of input.nml neccessary

run magic

- prepare sbatch (script):

```
cp /scratch/guest<ID>/ws-files/slurm/magic-gcc-sbatch.sh ${PROJDIR}/dynamo_benchmark
cd ${PROJDIR}/dynamo_benchmark
```

  open file in editor and change the strings tagged with <...> with ID of guest account
  Also set MAGIC_HOME variable

- SLURM job(s):
  - please always use the *reservationName* associated with your login ID
  - submit batch script

```
sbatch magic-gcc-sbatch.sh
```

  - check job status

```
squeue -u guest<ID>
```

  - stop/cancel running job

```
scancel <jobID>
```

  - start interactive session:

```
source module-gcc-13.2.sh
srun --account=planets --qos=swan_default --reservation=guest<ID>
     --time=02:00:00 -export=all bash
```

  - hint: for consecutive runs save (copy, move) and delete all *.test, *.log *.BIS files

verify results

- Check ‘log.test’: no keywords ‘error, fail’
  -> show well formed log in login session
  check SLURM std{err, out} files
- verify simulation results are correct:
  compare to reference files (reference.out → e_kin.test; referenceMag.out → e_mag_oc.test)

```
md5sum reference.out e_kin.test
        c34be15278c48b9e5355cab4a2bc24f6  reference.out
        c34be15278c48b9e5355cab4a2bc24f6  e_kin.test
md5sum referenceMag.out e_mag_oc.test
        76e759278c3bc1bea61568d2decac012  referenceMag.out
        76e759278c3bc1bea61568d2decac012  e_mag_oc.test
```

- act as system-test actually for
  - changed compiler version, options, optimizations
  - code changes or enhancements

Visualization (1/3)

- magIC provides own module for graphical visualization
- Based on serial FORTRAN code (→gfortran – compiler have to used)
- setup of python RTE utilizing with *conda* see next slide
- use the magIC visualization with help ipython
- any other bespoke framework can be used, too

visualization (2/3)

- postpressing/visualization implemented via python3 module
- package dependencies:
  ipython, pylab, numpy, scipy, matplotlib, meson
- setup conda (`python3`) environment
  - load conda framework

    ```
    module load Miniconda3
    source ${EBROOTMINICONDA3}/etc/profile.d/conda.sh
    export FC=gfortran
    ```
  - create conda environment + use

    ```
    conda create -n <envName>
    conda activate <envName>
    ```
  - install packages

    via tested package list

    ```
    conda install --file /scratch/guest<ID>/ws-files/conda/pkgs4magic.txt
    ```

    via dependency list

    ```
    conda install ipython numpy scipy matplotlib meson
    ```
- alternatives : use `pip` or other pkg-manager or ‘hand crafted/bespoke’ - installation

visualization (3/3)

- active `magIC´ environment, extend PYTHONPATH to

- interactive example session with ipython

```
cd ${PROJDIR}/dynamo_benchmark
ipython --pylab                          use ipython
from magic import *                      load magic module
    FIRST execution: will create magic python modules from fortLIB f90 files
MagicTs(field="e_mag_oc")                plot magnetization
MagicTs(field="dipole")                  plot dipole behaviour vs. time
S=Surf()                                 create surface object
S.surf(field="Br",r=1)                   Plot radial field at surface
```

save your work

- create `tar` – archive of your account or project directory(ies), better relevant data only

  → `tar jcpvf /home/guest<ID>/magicWS-guest<id>.tar.bz2  \`
  `/scratch/guest<id>/<projectDir>`

- transfer via `scp` to your institute/home ‘login’ node
- we provide cloud download:
  - send e-mail to heckes@mps.mpg.de containing full path to archive name, your full name and guest account name
  - plan B only

Layout template

- Template text