PLASMA PEPSC

DRESDEN concept

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

Hands-on training for PIConGPU users

getting started with PIConGPU on LUMI

Richard Pausch[1], Sergey Bastrakov[1], Michael Bussmann[1,2], Finn-Ole Carstens[1], Alexander Debus[1], Fabia Dietrich[1], Sergey Ermakov[1,3], Julian Lenz[1,2], Brian Marre[1], Tapish Narwal[1,2], Pavel Ordyna[1], Klaus Steiniger[1,2], Jessica Tiebel[1], René Widera[1]

[1] *Helmholtz-Zentrum Dresden – Rossendorf*

[2] *CASUS - Center for Advanced Systems Understanding*

[3] *ETH Zürich*

2024-10-25

Introduction

What will we do today?

1. check your access to LUMI
2. setup and run first PIConGPU simulation
3. data analysis of simulation using openPMD and Jupyter
4. dive deeper into how to setup a PIConGPU simulation
5. time for questions
6. start another project on your own

DRESDEN concept

HZDR
HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

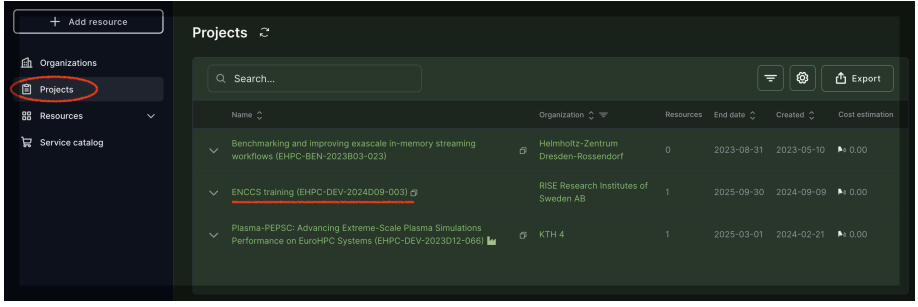
Part I

Check LUMI access

Accessing LUMI

Checking you are part of the workshop project

- go to: `https://my.lumi-supercomputer.eu/`
- login with your credentials
- go to projects and check whether you have access to **ENCCS training (EHPC-DEV-2024D09-003)**



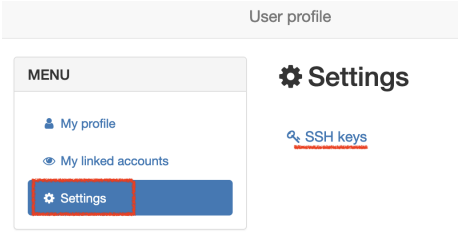
Accessing LUMI

Setting up an SSH key pair

- create a (LUMI specific) key pair:

```
ssh-keygen -t ed25519 -f /home/username/.ssh/id_juwels_ed25519
```

- now you should have:
  - your private key: `id_lumi_ed25519`
  - and your public key: `id_lumi_ed25519.pub`
- go to: https://mms.myaccessid.org/profile/



- upload your **public** key

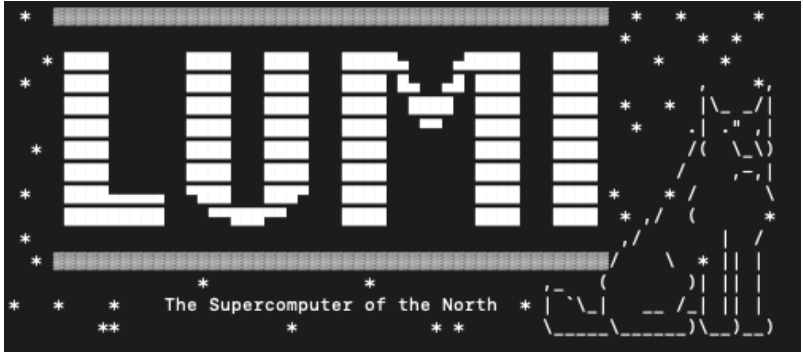
Accessing LUMI

ssh into LUMI

- execute:

```
ssh -i  /.ssh/id_lumi_ed25519 your_username@lumi.csc.fi
```

- you should see:



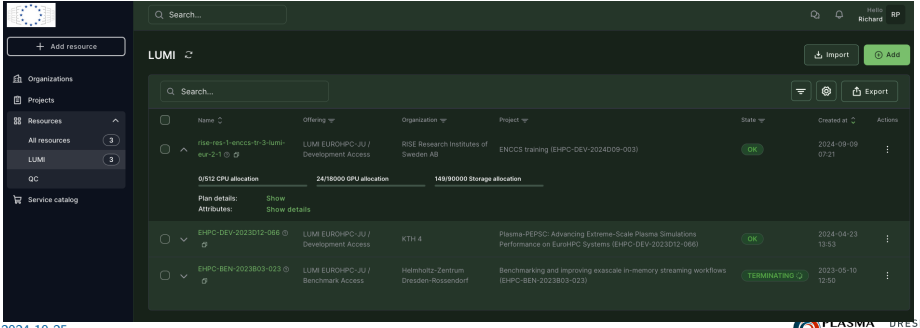
Accessing LUMI
your workshop project

- check which projects you have access to:

```
lumi-allocations
```

```
Data updated: 2024-10-21 17:03:48
Project            |                CPU (used/allocated)|                GPU (used/allocated)|          Storage (used/allocated)
-------------------------------------------------------------------------------------------------------------------------------
project_465001131  |              0/1000   (0.0%) core/hours|          5/40000   (0.0%) gpu/hours|          0/90000   (0.0%) TB/hours
project_465001310  |            113/512000 (0.0%) core/hours|         24/18000   (0.1%) gpu/hours|        145/90000   (0.2%) TB/hours
```

- compare to:



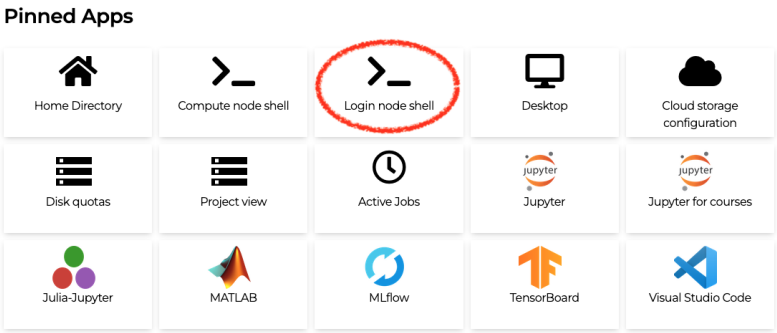
Accessing LUMI

alternative access via browser

- LUMI offers a web interface

  `www.lumi.csc.fi`

- login with you credentials
- select compute node shell



Accessing LUMI

creating your working directories

- we are working with

  `project_465001310`

- project directory

  `/projappl/project_465001310`

- please create a directory there

```
cd /projappl/project_465001310
mkdir $(whoami)
```

- simulation directory

  `/scratch/project_465001310`

- please create a directory there

```
cd /scratch/project_465001310
mkdir $(whoami)
```

DRESDEN concept

HZDR
HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

Part II

First PIConGPU simulation

Setting up PIConGPU (on LUMI)

There is a manual!

`https://picongpu.readthedocs.io`

is your first source of help!

The steps we follow now are similar to the 'PIConGPU in 5 Minutes on Hemera' tutorial there.

Performing PIConGPU simulations

The general workflow

1. **Get PIConGPU**

```
$ git clone https://github.com/ComputationalRadiationPhysics/picongpu.git
```

2. **Prepare and load environment**

```
$ source picongpu.profile
```

3. **Create simulation setup**

```
$ pic-create $PIC_EXAMPLES/LaserWakefield myLWFA
```

4. **Adjust setup parameters**

```
$ vim simulation.param
```

5. **Compile setup**

```
$ pic-build
```

6. **Run setup (submit to batch system)**

```
$ tbg -s -t -c etc/picongpu/*.cfg mySimOutputDirectory
```

7. **Analyze results**

```
$ jupyter-lab
```

Setting up PIConGPU on LUMI

(1) get the source code

- download the PIConGPU source code:

  in home directory

```
cd $HOME/
mkdir src
cd src
```

- get source code

```
git clone https://github.com/ComputationalRadiationPhysics/picongpu.git
```

Setting up PIConGPU on LUMI

(2) prepare and load environment

- copy default setup file:

```
cp picongpu/etc/picongpu/lumi-eurohpc/lumi-G_hipcc_picongpu.profile.example
$HOME/lumi-G_hipcc_picongpu.profile
```

- either use your editor of choice in the terminal (vim, emacs, nano) or use the web interface
- adjust the following lines:

```
 8 export MY_MAILNOTIFY="ALL"
 9 export MY_MAIL="your.mail@server.gov"
16 export PROJID=project_465001310
17 export PROJECT_DIR=/projappl/$PROJID/
18 export SCRATCH=/scratch/$PROJID/
27 export EDITOR="emacs -nw"
72 export PIC_LIBS=$PROJECT_DIR/workshop_software/local
81 export PICSRC=$HOME/src/picongpu
```

```
source $HOME/lumi-G_hipcc_picongpu.profile
```

Setting up PIConGPU on LUMI

(3) Create simulation setup

```
cd $HOME
mkdir -p picInputs
cd picInputs
pic-create LaserWakefield LWFA
cd LWFA
```

- ...
- peek into `include/picongpu/param`:

  ```
  $ ls -l include/picongpu/param
  ```

  ⇒ The `*.param` files largely define the simulation setup

Setting up PIConGPU on LUMI

(4.1) Adjust setup parameters

- for now we will not go into details
- just to follow the workflow, we will adjust **one compile-time** parameter
- compile-time parameters are in `include/picongpu/param`
- let's see where the parameter files are:

```
cd include/picongpu/param
ls -l
```

- let's edit the laser configuration

```
vim incidentField.param
```

66 change a_0 from 8.0 to 5.0

Setting up PIConGPU on LUMI

(4.2) Adjust setup parameters

- just to follow the workflow, we will adjust **one run-time** parameter
- run-time parameters are in `etc/picongpu/*.cfg`
- let's see where the parameter files are:

```
cd $HOME/picInputs/LWFA
cd etc/picongpu/
ls -l
```

- let's edit the `4.cfg`

```
vim 4.cfg
```

68 remove checkpointing and add mapped memory strategy

```
TBG_openPMD="--openPMD.period 100 \\
             --openPMD.file simData \\
             --openPMD.ext bp \\
             --openPMD.dataPreparationStrategy mappedMemory"
```

Setting up PIConGPU on LUMI
(5) compile setup

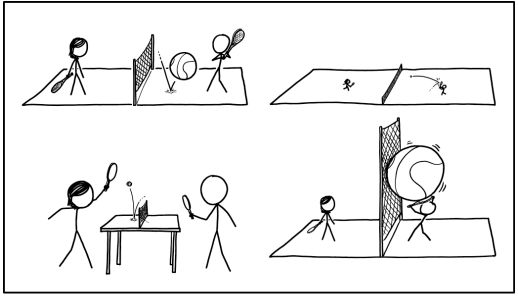
- go back to the root of your LWFA example
```
cd $HOME/picInputs/LWFA
```
- get compute resources for compiling:
```
getDevice
```
- compile your setup
```
pic-build
```
- better:
```
pic-build -f 2> err.txt
```

  - Allows to read errors in `err.txt` (after compiling)
  - Using option `-f` ensures we do not use cached `*.param` files but the original ones during compile. Accidentally using cached files is a common source of errors.
  - `-c ...` fixes a current issue with CMake

Setting up PIConGPU on LUMI

You can spend compile time to rethink your parameter choices (or check the latest xkcd...)

```
pic-build -f 2> err.txt
```



PARAMETERBALL IS A RAQUET GAME DIVIDED INTO FOUR QUARTERS, WITH BALL SIZE, COURT SIZE, AND NET HEIGHT RANDOMIZED EACH QUARTER.

Setting up PIConGPU on LUMI

We have a reservation

- in order to get your simulation to start, please use today's reservation
- the reservation is called:

  `ENCCS_day3`

- adjust or `tbg` template file

  ```
  vim etc/picongpu/lumi-eurohpc/standard-g.tpl
  ```

  replace line 25:

  ```
  #SBATCH --partition=!TBG_queue
  ```

  with:

  ```
  #SBATCH --reservation=ENCCS_day3
  #SBATCH --exclusive
  ```

Setting up PIConGPU on LUMI
(6) Run setup

- **go back to LUMI head node - do not run `tbg` from a compute node**
- submit job using `tbg` (assuming you changed the `standard-g.tpl`)

```
tbg -s -t -c etc/picongpu/4.cfg $SCRATCH/$(whoami)/01_LWFA
```

- `tbg` stand for Template Batch Generator and is our abstraction for setups for various clusters
- it sets a submit command via `-s` (default on LUMI: `sbatch`)
- it uses a template via `-t` (default on LUMI: `standard-g.tpl`)
- it uses a setup configuration
- does your job run?

```
squeue -u $USER
scontrol show jobid <Your job's id>
```

DRESDEN concept

HZDR
HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

Part III

Data analysis with openPMD and Jupyter

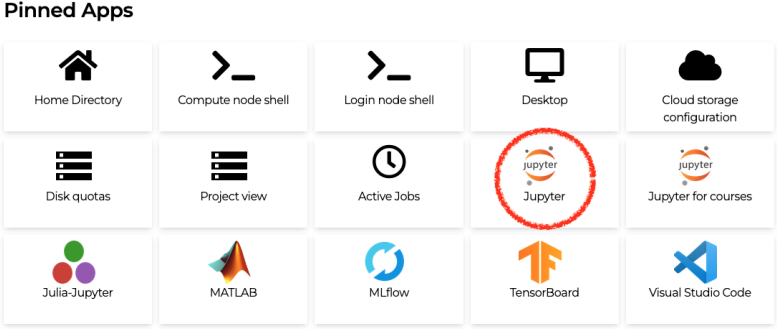
Analysing your PIConGPU simulation

using jupyter, openPMD-api, ...

- again use the LUMI web interface

  `www.lumi.csc.fi`

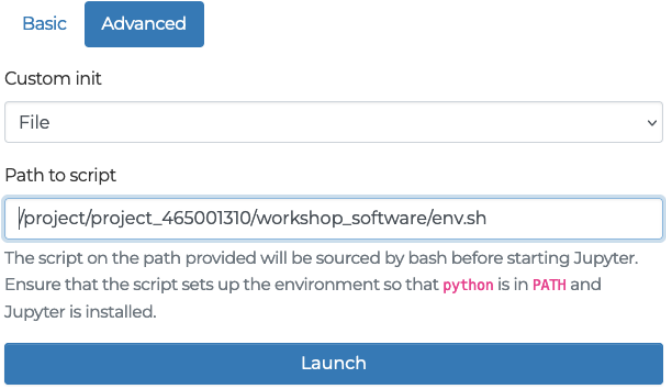
- login with your credentials
- select Jupyter



Analysing your PIConGPU simulation

getting a jupyter job

- set reservation
- use **2 cores**
- use advanced setting and use prepared python environment
- for now use your home directory as working directory



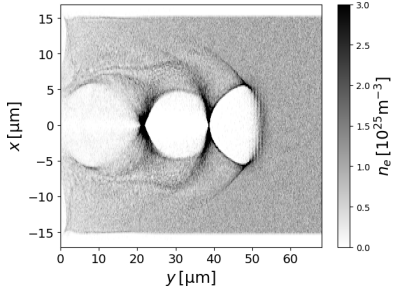
Analysing your PIConGPU simulation

run an example notebook

- copy notebook to your home directory

```
cp /projappl/project_465001310/workshop_software/notebooks/LWFA_analysis.ipynb $HOME/
```

- change the simulation directory to your simulation path



- execute it
- explore other data

DRESDEN concept

HZDR
HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

Part IV

Adjusting simulation parameters

PIConGPU `*.param` files

`simulation.param`

- contains grid layout
  Δx, Δy, Δz, Δt
- reference density
  `BASE_DENSITY_SI`
- number of particles per cell to be initialized
  `TYPICAL_PARTICLES_PER_CELL`

PIConGPU `*.param` files

`speciesInitialization.param`

- contains a pipeline that described how macro-particles are initialized

```
using InitPipeline = pmacc::mp_list<
    ...>;
```

- particles can be created based on a density profile

```
CreateDensity<densityProfiles::Gaussian,
    startPosition::Random,
    PIC_Electrons>
```

- or derived from an already existing particle distribution

```
Derive<PIC_Electrons, PIC_Ions>
```

- particle attributes can be manipulated to set charge states, temperatures, drifts, ...

PIConGPU *.param files

density.param

- LWFA contains only a Gaussian density profile
- let's have a look at it ...
- more density profiles are available, most flexible is:

```
struct FreeFormulaFunctor
{
    HDINLINE float_X operator()(const floatD_64& position_SI,
        const float3_64& cellSize_SI) {

        float_X s = 1.0_X - 5.0_X * math::abs(position_SI.y() - 0.0002_X);
        s *= float_X(s >= 0.0);
        return s;
    }
};
using FreeFormula = FreeFormulaImpl<FreeFormulaFunctor>;
```

PIConGPU *.param files

incidentField.param

- LWFA uses Gaussian laser profile
- at the very end:

```
using XMin = profiles::None;
using XMax = profiles::None;
using YMin = PARAM_LASERPROFILE;
using YMax = profiles::None;
using ZMin = profiles::None;
using ZMax = profiles::None;
```

- laser definition:

```
using GaussianProfile = profiles::GaussianPulse<GaussianPulseParam>;
```

- let's have a detailed look

Task

run your own mini-simulation campaign

- based on your initial LWFA setup, either:
- simulate half and double the density and study the influence on the wakefield
- simulate $a_0 = 1.0$ and $a_0 = 0.2$ and see the effect on the wakefield
- play around with any other parameter you like

DRESDEN concept

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

Part V

Time for your questions

- **What is ...?**
- **How do I do ...?**
- **Is it possible to ...?**
  Yes! (with enough time and effort ;-)
- **Where do I get more information on ...?**
  In order of precedence:
  - The manual: https://picongpu.readthedocs.io
  - Old issues
    https://github.com/ComputationalRadiationPhysics/picongpu/issues?q=is%3Aissue
  - Open a new issue

DRESDEN concept

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

Part VI

Starting from another examples

Explore other default examples

setup your own example - we are her to help you

- **Kelvin Helmholtz instability**
  `KelvinHelmholtz`
  use `4.cfg` and add openPMD output
  https://github.com/ComputationalRadiationPhysics/picongpu/tree/dev/share/picongpu/examples/KelvinHelmholtz

- **Solid density foil target for proton acceleration**
  `FoilLCT`
  use `4.cfg` or `8.cfg`
  only uses 2D
  https://github.com/ComputationalRadiationPhysics/picongpu/tree/dev/share/picongpu/examples/FoilLCT