



Interactive Computing with Jupyter on Piz Daint, using Python, ParaView and Julia

Tim Robinson, Jean Favre and Harmen Stoppels, CSCS

Apr 26, 2021

Agenda

09:00 – 10:45	Jupyter, JupyterLab and JupyterHub on Piz Daint
10:45 – 11:00	<i>Break</i>
11:00 – 12:00	ParaView and Jupyter, Part I
12:00 – 13:00	<i>Lunch</i>
13:00 – 14:00	ParaView and Jupyter, Part II
14:00 – 14:15	<i>Break</i>
14:15 – 16:15	Programming in Julia



CSCS

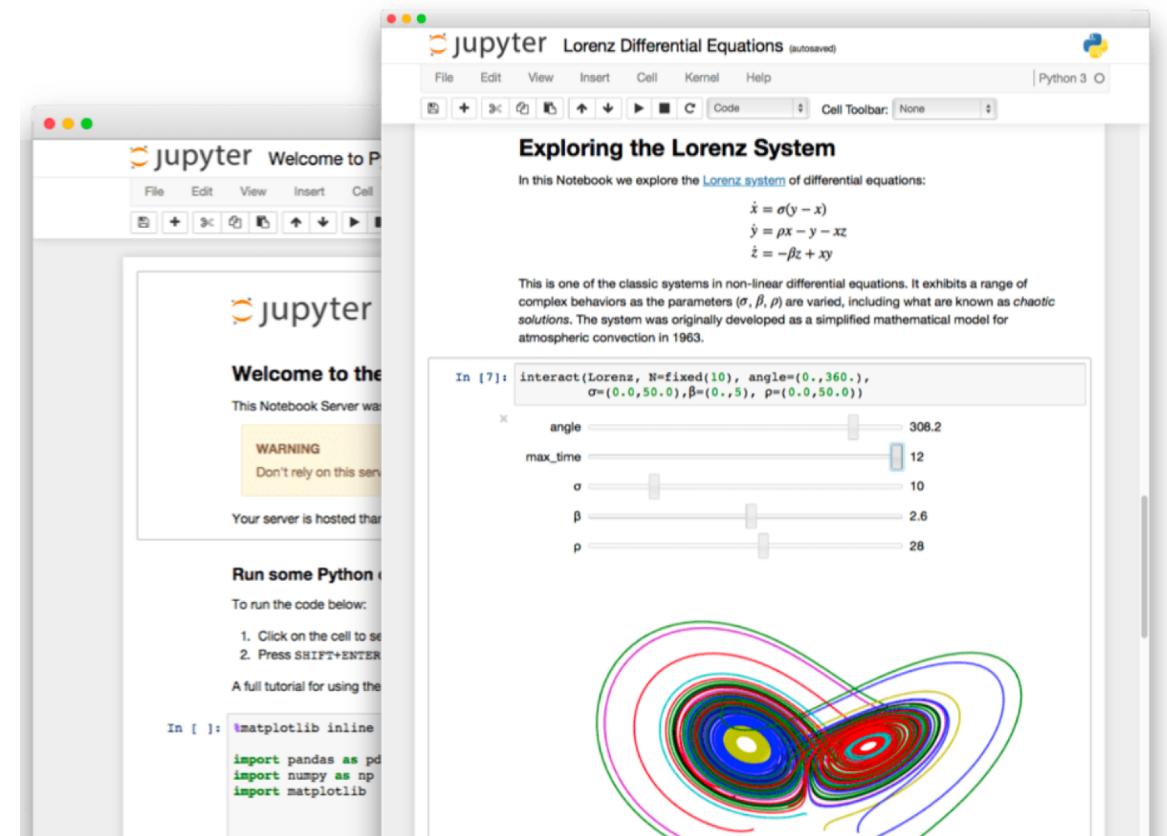
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETHzürich

Part I: Jupyter: Jupyter Notebook, JupyterLab and JupyterHub

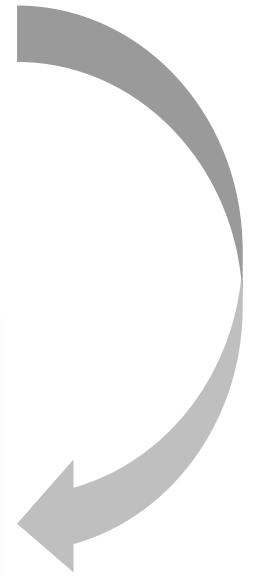
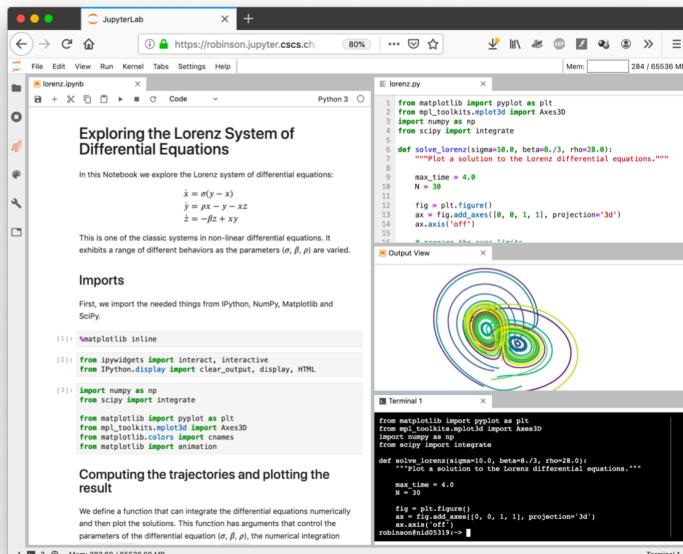
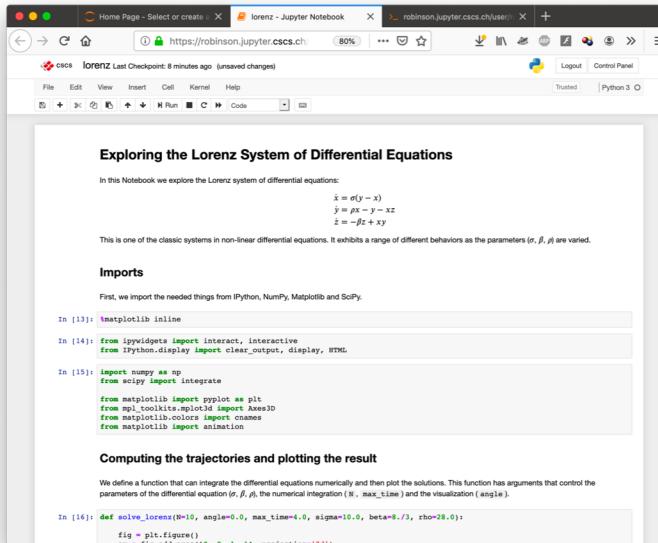
Jupyter Notebook

- Jupyter Notebook is an open-source web app for creating reproducible computational narratives
- Documents can contain live code, equations, narrative text, visualizations, and rich media
- Documents can be shared or exported to PDF, HTML, LaTeX, etc.
- Working environment includes
 - File browser
 - Terminal
 - Support for many languages: Python, R, Julia, C++, ...
 - Extensible design
 - Many server/client plugins
- Notebook is attached to a “kernel”, which does the actual computation



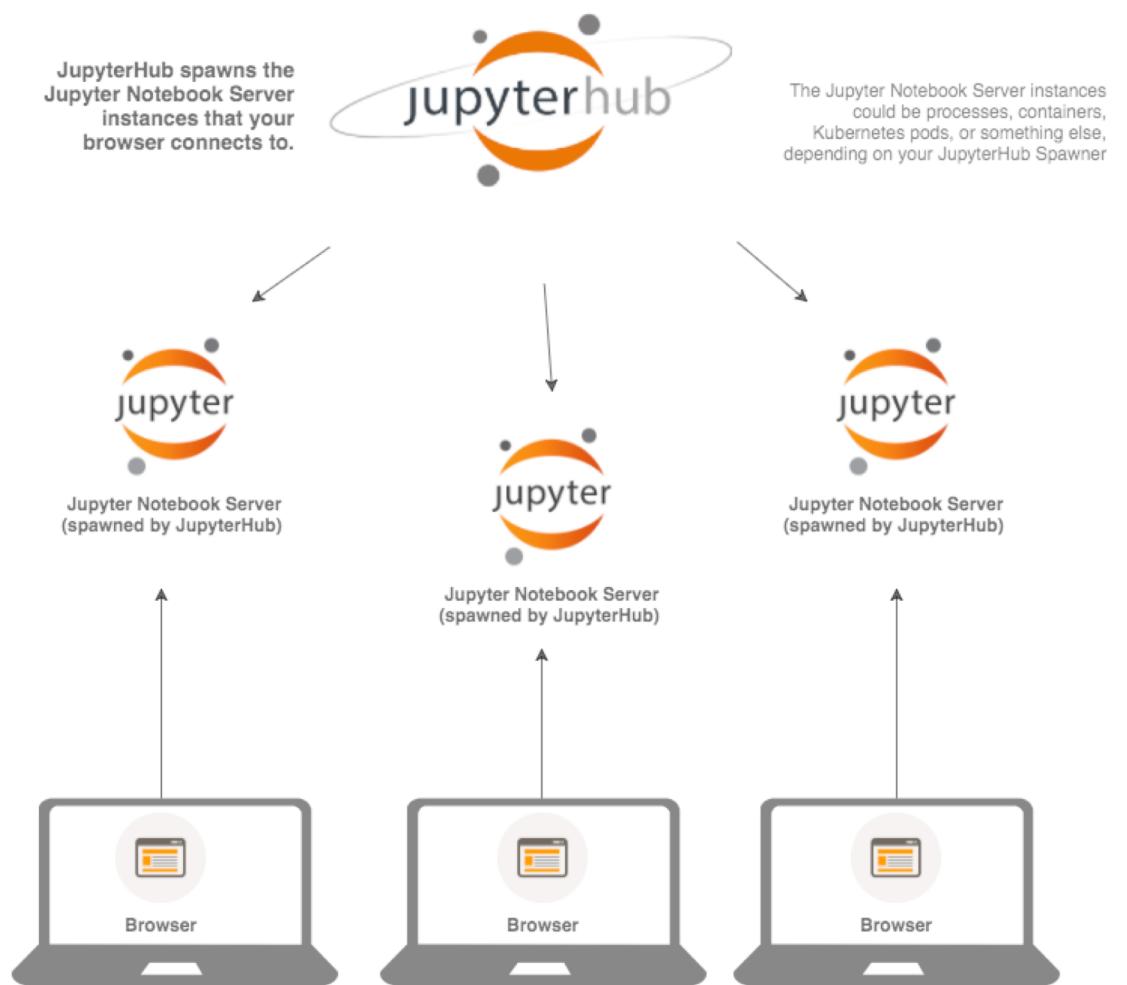
JupyterLab

- JupyterLab is the next-generation user interface for Jupyter
 - Higher degree of interaction between notebooks, documents, text editors and other activities (arrange with tabs/splitters)
 - More advanced interactive development environment
 - Uses the same notebook document format
 - JupyterLab is the default interface at CSCS
 - Classic notebook interface is still accessible by modifying the URL from /lab to /tree
 - Or: Help -> Launch Classic Notebook



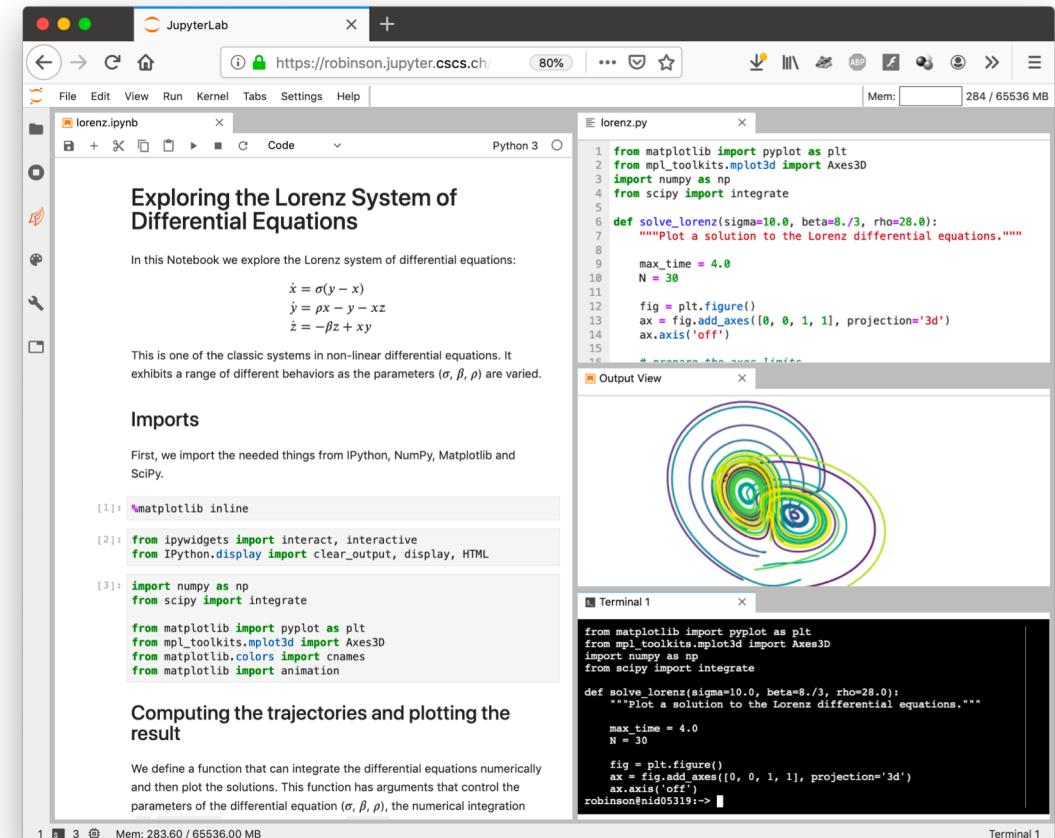
JupyterHub

- Multi-user server for Jupyter Notebooks (designed for classrooms, research labs, Universities...)
- Spawns, manages and proxies multiple instances of the single-user Jupyter Notebook server
- Main components:
 - **Hub** (tornado process)
 - **configurable http proxy** (node-http-proxy) that receives the requests from the client's browser
 - multiple **single-user Jupyter notebook servers** (Python/IPython/tornado) that are monitored by Spawners
 - an **authentication class** that manages how users can access the system



JupyterHub for Piz Daint

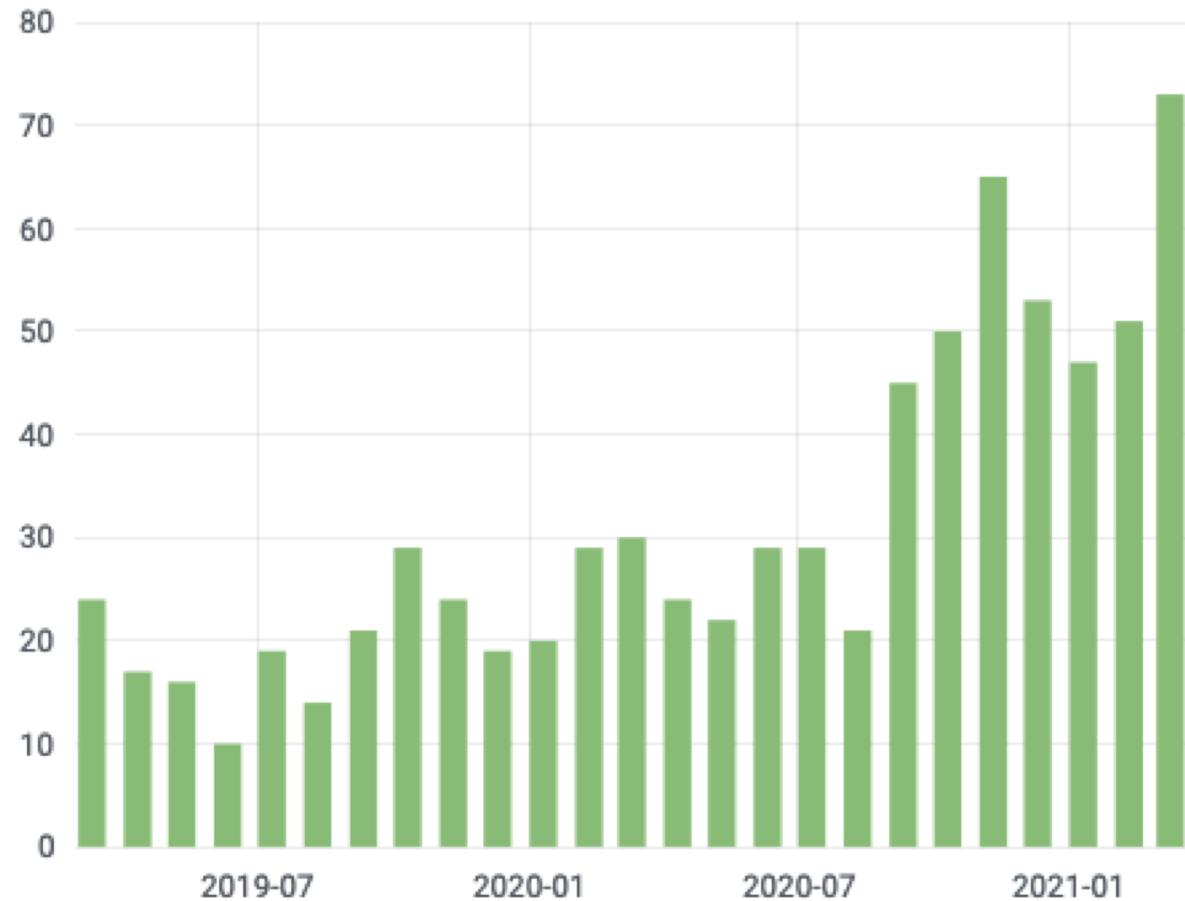
- JupyterHub for Piz Daint is accessible at:
<https://jupyter.csccs.ch>
- Available to anyone with a compute allocation on Piz Daint
- Servers are launched on the compute nodes (GPU or multicore)
- You have dedicated access to the full compute capability of the node(s)
- Special Slurm reservations for interactive computing grow and shrink automatically according to demand
- In most cases you should have to wait less than five minutes for your allocation



The screenshot shows a JupyterLab interface with the following components:

- Top Bar:** Shows the URL <https://robinson.jupyter.csccs.ch/>, a memory usage of 284 / 65536 MB, and various browser controls.
- Left Sidebar:** Includes icons for file operations (New, Open, Save, etc.), a search bar, and a "Code" tab.
- Central Area:**
 - Code Cell:** Displays Python code for solving the Lorenz system. It includes imports for matplotlib, ipywidgets, numpy, scipy, and animation, along with a function definition for `solve_lorenz`.
 - Text Cell:** Describes the Lorenz system and its classic chaotic behavior.
 - Output View:** Shows a 3D plot of the Lorenz attractor, which is a set of chaotic trajectories.
- Bottom Area:** A terminal window titled "Terminal 1" containing the same Python code and output, showing the execution of the script and the resulting 3D plot.

JupyterHub monthly users on Piz Daint



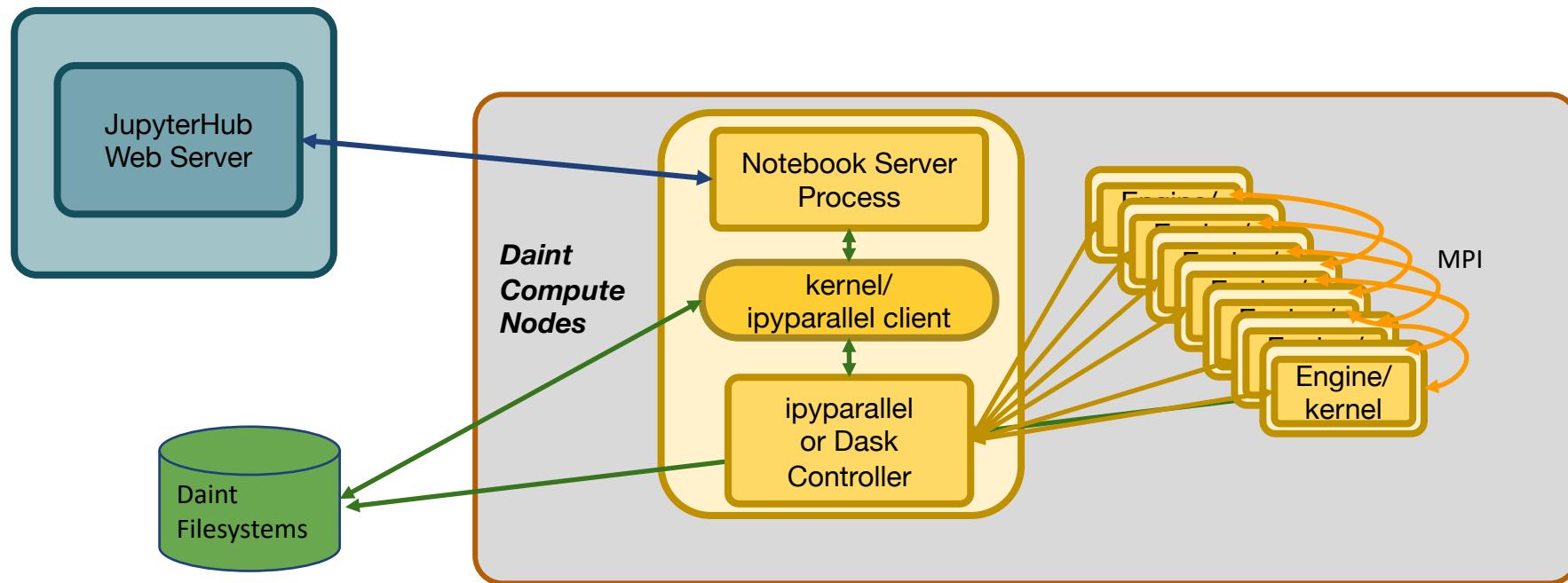
Challenges with interactive computing on HPC

- Shared parallel filesystems
- Network connectivity
- Slurm
- Warm swapping
- All kinds of other possible issues - usually these are temporal!

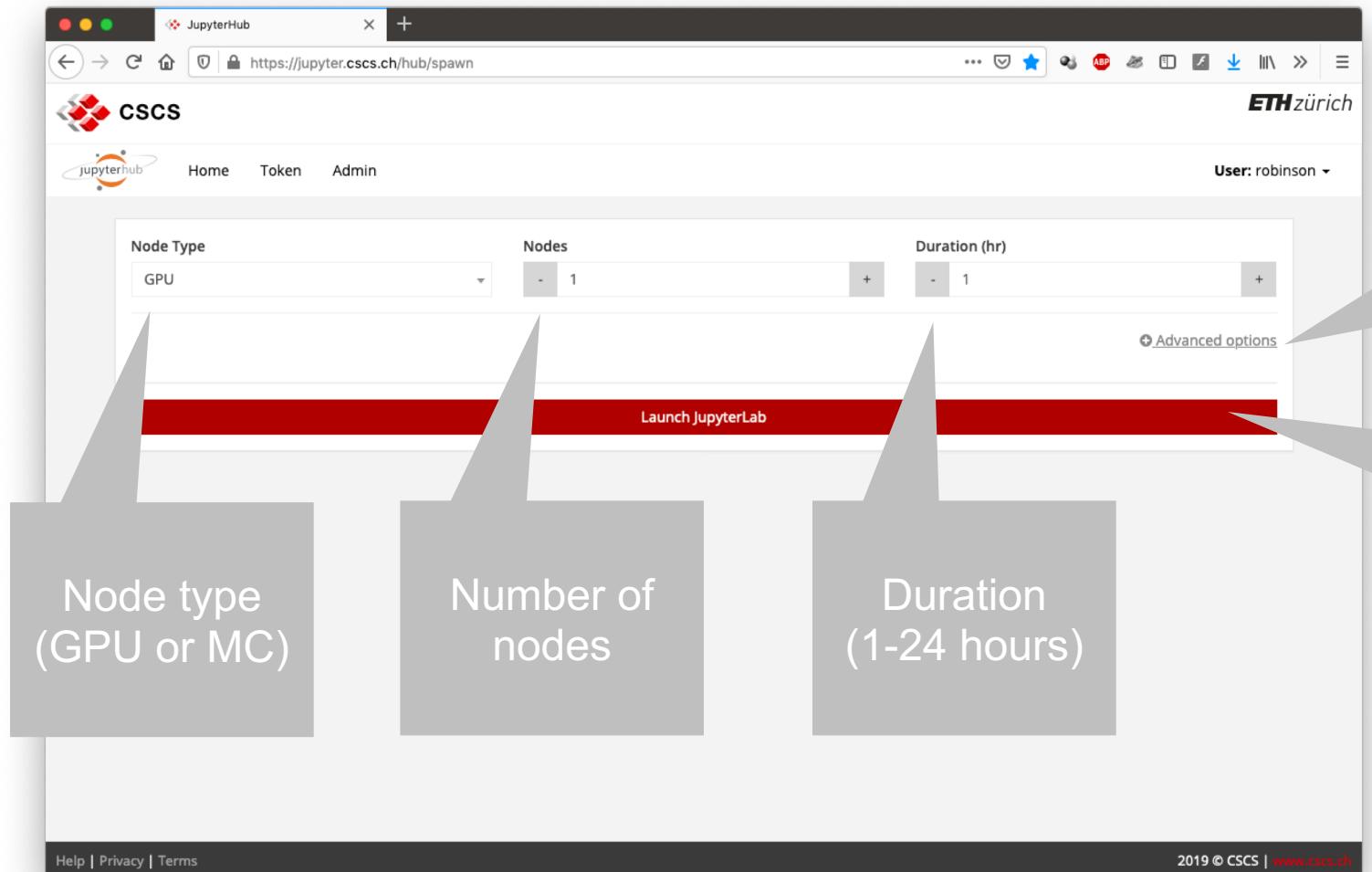
- Debugging problems:
 - Check for any maintenance announcements on the spawner page
 - Check your own internet connectivity
 - Check for old JupyterHub sessions in browser tabs (especially if kernels are not starting or are slow to connect or restart)
 - Inspect the Jupyter log file in your \$SCRATCH directory
 - If problems persist, submit a ticket at <https://support.cscs.ch>
 - Include as much detailed information as possible – precise time periods, Slurm jobid, what was working, what was not working...

JupyterHub implementation at CSCS

jupyter.cscs.ch



Basic options



Advanced options

The screenshot shows the JupyterHub spawn interface on a web browser. The URL is <https://jupyter.cscs.ch/hub/spawn>. The interface includes fields for Node Type (GPU), Nodes (1), Duration (hr) (1), Queue (Dedicated Queue (Max. 4 Nodes)), Project Id (leave empty for default), JupyterLab Version (1.1.1), and MPI Processes Per Node (1). There are also checkboxes for starting IPyParallel and Distributed Dask clusters.

Annotations on the left side:

- Queue**: Points to the Queue dropdown.
- Reservation**: Points to the Advanced Reservation section.
- IPyParallel (multi-node notebooks)**
Replaced by ipcluster magic

Annotations on the right side:

- Project Id (account)**: Points to the Project Id field.
- JupyterLab version**: Points to the JupyterLab Version field.

Annotations at the bottom:

Distributed Dask cluster (multi-node notebooks)
Use dask dashboard or try dask-jobqueue

JupyterLab interface

Standard output and stderr written to:

```
$SCRATCH/jupyterhub_slurmspawner_<jobid>.log
```

File Browser

`$HOME` is mounted by default

To access `$SCRATCH` and `$PROJECT` create sym links by issuing the following commands in a notebook cell:

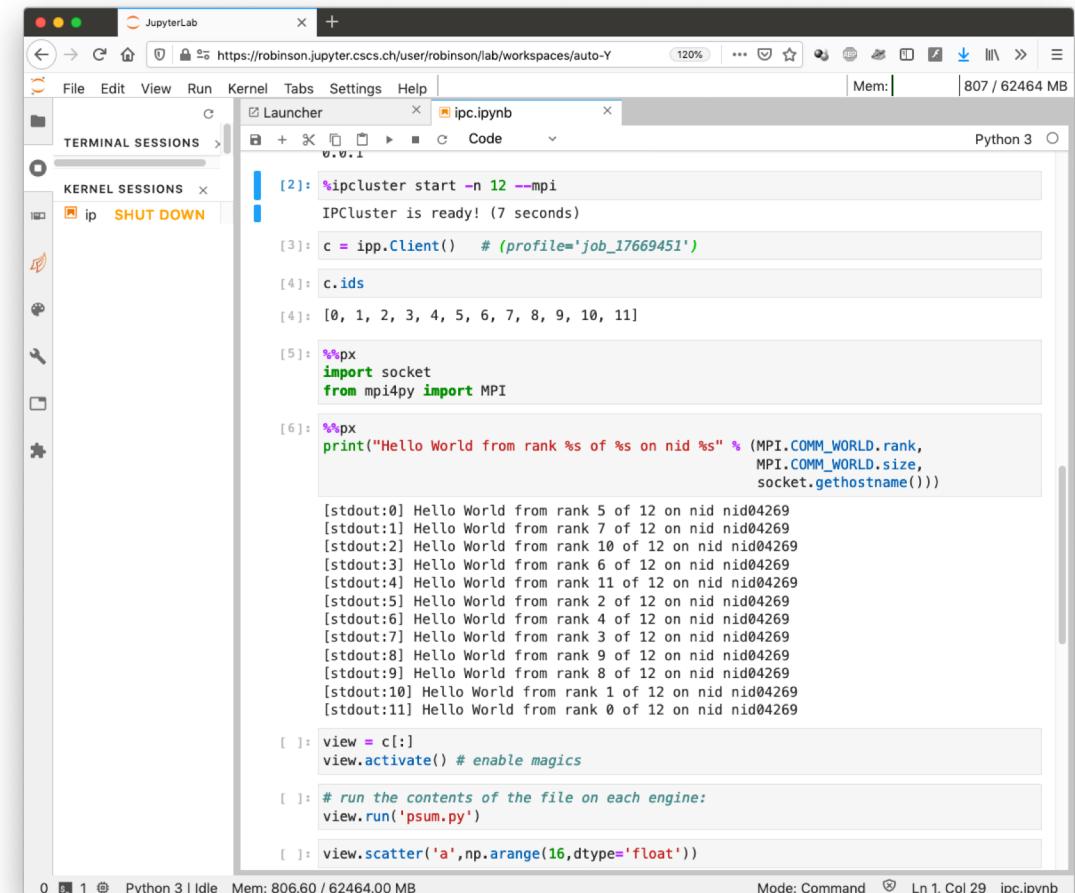
```
!ln -s $SCRATCH ./scratch  
!ln -s $PROJECT ./project
```

Getting the Python packages you need

- **pip install**, direct from the notebook (simple)
- **Virtual environments and kernels** (better)
- **module load** commands in \$HOME/.jupyterhub.env (can be useful)
- Install **miniconda** and use **conda** environments (if you prefer conda)
- **Containers** (work in progress; potentially offers more flexibility)

IPyParallel and ipcluster magic

- IPCluster magic commands for automatically starting and stopping ipyparallel clusters (ipcontroller and ipengines)
- `module load ipcmagic` in `$HOME/.jupyterhub.env`
- `import ipcmagic` in notebook
- Makes it easier to use MPI directly in the notebook

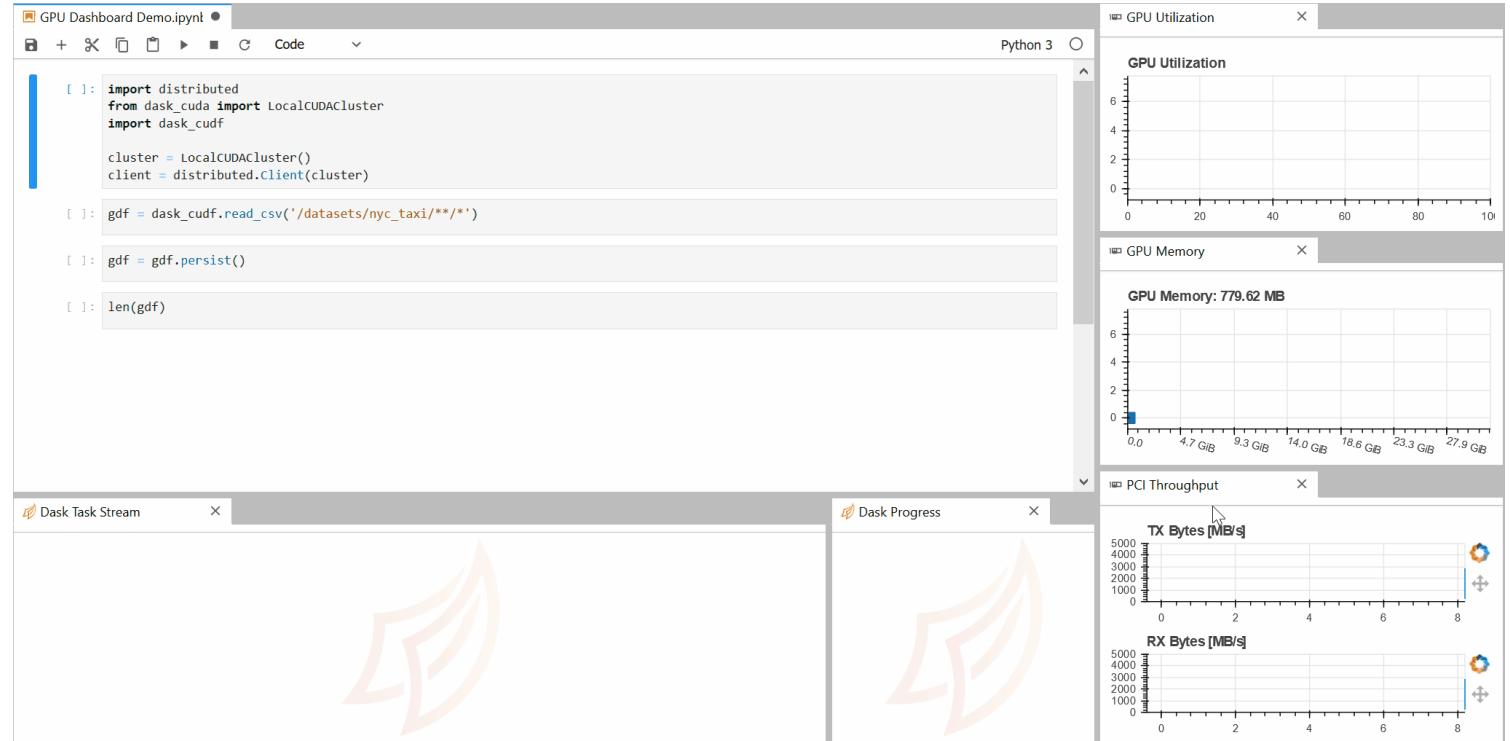


The screenshot shows a JupyterLab interface with a terminal session titled 'ipc.ipynb'. The session output displays the following sequence of commands and their results:

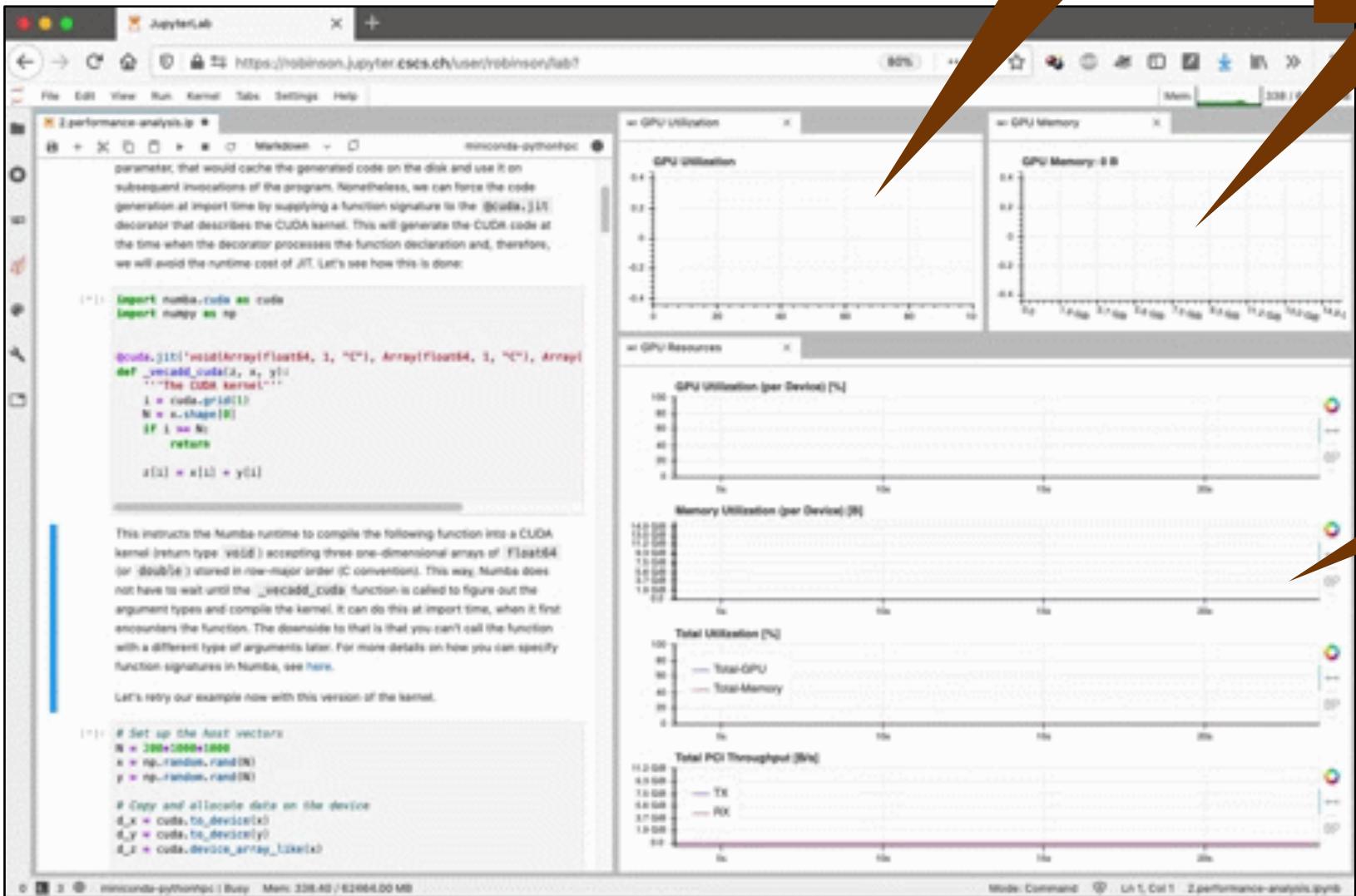
```
[2]: %ipcluster start -n 12 --mpi
IPCluster is ready! (7 seconds)
[3]: c = ipp.Client() # (profile='job_17669451')
[4]: c.ids
[4]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
[5]: %px
import socket
from mpi4py import MPI
[6]: %px
print("Hello World from rank %s of %s on nid %s" % (MPI.COMM_WORLD.rank,
MPI.COMM_WORLD.size,
socket.gethostname()))
[stdout:0] Hello World from rank 5 of 12 on nid nid04269
[stdout:1] Hello World from rank 7 of 12 on nid nid04269
[stdout:2] Hello World from rank 10 of 12 on nid nid04269
[stdout:3] Hello World from rank 6 of 12 on nid nid04269
[stdout:4] Hello World from rank 11 of 12 on nid nid04269
[stdout:5] Hello World from rank 2 of 12 on nid nid04269
[stdout:6] Hello World from rank 4 of 12 on nid nid04269
[stdout:7] Hello World from rank 3 of 12 on nid nid04269
[stdout:8] Hello World from rank 9 of 12 on nid nid04269
[stdout:9] Hello World from rank 8 of 12 on nid nid04269
[stdout:10] Hello World from rank 1 of 12 on nid nid04269
[stdout:11] Hello World from rank 0 of 12 on nid nid04269
[ ]: view = c[:]
view.activate() # enable magics
[ ]: # run the contents of the file on each engine:
view.run('psum.py')
[ ]: view.scatter('a',np.arange(16,dtype='float'))
```

NVDashboard and Dask Distributed JupyterLab Extensions

- NVDashboard is a JupyterLab extension for displaying dashboards of GPU usage (from the NVIDIA RAPIDS team)
- Dask JupyterLab extension provides a way to manage Dask clusters, as well as embedding Dask's dashboard plots directly into JupyterLab panes



GPU dashboards



Compute

Memory

Timeline:
- Compute
- Memory
- Total utilization
- PCIe throughput



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETHzürich

Let's crack on...
