



Training for Brilliant Minds

# Machine Learning on Puhti

Part 1: Getting started

June 3, 2020

Mats Sjöberg – [mats.sjoberg@csc.fi](mailto:mats.sjoberg@csc.fi)



# Overview

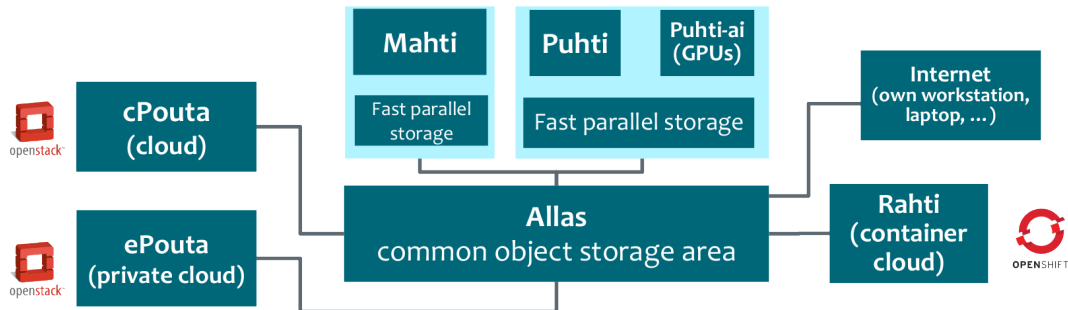
What CSC service to use?

Puhti supercomputer

Data storage

What CSC service to use?

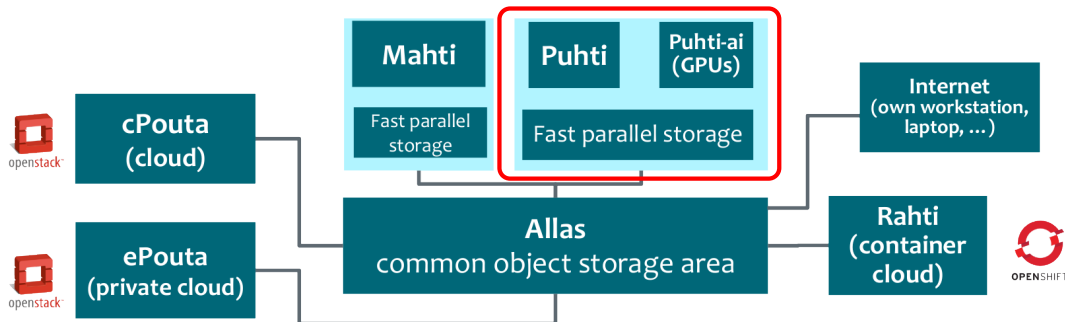
# What CSC service to use?



# What CSC service to use?

## Puhti

- Supercomputer cluster
- GPU-accelerated nodes
- Multi-user system



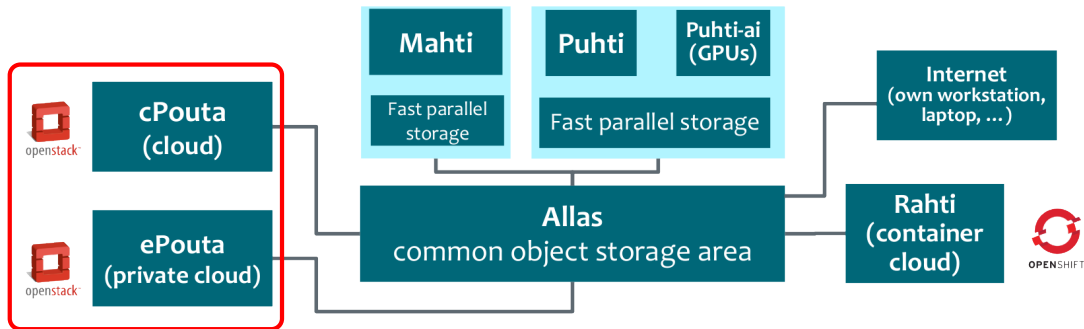
# What CSC service to use?

## Pouta

- Your “own” (virtual) server
- Less powerful than Puhti

## Puhti

- Supercomputer cluster
- GPU-accelerated nodes
- Multi-user system



# What CSC service to use?

## Pouta

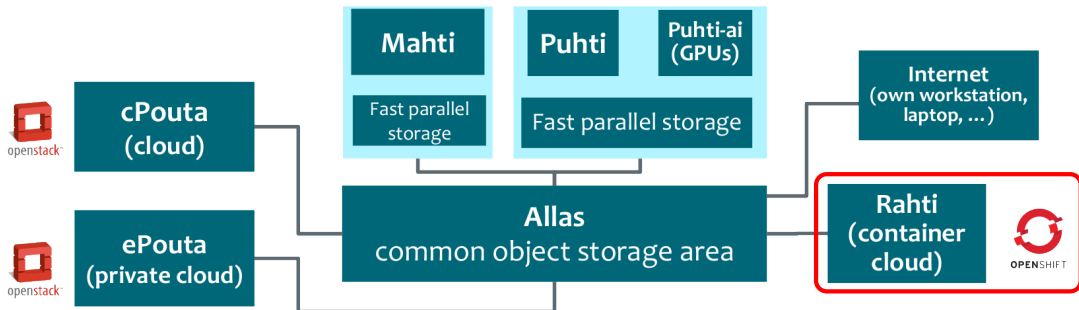
- Your “own” (virtual) server
- Less powerful than Puhti

## Puhti

- Supercomputer cluster
- GPU-accelerated nodes
- Multi-user system

## Rahti

- Container cloud
- No GPUs yet



# What CSC service to use?

## Pouta

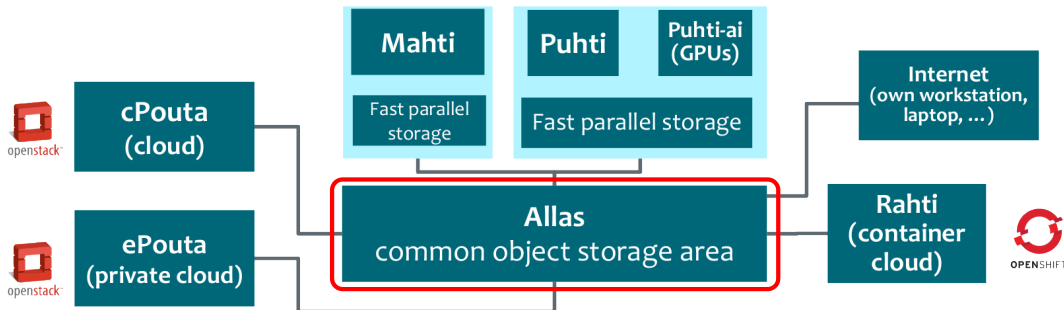
- Your “own” (virtual) server
- Less powerful than Puhti

## Puhti

- Supercomputer cluster
- GPU-accelerated nodes
- Multi-user system

## Rahti

- Container cloud
- No GPUs yet





# What CSC service to use?

## Pouta

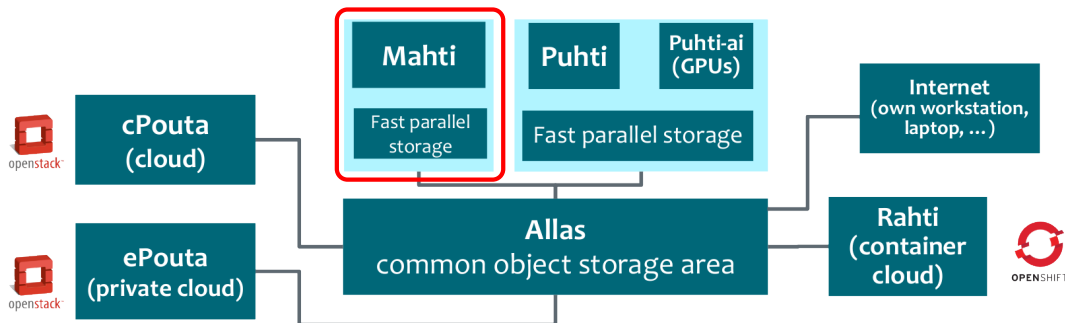
- Your “own” (virtual) server
- Less powerful than Puhti

## Puhti

- Supercomputer cluster
- GPU-accelerated nodes
- Multi-user system

## Rahti

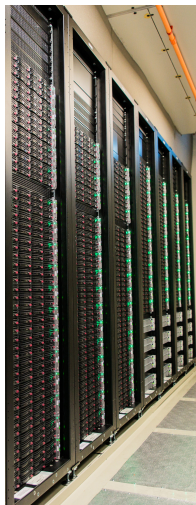
- Container cloud
- No GPUs yet



Getting access to Puhti

# Puhti supercomputer

- *Puhti-AI*, cluster with 80 nodes with 4 GPUs each  $\rightarrow$  320 GPUs in total
- Latest generation Nvidia V100 GPUs (Volta) with 32 GB of memory
- Fast network:  $2 \times 100$  Gbps links to each node
- Each node has a fast 3.2 TB local NVME disk



# Getting access to Puhti

<https://docs.csc.fi/computing/overview/>

To use Puhti you need to:

- Have a CSC account
- Be member of a CSC project, either by
  - creating a new project, or
  - joining an existing project (ask the PI to add you!)
- Finally, the project needs to have Puhti access

→ MyCSC portal: <https://my.csc.fi/>

# Accessing Puhti

- Using an ssh client such as OpenSSH or PuTTY
- Basic Linux skills are required!
- More info: <https://docs.csc.fi/computing/connecting/>

```
$ ssh <csc_username>@puhti.csc.fi
```

```
$ ssh <csc_username>@puhti-login2.csc.fi
```

Running jobs on Puhti

# Supported frameworks

We currently support:

- **Python Data** – collection of Python libraries for data analytics and machine learning
- **TensorFlow** – deep learning library for Python
- **PyTorch** – machine learning framework for Python
- **MXNet** – deep learning library for Python
- **RAPIDS** – suite of libraries for data analytics and machine learning on GPUs

<https://docs.csc.fi/apps/#data-analytics-and-machine-learning>

# Example: TensorFlow

- First check the application page for instructions:  
<https://docs.csc.fi/apps/tensorflow/>
- Load the default version:  
`module load tensorflow`
- or specific version:  
`module load tensorflow/2.0.0`
- **Note:** some modules are *Singularity-based*!



# What if some package is missing?

If you are using our module, but a trivial package is missing ...

- install it yourself, e.g.,  
`pip install --user <packagename>`
- ...or if it might be generally useful, send an email to [servicedesk@csc.fi](mailto:servicedesk@csc.fi) – we can install it for you!

# What if some package is missing?

If you need a specific setup, and our modules are not right for you ...

- use a virtualenv:

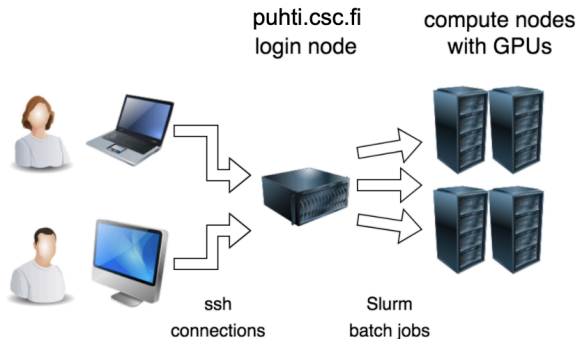
```
$ python3 -m venv myenv  
$ source myenv/bin/activate  
$ pip install ...
```

- use conda: <https://docs.csc.fi/support/tutorials/conda/>
- use singularity containers:  
<https://docs.csc.fi/computing/containers/run-existing/>
- or if generally useful, send an email to [servicedesk@csc.fi](mailto:servicedesk@csc.fi)

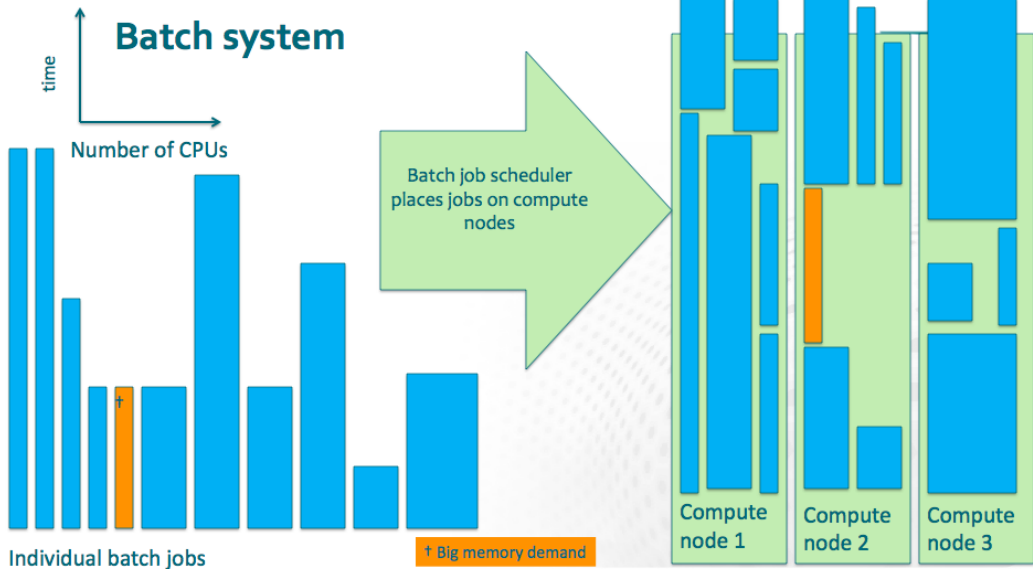
# Running a job on Puhti

Don't run heavy computing jobs in the login nodes!

- Puhti uses the *Slurm* batch job system
- Jobs do not run instantly but are put in a *queue*
- Resources (runtime, memory, number of cores) need to be specified



# Running a job on Puhti



# Running a job on Puhti

Create a job script, for example `run.sh`:

```
#!/bin/bash
#SBATCH --account=<project>
#SBATCH --partition=gpu
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=10
#SBATCH --mem=64G
#SBATCH --time=1:00:00
#SBATCH --gres=gpu:v100:1

module load tensorflow/2.0.0
srun python3 myprog.py <options>
```

<https://docs.csc.fi/computing/running/creating-job-scripts/>

# Running a job on Puhti

Example job script for Singularity-based modules:

```
#!/bin/bash
#SBATCH --account=<project>
#SBATCH --partition=gpu
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=10
#SBATCH --mem=64G
#SBATCH --time=1:00:00
#SBATCH --gres=gpu:v100:1
```

```
module load tensorflow/nvidia-20.03-tf2-py3
```

```
srun singularity_wrapper exec python3 myprog.py <options>
```

# Running a job on Puhti

Submit the job:

```
sbatch run.sh
```

Check the queue:

```
squeue -l -u $USER
```

Cancel a job:

```
scancel <jobid>
```

<https://docs.csc.fi/computing/running/submitting-jobs/>

Data storage



# Data storage on Puhti

- Disk space and *number of files* are limited on Puhti!  
→ We want to ensure that the shared (Lustre) filesystem works efficiently for everyone!
- Useful command: `csc-workspaces`

	Owner	Path	Capacity	Number of files	Cleaning
home	Personal	/users/<user-name>	10 GiB	100 000 files	No
projappl	Project	/projappl/<project>	50 GiB	100 000 files	No
scratch	Project	/scratch/<project>	1 TiB	1 000 000 files	Yes - 90 days

Data quotas can be increased via MyCSC!

<https://docs.csc.fi/computing/disk/>

# Using Allas

- store big datasets in Allas, CSC's object storage
- download them to project scratch prior to computation
- you can also upload trained models (or keep in projappl)

```
$ module load allas
$ allas-conf
$ cd /scratch/<your-project>
$ swift download <bucket-name> your-dataset.tar
```

# Large number of files

- Many datasets contain a large number of small files
- Shared filesystem (Lustre) performs poorly in this scenario  
→ noticable slowdowns for all Puhti users!

Consider alternatives:

- packaging your dataset into larger files
- use NVME fast local storage on GPU nodes

# Using more efficient data formats

Instead of many small files, use one or a few bigger files.

Examples:

- TensorFlow's TFRecord format
- HDF5
- LMDB
- ZIP, for example via Python's `zipfile` library

# Fast local NVME drive

- All GPU nodes have a local NVME drive
- Just add `nvme:<number-of-GB>` to sbatch `--gres` flag

```
#!/bin/bash
#SBATCH --account=<project>
#SBATCH --partition=gpu
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=10
#SBATCH --mem=64G
#SBATCH --time=1:00:00
#SBATCH --gres=gpu:v100:1,nvme:100
```

```
tar xf /scratch/<your-project>/your-dataset.tar -C $LOCAL_SCRATCH
```

```
srun python3 myprog.py --data_dir=$LOCAL_SCRATCH <options>
```

Thank you!

# Don't forget part 2 of this webinar!

## Machine Learning on Puhti

### Part 2: Scaling up and using resources efficiently

- GPU utilization
- Multi-GPU and multi-node jobs
- Singularity containers

**Time:** Wednesday, June 10, 2020 at 14:00-15:00

**Place:** <https://ssl.eventilla.com/event/jJRkz>