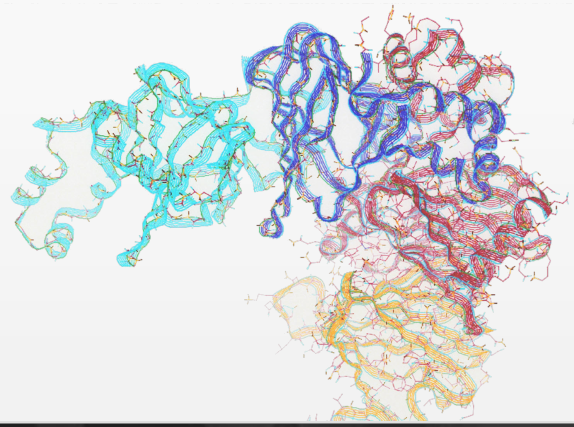
Basics of High-Performance Computing - Dr. S. Bacchio

In today's training

Basics of High-Performance Computing

- Performance analysis
- Computing paradigms
- Parallel programming
- Tips and hands on!

GitHub



The Cyprus Institute
RESEARCH·TECHNOLOGY·INNOVATION

Before starting…

- Examples and slides are available on Github
  - https://github.com/CaSToRC-CyI/NCC-Beginner-Training-2021
  - What is Git and Github? (next slide)

- In the examples we will use Python
  - Why Python? (next next slide)
  - On Wednesday examples with C
  - At the intermediate training we will talk about “Python for HPC”

- Examples may not work on Windows (consider installing Ubuntu on Windows 10)
  - Tomorrow you will get access to our local cluster

- **Please interrupt me for questions at any moment!**

Before starting… What is Git and Github?

Git is a version control system that allows you to:

- **keep track of the changes** in your project,
- work in group or in a community
- develop in parallel an application
- perform a review process on the changes

GitHub is the most popular online platform for version control and

- it is based on Git
- most of its services are for free
- you can create organizations and manage projects
- Github Actions for continuous integration

Hands on!

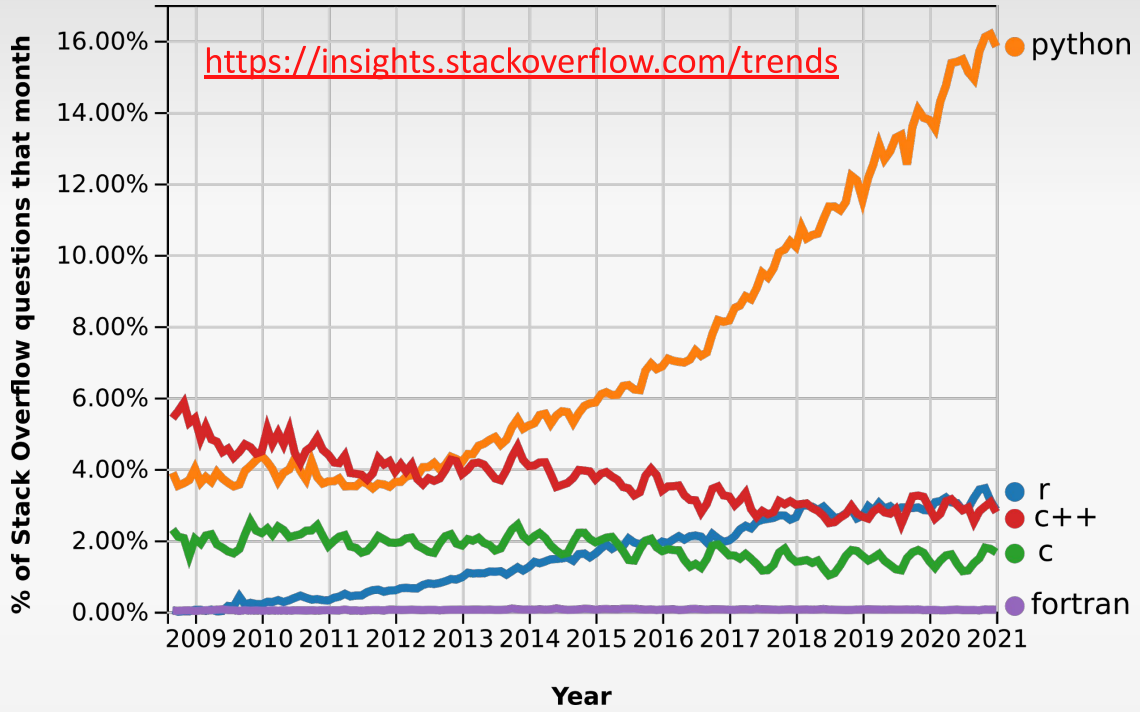
```
> # Install Git:
https://github.com/git-guides/install-git

> git clone
https://github.com/CaSToRC-CyI/NCC-Beginner-Training-2021
> git status
> git log

> # Edit a file

> git status
> git commit -am “Describe the change”
> git push
```

Please consider to put your open source software online on a version control platform (e.g. Github)

Further reading: https://dev.to/ravirajthedeveloper/what-is-git-and-github-and-how-to-use-github-2mb1

Before starting… Why Python?

- **Interpreted** and object oriented programming language
- Science- and data-oriented
- Easy to Learn and Use
- Huge community
- Hundreds of Python Libraries and Frameworks
- First choice for Big Data and **Machine learning**
- User-friendly and great **APIs**
- Easy deployment of software (PyPI)
- Build with a scientific approach (PEPs)
- **Performance issues?** They can be overcome



The Cyprus Institute
RESEARCH·TECHNOLOGY·INNOVATION

Performance analysis

The main focus of High-Performance Computing is…

PERFORMANCE!

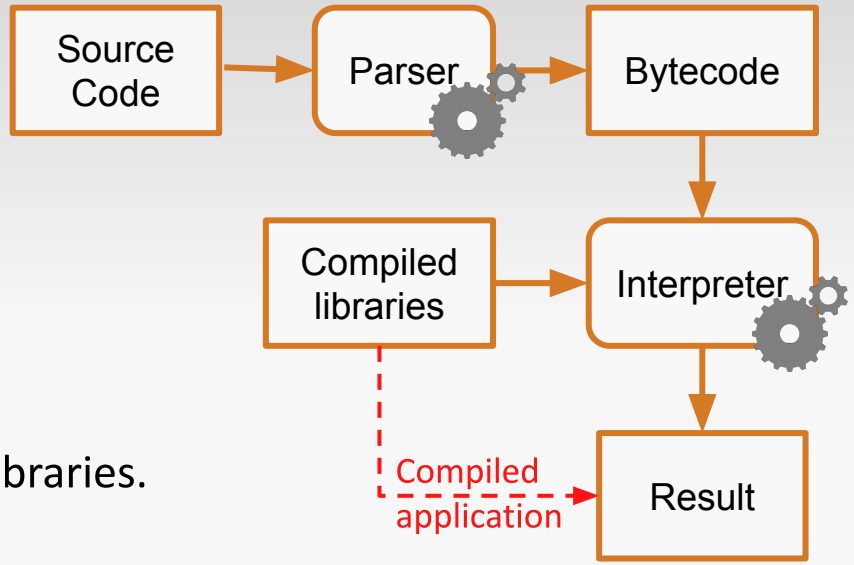
- Computational Performance is measured with **FLOPS:** Floating-point operations per seconds
  - +, -, * count as 1 flop
  - /, `sqrt`, `sin`, etc… count as 2 or more flops depending on the architecture
  - **How?** FLOPS = “Theoretical number of operations” / “measured time”
    - ➢ e.g. `x*y`: 1 flop if real, 6 flops if complex!

- Memory Performance is measured with **Bandwidth:** Bytes read+written per seconds

Performance in Python

Python is a very powerful and flexible programming language, but…

- interpreted = bad (computational) performance
- it is important to know the strengths and the weaknesses!
- By its own it is not mean for High-Performance computing.



Built-in functions and HPC modules are based on **compiled** and **optimized** libraries.

Use as much as possible:

- built-in functions
- numerical modules (Numpy, Scipy, Pandas, …)
- compile your kernels (Cython, Pythran, Numba, …)

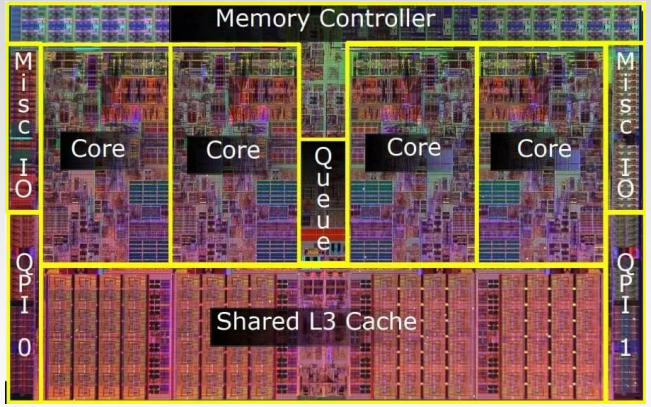
NEVER do for-loops on data!

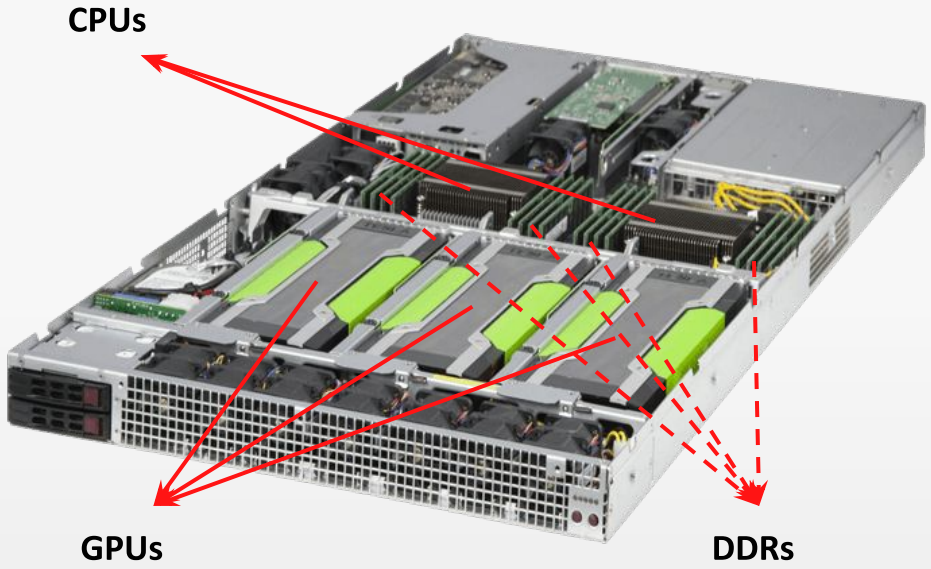
Hands on!

```
> Notebook-01: performance
```

Computing paradigms

- **Multithreading:**
  - shared data
  - distributed operations
  - physical / virtual cores
- **(GPU-)Accelerated computing:**
  - host / device memory
  - host / device tasks
  - dedicated compilation
- **Distributed computing:**
  - distributed data
  - distributed tasks
  - communication protocol





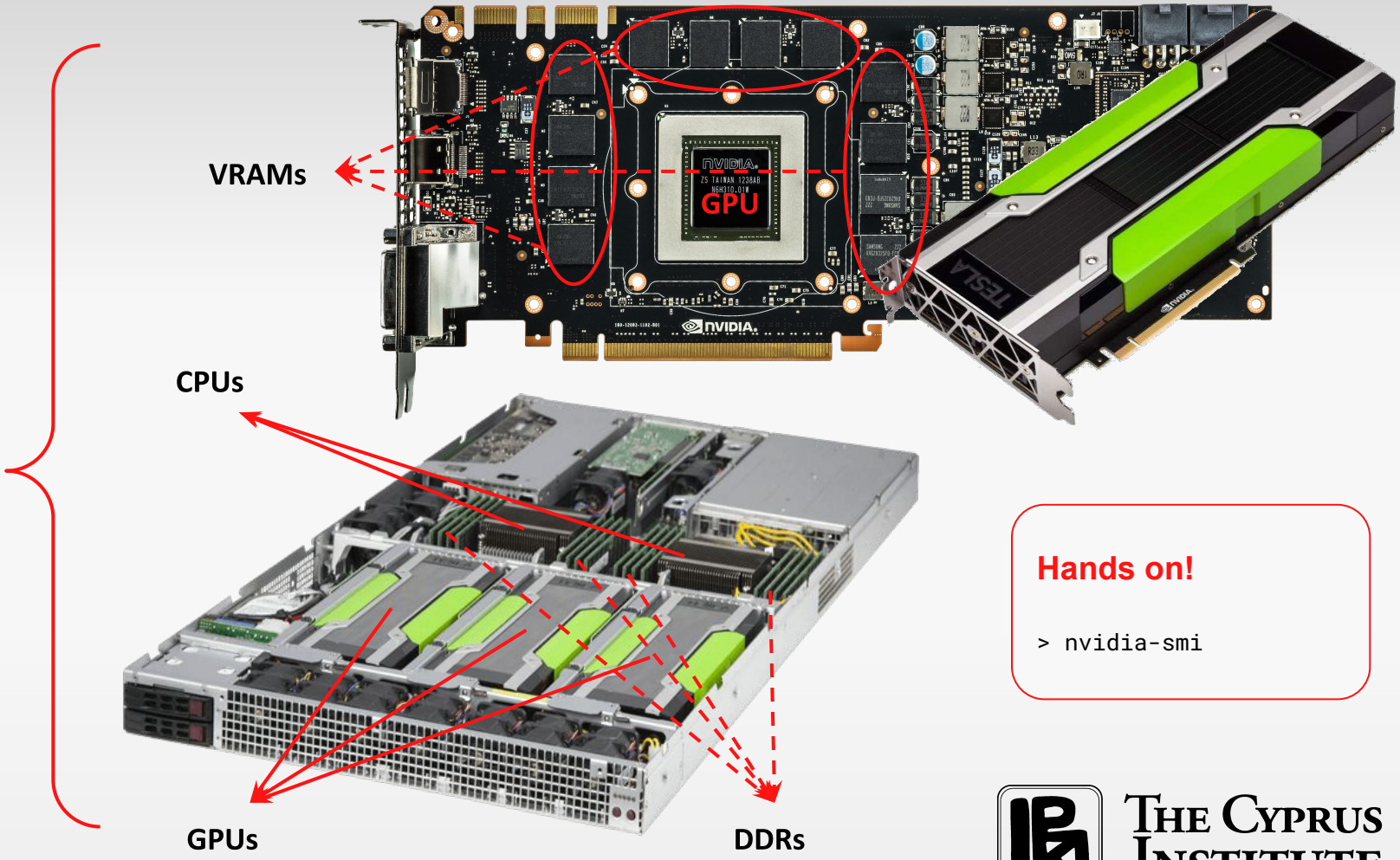
Hands on!

```
> less /proc/cpuinfo
> less /proc/meminfo
```

The Cyprus Institute
RESEARCH·TECHNOLOGY·INNOVATION

Computing paradigms

- **Multithreading:**
  - shared data
  - distributed operations
  - physical / virtual cores
- **(GPU-)Accelerated computing:**
  - host / device memory
  - host / device tasks
  - dedicated compilation
- **Distributed computing:**
  - distributed data
  - distributed tasks
  - communication protocol

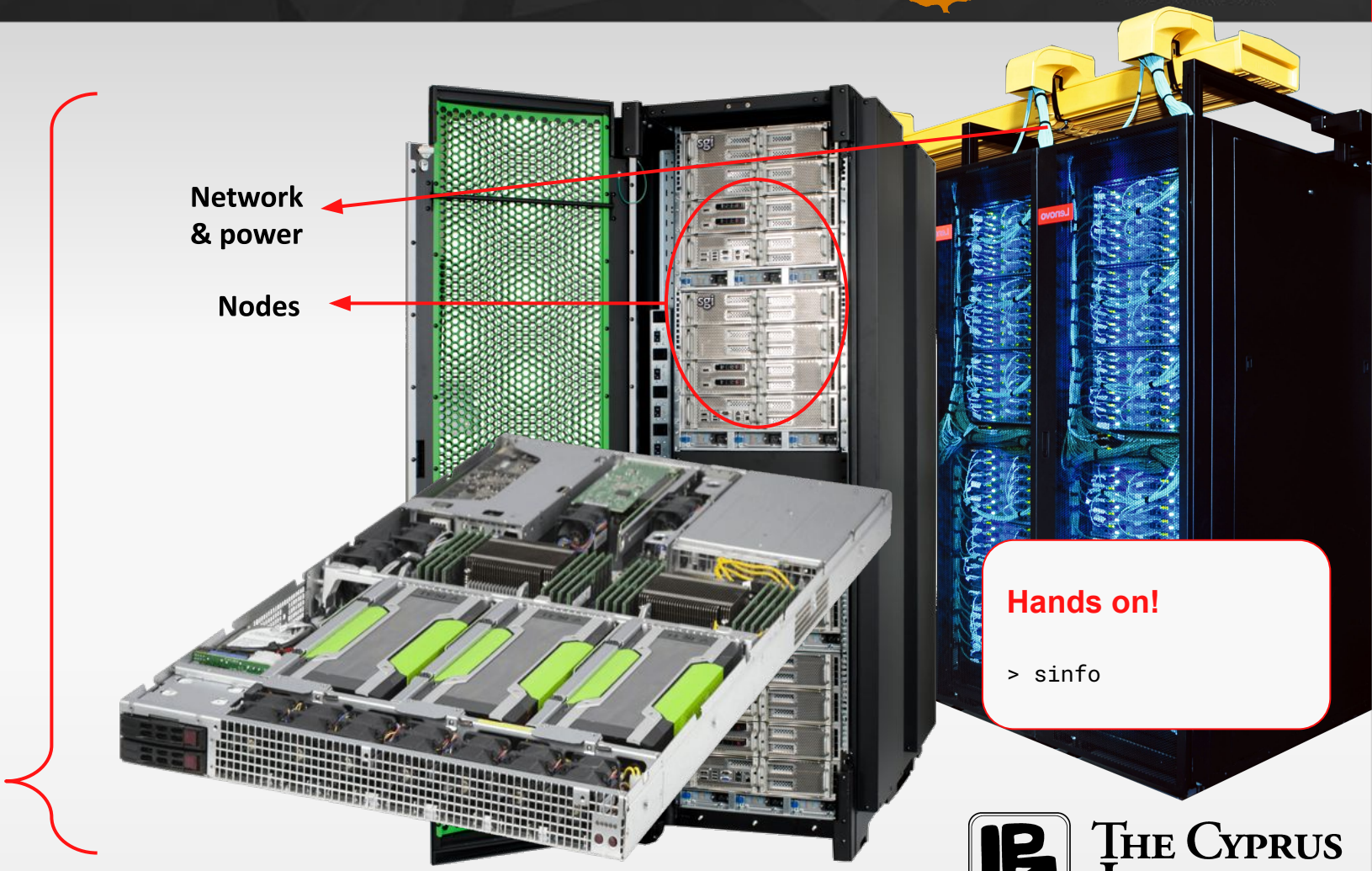


Hands on!

```
> nvidia-smi
```

Computing paradigms

- **Multithreading:**
  - shared data
  - distributed operations
  - physical / virtual cores
- **(GPU-)Accelerated computing:**
  - host / device memory
  - host / device tasks
  - dedicated compilation
- **Distributed computing:**
  - distributed data
  - distributed tasks
  - communication protocol



The Cyprus Institute
RESEARCH·TECHNOLOGY·INNOVATION

Multithreading

How?

- **OpenMP:**
  - Popular high-level implementation
  - Available for C/C++ and Fortran
  - Topic of Wednesday class
- **Low-level implementations:**
  - POSIX threads: pthread (C, fortran)
  - std::threads (C++)
  - threading (Python)

Main Feature:

- Shared data!
- Multiprocessing is not multithreading because data are not shared

Example OpenMP

```
#omp parallel for
for(int i=0; i<100; i++) {
      ...
}
```

Hands on!

```
> Notebook: multithreading
```

Example POSIX thread

```
#include <pthread.h>

pthread_t threads[NUM_THREADS];

int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
                   void *(*start_routine) (void *), void *arg);

int pthread_join(pthread_t thread, void **retval);
```

Parallel computing with MPI

MPI - Message Passing Interface

- Commonly used for parallel applications with C/C++, Fortran, Python, MATLAB, Julia, R, …
- Standard for Message Passing developed by the MPI Forum
- Various implementations: OpenMPI, MPICH, IMPI, …
- `mpi4py`: Python interface

Main features:

- Communication protocols: send/recv, reductions, scatter, …
- Parallel I/O
- Non-blocking operations
- Dynamic process management

Hands on!

```
> Notebook-02: Server and Client
```

Thank you!

Thank you for you attention!

The Cyprus Institute
RESEARCH·TECHNOLOGY·INNOVATION