United Nations
Educational, Scientific and
Cultural Organization

ICTP - East African Institute
for Fundamental Research
under the auspices of UNESCO

An ICTP Partner Institute

UNIVERSITY of RWANDA

Republic of Rwanda
Ministry of Education

Introduction to HPC for Scientists & Engineers

Edward Ntim Gasu, Ph.D.

(ICTP-East African Institute for Fundamental Research, Kigali, Rwanda)

Elliot Sarpong Menkah, Ph.D.

(Kwame Nkrumah University of Science and Tech. Kumasi, Ghana)

High Performance Computing (HPC)?

- No real definition, depends on the prospective:
  - HPC is when I care **how fast** I get an answer
  - HPC is when I foresee my problem to get **bigger and bigger**
  - **Scientific computing is NOT an IT Service?**

- Thus **HPC can happen on**:
  - A supercomputer
  - A Linux Cluster
  - A grid or a cloud
  - A workstation, desktop, laptop, smartphone!
  - **Cyberinfrastructure** = any combination of the above
- **HPC** means also High-Productivity Computing

Scientific Computing @ EAIFR

- TWAS-1 and QUEVEDO Clusters

- TWAS-1: 1 Master, with 3 Compute Nodes (80 CPUs, 92 GB RAM Each)
- Total of 320 CPUs

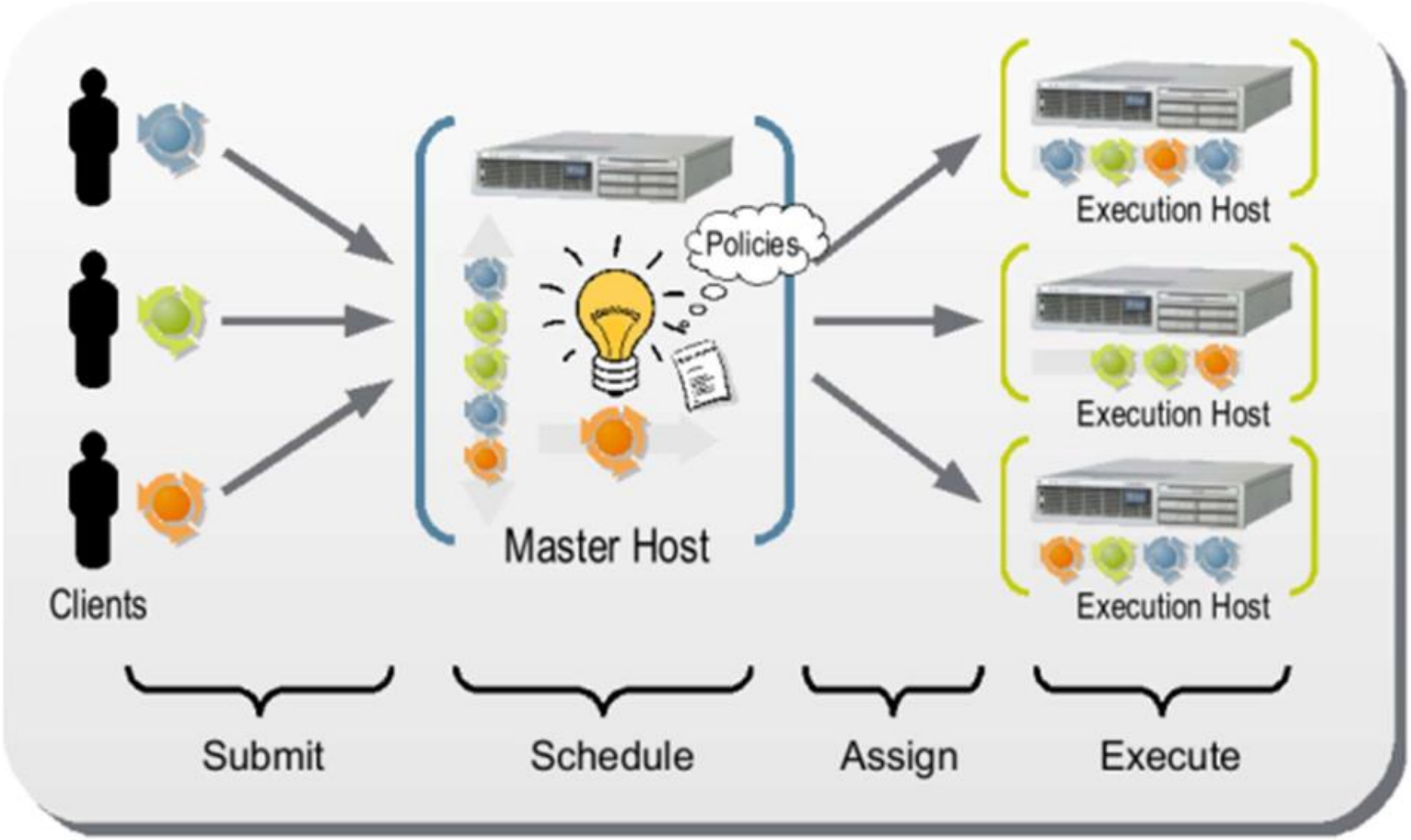
- QUEVEDO: 1 Master and 8 Compute
- Total of 316 CPUs

- For small scale Production and Education

- Scientific communities contribute to the development work

- Close collaboration with the Computer Science and IoT Departments

Scientific Computing @ EAIFR



“Shut-up and compute ! …”

- Workloads, such as running calculations, are usually "on demand".
- When a user needs something, s/he tells the server, and the server does it.

- **When it's done, it's done!**

- For the most part it doesn't matter on which particular machine the calculations are run.
- All that matters is that the **user can get the results**.
- This kind of work is often called **batch**, offline, or interactive **work**.
- Sometimes batch work is called a **job**. Typical jobs include processing of accounting files, **rendering images or movies, running simulations, processing input data, modeling chemical or mechanical interactions, and data mining**.
- Many organizations have hundreds, thousands, or even tens of thousands of machines devoted to running jobs.

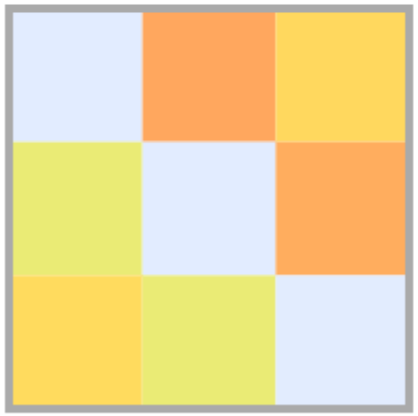
Demands on QUEVEDO & TWAS-1

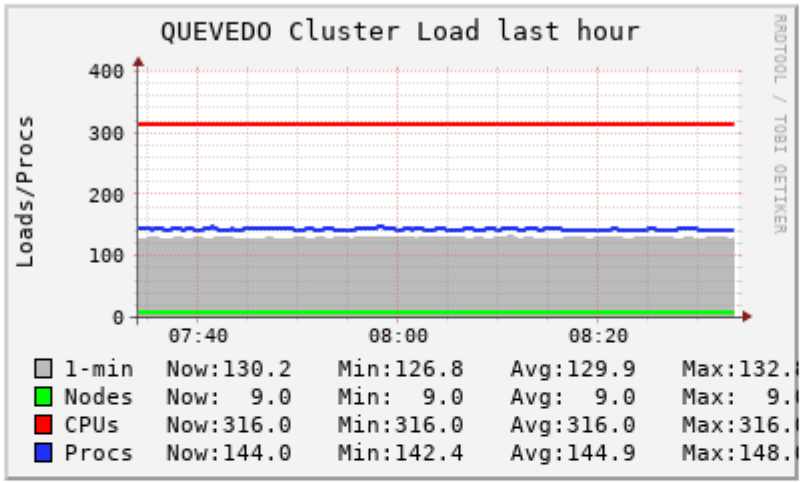
Overview of QUEVEDO @ 2025-10-17 06:33

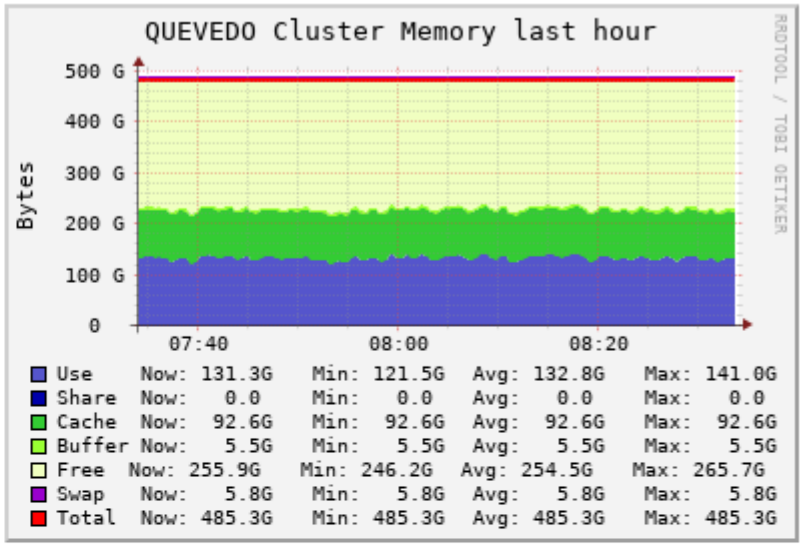
CPUs Total: **316**
Hosts up: **9**
Hosts down: **0**

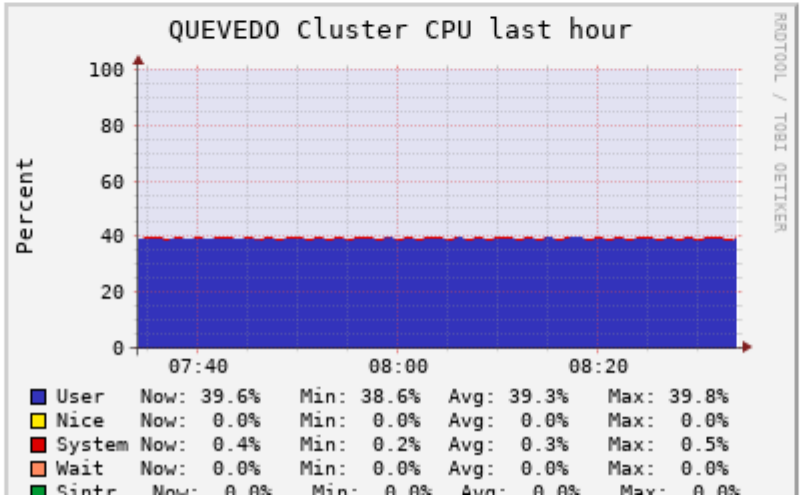
Current Load Avg (15, 5, 1m):
41%, 41%, 41%
Avg Utilization (last hour):
41%

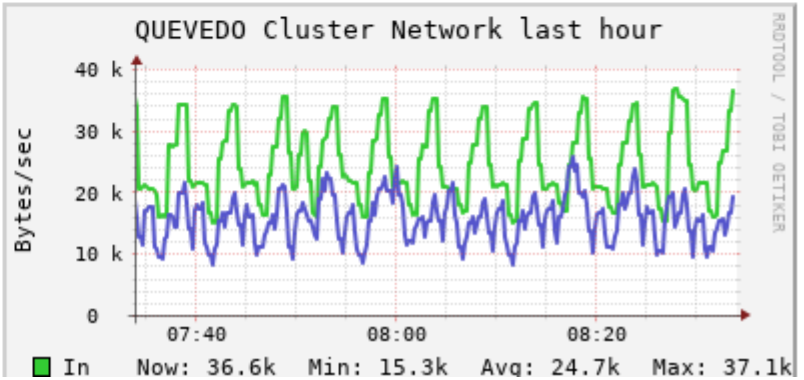
Server Load Distribution









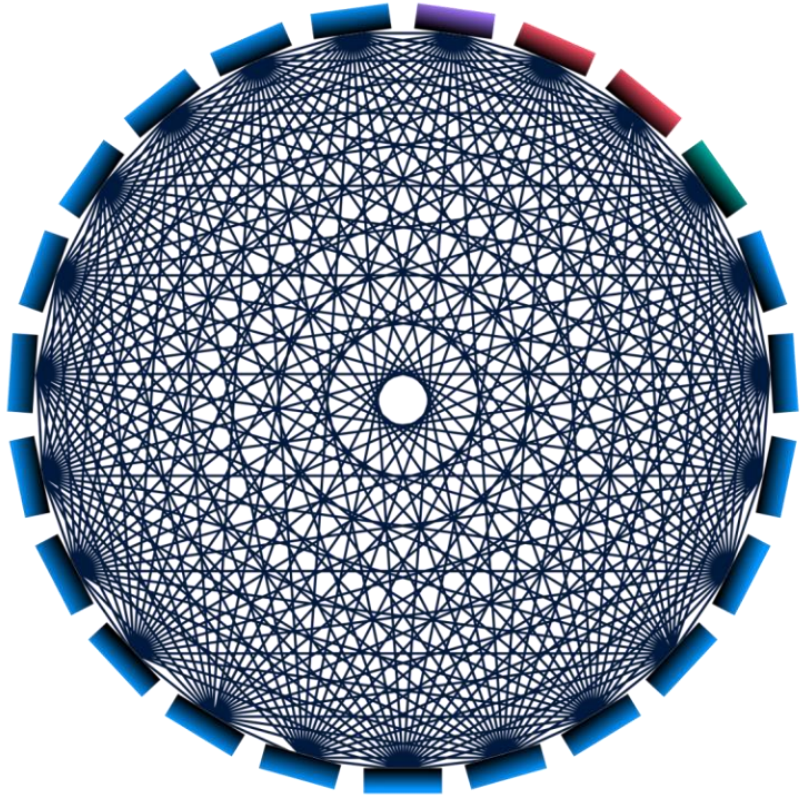


Welcome to:

```
*******************************************************************************
* Red Hat Enterprise Linux 8.7 (Ootpa)                                        *
*                                                                             *
* Booster module:                                                             *
* Atos Bull Sequana X2135 "Da Vinci" Blade                                    *
* 3456 compute nodes with:                                                    *
*       - 32 cores Ice Lake at 2.60 GHz                                       *
*       - 4 x NVIDIA Ampere A100 GPUs, 64GB                                   *
*       - 512 GB RAM                                                          *
*                                                                             *
* DataCentric General Purpose module (DCGP):                                  *
* Atos BullSequana X2140 Blade                                                *
* 1536 compute nodes with:                                                    *
*       - 2x56 cores Intel Sapphire Rapids at 2.00 GHz                        *
*       - 512 GB RAM                                                          *
*                                                                             *
* Internal Network: 200G HDR Infiniband Dragonfly+                            *
* Workload Manager: SLURM 23.11                                               *
*                                                                             *
```

Leonardo HPC Supercomputer

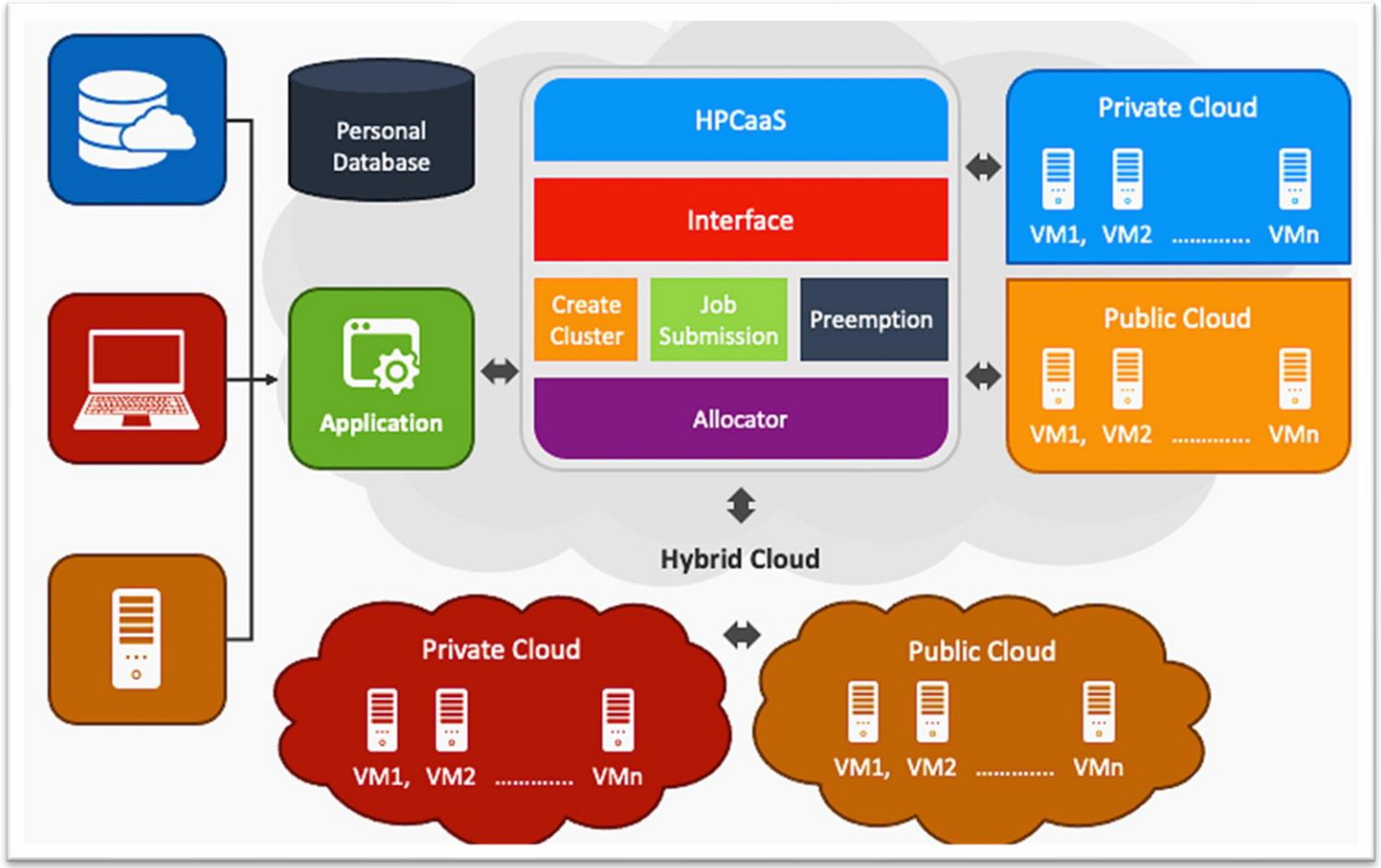
- **Leonardo** features a state-of-the-art interconnect system tailored for high-performance computing (HPC). It delivers ***low latency*** **and** ***high bandwidth*** by leveraging **NVIDIA Mellanox InfiniBand HDR** (High Data Rate) technology, powered by NVIDIA QUANTUM QM8700 Smart Switches, and a Dragonfly+ topology.
- **System Composition:**
  - **19** cells dedicated to the *Booster* **partition.**
  - **2** cells for the *DCGP* (Data-Centric General Purpose) **partition.**
  - **1** hybrid cell with both accelerated (**36 Booster nodes**) and conventional (**288 DCGP nodes**) compute resources.
  - **1** cell allocated for **management, storage, and login services.**
- **Adaptive Routing:** The network employs adaptive routing, dynamically optimizing data paths to alleviate congestion and maintain performance under load.



#10 @ top500.org

YouTube

Modern Cyberinfrastructure & Application



Modern HPC Architecture

Where to use HPC

Accessing Remote HPC Resources.

cecam
Centre Européen de Calcul Atomique et Moléculaire

MENU

COMP AF
computing africa

Access to computational resources for African-based researchers

With support from:

ICTP The Abdus Salam International Centre for Theoretical Physics

IAEA

unesco

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

CINECA

Accessing Remote HPC Resources

QUEVEDO Cluster (SGE Scheduler)

Docs: Secure shell (SSH) and job submission basics

Account Registration & Authentication

→ Request user account via institutional email

→ Admin adds you to project group **grp$N {$N = 01..08}**

→ Authentication: Password & SSH (RSA/ECDSA keys supported

Login Procedure **Password: passcode**

Use SSH to connect to QUEVEDO:

```
$ ssh username@quevedo.eaifr.org
```

Storage available in: $HOME

Tip: Always use secure network connection

```
ed@DESKTOP-DV2U4QD:~$ ssh engasu@quevedo.eaifr.org
Password:
You are required to change your password immediately (root enforced)
Changing password for engasu.
(current) UNIX password:
New password:
BAD PASSWORD: The password is too similar to the old one
New password:
Retype new password:
Rocks 7.0 (Manzanita)
Profile built 10:27 10-Sep-2018

Kickstarted 12:07 10-Sep-2018
 ##################################################################
 #***ANNOUNCEMENT: Monthly Cluster Maintenance Begins!!           #
 #                THIS WILL INCLUDE CLEANING OLD DATA.            #
 #                Kindly back-up all data                         #
 #                Beware: Dormant accounts will be removed.       #
 ##################################################################
```

Submitting Jobs (SGE)

Prepare a job script (job.sh):

```
#!/bin/bash
#$ -N my_job
#$ -l h_rt=24:00:00
#$ -l hostname=compute-0-1|compute-0-5
#$ -pe mpi 64
#$ -j y
#$ -o out_err.txt
#$ -cwd
mpirun -np 64 ./my_program
```

Submit: qsub job.sh

Check queue: qstat -f or qstat -u $USER or qstat -u <username>

Cancel job: qdel <job_id>

File Transfer

Upload files to QUEVEDO (Local → Cluster) using scp (secure copy over SSH):

```
$ scp   local-file   username@quevedo.eaifr.org:/home/user/dest
```

```
$ scp  -r  local-dir  username@quevedo.eaifr.org:/home/user/dest
```

Download files from QUEVEDO (Cluster → Local):

```
$ scp  username@quevedo.eaifr.org:/home/user/remote-file  ./
```

```
$ scp -r username@quevedo.eaifr.org:/home/user/remote-dir ./
```

Synchronize efficiently using rsync -avz (archive, verbose, compress):

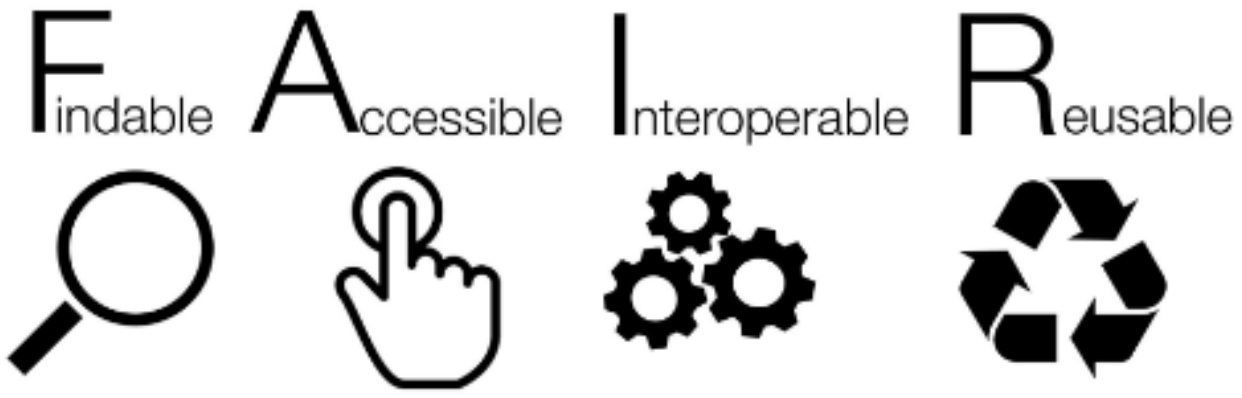
```
$ rsync -avz local-dir username@quevedo.eaifr.org:/home/user
```

```
$ rsync -avz username@quevedo.eaifr.org:/home/user/remote-dir ./
```

Tip: Use VPN if connecting from offsite for secure transfer

Data Management

- Good data management ensures reproducibility and long-term usability



Core FAIR Principles:
→ **Findable**: Data and metadata are easy to locate,
→ **Accessible**: Data can be retrieved using standardized protocols,
→ **Interoperable**: Data can be integrated with other datasets,
→ **Reusable**: Data is well-described and usable for future studies.

- Use **consistent file naming**, **metadata standards**, and **secure storage** !
- **Documentation ! Documentation ! Documentation ! Documentation !**

Running Computations

Batch vs Interactive Execution

Batch mode: run jobs non-interactively in the queue

Interactive mode: obtain a live session on a compute node

Example Interactive Job Requests:

SLURM:

```
$ salloc --ntasks=4 --time=00:30:00
```

PBS:

```
 $ qsub -I -l select=1:ncpus=4:mem=8gb -l walltime=00:30:00
```

SGE:

```
 $ qlogin -pe mpi 4 -l h_vmem=2G -l h_rt=00:30:00
```

Interactive GPU session on PBS:

```
$ qsub -I -q gpu_1 -P PRJT1234 -l select=1:ncpus=9:ngpus=1
```

Tip: Interactive mode is useful for testing code before full batch runs

Serial Job (SLURM)

```
#SBATCH --job-name=serial_job          (job name)
#SBATCH --time=00:30:00                (max runtime hh:mm:ss)
#SBATCH --nodes=1                       (request 1 node)
#SBATCH --ntasks=1                      (single task/process)
#SBATCH --cpus-per-task=1               (1 CPU core per task)
#SBATCH --partition=<partition_name>   (queue/partition)
#SBATCH --mem=2G                        (memory per node)
#SBATCH --output=serialJob.out          (output file)
#SBATCH --account=<project_account>     (project/project account)
```

MPI Job (SLURM)

```
#SBATCH --job-name=jobMPI
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=1
#SBATCH --mem=<mem_per_node>
#SBATCH --partition=<partition_name>
#SBATCH --account=<project_account>
```

```
module load intel intelmpi              (load MPI libraries)

srun myprogram < myinput > myoutput     (run MPI program across nodes)
```

OpenMP Job (SLURM)

```
#SBATCH --job-name=openmp_job
#SBATCH --time=01:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=48          (number of OpenMP threads)
#SBATCH --mem=<mem_per_node>
#SBATCH --partition=<partition_name>
#SBATCH --account=<project_account>

module load intel                        (load Intel compiler/libraries)
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK  (set OpenMP threads)
srun ./myprogram < myinput > myoutput         (run program)
```

Multi-GPU Job (SLURM)

```
#SBATCH --job-name=multi_gpu_job
#SBATCH --time=04:00:00
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=4    (MPI tasks per node, 1 per GPU)
#SBATCH --cpus-per-task=10
#SBATCH --gres=gpu:4           (GPUs per node)
#SBATCH --partition=<gpu_partition>
#SBATCH --qos=<qos_priority>
#SBATCH --account=<project_account>

module load cuda/12.2 openmpi ...   (load CUDA, MPI, dependencies)
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
export NCCL_DEBUG=INFO              (NCCL debug for multi-GPU)
srun --mpi=pmix ./my_distributed_gpu_app --config config.yaml
```

Running GPU Jobs on PBS

Example PBS batch script (gpu_job.pbs):

```
#!/bin/bash
#PBS -N nameyourjob                    (job name)
#PBS -q gpu_1                          (queue with GPUs)
#PBS -l select=1:ncpus=4:ngpus=1       (resources: 1 node, 4 CPUs, 1 GPU)
#PBS -P PRJT1234                        (project ID)
#PBS -l walltime=4:00:00                (max runtime)
#PBS -m abe                             (send email on abort, begin, end)
#PBS -M your.email@address              (notification email)

cd /mnt/lustre/users/USERNAME/cuda_test   (change to working directory)
echo `date`: executing CUDA job on host ${HOSTNAME}
echo Available GPU devices: $CUDA_VISIBLE_DEVICES

./hello\_cuda                                (run CUDA program)
```

Quantum ESPRESSO Job (SGE)

```
#$ -N qe_scf                    (job name)
#$ -cwd                         (run in current working directory)
#$ -pe mpi 4                    (parallel environment with 4 cores)
#$ -l h_rt=24:00:00            (max runtime hh:mm:ss)
#$ -l hostname=compute-0-5  (submit to specific node)
#$ -j y                         (merge stdout and stderr)
#$ -o qe_job.txt               (output file)
module load openmpi/openmpi_3.1.3_intel_2018   (load MPI module)
module load espresso/6.4.1                     (load Quantum ESPRESSO module)
#mpirun -mca btl_openib_allow_ib 1 -np 4 pw.x < scf.in > scf.out
```

Benchmarking in HPC

- Computer performance:
    - CPU operations {sometimes depends on the programming}
    - Network
- Time taken to run code vs. varying workload = time to solution
    - Algorithmic optimization

- Floating point precision

- Memory efficiency
- Energy efficiency

Numerical Precision

Machine epsilon defines the smallest detectable change in floating-point arithmetic.

Machine Epsilon (ε):

```
int main() {
    double h = 1.0;
    while ((1.0 + h / 2.0) > 1.0)
        h /= 2.0;
    printf("Manually Estimated Epsilon: %.10e\n", h);
    printf("Standard Library Epsilon: %.10e\n", DBL_EPSILON);
    return 0;
}
```

Output:

```
Manually Estimated Epsilon: 2.2204460493e-16
Standard Library Epsilon: 2.2204460493e-16
```

Floating-Point Overflow

Overflow demonstrates the upper numerical limits in floating-point computation.

```
    /* Code to compute an overflow */
    double dm = 1.0, dm_max = 0.0, m_overflow = 0.0, overflow = 0.0;
    double hd_m = 0.0;
    while (dm < 1.0e310) {
        if (1.0 / dm != hd_m) {
            dm_max = dm;
        } else {
            printf(" Maximum finite positive double is: %g \n", dm);
        }
        dm *= 2.0;
    }
```

Output:

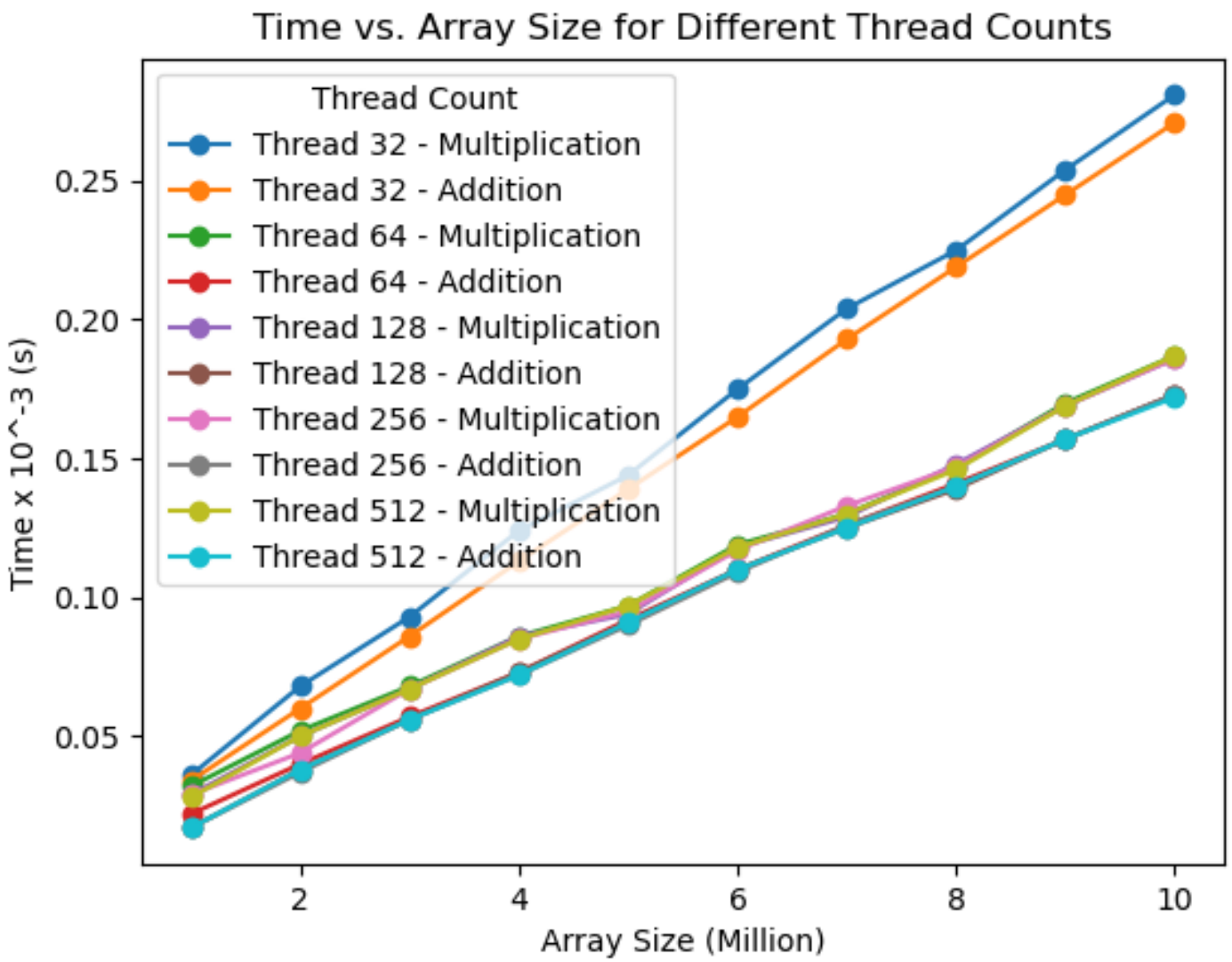
Maximum Finite Positive Double is: 8.98847e+307 | Standard Library: 1.79769e+308

Manually Estimated Overflow: inf | Standard Library Overflow: inf

Key Insight:

It’s crucial in HPC benchmarking to verify numerical stability before scaling workloads.

Benchmarking for Performance Evaluation



Thank You!