EURO Greece

How to access the Greek HPC Infrastructure ARIS

Nikos Triantafyllis

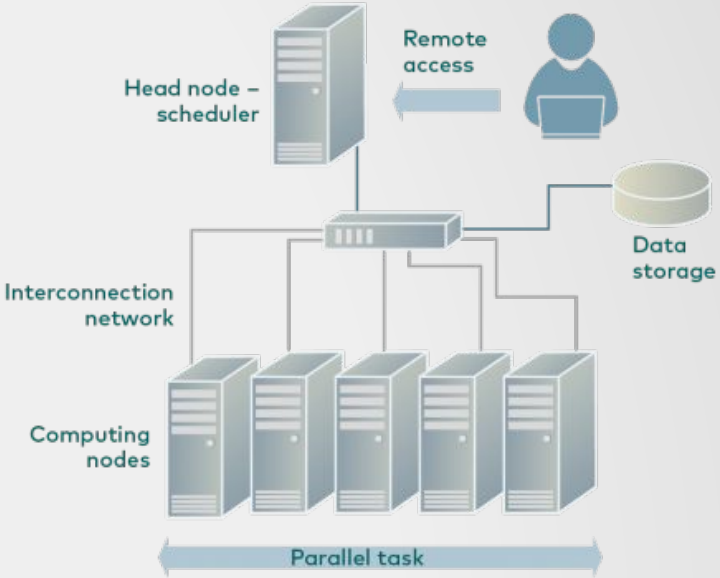
ntriantafyl@admin.grnet.gr

What is HPC?

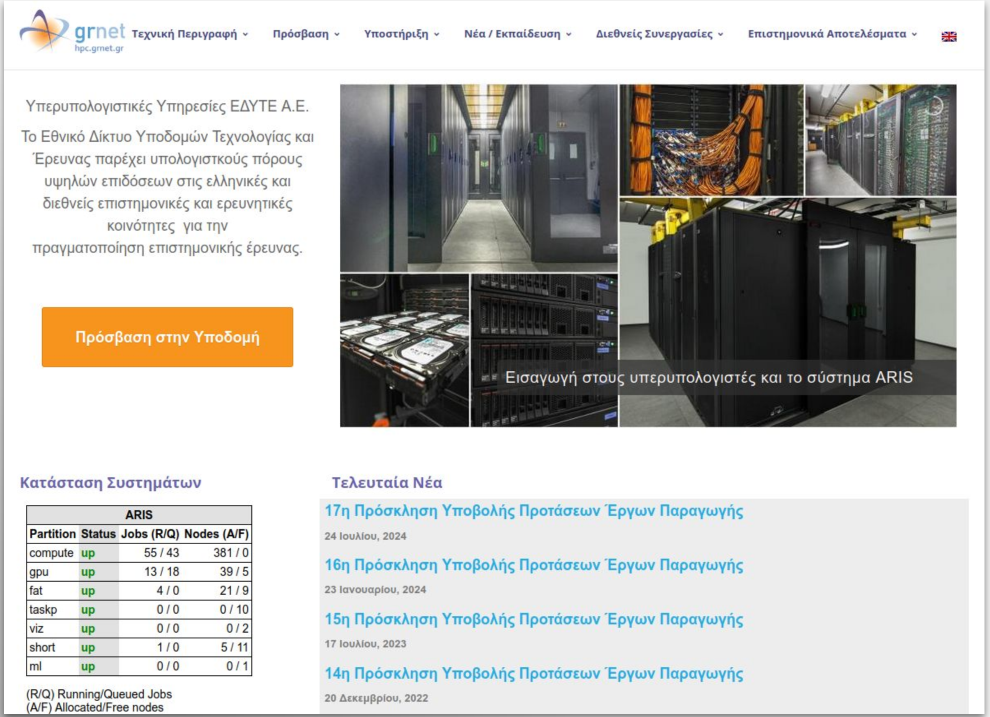
- High-Performance Computing (HPC) is the ability to perform sophisticated calculations at high speeds.

- An HPC cluster consists of hundreds or thousands of compute servers, so-called nodes. The nodes in each cluster work in parallel with each other.

- HPC solves large problems in science, engineering, or business, that are too complex for a PC. On typical PC it might take e.g. hours, days, weeks to perform the computations, but if you use an HPC Cluster, it might only take minutes, hours, days, respectively.



GRNET HPC – ARIS



Υπερυπολογιστικές Υπηρεσίες ΕΔΥΤΕ Α.Ε.

Το Εθνικό Δίκτυο Υποδομών Τεχνολογίας και Έρευνας παρέχει υπολογιστικούς πόρους υψηλών επιδόσεων στις ελληνικές και διεθνείς επιστημονικές και ερευνητικές κοινότητες για την πραγματοποίηση επιστημονικής έρευνας.

Εισαγωγή στους υπερυπολογιστές και το σύστημα ARIS

Κατάσταση Συστημάτων

ARIS			
Partition	Status	Jobs (R/Q)	Nodes (A/F)
compute	up	55 / 43	381 / 0
gpu	up	13 / 18	39 / 5
fat	up	4 / 0	21 / 9
taskp	up	0 / 0	0 / 10
viz	up	0 / 0	0 / 2
short	up	1 / 0	5 / 11
ml	up	0 / 0	0 / 1

(R/Q) Running/Queued Jobs
(A/F) Allocated/Free nodes

Τελευταία Νέα

17η Πρόσκληση Υποβολής Προτάσεων Έργων Παραγωγής

24 Ιουλίου, 2024

16η Πρόσκληση Υποβολής Προτάσεων Έργων Παραγωγής

23 Ιανουαρίου, 2024

15η Πρόσκληση Υποβολής Προτάσεων Έργων Παραγωγής

17 Ιουλίου, 2023

14η Πρόσκληση Υποβολής Προτάσεων Έργων Παραγωγής

20 Δεκεμβρίου, 2022

https://www.hpc.grnet.gr

GRNET HPC – ARIS: Access

Who Can Access the System?

- Scientists and researchers affiliated with Greek academic and research institutions
- The system is free to use

How to Gain Access?

- Researchers submit project proposals to gain access
- Proposals can be submitted as:
  - **Preparatory/Development** projects, through an ongoing open call
  - **Production** projects during periodic calls

GRNET HPC – ARIS: Access

EURO Greece

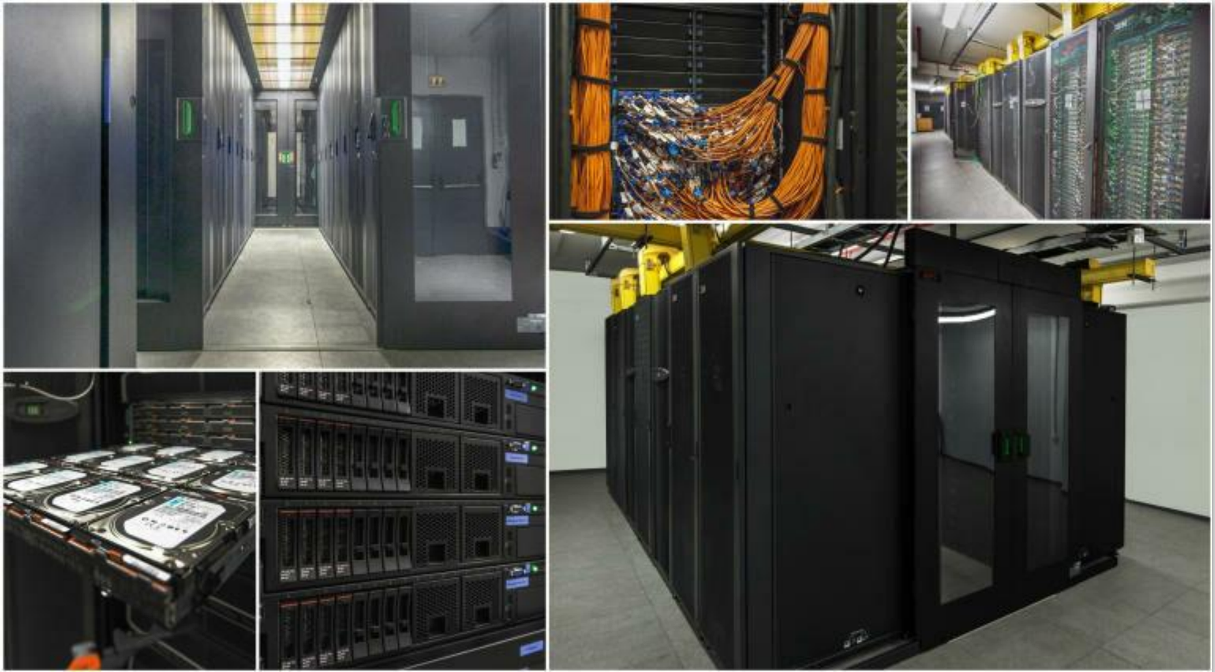
What is required?

- A clear description of the intended application
- Justification for the need for an HPC system
- Specific computational resource requirements (e.g., number of processors, memory size)
- Expected scientific benefits

Which Applications are to be used?

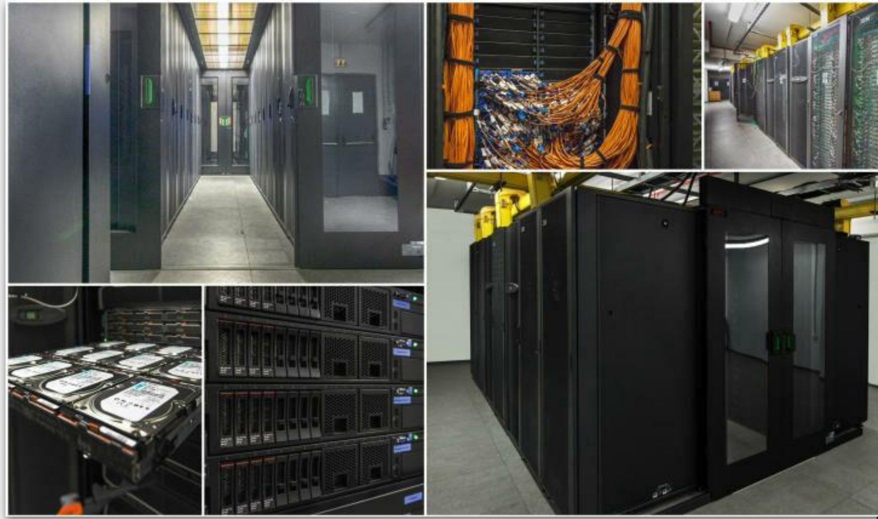
- Scientific applications using parallel processing methodologies such as:
  - Distributed memory on multiple nodes (MPI)
  - Shared memory on single nodes (OpenMP)
  - CUDA (for GPU acceleration)
- Applications simulate physical phenomena requiring extensive mathematical computations
- Machine Learning (ML) and neural network training benefiting from GPU acceleration

GRNET HPC – ARIS: Infrastructure

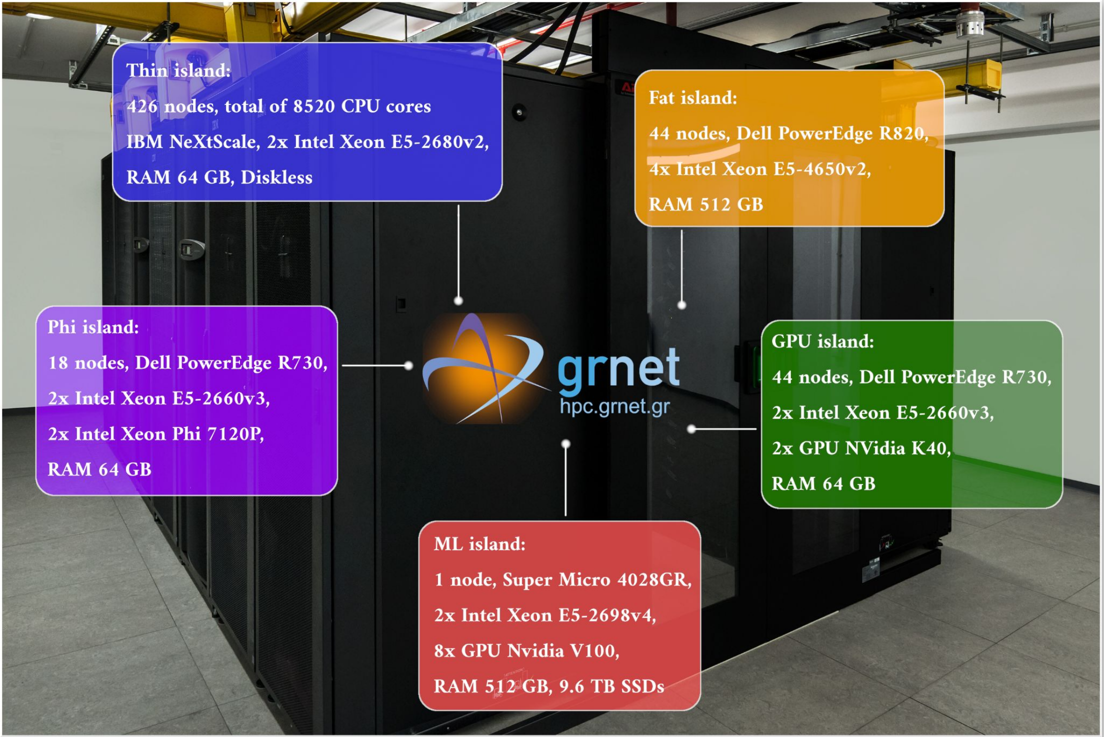


GRNET HPC - ARIS: Infrastructure

- 533 compute nodes organized in 5 partitions/islands (node groups)
- Resource Manager: Slurm v. 16.05.11
- Operating System: Red Hat Enterprise Linux 6 & 7
- File System: 2PB IBM GPFS
- Interconnection network: Infiniband 56 Gbps
- Processing capability: 535 TFlops
- No. 468 in the Top 500 list of June of 2015



GRNET HPC – ARIS: Infrastructure



Thin island:

426 nodes, total of 8520 CPU cores

IBM NeXtScale, 2x Intel Xeon E5-2680v2,

RAM 64 GB, Diskless

Fat island:

44 nodes, Dell PowerEdge R820,

4x Intel Xeon E5-4650v2,

RAM 512 GB

Phi island:

18 nodes, Dell PowerEdge R730,

2x Intel Xeon E5-2660v3,

2x Intel Xeon Phi 7120P,

RAM 64 GB

GPU island:

44 nodes, Dell PowerEdge R730,

2x Intel Xeon E5-2660v3,

2x GPU NVidia K40,

RAM 64 GB

ML island:

1 node, Super Micro 4028GR,

2x Intel Xeon E5-2698v4,

8x GPU Nvidia V100,

RAM 512 GB, 9.6 TB SSDs

Preparatory/Development Projects

- The call for preparatory/development projects is open continuously
- Evaluation results are provided within 10 business days of submission
- Selected projects start within 1 month after evaluation

- Provide access to ARIS for researchers in Greek institutions to:
  - Preparatory - Type A: Perform scalability tests
  - Development - Type B: Support code migration and optimization

- Duration: Up to 2 months for Type A, and 4 months for Type B

Preparatory/Development Projects

EURO Greece

- The maximum number of core hours provided in detail:
  a. **100,000 core hours** on the **Thin Node Island**
  b. **50,000 core hours** on the **GPU Island**
  c. **50,000 core hours** on the **Xeon Phi Island**
  d. **100,000 core hours** on the **Fat Node Island**
- The total requested core hours **must not exceed 100,000 core hours**

- Users should fill the submission form
- View form in PDF: preparatory.pdf
- Report after 2 months and within 30 days after project completion

Production Projects

- The call for production projects is periodically (2 times per year)
- Allocate up to 5 million core hours per project (total max 41 million core hours)
- The application must be completed in English
- PI must be affiliated with a Greek academic/research institution
- International collaborators allowed, but cannot be PIs
- Commitment to utilizing allocated resources and acknowledging ARIS in publications
- GRNET reserves the right to publish project summaries and performance results
- Access to the System ends 12 months after the acceptance/allocation date
- Final report for approved projects: 2 months after access ends

Production Projects

- **Selection Criteria**
  1. **Scientific Excellence (K1)**: Impact, novelty, and adherence to international standards
  2. **Need for Use (K2)**: Justification for using ARIS HPC resources
  3. **Adequacy (K3)**: Experience and expertise of PI and team
  4. **Applicability (K4)**: Compatibility with ARIS system and resource availability

- **Total: 41 million core hours** allocated as follows:
  - Thin nodes: 30 million core hours
  - Fat nodes: 7 million core hours
  - GPU nodes: 3 million core hours
  - Intel Xeon Phi nodes: 1 million core hours
  - Machine learning: 30,000 GPU-card hours

Production Projects

- **Evaluation Process**
  1. **Stage A: Eligibility & Completeness Check**
  2. **Stage B: Technical Evaluation** (Feasibility - K4)
  3. **Stage C: Scientific Evaluation** (K1, K2, K3) - Peer review
  4. **Stage D: Ranking & Resource Allocation** - Technical Committee decision
- Scores range from **Low (1) to Excellent (4)** for each **Ki** and **overall proposal**
- Proposals with "Low" overall quality score are rejected before final stage
- If resources are limited, the Committee may reduce allocations based on minimum evaluator recommendations to support more proposals
- Notifications sent via email with further instructions
- Users should fill the form (on active call period)
- View form in PDF: production.pdf
- Additional file to provide in proposal submission: detailed project document.doc
- Report submission required within 30 days after project completion

Useful Links

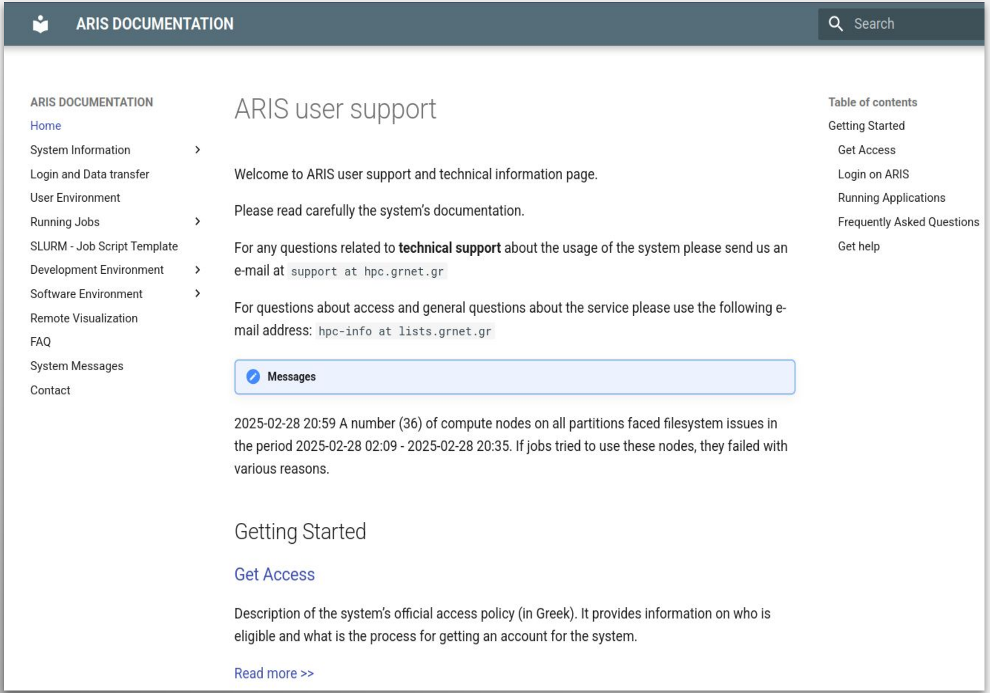
- The system's technical specifications are available in the Technical Description
- The access and usage policies are outlined in the ARIS Access Policy
- ARIS Terms of Use (Acceptable Usage Policy)
- User should accept the Privacy Policy
- For detailed information and announcements, register for the HPC Announcement List
- ARIS Documentation

How to submit a job via Slurm on an HPC cluster

Nikos Triantafyllis

ntriantafyl@admin.grnet.gr

GRNET HPC – ARIS



ARIS user support

Welcome to ARIS user support and technical information page.

Please read carefully the system's documentation.

For any questions related to **technical support** about the usage of the system please send us an e-mail at `support at hpc.grnet.gr`

For questions about access and general questions about the service please use the following e-mail address: `hpc-info at lists.grnet.gr`

Messages

2025-02-28 20:59 A number (36) of compute nodes on all partitions faced filesystem issues in the period 2025-02-28 02:09 - 2025-02-28 20:35. If jobs tried to use these nodes, they failed with various reasons.

Getting Started

Get Access

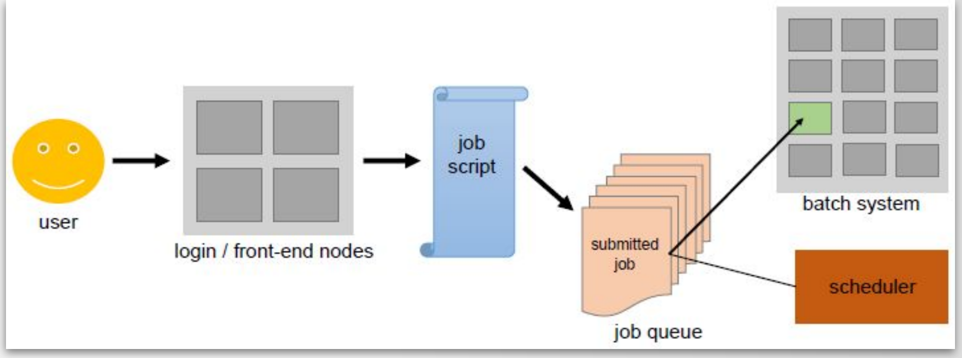
Description of the system's official access policy (in Greek). It provides information on who is eligible and what is the process for getting an account for the system.

Read more >>

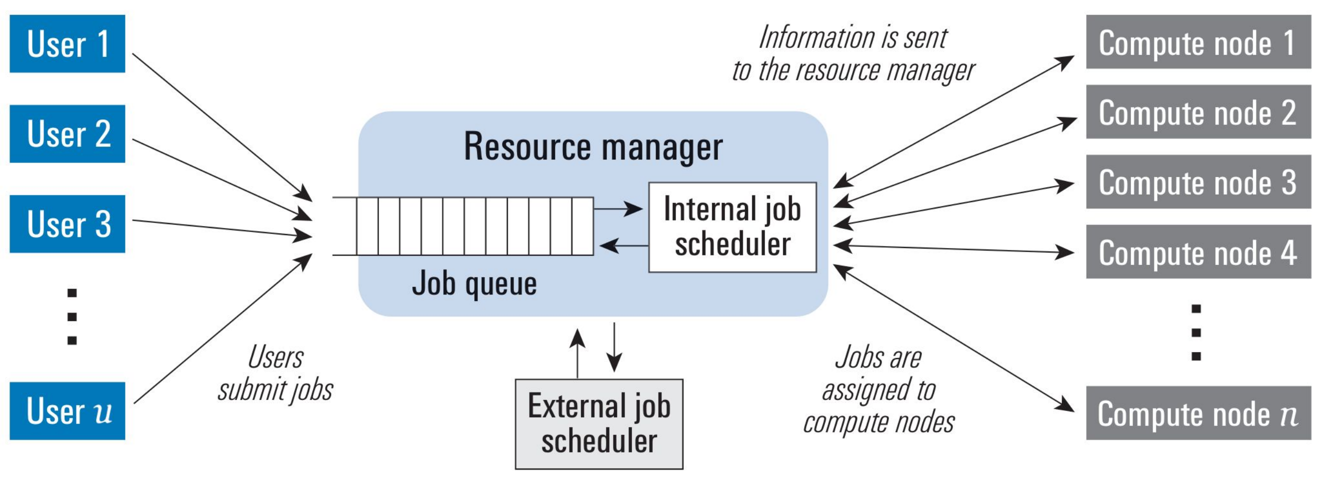
https://doc.aris.grnet.gr

Job Lifecycle

- **User Access**: User access HPC via ssh Job
- **Submission**: User submits a job using sbatch
- **Pending (PD)**: Job waits in queue for resources to become available
- **Scheduled**: SLURM assigns resources based on priority and availability
- **Running (R)**: Job executes on allocated resources
- **Completion (CD)**: Job finishes successfully or fails
- **Failure/Preemption (F)/(PR)**: Job may fail due to errors or get preempted by higher-priority jobs
- **Job Cleanup**: SLURM releases resources, logs results



SLURM - Resource and Job Management System



slurm workload manager software stack that runs on HPC infrastructure and operates resource management, job scheduling and accounting

SLURM Useful Commands

- sacct is used to report job accounting information
- sbatch is used to submit a job script for later execution
- scancel is used to cancel a pending or running job
- sinfo reports the state of partitions and nodes managed by SLURM
- squeue reports the state of jobs
- srun usually is executed inside the job script to run apps after job submission

More info https://slurm.schedmd.com

ARIS Modules

Managing Environment with Modules

- Modules control environment variables such as **PATH**, **LD_LIBRARY_PATH**
- Use **module** command to load, unload, and list modules

Module Commands

- **module avail**: List all available modules
- **module load <module>**: Load a module
- **module unload <module>**: Unload a module
- **module list**: List loaded modules
- **module switch**: Switch between module versions
- **module purge**: Remove all modules
- Example of loading a module with a certain version:

  ```
  > module load gnu/5.1.0
  > gcc --version
  gcc (GCC) 5.1.0
  ```

ARIS Compilers

- Available compilers : GNU, Intel, PGI, Sun(Oracle)
- Available MPI Flavors : IntelMPI, OpenMPI, MVapiCH.
- Best Compiler flags, more flags may be needed
- GNU : -O3 -mavx -march=ivybridge
- Intel : -O3 -xCORE-AVX-I
- PGI : -O4 -tp=sandybridge
- MPI :
    - IntelMPI (Intel): mpiicc, mpiicpc, mpiifort
    - OpenMPI(gnu/intel/pgi) : mpicc, mpicxx, mpif90

SSH Access

- Accessible from Internet via SSH at login nodes
- User must provide a username, SSH keys, and an IP range (at max /24) to be granted access
- Compute nodes are not directly accessible and they have no internet access
- File Transfer through secure protocols e.g. scp, sftp
- To connect, user need to have an SSH Client Software. For instance:
  a. Mac OS, Linux: OpenSSH (usually pre-installed)
  b. Windows: Putty, Bitvise, mobaXterm

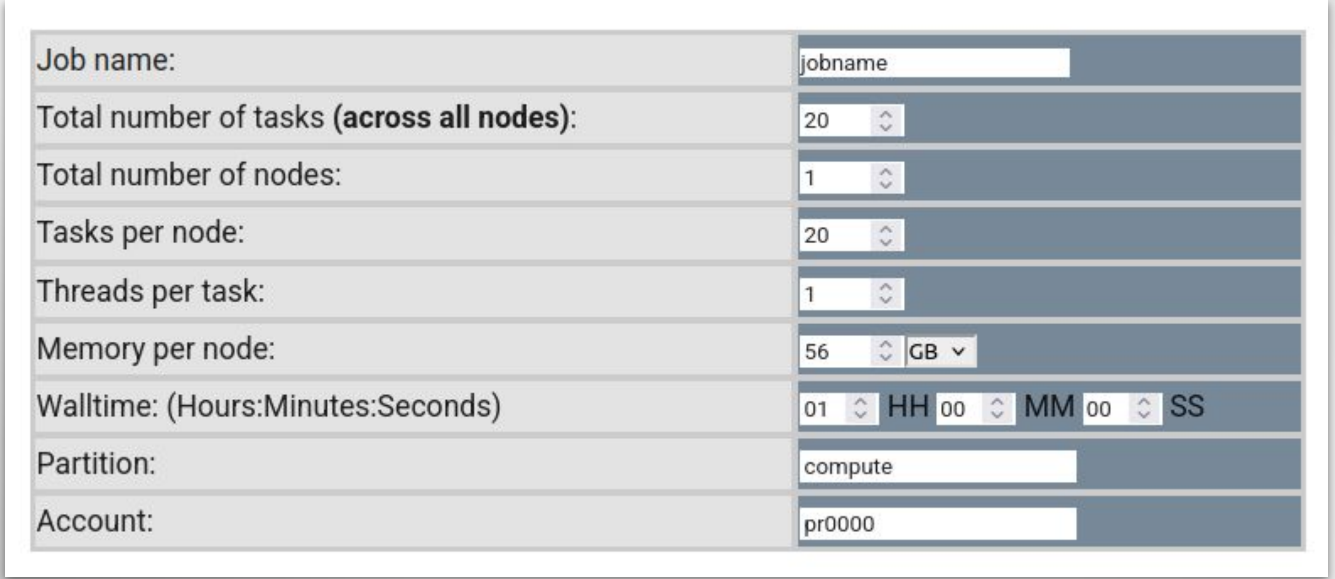
Transfer files from/to ARIS

- Put a local directory from your computer to ARIS

  scp -i /path/to/rsa -r localdir username@login.aris.grnet.gr:remotedir
- Get a remote directory from ARIS to your computer

  scp -i /path/to/rsa -r username@login.aris.grnet.gr:remotedir localdir
- Clients: scp (Linux/Mac), or PSCP (Windows), or WinSCP (Windows)

SLURM Directives



Job name:	jobname
Total number of tasks **(across all nodes)**:	20
Total number of nodes:	1
Tasks per node:	20
Threads per task:	1
Memory per node:	56 GB
Walltime: (Hours:Minutes:Seconds)	01 HH 00 MM 00 SS
Partition:	compute
Account:	pr0000

See Template: https://doc.aris.grnet.gr/scripttemplate

Submit MPI Job

```
#!/bin/bash -l

# Pure MPI job , using 80 procs on 4 nodes ,
# with 20 procs per node and 1 thread per MPI task

#SBATCH --job-name=mpijob # Job name
#SBATCH --output=mpijob.%j.out # Stdout (%j expands to jobId)
#SBATCH --error=mpijob.%j.err # Stderr (%j expands to jobId)
#SBATCH --ntasks=80 # Total number of tasks
#SBATCH --nodes=4 # Total number of nodes requested
#SBATCH --ntasks-per-node=20 # Tasks per node
#SBATCH --cpus-per-task=1 # Threads per task(=1) for pure MPI
#SBATCH --mem=56000 # Memory per job in MB
#SBATCH -t 01:30:00 # Run time (hh:mm:ss) - (max 48h)
#SBATCH --partition=compute # Submit queue
#SBATCH -A testproj # Accounting project

# Load any necessary modules

module load gnu
module load intel
module load intelmpi

# Launch the executable

srun EXE ARGS
```

Submit Hybrid MPI/OpenMP Job

```
#!/bin/bash -l

# Hybrid MPI/OpenMP job , using 80 procs on 4 nodes ,
# with 2 procs per node and 10 threads per MPI task.

#SBATCH --job-name=hybridjob # Job name
#SBATCH --output=hybridjob.%j.out # Stdout (%j expands to jobId)
#SBATCH --error=hybridjob.%j.err # Stderr (%j expands to jobId)
#SBATCH --ntasks=8 # Total number of tasks
#SBATCH --nodes=4 # Total number of nodes requested
#SBATCH --ntasks-per-node=2 # Tasks per node
#SBATCH --cpus-per-task=10 # Threads per task
#SBATCH --mem=56000 # Memory per job in MB
#SBATCH -t 01:30:00 # Run time (hh:mm:ss) - (max 48h)
#SBATCH --partition=compute # Submit queue
#SBATCH -A testproj # Accounting project

# Load any necessary modules

module load gnu
module load intel
module load intelmpi

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# Launch the executable
srun EXE ARGS
```

Submit GPU Job

```
#!/bin/bash -l

# GPU job , using 80 procs on 4 nodes ,
# with 2 gpus per node, 1 procs per node  and 20 threads per MPI task.

#SBATCH --job-name=gpujob # Job name
#SBATCH --output=gpujob.%j.out # Stdout (%j expands to jobId)
#SBATCH --error=gpujob.%j.err # Stderr (%j expands to jobId)
#SBATCH --ntasks=4 # Total number of tasks
#SBATCH --gres=gpu:2 # GPUs per node
#SBATCH --nodes=4 # Total number of nodes requested
#SBATCH --ntasks-per-node=1 # Tasks per node
#SBATCH --cpus-per-task=20 # Threads per task
#SBATCH --mem=56000 # Memory per job in MB
#SBATCH -t 01:30:00 # Run time (hh:mm:ss) - (max 48h)
#SBATCH --partition=gpu # Run on the GPU nodes queue
#SBATCH -A testproj # Accounting project

# Load any necessary modules

module load gnu
module load intel
module load intelmpi
module load cuda

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# Launch the executable
srun EXE ARGS
```

Submit multiple serial Job

```
#!/bin/bash -l

# Multiple Serial job , 5 tasks , requesting 1 node, 2800 MB of memory per task

#SBATCH --job-name=multiple-seraljob # Job name
#SBATCH --output=multiple-serialjob.%j.out # Stdout (%j expands to jobId)
#SBATCH --error=multiple-serialjob.%j.err # Stderr (%j expands to jobId)
#SBATCH --nodes=1 # Total number of nodes requested
#SBATCH --ntasks=5 # Total number of tasks
#SBATCH --ntasks-per-node=5 # Tasks per node
#SBATCH --cpus-per-task=1 # Threads per task
#SBATCH --mem-per-cpu=2800 # Memory per task in MB
#SBATCH -t 01:30:00 # Run time (hh:mm:ss) - (max 48h)
#SBATCH --partition=taskp # Submit queue
#SBATCH -A testproj # Accounting project

# Load any necessary modules
module load gnu
module load intel

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# Launch the executable a.out

srun -n 1 -c 1 ./a.out input0 &
srun -n 1 -c 1 ./a.out input1 &
srun -n 1 -c 1 ./a.out input2 &
srun -n 1 -c 1 ./a.out input3 &
srun -n 1 -c 1 ./a.out input4
wait
```

Usage Report

```
$ mybudget
========================================================================
Core Hours Allocation Information for account : testproj
========================================================================
Allocated Core Hours   :      2400000.00
Consumed  Core Hours   :                 15.00
Percentage of Consumed :                 0.00
========================================================================


$ myreport
----------------------------------------------------------------------------------------------------
Cluster/Account/User Utilization 2015-04-07T00:00:00 - 2015-10-07T23:59:59 (15897600 secs)
Time reported in CPU Hours
----------------------------------------------------------------------------------------------------
  Cluster            Account Login       Proper Name       Used       Energy
--------- --------------- --------- --------------- ---------- ----------------------------------------
 aris       testproj username            User Name          15          110
```

SLURM commands in action

- $ **sbatch script.sh**

  ```
  Submitted batch job 12345
  ```

- $ **squeue -j 12345**

  ```
  JOBID  PARTITION  NAME          USER  ST  TIME  NODES  NODELIST(REASON)
  12345  batch     my_job  user1  PD  0:00  20        (Resources)
  ```

- $ **squeue -j 12345**

  ```
  JOBID  PARTITION  NAME          USER  ST  TIME  NODES  NODELIST(REASON)
  12345  batch     my_job  user1  R 1:20  20           node[01-20]
  ```

- $ **sacct -j 12345 --format=JobID,JobName,Partition,Account,AllocCPUS,State,ExitCode**

  ```
  JobID     JobName  Partition          Account  AllocCPUS          State ExitCode
  ------------ ---------- ---------- ---------- ---------- ---------- --------
  12345     my_job  compute          my_acc            4          COMPLETED       0:0
  12345.batch        batch     compute          my_acc            4          COMPLETED       0:0
  12345.0            task1     compute          my_acc            2          COMPLETED       0:0
  12345.1            task2     compute          my_acc            2          COMPLETED       0:0
  ```

- $ **scancel 12345**

EURO Greece

Thanks!

Co-funded by the European Union

EuroHPC Joint Undertaking

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.