JULIA on HPC

What is Julia?

Julia is a high-level programming language for technical computing, with syntax that is familiar to users of other technical computing environments. It was designed to be used for numerical and scientific computing and is fast enough to be used for high-performance computing. Julia is free and open-source and available for use on various platforms, including Windows, macOS, and Linux. The language has a strong focus on performance and has been designed to be able to execute code at speeds comparable to compiled languages like C and Fortran. Julia also has a rich set of libraries and packages for a wide range of applications, including machine learning, data analysis, and scientific computing.

Links:

Official Website

Documentation

Distributed Computing

Princeton Documentation

Versions Available:

The following versions are available on the cluster:

- Julia-1.6.2

How to load JULIA?

To load JULIA, use the following commands:

```
#Load the JULIA module
module load julia/1.6.2
```

To verify if the module and dependencies are loaded correctly, use the following command.

```
#Show all the modules loaded
module list
```

This should list all the JULIA dependencies that are loaded- only Julia in this case.

How to use JULIA?

To demonstrate the usage of this programming language, use the following code snippet:

```
using Distributed

# launch worker processes
num_cores = parse(Int, ENV["SLURM_CPUS_PER_TASK"])
addprocs(num_cores)

println("Number of cores: ", nprocs())
println("Number of workers: ", nworkers())

# each worker gets its id, process id and hostname
for i in workers()
    id, pid, host = fetch(@spawnat i (myid(), getpid(),
gethostname()))
    println(id, " " , pid, " ", host)
end

# remove the workers
for i in workers()
    rmprocs(i)
end
# The code above launches worker processes on HPC. The number of cores
to use is obtained from #the SLURM_CPUS_PER_TASK environment variable
and passed to the addprocs function. This #function starts worker
processes on the specified number of cores. The println statements are
#then used to print out the number of cores, the number of worker
```

Make a distributed.jl script and paste the above code. To submit the job to SLURM scheduler, use the following template,

```
#!/bin/bash
#SBATCH --job-name=julia_worker
#SBATCH --output=julia_worker.out
#SBATCH --error=julia_worker.err
#SBATCH -p main
#SBATCH --qos main
#SBATCH --nodes=1
#SBATCH --ntasks 1
#SBATCH --cpus-per-task=4      # No of processor to assign

# Load the module
module load julia/1.6.2

julia distributed.jl
```

The output should be on julia_worker.out which looks like,

```
Number of cores: 5
Number of workers: 4
2 337478 compute-1-3.local
3 337480 compute-1-3.local
4 337481 compute-1-3.local
5 337482 compute-1-3.local
```

Where to find help?

If you are stuck on some part or need help at any point, please contact OIT at the following address.

https://ua-app01.ua.edu/researchComputingPortal/public/oitHelp