Johns Hopkins Engineering

Applied Machine Learning for Mechanical Engineers

Machine Learning Using Supercomputers, Part 1, A

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Supercomputing

- By the end of this lecture, you will be able to:
  - Describe computational intensity
  - Describe distributed computing
  - Describe multi-core parallel computing
  - Describe multi-node parallel computing
  - Describe Supercomputers
  - Describe MARCC

Supercomputing

- Computational Intensity
  - Rough estimation of required computation in time for a particular processing unit with certain speed
    - Required time to initiate the run
    - Required hours of run per one processing unit
    - Required time to end the run
  - Most of the time we can neglect the required time to initiate and end the run

Supercomputing

- Computational Intensity
  - Rough estimation of required computation in time for a particular processing unit with a certain speed
    - Example: Processor Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
    - It takes about 2.5 hours to complete the machine learning run for 1 RTT and 1 RRS
    - We have 5 RTTs and 100 RRSs
    - Total required time in hours for one CPU:

$$2.5\ hours \times 5\ RTT \times 100\ RRSs = \ 1{,}250\ hours$$

Supercomputing

- Multi-Core Parallel Computing
  - Rough estimation of required computation in time for a particular processing unit with certain speed
    - Example: Processor Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
    - Let's say our computer has 4 CPU cores
    - Then we should be able to divide the independednt computation between the 4 cores. Each RTT-RRS can be computed independently. Assign $100/4 = 25$ different RRS of 5 RTT to a different CPU core. Then the total time to run the code is about:

$$\frac{2.5\ hours \times 5\ RTT \times 100\ RRSs}{4} = 312.5\ hours$$

Supercomputing

- Multi-Core Parallel Computing

  - Rough estimation of required computation in time for a particular processing unit with certain speed
    - Example: Processor Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
    - It takes time to initiate parallel computing
    - Communication between cores are also taking time
    - So :

$$\frac{2.5\ hours \times 5\ RTT \times 100\ RRSs}{4} \times 1.25 \approx \ 400\ hours \approx 17\ days$$

Supercomputing

- Multi-Node Parallel Computing
  - Rough estimation of required computation in time for a particular processing unit with certain speed
    - Example: Processor Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
    - Let’s say we have access to 25 similar computers, aslo referred to as nodes.
    - we should be able to divide the independednt computation between the $4 \times 25 = 100$ cores. Each RTT-RRS can be computed independently. Assign $100/100 \ = 1$ different RRS of 5 RTT to a different CPU core. Then the total time to run the code is about (including the safety factor of 1.25 per each node):

$$\frac{2.5\ hours\ \times 5\ RTT\ \times 100\ RRSs}{100} \times 1.25 \approx\ 16\ hours \approx less\ than\ 1\ day$$

Supercomputing

- Supercomputers
  - Supercomputers are equipped with high-level clusters of nodes for accelerated and high-performance computing (HPC)
    - You can divide your computational needs over their multiple nodes and cores
    - Each node has multiple cores
    - There are nodes with multiple high-performance GPUs
    - There are nodes with large Random-Access Memory (RAM), in the scale of over 1 TB
    - One famous and powerful supercomputer is The Maryland Advanced Research Computing Center (MARCC)

IBM supercomputer

MARCC

https://www.marcc.jhu.edu/

https://en.wikipedia.org/wiki/File:IBM_Blue_Gene_P_supercomputer.jpg

Supercomputing

- Supercomputers
  - Every supercomputer usually has its way of operating and management
  - Once you are assigned an account, you should have an individual directory on their systems with certain amount of hard drive storage
  - You use your username and password to connect to the supercomputer login nodes
  - There might be some two step-verification process to connect to their systems

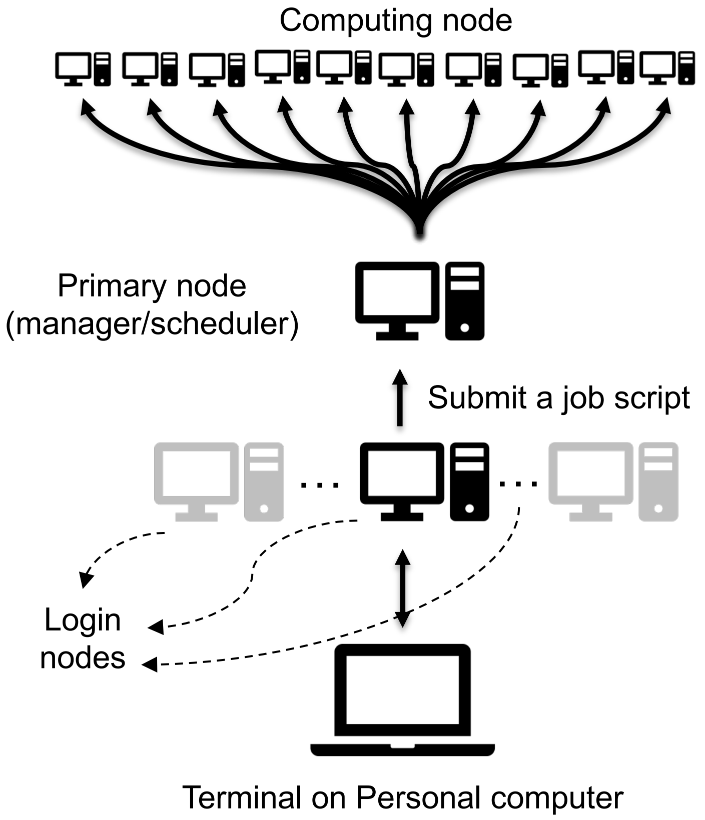
IBM supercomputer

MARCC

https://www.marcc.jhu.edu/
https://en.wikipedia.org/wiki/File:IBM_Blue_Gene_P_supercomputer.jpg

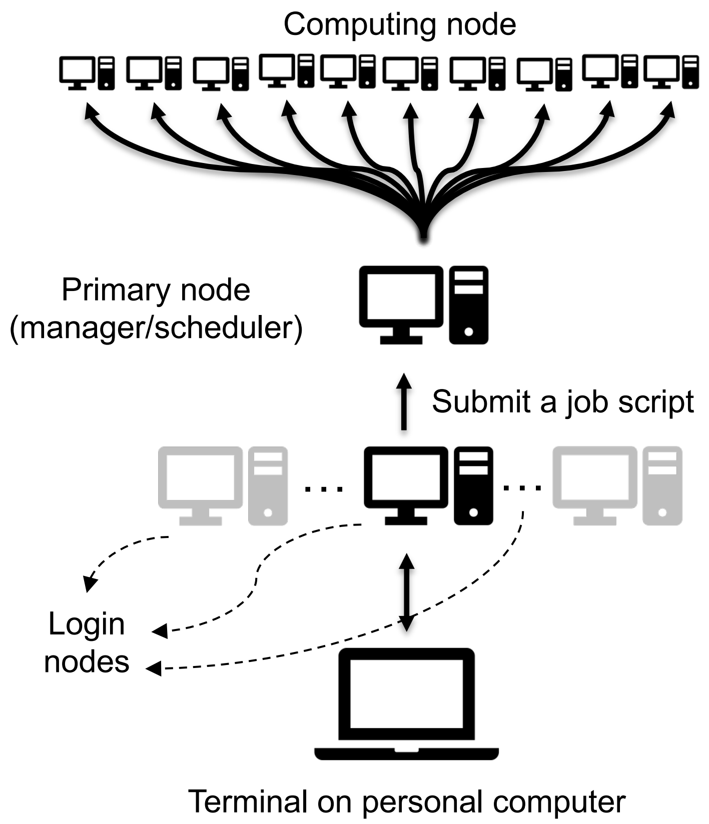
Supercomputing

- Supercomputers
  - Nodes are usually Linux-based
  - We usually connect to a supercomputer “login node” to manage our files and submit a parallel computing plan, similar to the example in previous slides
  - We need our personal computer or have a “terminal” software to be able to connect to the login node
  - Linux and macOS systems have terminals. On windows 10, you can install “ubuntu.”
  - We can also implement the terminal at google Colab, so we would not need to install anything extra on our computer.


Computing node
Primary node (manager/scheduler)
Submit a job script
Login nodes
Terminal on Personal computer

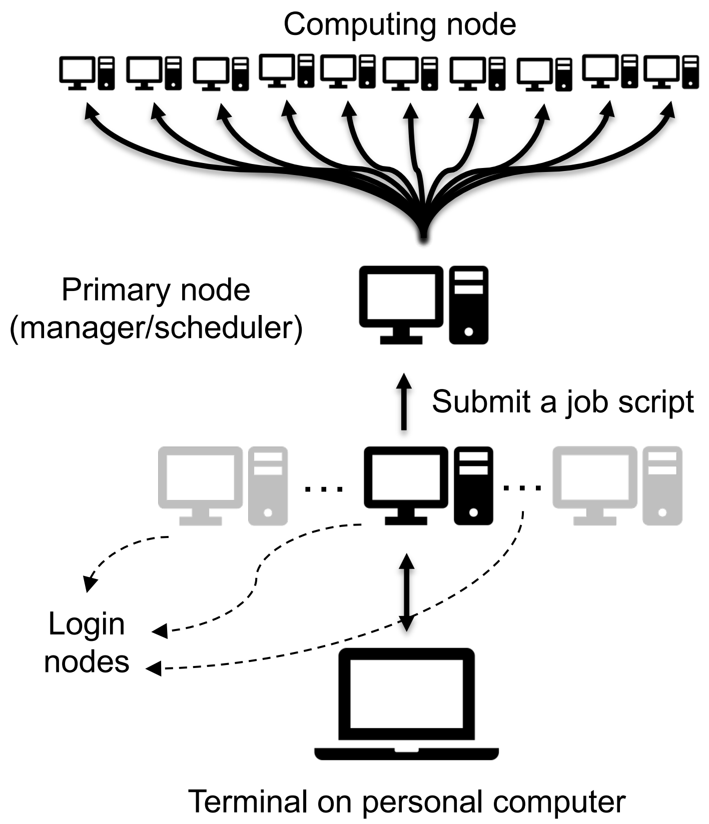
Supercomputing

- Supercomputers
  - You need to know some Linux-based commands
  - You can send files to your designated directory or download files/results to your personal computer
  - You can create directories, add files, remove files, and manage your codes
  - Every node on the supercomputer usually has access to your designated software and programming tools such as Python, R, MATLAB, etc., and necessary libraries such as TensorFlow, matplotlib, etc., all with necessary licenses.


Computing node
Primary node (manager/scheduler)
Submit a job script
Login nodes
Terminal on personal computer

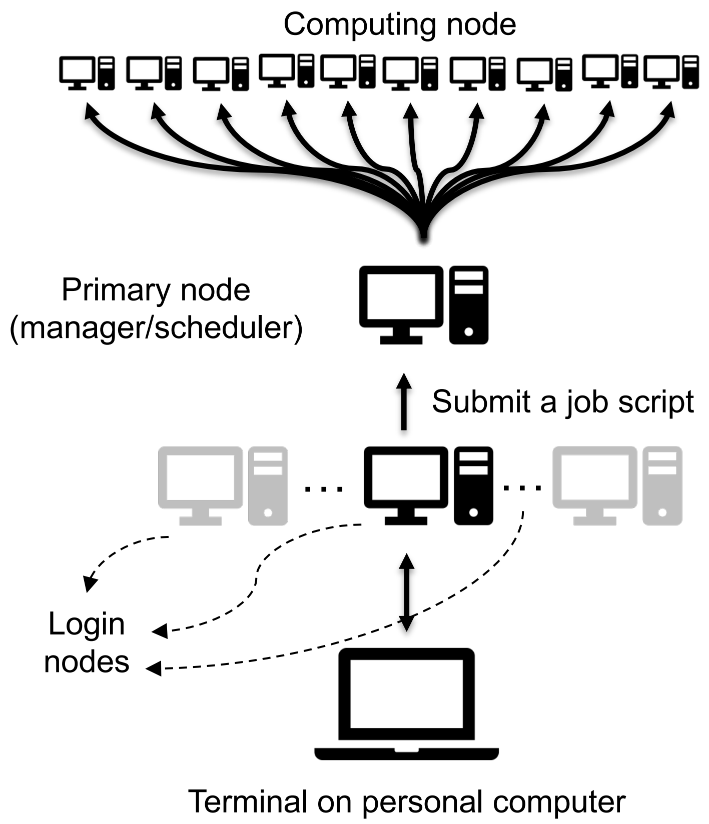
Supercomputing

- Supercomputers
  - You are not allowed to run your extensive codes on login nodes
  - Login nodes are just to manage your files and submit job scripts to the primary node
  - Job scripts are simply scripts that carry information such as how many nodes/cores you need, if GPU is needed, amount of required RAM, run time, and codes need to be run
  - The primary node then receives your job script and put your job in a queue.
  - Once nodes are available, based on your requested resources on the job script, the primary node assigns your job to the appropriate computing nodes.


Computing node
Primary node (manager/scheduler)
Submit a job script
Login nodes
Terminal on personal computer

Supercomputing

- Supercomputers
  - You can submit more than one job script
  - Once a job is completed, you may have access to your results in your designated directory
  - The runs are coming with logs, in case your code prints information as the outputs or if there is an error.



Supercomputing

- MARCC
  - MARCC is a shared computing facility located on the Bayview Campus of Johns Hopkins University and funded by a State of Maryland grant to Johns Hopkins University
  - MARCC is jointly managed by Johns Hopkins University and the University of Maryland College Park
  - Blue Crab is the main cluster at MARCC with over 23,000 cores (June 2018) and a combined theoretical performance of over 1.4 PFLOPs.
  - The compute nodes are a combination of Intel Ivy Bridge (large memory nodes), Haswell, Broadwell, and Skylake processors, and several Nvidia K80/P100 GPUS as of June 2018.
  - They keep upgrading their systems and hardware.

https://www.marcc.jhu.edu/about-marcc/about-us/

Supercomputing

- MARCC
  - Allocation policy: https://www.marcc.jhu.edu/allocation-policy/
  - The Blue Crab cluster controls access to the cluster with a “shares” system
  - Your principal investigator (PI), for example, may have access to 100,000 shares, which are shared between members of their group.
  - An allocation of 100,000 shares should allow your group to consume roughly 100,000 CPU-hours per quarter.
  - When you consume your allocation, you will no longer be able to use the hardware.

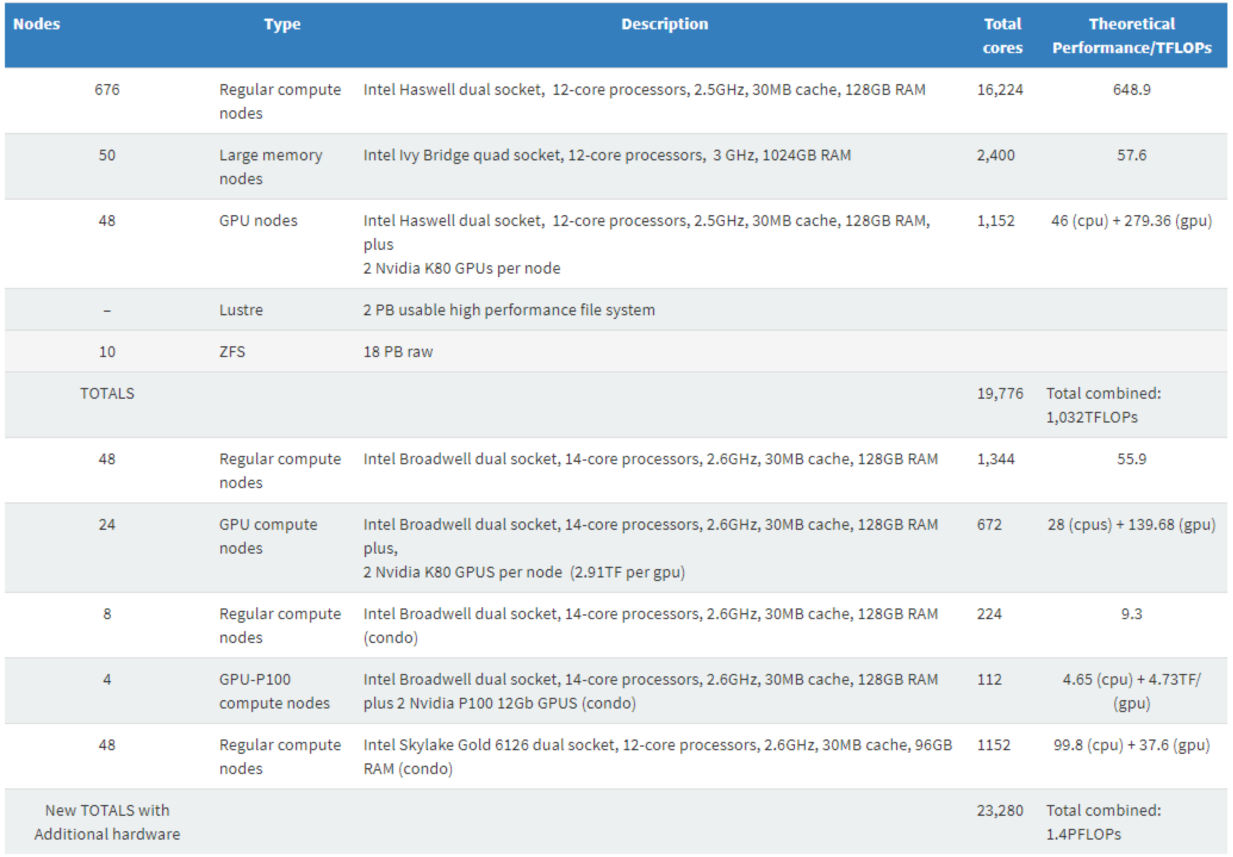
It is important to estimate the required computational intensity with a good precision to avoid loss of service and burning of the allocations

https://www.marcc.jhu.edu/hardware/

Supercomputing

- MARCC
  - Details of compute nodes

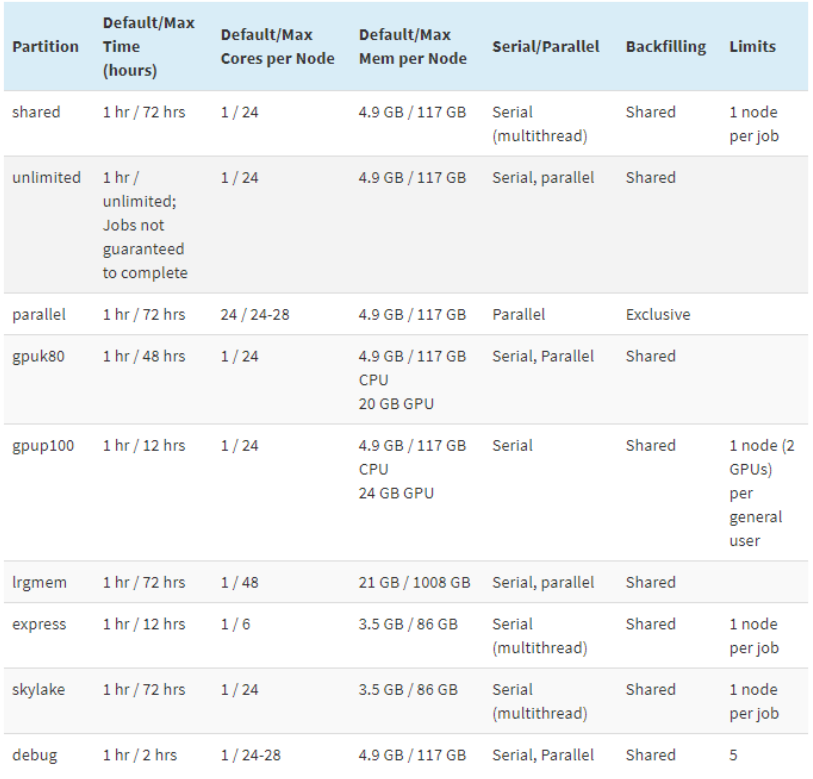
https://www.marcc.jhu.edu/hardware/



Nodes	Type	Description	Total cores	Theoretical Performance/TFLOPs
676	Regular compute nodes	Intel Haswell dual socket, 12-core processors, 2.5GHz, 30MB cache, 128GB RAM	16,224	648.9
50	Large memory nodes	Intel Ivy Bridge quad socket, 12-core processors, 3 GHz, 1024GB RAM	2,400	57.6
48	GPU nodes	Intel Haswell dual socket, 12-core processors, 2.5GHz, 30MB cache, 128GB RAM, plus 2 Nvidia K80 GPUs per node	1,152	46 (cpu) + 279.36 (gpu)
–	Lustre	2 PB usable high performance file system		
10	ZFS	18 PB raw		
TOTALS			19,776	Total combined: 1,032TFLOPs
48	Regular compute nodes	Intel Broadwell dual socket, 14-core processors, 2.6GHz, 30MB cache, 128GB RAM	1,344	55.9
24	GPU compute nodes	Intel Broadwell dual socket, 14-core processors, 2.6GHz, 30MB cache, 128GB RAM plus, 2 Nvidia K80 GPUS per node (2.91TF per gpu)	672	28 (cpus) + 139.68 (gpu)
8	Regular compute nodes	Intel Broadwell dual socket, 14-core processors, 2.6GHz, 30MB cache, 128GB RAM (condo)	224	9.3
4	GPU-P100 compute nodes	Intel Broadwell dual socket, 14-core processors, 2.6GHz, 30MB cache, 128GB RAM plus 2 Nvidia P100 12Gb GPUS (condo)	112	4.65 (cpu) + 4.73TF/(gpu)
48	Regular compute nodes	Intel Skylake Gold 6126 dual socket, 12-core processors, 2.6GHz, 30MB cache, 96GB RAM (condo)	1152	99.8 (cpu) + 37.6 (gpu)
New TOTALS with Additional hardware			23,280	Total combined: 1.4PFLOPs

Supercomputing

- MARCC
  - The primary node uses the Simple Linux Universal Resource Manager (SLURM) to manage resource scheduling and job submission
  - SLURM uses "partitions" to divide types of jobs
  - MARCC defines a few partitions that will allow sequential/shared computing and parallel, CPU, GPU, and large memory jobs.
  - The default partition is called "shared.“
  - Each partition might have limitations for its nodes



Partition	Default/Max Time (hours)	Default/Max Cores per Node	Default/Max Mem per Node	Serial/Parallel	Backfilling	Limits
shared	1 hr / 72 hrs	1 / 24	4.9 GB / 117 GB	Serial (multithread)	Shared	1 node per job
unlimited	1 hr / unlimited; Jobs not guaranteed to complete	1 / 24	4.9 GB / 117 GB	Serial, parallel	Shared	
parallel	1 hr / 72 hrs	24 / 24-28	4.9 GB / 117 GB	Parallel	Exclusive	
gpuk80	1 hr / 48 hrs	1 / 24	4.9 GB / 117 GB CPU 20 GB GPU	Serial, Parallel	Shared	
gpup100	1 hr / 12 hrs	1 / 24	4.9 GB / 117 GB CPU 24 GB GPU	Serial	Shared	1 node (2 GPUs) per general user
lrgmem	1 hr / 72 hrs	1 / 48	21 GB / 1008 GB	Serial, parallel	Shared	
express	1 hr / 12 hrs	1 / 6	3.5 GB / 86 GB	Serial (multithread)	Shared	1 node per job
skylake	1 hr / 72 hrs	1 / 24	3.5 GB / 86 GB	Serial (multithread)	Shared	1 node per job
debug	1 hr / 2 hrs	1 / 24-28	4.9 GB / 117 GB	Serial, Parallel	Shared	5

https://www.marcc.jhu.edu/getting-started/running-jobs/

Supercomputing

- MARCC
  - Example of a job script:
    - Create a file (no need for an extension such as .txt, etc.), rename it as say my_first_job and include these lines of script:
    - Use sbatch my_first_job to submit the job

```
#!/bin/bash

#SBATCH --job-name= my_job_name
#SBATCH –time = 00:20:00
#SBATCH –partition = shared
#SBATCH –nodes = 1
#SBATCH --cpus-per-task = 12
#SBATCH --mem-per-cpu = 4G
#SBATCH --mail-type=end
#SBATCH --mail-user=mrafiei1@jhu.edu
#SBATCH --requeue

# time mpiexec ./code-mvapich.x > OUT-24log

module load python/3.7.7
python -m venv ./venv
source ./venv/bin/activate
pip install tensorflow==2.2.0rc0
pip install matplotlib

python my_code01.py
python my_code02.py
```

Supercomputing

- MARCC
  - Details

```
#!/bin/bash

#SBATCH --job-name= my_job_name
#SBATCH –time = 00:20:00
#SBATCH –partition = shared
#SBATCH –nodes = 1
#SBATCH --cpus-per-task = 12
#SBATCH --mem-per-cpu = 4G
#SBATCH --mail-type = end
#SBATCH --mail-user = mrafiei1@jhu.edu

# time mpiexec ./code-mvapich.x > OUT-24log

module load python/3.7.7
python -m venv ./venv
source ./venv/bin/activate
pip install tensorflow==2.2.0rc0
pip install matplotlib

python my_code01.py
python my_code02.py
```

Must be included to tell the primary node to execute the following scripts

A name for your job

Time to run the job on the computation node/s

Name of the partition (shared, parallel, lrgmem, debug, etc.)

Number of nodes to run the codes in this job script

Number of CPU cores for each node

Memory required per CPU core

Send an email when the job is completed

Your email address

Include to have an output log in your job file directory

Load python/some-version into the compute node

Create a virtual environment for your python packages

Activate the created virtual environment

Install necessary packages (some aren't available some are)

Install necessary packages (some aren't available some are)

Include python codes to run in order

https://www.marcc.jhu.edu/getting-started/running-jobs/
https://www.marcc.jhu.edu/tensorflow-latest/

Supercomputing

- In this lecture, you learned about:
  - Computational intensity
  - Distributed computing
  - Multi-core parallel computing
  - Multi-node parallel computing
  - Supercomputers
  - MARCC
- In the next module, we will use MARCC resources to run some Python machine learning codes.

JOHNS HOPKINS

WHITING SCHOOL *of* ENGINEERING