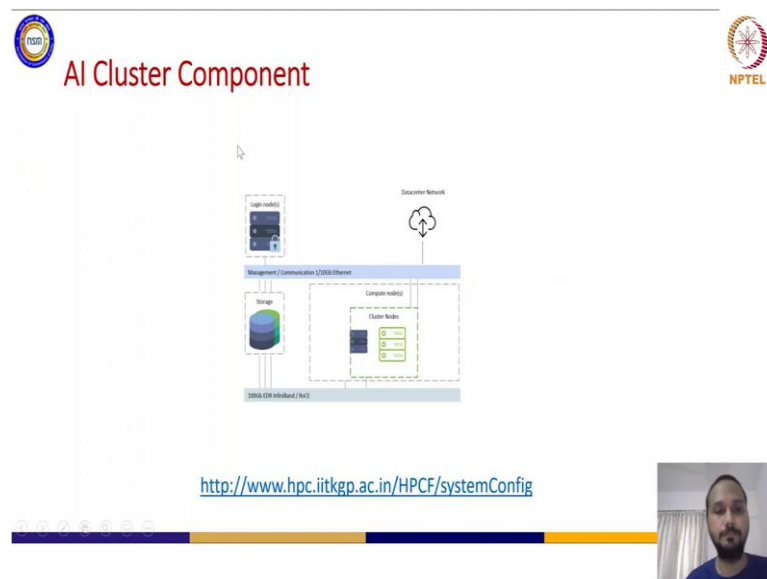


Applied Accelerated Artificial Intelligence
Prof. Bharatkumar Sharma
School of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 09
Scheduling and Resource Management Introduction to schedulers and
orchestration tools Part - 2

(Refer Slide Time: 00:50)



So, let us look at what all are the components of this particular cluster. So, I talked to you about a special networks, I said that we do have networks which are very special which are not your traditional 10 gig networks, but you have InfiniBand networks which core gives 100 gbps or 200 gbps also. So, that is one of the part of your overall cluster component so that you can communicate faster not just compute, but communicate faster across nodes also.

Obviously, they the users need to access it from outside. So, you need to have a network which is connecting it. You have your nodes where you are going to run all of your jobs which means you have the cluster nodes or the compute nodes where the actual computation will happen.

Like in an environment you will have a CPU – GPU nodes and you will have x number of them, and all these nodes are connected to each other via this high speed a InfiniBand

networks so that you can spread your work across those nodes and they can communicate faster and you can accumulate the results also.

You also need that every node does not have its own storage, but you have storage which is like a network file storage which means all the nodes see the same storage. So, if you access one node or versus the other node or the third node it does not matter, your home directory or your all of your files are actually mounted and they are there in this particular storage which is common for all the nodes.

And, as you can see here especially in AI also you require a very high speed storage because if you talk about AI you are not talking about even nowadays gigabytes of images or gigabytes of data. We are talking about terabytes of data that needs to be trained or analyzed on. So, you need also very high speed storage so that you can basically access not just compute faster not just communicate faster, but also read the data or the storage very very fast.

And, finally, the login node which might not be something new to you, but all the users they basically do not access the node directly they are given access to something called as a login node. In this login node is where you ask for resources and based on the available resources you get access to one of the nodes. So, we are not directly going to go and access the compute nodes.

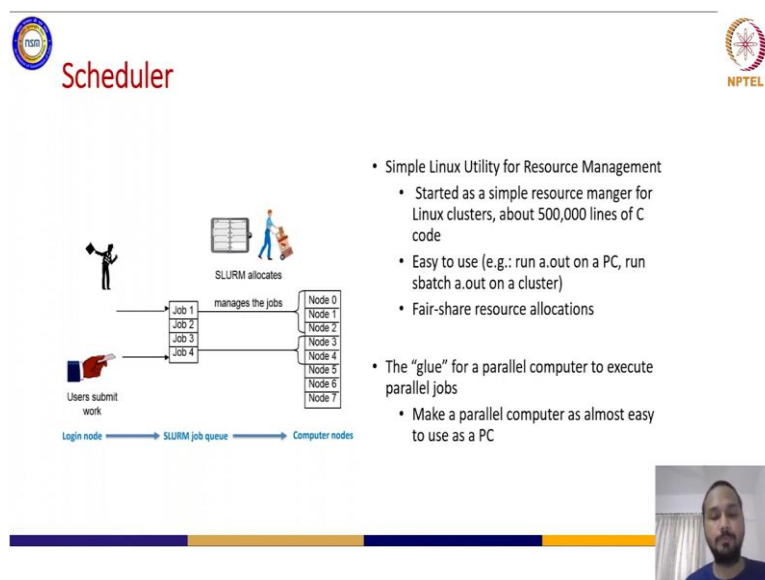
So, every user comes to the login node, ask for the resources then they are given that compute nodes and then they execute their work and then they are done. And, this is what generally the component of any cluster even in the cloud environment you will see almost the same setup, just that there will be some additional software component to maintain it which we will talk about.

So, I talked about all of the components of this cluster environment. Now, let us move on to the second part I said that I am going to ask for resources and I need to ask it. Now, what is that software that I require to do this? In order for you to get a feeling of how the cluster looks like, in the end if you get time I will also show you this.

I put one of the NSM cluster link here which kind of shows how the national supercomputing mission cluster configuration or the cluster environment looks like, how many compute nodes are there, how many login nodes are there, how much is the

storage, how are they connected to each other via the InfiniBand network. So, you can go and look at this specification and see how this clusters, traditionally look like in case you do not have access to one cluster at this point of time.

(Refer Slide Time: 04:48)



So, I told you that you need to access this hardware; right and you need to ask for it. It is not by default given to you and that is done by a software or a framework which is called as scheduler and as the name says the job is kind of very clear, right.

So, the idea is simple that you have a x number of compute resources which are known to the scheduler or its part of the configuration which means that I know that how many nodes I have, these nodes have how many CPU cores, how much memory, how many GPUs. All of those part are known to the scheduler which is the manager.

The user who comes to this clusters submits a work or a job as you can see there will be multiple users. Every user submits a job what is this job containing I will come to that part. And, then the SLURM is the one the manager basically sees what do you require and based on what is free and what provision has been provided to you like if you can only ask for one node then you will get only one node.

And, the Slurm basically takes your job and allocates a node for you so that you can go ahead and start executing on that particular node. Now, here you can ask for 5 nodes, 8 nodes based on what is the computer requirement from your side. So, you see that there

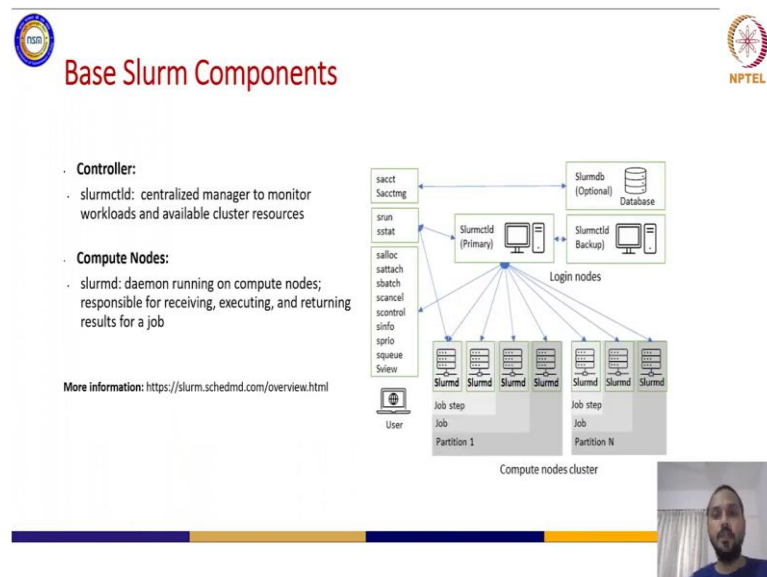
is a in between scheduler as a manager which comes into the picture making sure that all the policies are also met like everybody gets a fair share and there is no coordination which is required because everything has been done via the Slurm.

One of the most popular scheduler is called as Slurm which is simple Linux Utility for Resource Management . It started with a simple resource manager. It has almost like 5,00,000 lines of C code. It is very easy to use. You can run it on your PC if you want to by the way. If you would like to do that you can do it.

Generally, it is used in a cluster environment, but if you would like to test out on your own laptop, you can run both the manager and the compute node. You can make your laptop as both of it and try out if you would like to. Just to get a feeling, but we recommend you to leave this job of management and installation of Slurm and everything for the cluster admins. And, if you are a cluster admin yes please do go ahead and do that .

It is the glue for a parallel computer as I said to execute parallel jobs and that is what is the role of the scheduler.

(Refer Slide Time: 07:31)



If I look at it from architecture point of view the Slurm basically consist of two main components. The first one is called as a controller; the second one is your compute

nodes. So, as you can see here this is your login node, I told you there are two types of nodes when I showed you the cluster – the login node and the compute nodes.



The login node is the one where you have the controller. It is run as a daemon. The daemon name is `slurmctld`. So, you can see here it runs as a daemon and basically a user is going to ask this daemon which is running to basically allocate certain compute resources.

The compute nodes is going to run another daemon which is called as `slurmd`. This `slurmd` is basically the daemon which is going to listen to the manager and basically the `slurmd` will tell the manager that I am free or how many resources are free and all those things and based on what you have asked for the slurm can basically based on the policy defined assign that particular node for the job that you have in mind.

There are some optional component also which I am not covering in this particular session, but I hope you get the idea. So, it is like a client server right or a manager and a slave like a model where you are going to have a master and the master is going to give the job to the slaves who are going to basically do it and then give the results back to the master and they basically work on it.

So, I have provided the link for more information if anybody who would like to see how this Slurm functions into more detail of this Slurm architecture.



(Refer Slide Time: 09:17)



Job Submission

- To run your job, you will need to specify what resources you need.
- These can be memory, cores, nodes, GPUs, etc.
- There is a lot of flexibility in the scheduler to get specifically the resources you need.

Options	Description
--nodes	The number of nodes for the job (computers)
--mem	The amount of memory per node that your job need
-n	The total number of tasks you job requires
--gres gpu:#	The number of GPUs per node you need in you job
--qos	the QOS you want to run in, currently normal or debug
--mem-per-cpu=	The amount of memory per cpu your job requires
-N	The minimum (and maximum) number of nodes required
--ntasks-per-node=#	tasks per node.
--exclusive	this will get you exclusive use of the node




So, what can you ask for to this manager, right? So, you can ask different things. You can ask a node which means how many compute nodes? Do you need one computer, two computer, three computers, four computer? How much memory do you require for node? Like do you need 1 gb, 2 gb, 4 gb, 8 gb?

You can ask for the number of parallel task that you are going to run like if you are going to say you know that you have a task which can utilize this only four cores. So, you can say I need within a node I need one node and within a node I need only four cores. So, if your cluster if your node had 16 cores, it will take away those four cores and give it to you while the rest all will be free for somebody else to use.


You can also ask the number of GPUs you want that in that particular node. You can also define some quality of service which will skip for this session; it is more towards the admin side. And, also you can define other parameters number of tasks that you need per node and do you want exclusive access to it or you can share those resources with somebody else and there are different things that you can ask.

I am going to show you a live demo of this so that you get an idea of how this overall thing works.

(Refer Slide Time: 10:31)




Slurm Common Commands



• For full list of common Slurm command, follow this [link](#).

Command	Description	Detailed
sinfo	Reports the state of the partitions and nodes managed by Slurm	PARTITION: the name of the partition AVAIL: whether the partition is up or down TIMELIMIT: the maximum length a job will run in the format Days:Hours:Minutes:Seconds NODES: the number of nodes of that configuration STATE: down* if jobs cannot be ran, idle if it is available for jobs, alloc if all the CPUs in the partition are allocated to jobs, or mix if some CPUs on the nodes are allocated and others are idle. NODELIST: specific nodes associated with that partition.
sacct	Lists the jobs that are running or have been run.	
squeue	Lists the state of all jobs being run or scheduled to run. Use squeue -u username to view only the jobs from a specific user	JOBID: number id associated with the job PARTITION: name of partition running the job NAME: name of the job ran with sbatch or sinteractive USER: who ordered the job to be ran ST: State of the job, PD for pending, R for running TIME: how long the job has been running in the format Days:Hours: Minutes:Seconds NODES: number of nodes allocated to the job NODELIST(REASON): either the name of the node running the job of the reason the job is not running such as JobHeldAdmin (job is prevented from running by the administrator).
scancel	Signals or cancels a job. One or more jobs separated by spaces may be specified.	

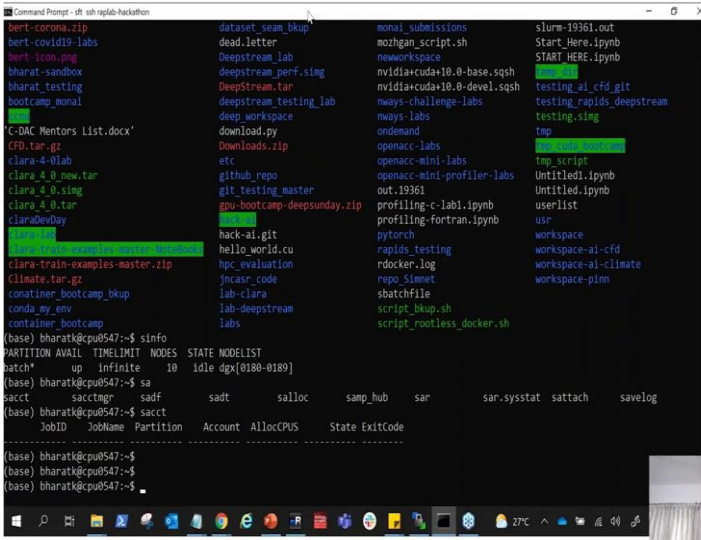


There are certain commands which are required for you to ask and find more details about this overall thing. Like, there is a command called as sinfo; sinfo will tell you all

the resources which are there on your cluster so that you can get a feeling of how many resources are there, how many of them are free, how many of them are used and who is using it and all those things.

Similarly, you can check the jobs that you have submitted are they done or are they not done via commands like squeue and I am going to show you a view of the same. Can one of the admins basically confirm if you are able to see the terminal?

(Refer Slide Time: 11:17)



The screenshot shows a terminal window with the following content:

```

bert-corona.zip
bert-covid19-labs
bert-icon.png
bharat-sandbox
bharat_testing
bootcamp_monai
C-BAC Mentors List.docx
CFD.tar.gz
clara-4-0-lab
clara_4_0_new.tar
clara_4_0_sing
clara_4_0.tar
claraDevDay
claraTrain
clara-train-examples-master.zip
Climate.tar.gz
container_bootcamp_bkup
conda_env
container_bootcamp
dataset_scam_bkup
dead.letter
Deepstream_lab
deepstream_perf.sing
DeepStream.tar
deepstream_testing_lab
deep_workspace
Download.py
Downloads.zip
etc
github_repo
git_testing_master
gpu-bootcamp-deepsunday.zip
hack-ai.git
hello_world.cu
hpc_evaluation
jncasr_code
lab-clara
lab-deepstream
labs
monai_submissions
mozghan_script.sh
newworkspace
nvidia+cuda+10.0-base.sqsh
nvidia+cuda+10.0-devel.sqsh
nways-challenge-labs
nways-labs
ondemand
openacc-labs
openacc-mini-labs
openacc-mini-profiler-labs
out.19361
profiling-c-lab1.ipynb
profiling-fortran.ipynb
pytorch
rapids_testing
rdoctest.log
repo_Slmmet
sbatchfile
script_bkup.sh
script_rootless_docker.sh
slurm-19361.out
Start Here.ipynb
START_HERE.ipynb
testing_ai_cfd_git
testing_rapids_deepstream
testing.sing
tmp
tmp_script
Untitled1.ipynb
Untitled.ipynb
userlist
usr
workspace
workspace-ai-cfd
workspace-ai-climate
workspace-pinn

```

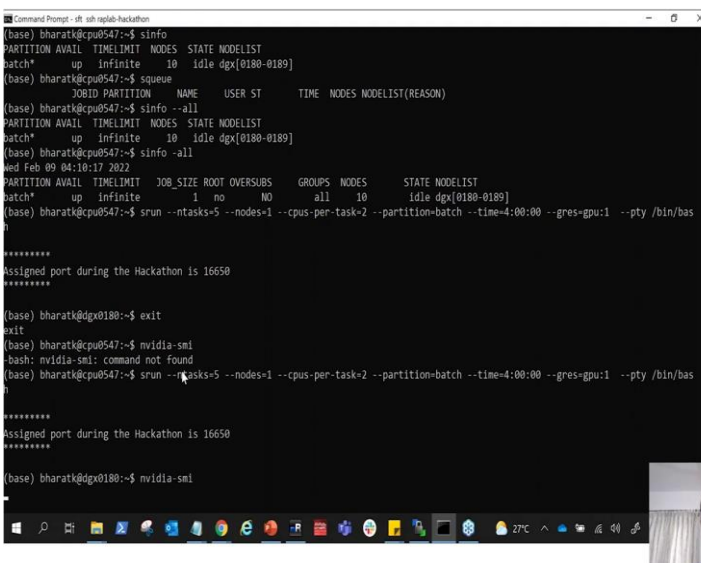
Below the file listings, the user runs the following commands and receives the output:

```

(base) bharatk@cpu0547:~$ sinfo
PARTITION AVAIL TIMELIMIT NODS STATE NODELIST
batch* up infinite 10 idle dgx[0180-0189]
(base) bharatk@cpu0547:~$ sa
sacct sacctmgr sadf sadt salloc samp_hub sar sar.sysstat sattach savelog
(base) bharatk@cpu0547:~$ squeue
JobID JobName Partition Account AllocCPUS State ExitCode
-----

```

(Refer Slide Time: 11:20)



The screenshot shows a terminal window with the following content:

```

(base) bharatk@cpu0547:~$ sinfo
PARTITION AVAIL TIMELIMIT NODS STATE NODELIST
batch* up infinite 10 idle dgx[0180-0189]
(base) bharatk@cpu0547:~$ squeue
JobID Partition Name User ST Time NODS NODELIST(REASON)
-----
(base) bharatk@cpu0547:~$ sinfo --all
PARTITION AVAIL TIMELIMIT NODS STATE NODELIST
batch* up infinite 10 idle dgx[0180-0189]
(base) bharatk@cpu0547:~$ sinfo -all
Wed Feb 09 04:10:17 2022
PARTITION AVAIL TIMELIMIT JOB_SIZE ROOT OVERSUBS GROUPS NODS STATE NODELIST
batch* up infinite 1 no NO all 10 idle dgx[0180-0189]
(base) bharatk@cpu0547:~$ srun --tasks=5 --nodes=1 --cpus-per-task=2 --partition=batch --time=4:00:00 --gres=gpu:1 --pty /bin/bash
*****
Assigned port during the Hackathon is 16650
*****
(base) bharatk@dgx0180:~$ exit
exit
(base) bharatk@cpu0547:~$ nvidia-smi
-bash: nvidia-smi: command not found
(base) bharatk@cpu0547:~$ srun --tasks=5 --nodes=1 --cpus-per-task=2 --partition=batch --time=4:00:00 --gres=gpu:1 --pty /bin/bash
*****
Assigned port during the Hackathon is 16650
*****
(base) bharatk@dgx0180:~$ nvidia-smi

```

Yes, Bharat, it is visible.

Ok. Thank you. So, I hope you are able to see this screen and I am basically logged in to a cluster. This is our internal cluster one of the requirements for you even though we are showing a demo is that in case you would like to do a hands-on on the system you should have access to one of these clusters. And, there are various clusters which are part of nsm and all or if you are if your college has one of the cluster you can do the same demo there as well.

So, the first thing I am going to check is running a command called as `sinfo`. You can see here this `sinfo` has given me details about my number of nodes. So, you can see here it says that I have nodes the node name is `dgx 0180` to `0189` and there are 10 nodes. And, all of these 10 nodes are idle at this point of time which means there is literally no work going on this machine.

How can I check it? I can again do `squeue` and you can see here that I see that there is literally no job which is running at this point of time in the cluster. So, I can also do. So, it is I am privileged not to see all of these things, but you can get more information about it like what is the time limit and they were up from how much time, do you have do I have the root privilege and all.

But, let me give you an example of how I can actually run a particular job and ask for the details of this particular node. So, what I am going to do is that, this slides will be available to you in case you want to do that, but I am going to show you a demo of how I can ask for a node and run some tasks, right.

So, you can see here I am saying `srun`. `srun` is a way for me to access and ask for a node how many nodes I am asking? I am asking that I need only one node at this particular point of time and I am need only 5 cores or I am going to run only 5 parallel tasks in it, so, I need only 5 and then I am also seeing other things. So, you can skip this particular part.

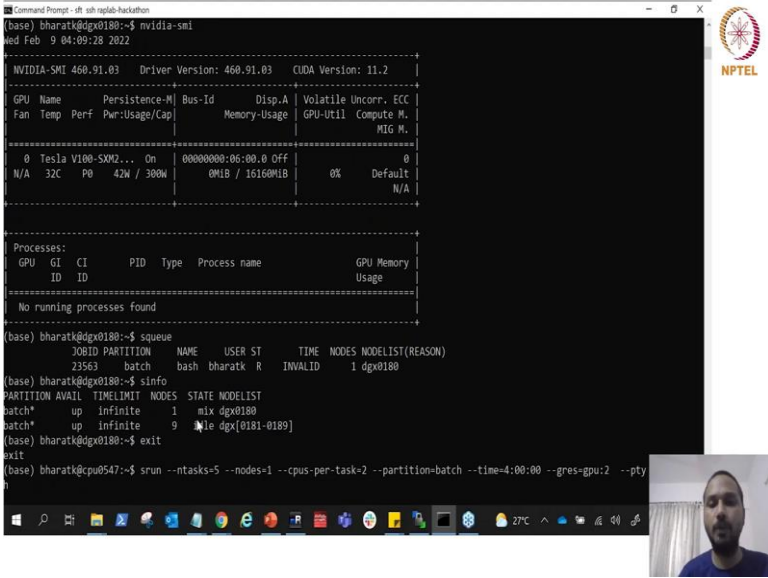
But, then I am also seeing that I believe my work will get over in 4 hours, so, I need it only for 4 hours. I am also mentioning it that I need one gpu because I am going to run a accelerated computing job and I need a GPU to make sure my network kind of trains much more faster.

So, I am asking it for one node having 5 cores and then also the time for which I needed and also the number of GPUs I needed and I am saying that I need a basically a terminal which can be opened. So, once I enter this basically you can see here that I moved from my login node.

One thing which I forgot to tell you is that you can see here this is the login node and when I run this command I have moved away from the login node to this node which is dgx 0180 which was my compute node. So, let me exit once more and show you that see my login node might not even have a GPU.

If I do nvidia – smi which is the command to find out if I have any GPUs or not you can see here there is no GPU which is available and I should not run anything on the login node.

(Refer Slide Time: 14:55)



```

Command Prompt - ssh-nvidia-hackathon
(base) bharatk@dgx0180:~$ nvidia-smi
Wed Feb  9 04:09:28 2022

+-----+
| NVIDIA-SMI 460.91.03   | Driver Version: 460.91.03   | CUDA Version: 11.2   |
+-----+-----+
| GPU Name               | Persistence-M| Bus-Id        | Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|  Memory Usage | GPU-Util  Compute M. |
|                           |               | Memory       |          |                       |
+-----+-----+
| 0 Tesla V100-SXM2...  | On          | 00000000:06:00:0 Off |   0    |          0%   Default |
| N/A   32C   P0   42W / 300W |  0MiB / 16160MiB |              |         | MIG M. |
+-----+-----+

Processes:
+-----+
| GPU  GI  CI       PID   Type   Process name          | GPU Memory |
| ID   ID  ID               |            | Usage         |
+-----+
| No running processes found |
+-----+

(base) bharatk@dgx0180:~$ squeue
JOBID PARTITION NAME USER ST TIME NODES MODELLIST(REASON)
23563 batch bash bharatk R INVALID 1 dgx0180

(base) bharatk@dgx0180:~$ sinfo
PARTITION AVAIL TIMELIMIT NNODES STATE MODELLIST
batch* up infinite 1 mix dgx0180
batch* up infinite 9 file dgx[0181-0189]

(base) bharatk@dgx0180:~$ exit
exit
(base) bharatk@cpu0547:~$ srun --ntasks=5 --nodes=1 --cpus-per-task=2 --partition=batch --time=4:00:00 --gres=gpu:2 --pty

```

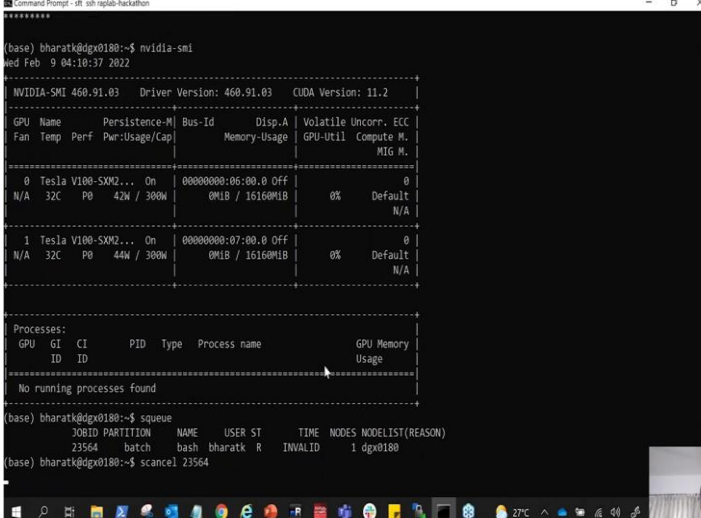
So, I am going to ask it for resources and then if I do nvidia – smi I can basically see that now on that node I have asked for one gpu and I am basically getting one nvidia Tesla V100 card which is given to me at this time, right.

And if I do a squeue also you would see that it is telling me that I am basically running one particular job and that job has been allocated on dgx 0180, right. So, it will give me all of those details and now, you can see earlier all of my machines were empty, but if you see now sinfo you can clearly see that now only 9 machines are empty and one of

this machine is actually in mixed state which means some of the part of this machine is used while some of this part is still available for me to be given to some other resources.

So, this is how you can see. In fact, let me just give you another example and ask for two GPUs instead of one GPU.

(Refer Slide Time: 16:01)



```
(base) bharatk@dgx0180:~$ nvidia-smi
Wed Feb 9 04:10:37 2022

+-----+
| NVIDIA-SMI 460.91.03   Driver Version: 460.91.03   CUDA Version: 11.2   |
+-----+-----+
| GPU Name               Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
|                               |              | MIG M.   |
+-----+-----+
| 0  Tesla V100-SXM2...  On          | 00000000:06:00:0 Off |           0         |
| N/A   32C   P0   42W / 300W |  0MiB / 16160MiB |      0%   Default   |
+-----+-----+
| 1  Tesla V100-SXM2...  On          | 00000000:07:00:0 Off |           0         |
| N/A   32C   P0   44W / 300W |  0MiB / 16160MiB |      0%   Default   |
+-----+-----+

Processes:
+-----+-----+
| GPU  GI  CI           PID  Type  Process name          GPU Memory |
| ID   ID  ID           |              |           | Usage        |
+-----+-----+
| No running processes found |
+-----+-----+

(base) bharatk@dgx0180:~$ squeue
JOBID PARTITION NAME USER ST TIME NODES MODELIST(REASON)
23564 batch bash bharatk R INVALID 1 dgx0180

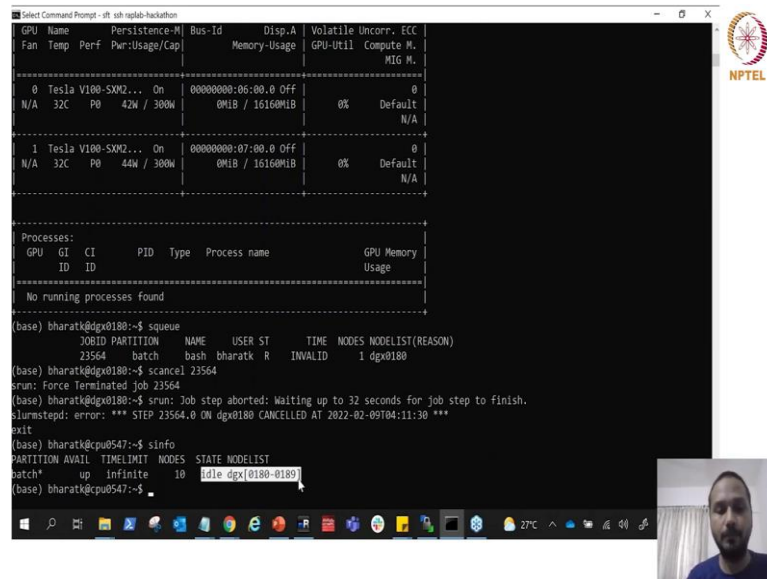
(base) bharatk@dgx0180:~$ scancel 23564
```

And, if I am lucky I might get that in case the resources are available and I saw that there are resources which are available.

Now, you can see here that again I am in the same node which is dgx0180. Earlier when I did nvidia – smi I saw only one GPU, but now I asked the resources to give me two GPUs and suddenly I start seeing two GPUs. Now, I can run a job which can run across two GPUs or in parallel and this is how the overall basically cluster ease works, right.

So, this is how multiple users can actually access and ask for more and more resources. Today, I have ten nodes, tomorrow I can add two more nodes and then it will just add it to my existing list and I can also cancel the job. Like you can see here there is squeue I can do scancel and give the ID which is assigned to that particular job and it will basically cancel that job.

(Refer Slide Time: 16:59)



The terminal window shows the output of the `nvidia-smi` command, displaying GPU status for two Tesla V100-SXM2 GPUs. Below this, the `squeue` command shows a job (ID 23564) in a 'Terminated' state. The `scancel 23564` command is used to cancel the job. Finally, the `sinfo` command shows the available resources, including a node with 10 idle GPUs (dgp[0180-0189]).

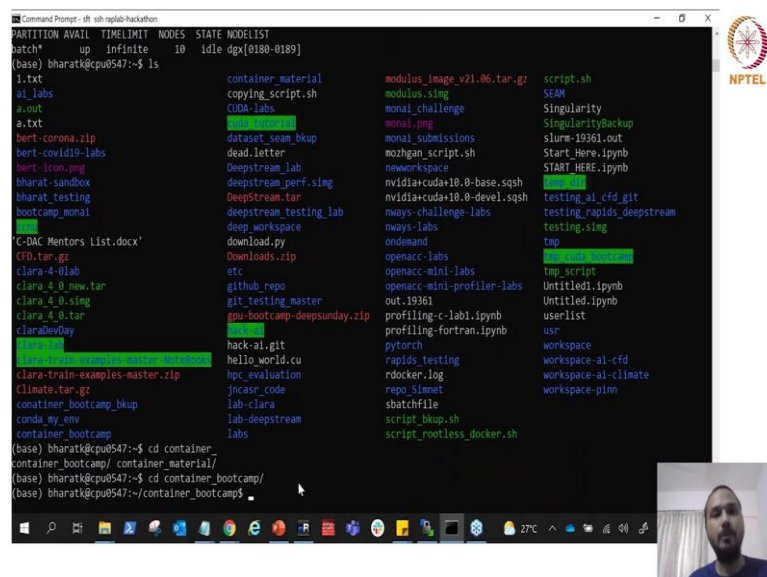
```
Select Command Prompt - ssh-nslab-hackathon
GPU Name Persistence-M Bus-Id Disp.A Volatile Uncorr. ECC
Fan Temp Perf Pwr:Usage/Cap Memory-Usage GPU-Util Compute M.
-----
0 Tesla V100-SXM2... On 00000000:06:00.0 Off 0% Default 0
N/A 32C P0 42W / 300W 0MiB / 16160MiB
-----
1 Tesla V100-SXM2... On 00000000:07:00.0 Off 0% Default 0
N/A 32C P0 44W / 300W 0MiB / 16160MiB
-----

Processes:
GPU GI CI PID Type Process name GPU Memory
ID ID
-----
No running processes found

(base) bharkat@dgp0180:~$ squeue
JOBID PARTITION NAME USER ST TIME NODES MODELIST(REASON)
-----
23564 batch bash bharkat R INVALID 1 dgp0180
(base) bharkat@dgp0180:~$ scancel 23564
srun: Force Terminated job 23564
(base) bharkat@dgp0180:~$ sinfo
PARTITION AVAIL TIMELIMIT NODES STATE MODELIST
-----
batch* up infinite 10 idle dgp[0180-0189]
(base) bharkat@dgp0180:~$
```

And, you can see as I cancel this job I came out of the compute node and I am back on my CPU node and there is no job which is running and all of the machines are again free for somebody else to use. Now, you can see here that how easy it was for me to basically just use this resources and in keep on increasing this resources and I can submit a job as much as I need it for being accessible, right .

(Refer Slide Time: 17:33)

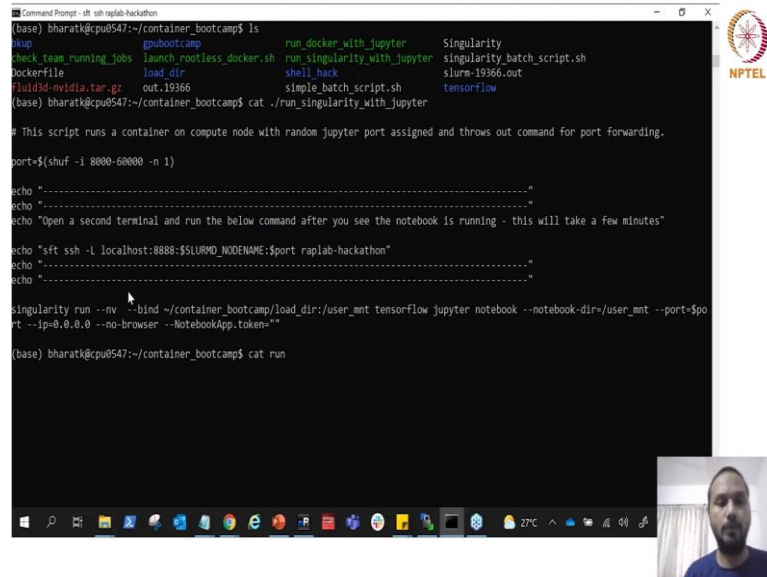


The terminal window shows the output of the `ls` command, listing files and directories in the current directory. The user then navigates to the `container_bootcamp` directory using the `cd` command.

```
Command Prompt - ssh-nslab-hackathon
PARTITION AVAIL TIMELIMIT NODES STATE MODELIST
-----
batch* up infinite 10 idle dgp[0180-0189]
(base) bharkat@dgp0180:~$ ls
1.txt container_material modulus_image.v21.06.tar.gz script.sh
ai_labs copying_script.sh modulus.sing SEAM
a.out CUDA-labs monai_challenge Singularity
a.txt singularity SingularityBackup
bert-corona.zip dataset_seam_bkup monai_submissions slurm-19361.out
bert-covid19-labs dead.letter mozhan_script.sh Start_Here.ipynb
bert-icon.png deepstream_lab newWorkspace START_HERE.ipynb
bharat_sandbox deepstream_perf.sing nvidia_cuda10.0-base.sqsh workspace
bharat_testing DeepStream.tar nvidia_cuda10.0-devel.sqsh testing_ai_cfd.git
bootcamp_monai deepstream_testing_lab nways-challenge-labs testing_rapids_deepstream
C-DAC Mentors List.docx deep workspace testing.sing
CFD.tar.gz download.py ondemand tmp
clara-4-0lab Downloads.zip openacc-labs tmp_script
clara-4-0_new.tar etc openacc-mini-labs tmp_script
clara_4_0.sing github_repo openacc-mini-profiler-labs Untitled1.ipynb
clara_4_0.tar git_testing_master out.19361 Untitled1.ipynb
claraDevDay gpu-bootcamp-deepsunday.zip profiling-c-lab1.ipynb userlist
claraDevDay profiling-fortran.ipynb workspace
clara_train_examples-master.zip hark-al.git workspace-ai-cfd
climate.tar.gz hello_world.cu rdocker.log workspace-ai-climate
conatiner_bootcamp_bkup jncsr_code repo_Simmet workspace-pinn
conda_my_env lab-clara sbatchfile
container_bootcamp lab-deepstream script_bkup.sh
container_bootcamp labs script_rootless_docker.sh
(base) bharkat@dgp0180:~$ cd container-
container_bootcamp/ container_material/
(base) bharkat@dgp0180:~$ cd container_bootcamp/
(base) bharkat@dgp0180:~/container_bootcamp$
```

And, let me also show you another example of how I put on this cluster also run a container job.

(Refer Slide Time: 17:42)



```
Command Prompt - ssh raplab-hackathon
(base) bharatk@cpu0547:~/container_bootcamp$ ls
singularity
check_team_running_jobs launch_rootless_docker.sh run_docker_with_jupyter singularity hatch_script.sh
dockerfile load_dir run_singularity_with_jupyter singularity hatch_script.sh
fluid3d-nvidia.tar.gz out_19366 shell hack slurm-19366.out
(base) bharatk@cpu0547:~/container_bootcamp$ cat run

# This script runs a container on compute node with random jupyter port assigned and throws out command for port forwarding.

port=$(shuf -i 8000-60000 -n 1)

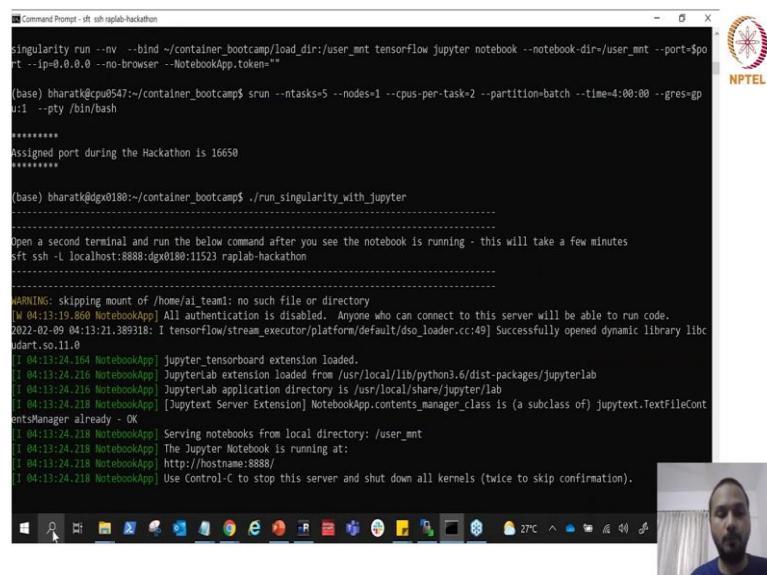
echo "-----"
echo "-----"
echo "Open a second terminal and run the below command after you see the notebook is running - this will take a few minutes"
echo "-----"
echo "sft ssh -L localhost:8888:$SLURMD_NODENAME:$port raplab-hackathon"
echo "-----"
echo "-----"

singularity run --nv --bind ~/container_bootcamp/load_dir:/user_mnt tensorflow jupyter notebook --notebook-dir=/user_mnt --port=$port --ip=0.0.0.0 --no-browser --NotebookApp.token=""

(base) bharatk@cpu0547:~/container_bootcamp$ cat run
```

So, I am going to run a container job I already have a script you already had a session on how to run containers. So, I am not going to run a container I am not going to show you that command, but the script will be provided to you in case you want it to run, but I am running a container job and that container job is going to basically give me certain resources.

(Refer Slide Time: 18:22)



```
Command Prompt - ssh raplab-hackathon
singularity run --nv --bind ~/container_bootcamp/load_dir:/user_mnt tensorflow jupyter notebook --notebook-dir=/user_mnt --port=$port --ip=0.0.0.0 --no-browser --NotebookApp.token=""

(base) bharatk@cpu0547:~/container_bootcamp$ srun --ntasks=5 --nodes=1 --cpus-per-task=2 --partition=batch --time=4:00:00 --gres=gpu:1 --pty /bin/bash

*****
Assigned port during the Hackathon is 16650
*****

(base) bharatk@gpx0180:~/container_bootcamp$ ./run_singularity_with_jupyter

Open a second terminal and run the below command after you see the notebook is running - this will take a few minutes
sft ssh -L localhost:8888:gpx0180:11523 raplab-hackathon

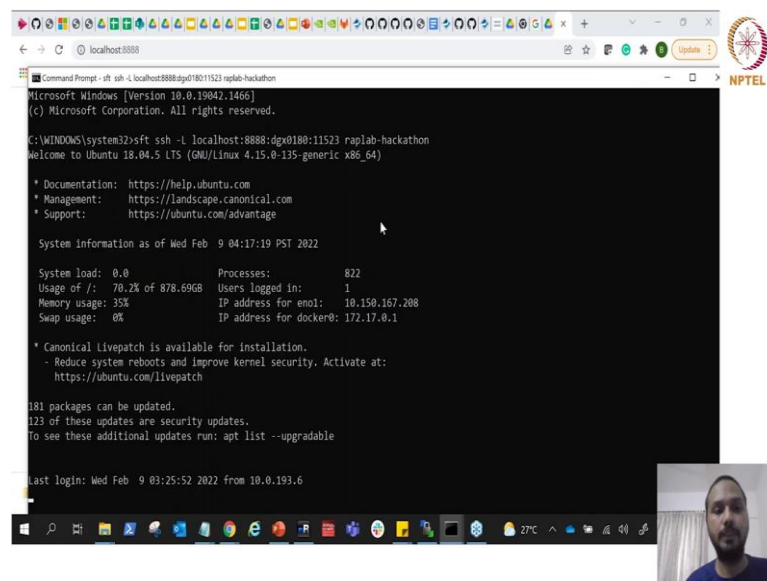
*****
WARNING: skipping mount of /home/ai_team1: no such file or directory
[04:13:19.880 NotebookApp] All authentication is disabled. Anyone who can connect to this server will be able to run code.
[04:13:21.38918: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudart.so.11.0
[04:13:24.164 NotebookApp] Jupyter TensorBoard extension loaded.
[04:13:24.216 NotebookApp] JupyterLab extension loaded from /usr/local/lib/python3.6/dist-packages/jupyterlab
[04:13:24.216 NotebookApp] JupyterLab application directory is /usr/local/share/jupyter/lab
[04:13:24.218 NotebookApp] [Jupyter Server Extension] NotebookApp.contents_manager_class is a subclass of jupyter.TextFileContentsManager already - OK
[04:13:24.218 NotebookApp] Serving notebooks from local directory: /user_mnt
[04:13:24.218 NotebookApp] The Jupyter Notebook is running at:
[04:13:24.218 NotebookApp] http://hostname:8888/
[04:13:24.218 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```

Oh, I am sorry. This is my mistake. I need to first allocate a resource for myself. So, the first thing I am doing is I am getting a GPU node. You can see here that I am getting a

GPU node which is dgx0180. Now, I am going to submit the job of running the container for making sure that I get access to the node. I have got access to the node you can see here I am in dgx machine which is having the GPU and I am running a container and let me just do one more step.

This step is something which is required for port forwarding.

(Refer Slide Time: 19:04)



```
Microsoft Windows [Version 10.0.19042.1466]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ssh -l localhost:8888-dgx0180:11523 raplab-hackathon
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-135-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Wed Feb  9 04:17:19 PST 2022

System load:  0.0               Processes:            822
Usage of /:   70.2% of 878.69GB  Users logged in:     1
Memory usage: 35%              IP address for eno1:  10.150.167.208
Swap usage:   0%               IP address for docker0: 172.17.0.1

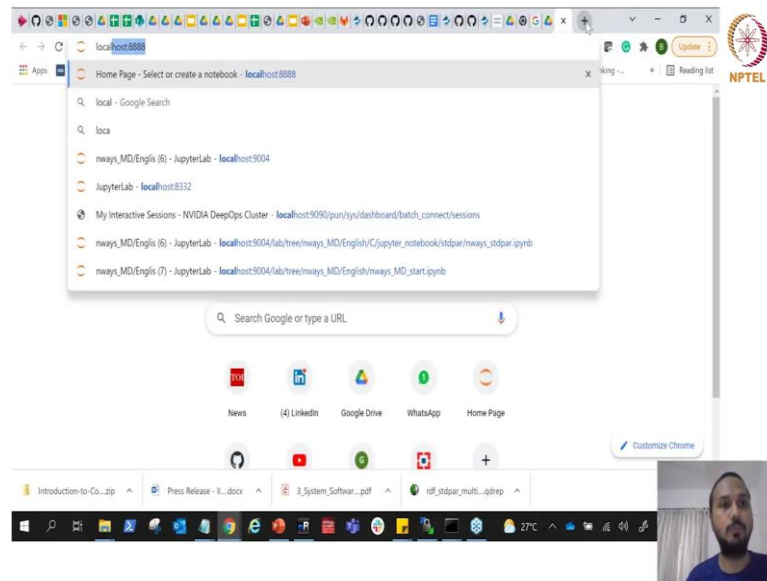
 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

181 packages can be updated.
123 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Last login: Wed Feb  9 03:25:52 2022 from 10.0.193.6
```

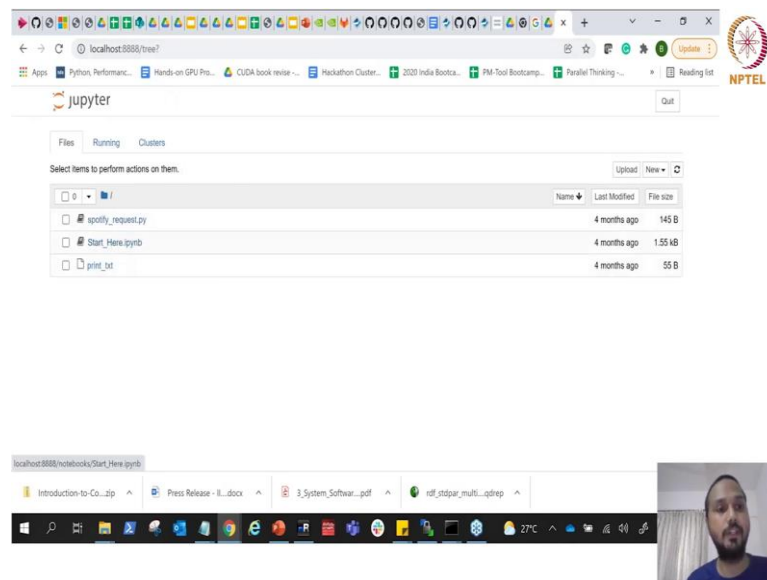
For now, you can do not need to worry about this step and you can forget about what it means and so, I just ran for particular container. And, in this particular container I am going to basically run a TensorFlow job.

(Refer Slide Time: 19:25)



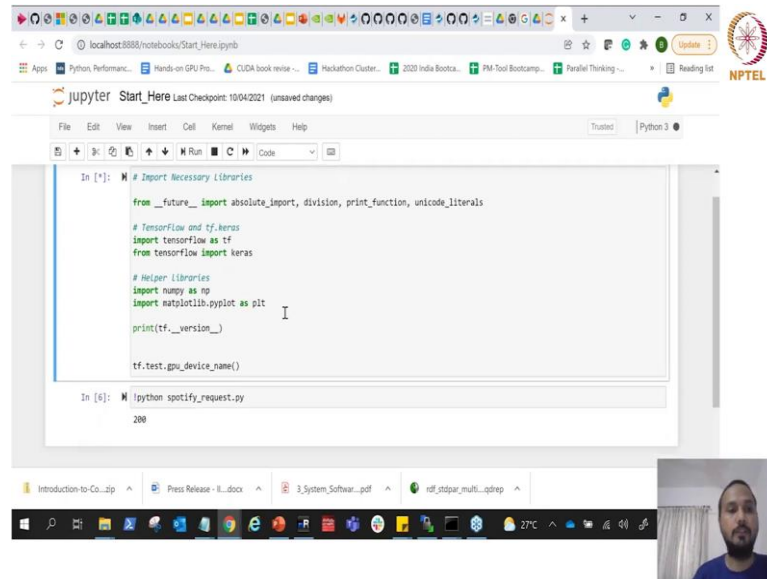
So, let me open my Jupyter notebook it was showed as a demo to you hopefully I have got the resources and I am able to like not yet. Yes.

(Refer Slide Time: 19:44)



So, you can see here that after I launched the job in the container I am running this Jupyter notebook inside the container notebook and I can now start doing anything in this particular Jupyter notebook which is going to run on the machine.

(Refer Slide Time: 19:57)



```
In [1]: # Import Necessary Libraries
from __future__ import absolute_import, division, print_function, unicode_literals

# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)

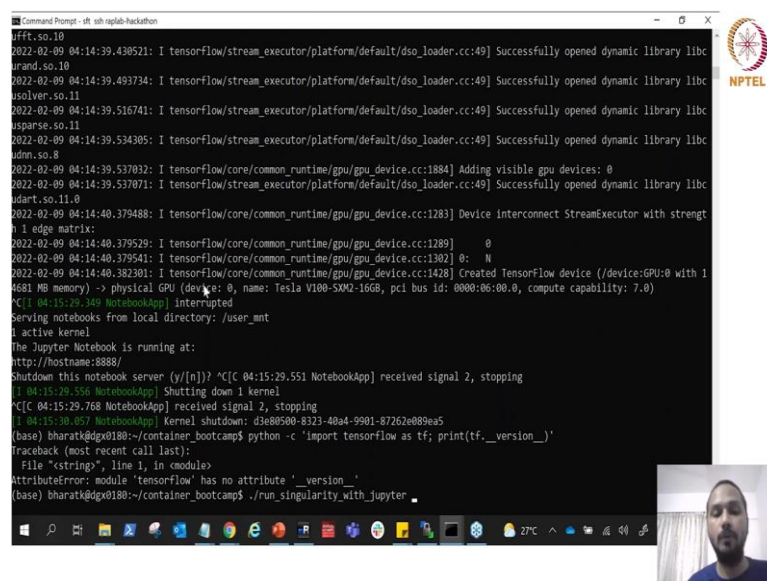
tf.test.gpu_device_name()

In [6]: !python spotify_request.py
200
```

Like, this particular container is a TensorFlow container and I can just run the TensorFlow here. You can see here that I am importing TensorFlow as tf. I am using keras and then I am asking for the TensorFlow version which is 2.3.1 and I am asking if I have a GPU with me or not inside this container and you can see here it says that I have a GPU allocated to this particular container.

And, you can see here that I am able to do this.

(Refer Slide Time: 20:34)



```
Command Prompt - C:\Users\Bharat> docker run --rm -it singularity-hackathon
jfft:50.10
2022-02-09 04:14:39.438521: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libc
brand.so.10
2022-02-09 04:14:39.493734: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libc
solver.so.11
2022-02-09 04:14:39.516741: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libc
sparse.so.11
2022-02-09 04:14:39.534385: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libc
udnn.so.8
2022-02-09 04:14:39.537032: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1884] Adding visible gpu devices: 0
2022-02-09 04:14:39.537071: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libc
udart.so.11.0
2022-02-09 04:14:40.379488: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1283] Device Interconnect StreamExecutor with strengt
n 1 edge matrix:
2022-02-09 04:14:40.378529: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1289] 0
2022-02-09 04:14:40.378541: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1302] 0: N
2022-02-09 04:14:40.382301: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1428] Created TensorFlow device (/device:GPU:0 with 1
4681 MB memory) -> physical GPU (device: 0, name: Tesla V100-SXM2-16GB, pci bus id: 0000:06:00.0, compute capability: 7.0)
^C[I 04:15:29.349 NotebookApp] Interrupted
Serving notebooks from local directory: /user_mnt
1 active kernel
The Jupyter Notebook is running at:
http://hostname:8888/
Shut down this notebook server (y/[n])? ^C[I 04:15:29.551 NotebookApp] received signal 2, stopping
[I 04:15:29.556 NotebookApp] Shutting down 1 kernel
^C[I 04:15:29.768 NotebookApp] received signal 2, stopping
[I 04:15:30.657 NotebookApp] Kernel shutdown: d3e80580-8323-48a4-9901-87262e089e5
(base) bharatk@gx0180:~/container_bootcamp$ python -c 'import tensorflow as tf; print(tf.__version__)'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
AttributeError: module 'tensorflow' has no attribute '_version_'
(base) bharatk@gx0180:~/container_bootcamp$ ./run_singularity_with_jupyter
```



So, again just to repeat I showed you a demo where I went to the GPU and after that I ran the container on the gpu node and it was I hope I ran within the container the Jupyter notebook and I could run this Jupyter notebook and run TensorFlow inside it. Now, some of you were asking in the last lecture on how does it matter, whether I need to do any installation or let me just show you one part.

On this machine on this, sorry about this. On this machine there is no there is no more TensorFlow installed. There is practically no TensorFlow which is there on this machine I am not done any kind of installation. In fact, it says that module TensorFlow has no attribute called as version. So, it is not having any installation of TensorFlow on this machine.

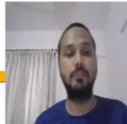
In order for me without to do any installation I just imported a container and I started running it in form of a container and I started accessing it on my Jupyter notebook. So, this is the advantage so that you can keep your cluster clean. I basically do not have to install TensorFlow Keras or any other software. I can just have containers on my machine and I can just give it to the cluster to run and keep on accessing it.

And, tomorrow if any other if I want to transfer it from this cluster to the cloud which is the example which I showed of cloud bursting I can take this container and deploy on the cloud of Amazon and all and just start running it there without having to do any kind of a installation and that is the advantage that you get with the container environment.

(Refer Slide Time: 22:21)

**Demo**

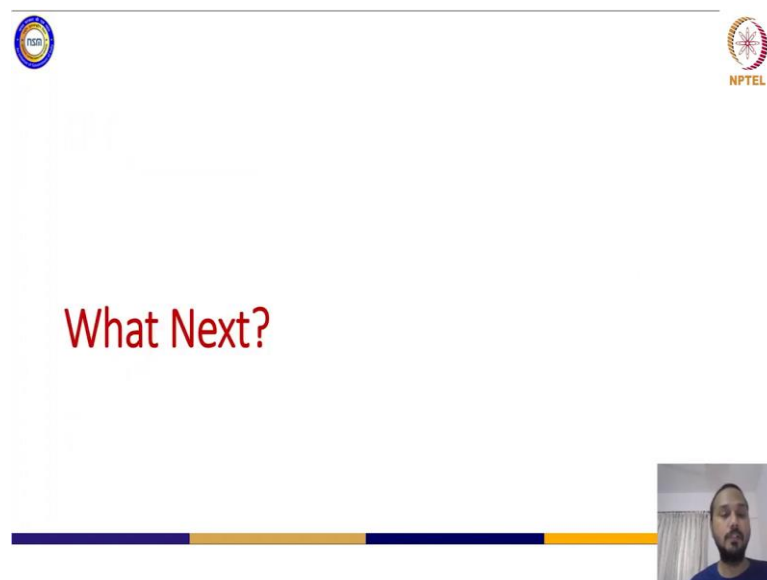
- Check the nodes status
 - `$sinfo`
- Submit an interactive job
 - `$srun --ntasks=5 --nodes=1 --cpus-per-task=2 --partition=batch --time=4:00:00 --gres=gpu:1 --pty /bin/bash`
 - `$nvidia-smi`
 - `$exit`
- Cheque status of the running jobs
 - `$squeue`
- Cancel the submitted job
 - `$scancel <Job Id>`



So, this was a quick demo in terms of what a cluster is, what is the advantage of using a cluster, how can you ask for resources, how you can see how many resources are free, how you can cancel. Once you have got the resources you can run a container which you saw in the previous lecture and then open Jupyter notebooks or whatever you want to do.

Have a very clean installation, not having messed up my system which is a cluster via the containers and can deploy it anywhere on the cluster on my machine on the cloud and keep it as clean as possible. Now, this is very important especially if I talk about enterprise computing or if you are trying to run all of these jobs where you need what quality of service.



(Refer Slide Time: 23:04)



So, far what you have seen was the part that we have looked at was the need for accelerated computing, the need for software stacks like virtualization, the need for containers that we talked about the need for not just one system, but multiple systems connected to each other in form of a cluster and making things happen.


But, what I am going to talk about now is kind of a teaser for the upcoming lectures which will be delivered in the next set of in this particular series, right.

(Refer Slide Time: 23:42)



Container orchestration for clusters

- **Resource limit control.** This feature reserves the CPU and memory for a container, which restrains interference among containers and provides information for scheduling decisions;
- **Scheduling.** Determine the policies on how to optimise the placement of containers on specific nodes;
- **Load balancing.** Distribute the load among container instances;
- **Health check.** Verify if a faulty container needs to be destroyed or replaced;
- **Fault tolerance.** Create containers automatically if applications or nodes fail;
- **Auto-scaling.** Add or remove containers automatically.



So, when I showed you in code an example what I did was I took the resources via the scheduler and then I started looking I just started running it and ran the Jupyter notebook, right. So, what it did what it did some amount of reservation for your CPU and memory, right.

But, as I said previously also that the containers had certain restraints so, you need more limit controls. You need better scheduling. You need better policies of scheduling which is there like Slurm already provides the scheduling policies like fair share and all those other kinds of scheduling.

What you might also require is some kind of a load balancing right because you might not have in instance where you are putting a lot of load on only one node or one part of your cluster while the other cluster is almost free. So, you need to distribute the load among the container instances also.

You need to do certain amount of health checking. Like if there is any problem or if there is a faulty container that needs to be destroyed or replaced. So, certain amount of health checking. What is the fault tolerance? Right. If the you can create container automatically if a particular node fails. If you see even if one particular node of Google or when you do search fails does not mean that your, the whole network is down, right. You will still be able to do that.

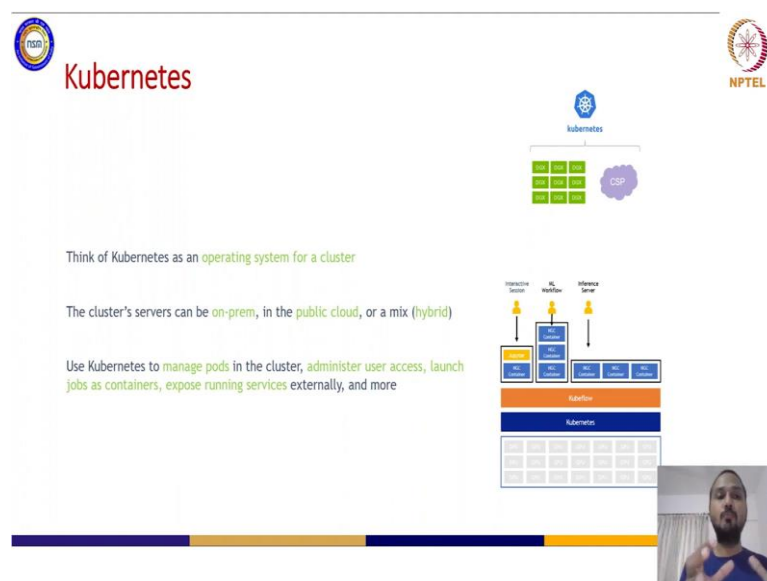
What it does it shifts that particular instance to another node and that happens behind the scenes for you without you having to worry about this fault tolerance part, but this is something which happens behind the scene. Also you can scale it automatically right like a container can actually can be added or removed automatically. So, you can do auto scaling.

So, if I talk about a particular environment in AI, I am not just talking about scheduling. Scheduling is one small part of the real world environment where all of these things happen. You need to do load balancing, you need to do health check, you need to do fall tolerance, you need to do auto scaling, you need to do resource limit control. Now, all of this is happens via this tool called as orchestration.

So, what you are trying to do is not just schedule, but you are trying to orchestrate to make it really having a quality of service and having a lot of automation and on demand access like you get in the cloud, right. That is what you see in the cloud without you having to know about all of this. Everything behind the scenes is running when you do any search.

There are machine learning and deep learning algorithms running on the Google when you do any search or anything like that, right. Or if whatever translation like if I say something it translates from one language to the other, all that is happening behind the scene in this particular cluster environment for you. What we are trying to do in this series is to expose you on what those things are.

(Refer Slide Time: 26:45)



Now, that overall thing is called as orchestration, which means you are trying to orchestrate all of those features and making sure you are following all of those principles and optimally utilizing the resources.

One of the orchestration layer the most famous one out there is called as Kubernetes. Kubernetes you can think of it as an operating system for your cluster. Currently, you have heard this term operating system where you have a single machine and you are trying to access it. Now, for a cluster you can think of Kubernetes which is doing many of the things that operating system does, but for all the resources which are across the nodes and not just on one particular machine.

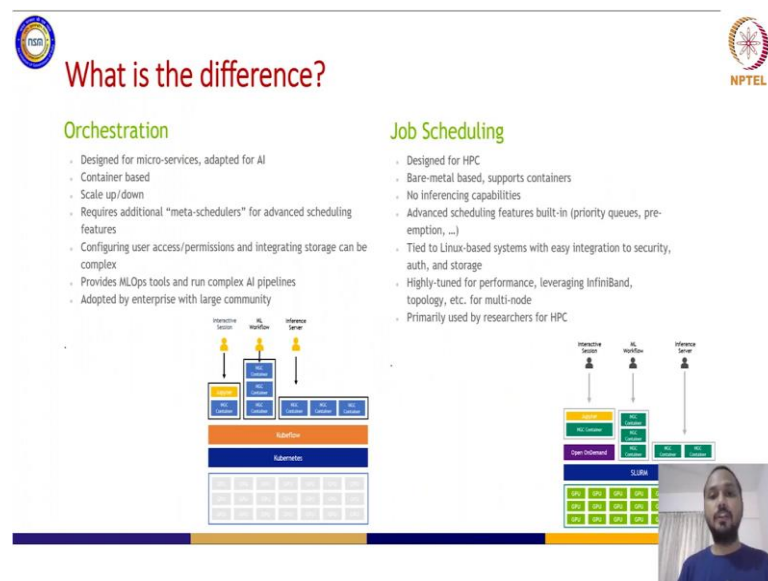
You can think of it as a higher level operating system which takes into consideration all of the resources of your cluster. These clusters can be on premise. We looked at one example which is in your premise. It can be a laptop also as one of the component of that cluster, it can be in the cloud or it can be hybrid we looked at the cloud bursting part, right.

So, Kubernetes as an orchestration good look that if it is happening on premise if you have resources you can keep it here, but if you are exiting the resources it can do a burst to the cloud and run the same thing on the cloud also without you having to worry about it. So, it can take this particular complex scenarios that we talked about sometime back.

It can be helped to manage all of these clusters, administer user access, launch containers and expose all of this as services.

Now, that is the part which is used and very heavily deployed in any of the installations that you see of artificial intelligence. So, it is more than a cluster it is not just about scheduling and it is much more than job scheduling, it is about orchestration.

(Refer Slide Time: 28:39)



Traditionally, the cluster has been very heavily tied down to high performance computing applications like molecular dynamics, computational fluid dynamics and all while orchestration is still very much relevant to the artificial intelligence space and it is end especially adopted by the enterprise segment with large across all of them. It provides you and helps you in deploying very complex AI pipelines starting from training to influencing.

It provides you meta-schedulers where you can define your own scale like Slurm can be one of the scheduler which is plugged to Kubernetes to do fair scheduling. You can scale up and scale down and all those features are part of a orchestration while a simple job scheduler can do only few things like it gives you bare metal performance, but it does not have this capability of scale up or scale down.

It is tied down and it is very highly tuned for leveraging the best performance which is primarily used by the researchers working in the high performance computing domain.

So, in the next set of lectures what you are going to see is how to use Kubernetes and how this AI behind the scene works and whatever you do what happens on there. With that I am done with the lecture which I had.