Who, What & How of NEF

STARS Team, INRIA Sophia

Brace Yourself

- There is a lot of important information in this presentation.
- It took a lot of time and energy in creating this presentation.
- Please respect that.
- Please refer to this presentation before deciding to come for personal help.
- We assist. We do not serve. Know the difference.
- Be willing to learn. That is the best weapon against all problems.

What is NEF ?

A collection of high-performance computers which can be utilized for large-scale or complicated CPU/GPU calculations.

Perpetual reminder to yourself

- NEF is a **shared** resource : It is to be used by everyone.
- NEF is a **limited** resource : It has limited CPU/GPU/memory resources.

Getting an account

- Fill out the form at this link with necessary information to get an account.
  - The form asks for you to provide your ssh Key
  - See if you have a file named *id_rsa.pub* in your ~/.ssh directory.
  - If you have, just upload that file.
  - If you do not have it (highly unlikely !!), then generate one by using information on this link.
  - Once, you upload that file, do not change the key or its filename ever.
  - For security reasons, every NEF account is associated with a set of ssh keys which basically identify the system you are using to access the NEF.
  - If you ever change your computer, you must request the admins to copy the ssh key of the new computer to your NEF account and delete the old key.

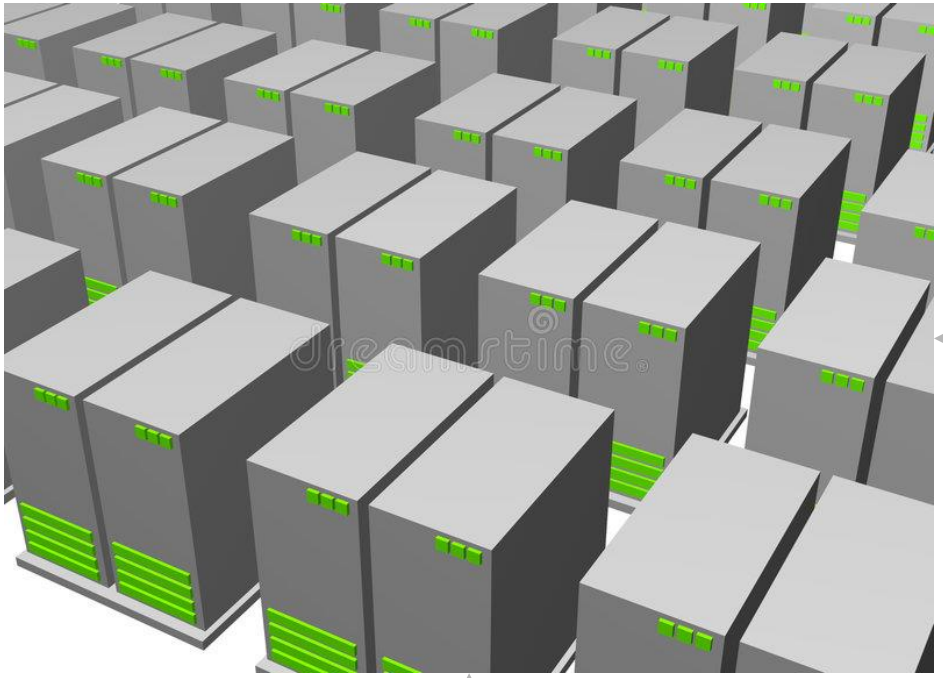
How to use NEF

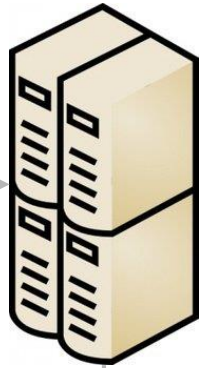
There are 2 components of NEF usage :

- **Data handling** : How to store data effectively and efficiently.
- **Resource utilization** : How to use resources in a judicious manner.

We review these two components one by one in the subsequent slides.
However first understand the structure of NEF.

NEF Structure.





You connect to nef-devel2

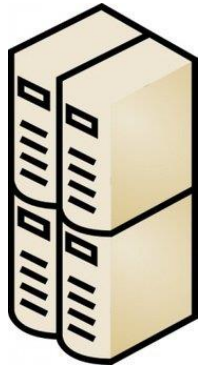
Nef-devel2 connects to compute nodes

You & your lab system

You cannot directly access the compute nodes

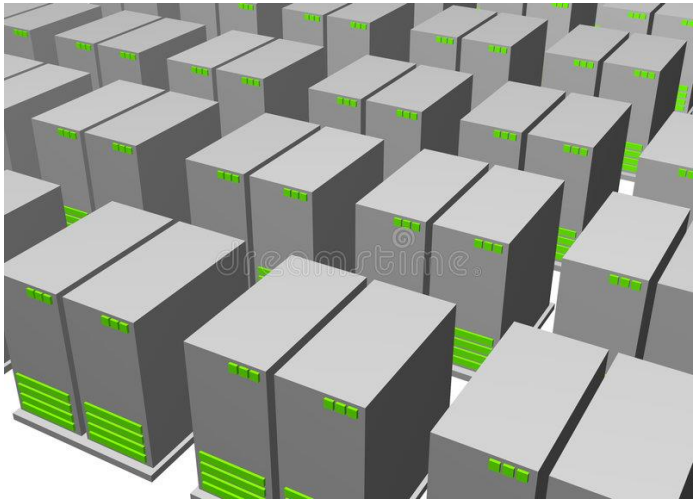
Nef-devel2

- It is just an access point to high-performance nodes.
- It is not good for computations.
  - You hurt others' jobs by running heavy computations on it.
- When you want to access high-performance nodes
  - There is only one way. To go through nef-devel2.



Compute Nodes

- They are here to serve your compute needs.
- They are a diverse family
  - Some of them don’t have GPUs.
  - Some of them have GPUs.
  - Actually most of them do not have GPUs.
- Use them for heavy computations.
- But to use us you must come through nef-devel2.
  - Sorry, but you cannot directly access compute nodes.



Data Handling in NEF

Basics of Folder structure in NEF

There are 4 separate types of folders in NEF

1. Home Folder (If you know linux, you should know this. If you don't you need to learn some linux first.)
2. Data Folder (/data/stars)
3. SSD Folder (/local/) (*It is available only on GPU Nodes*)
4. Temporary Folder (/tmp)

Home Folder (1 of 3)

- Every home folder has a quota.
  - In simple words :
    - How many files you can have in your home folder.
    - How much space you can use in your home folder.
  - This quota is enforced for efficient operation of the NEF cluster. **Please do not intellectualize this !!!**
  - To get the size of your quota use the following command :

```
(tensorflow) -bash-4.2$ cd
(tensorflow) -bash-4.2$ quota -s
Disk quotas for user uujjwal (uid 653079):
     Filesystem   space   quota   limit   grace   files   quota   limit   grace
nef-storage.ibnef:/export/home
                   110G    150G    600G           1105k  10000k  50000k
nef-storage.ibnef:/export/misc
                   110G    150G    600G           1105k  10000k  50000k
(tensorflow) -bash-4.2$
```

Home Folder (2 of 3)

- In the previous example
  - The space used is 110GB while the quota is 150GB (maximum usable space) and limit is 600GB.
  - If I use 151GB, the grace column would become active.
  - Grace refers to a period (21 days), which is given to you for reducing your usage to under 150GB. It is like a counter. If you do not fall in line, then after 21 days, admins would delete the most recent data so that your usage goes under 150GB.
  - Under the limits of grace period I can use a maximum of 600GB in home folder. More than that is not possible.
- The same arguments as above apply to the number of files as well.
- Remember that restrictions on number of files and space are mutually exclusive.
  - Exceeding any one of them would prevent you for storing or creating any new files in home folder.

Home Folder (3 of 3)

- It is a good idea to use home folder to :
    a. Store codes.
    b. Install anaconda.
    c. Store very small datasets (e.g : CIFAR10, CIFAR100, MNIST)

Data Folder (1 of 3)

- Data folder is located in /data/stars.
- It has two parts :
  - /data/stars/share
  - /data/stars/user
- The difference between these subfolders is purely utilitarian. There are no differences in terms of permissions or quotas.
- /data/stars/share should be used for storing datasets (as will be discussed later !!).
- /data/stars/user/<USERNAME> is a folder created by admins for you.
  - It is meant to store most of your experimental results and any large amounts of data you might need.

Data Folder (2 of 3)

- The quota of /data/stars is 18TB. This quota is shared by the entire STARS team.
  - So, please do not consider it a personal quota.
- Just to make a point :
  - If 3 STARS user end up storing a total of 16 TB of data in data folder, the rest of the STARS team has to share only 2 TB among themselves.
  - Generally during large-scale experiments it is not uncommon to see around 2-3 TB of data generated by one user. So, please keep in mind the quota of the STARS team when generating or storing data in the data folder.
- Later on we will see some other details about how to handle very large amounts of experimental data.

Data Folder (3 of 3)

Data folder is suitable for :

1. Storing datasets (More on this later !!)
2. Storing data generated during experiments.
3. Compiling very large libraries for personal use.
    a. Remember that compilation of a large library like Boost or OpenCV, generate a huge amount of intermediate data.

SSD Folder (1 of 3)

- SSD folders are situated only on GPU nodes.
- You cannot access a SSD folder without booking a GPU node.
- In a GPU node, the SSD folder is located in /local.

Node name	Funding team	Number of GPU cards	Node CPU	Node RAM	Node storage	Hyper threading active ?
nefgpu07	ASCLEPIOS	2x Titan X	2x E5-2620v3	128 GB	2x 1TB SATA 7.2kRPM	no
nefgpu08	ZENITH	4x GTX 1080 Ti	2x E5-2630v3	64 GB	2x 300GB SAS 15kRPM + 1x 800GB SSD + 2x 1.92TB SSD read intensive	no
nefgpu09	GRAPHDECO	4x Titan X	2x E5-2630v4	48 GB	2x 1TB SATA 7.2kRPM + 1x 400GB SSD	no
nefgpu10	STARS	4x Titan X	2x E5-2630v4	128 GB	2x 1TB SATA 7.2kRPM + 1x 1.6TB SSD	no
nefgpu11	STARS	4x GTX 1080	2x E5-2630v4	128 GB	2x 1TB SATA 7.2kRPM + 1x 1.6TB SSD	no
nefgpu12	STARS	4x GTX 1080	2x E5-2630v4	128 GB	2x 1TB SATA 7.2kRPM + 1x 1.6TB SSD	yes
nefgpu13	GRAPHDECO	2x GTX 1080 Ti	2x E5-2650v4	64 GB	2x 1TB SATA 7.2kRPM + 1x 400GB SSD	yes
nefgpu14	STARS	4x GTX 1080 Ti	2x E5-2620v4	128 GB	2x 1TB SATA 7.2kRPM + 1x 400GB SSD	yes
nefgpu15	STARS	4x GTX 1080 Ti	2x E5-2620v4	128 GB	2x 1TB SATA 7.2kRPM + 1x 400GB SSD	yes
nefgpu16	ASCLEPIOS	4x GTX 1080 Ti	2x E5-2630v4	128 GB	2x 600GB SAS 10kRPM + 1x 1.6TB SSD	yes
nefgpu17	ZENITH	4x GTX 1080 Ti	2x E5-2630v4	64 GB	2x 600GB SAS 10kRPM + 1x 1.6TB SSD + 2x 1.92TB SSD read intensive	yes
nefgpu18	common	4x GTX 1080 Ti	2x E5-2630v4	128 GB	2x 600GB SAS 10kRPM + 1x 1.6TB SSD	yes
nefgpu19	common	4x GTX 1080 Ti	2x E5-2630v4	128 GB	2x 600GB SAS 10kRPM + 1x 1.6TB SSD	yes
nefgpu20	common	4x GTX 1080 Ti	2x E5-2630v4	128 GB	2x 600GB SAS 10kRPM + 1x 1.6TB SSD	yes
nefgpu21	STARS	4x GTX 1080 Ti	2x E5-2620v4	128 GB	2x 600GB SAS 10kRPM + 1x 480GB SSD	yes
nefgpu22	STARS	4x GTX 1080 Ti	2x E5-2620v4	128 GB	2x 600GB SAS 10kRPM + 1x 480GB SSD	yes
nefgpu23	TITANE-EPITOME	4x GTX 1080 Ti	2x E5-2630v4	64 GB	2x 600GB SAS 10kRPM + 1x 1.6TB SSD	yes

SSD Folder (2 of 3)

- SSD folders have no quota.
  - They can be used until their capacity is full.
- SSD folders are accessible to any individual who books a GPU node (provided, it contains a SSD).
- SSD capacity typically is much smaller. So, large-scale long-term storage is neither possible nor recommended.

SSD Folder (3 of 3)

SSD Folder is good for :

1. Storing a dataset when training a deep network.

No other use of SSD should be ever carried out.

Temporary Folder (1 of 2)

- A temporary folder is present on every node.
- The path to this folder is /tmp.
- This folder does not have a quota. It usually has a large size.
- However, as soon as you are disconnected from a node, your data on /tmp is automatically deleted.
    - So, do not use it for long-term storage.

Temporary Folder (2 of 2)

A temporary folder should be used for :

1. Generating any temporary or intermediary data generated during experiments.
2. It is generally very fast and hence, it can also be used to store your results until the end of the experiment.
    a. Afterwards, it can be immediately copied to /data/stars.

Scratch Space

- Scratch is a linux group on NEF.
- When you change the group of a file on /data/stars/ to scratch,
  - That file is no longer counted as contributing to the 18TB original quota of /data/stars.
  - However admins are free to remove scratch data as and when it gets full.
- Scratch group is suitable when you end up generating a large amount of data such as 3-4TB or more, and are not sure about how much of that you will actually need in the long term.
- So in such cases, you do not alarm people about data space scarcity, while doing your work effectively without disturbing others.
- In order to know how to use scratch please refer to this link.

Tips for data handling.

- One very large file is better than many small sized files.
  - If you have such a scenario, then bundle such files into one zip or tar.gz archive.
  - It can happen for example during experiments where you generate multiple images for one input image during analysis and would like to save it for future reference.
- Do not duplicate data.
  - There should be only one copy of a dataset at /data/stars/share/STARSDATASETS
  - Any new dataset should be copied there and all people must use it.
  - You can find details of datasets and responsible people at this link.
- If you do not absolutely need some data, just delete it.
- When generating massive amounts of data, just put it on scratch and later selectively transfer some of it which you need to /data/stars.

Resource utilization in NEF

How to book a resource.

- The booking of a resource is done using a program known as OAR.
- Before learning how to book a resource (see examples here), understand following ideas first:
  - Interactive submission.
  - Passive submission.

Interactive Submission

- You have a direct control of a terminal on a compute node.
- It feels like working on a personal system through terminal.
- Very good interactive experience.
- Invoked by putting a -I switch at the end of job submission command.
- It takes a lot of network bandwidth.
- It also generally slows down the job submission a little bit (although not noticeable generally !!)

Passive Submission

- You do not have a direct control of a compute node.
- You provide a script for running your codes and the scheduler (OAR) runs it for you on a compute node.
- It is not interactive. It is like tendering a job to a contractor for getting it done.
- It is a preferred way of submission as it does not need a lot of network bandwidth and the node is completely dedicated to processing.

Preferential Access

- When a team (e.g-STARS) pays money to buy a resource, that team has preferential access to that resource.
  - So, a member from another team (e.g-Graphdeco) using a STARS GPU will be evicted if you request for the same GPU.
- Members from team **A** can book resources from team **B** by using -t besteffort option during job submission.
  - They will be evicted as soon as another member from team **B** requests those resources.
- Members from STARS team can book STARS GPU with besteffort mode as well.
  - They will be evicted if another STARS team member requests for those resources.
  - This is useful if you are not running a very critical experiment at a time of submission when multiple STARS members are also conducting experiments.
  - By following this approach you do not block any user while making resources count.

Information needed before job submission

- Do you need a specific host ? (e.g - nefgpu11.inria.fr ?)
- Do you need a GPU ?
- How many nodes do you need ?
- How many GPUs do you need ?
- How long do you want to book a resource for ?

Information about how to use the oar command to book a resource is available at this link.

Walltime

- Walltime is an important issue with respect to interactive Vs. Passive submissions.
  - If you book a resource in interactive mode for time T, then unless you explicitly quit the resource, the resource is with you for time T.
  - In the case of passive mode however, if your script finishes before T, then the resource will be freed.
- Thus, passive submission is also important for judicious use of resources.
- Do not book a node in interactive mode for long hours.
- Use Passive mode as much as is possible.

Examples of Job submission

1. oarsub -p “gpu=’YES’ and dedicated=’stars’ “ -l/gpunum=2,walltime=2 -S ./script.sh
    a. This books 2 GPUs on a STARS team resource for 2 hours in passive mode. The 2 GPUs are not guaranteed to be on the same node.
2. oarsub -p “gpu=’YES’ and dedicated=’stars’ “ -l/nodes=1/gpunum=2,walltime=2 -S ./script.sh
    a. The same submission as 1, but the 2 GPUs are guaranteed to be on the same node.
3. oarsub -p “gpu=’YES’ and dedicated=’stars’ “ -l/gpunum=2,walltime=2 -I
    a. The same submission as 1 but in interactive mode
4. oarsub -p “gpu=’YES’ and host=’nefgpu12.inria.fr’ “ -l/gpunum=2,walltime=2 -S ./script.sh
    a. The same submission as 1, but on nefgpu12. So, the 2 GPUs are guaranteed to be on the same node i.e nefgpu12.inria.fr

Monitoring your job

- When in interactive mode, you can directly monitor a job.
- In passive mode, you can do that by using the following command :
  - oarsub -C <JOBID>
  - Remember that on every interactive or passive resource booking, a JOB ID will be generated.

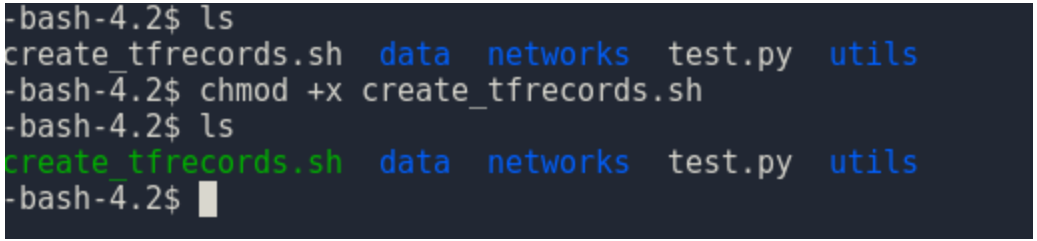
Running jobs on NEF from one file

Step 1 : Create a bash script with OAR options followed by the commands you want to execute.

```
#/bin/bash

#OAR -p gpu='YES' and dedicated='stars'

#OAR -l /gpunum=4,walltime=5

#OAR --name ilsvrc_tfcreator

module load cuda/9.1 cudnn/7.0-cuda-9.1

source activate tensorflow

python utils/ilsvrc_tfrecord_creator.py

zip -r /data/stars/share/STARSDATASETS/ilsvrc2017_clsloc_tfrecords.zip /data/stars/share/STARSDATASETS/IL
SVRC2017/tfrecords

rm -rf /data/stars/share/STARSDATASETS/ILSVRC2017/tfrecords
```

Running jobs on NEF from one file

Step 2 : Save the file and make it executable



Running jobs on NEF from one file

Step 3 : Submit the job as follows

```
-bash-4.2$ oarsub -S ./create_tfrecords.sh
[ADMISSION RULE] Modify resource description with type constraints
[ADMISSION RULE] Automatically add constraint to go on nodes permitted for the user.
OAR_JOB_ID=6445778
-bash-4.2$
```

You can see the Job ID of this job. This is in passive mode.

- In this way I can easily run my jobs in passive mode without having to remember the submission commands everytime.

Monitoring jobs

1. **Method 1**
   a. **oarstat|grep <nef-user-name>**

```
-bash-4.2$ oarstat | grep uujjwal
6445778    ilsvrc_tfcreat uujjwal       2018-05-08 14:59:53 R default
-bash-4.2$
```

2. **Method 2**
   a. **oarsub -C <JOBID>**

3. **Method 3**
   a. **Launch https://nef-frontal.inria.fr/monika (Monika will be useful only to check which resources are free so that you can make more targeted resource requests.)**

Using Modules

- Module files are a useful way to maintain different packages in Linux.
- Invoking a module file simply adds the necessary paths to the system path.
- In NEF there are module files for a vast number of packages.
  - So, before using Cuda 9.1, I type “ module load cuda/9.1”
  - I can also invoke multiple module files at the same time -- “module load cuda/9.1 cudnn/7.0-cuda-9.1”
  - Without invoking these, I cannot use these packages.
- To see a complete list of packages available type “module avail”
- To see which packages you have loaded type “module list”
- To unload all module files invoked by you type “module purge”
- To unload a specific module file type “module rm <Pkg_name>”

Making your own module files

- Users can create their own module files. This however is discouraged.
- Even if you do, do not share them with others.
  - It makes others dependent on those modules and prevents them to learn.
  - When you leave, those users will find themselves stranded.
- Module files should be maintained and used only by the admins.
- My experience :
  - “ If you are using your own module files, it simply means that you are doing something wrong and will create problem for others in the long run.”
- It is not very easy to maintain module files.

GPU Usage

- Do not book multiple GPUs when you do not need them.
- If a code runs on one GPU, then booking 4 GPUs will not necessarily speed your processing.
- If you are unsure that your code is actually using multiple GPUs then most probably it runs only on one GPU.
- How to scale a code to multiple GPUs depends upon the library you are using and of course over your own coding skills.

GPUs over multiple nodes

- Unless you are a very good programmer who knows what he/she is doing, do not utilize GPUs over multiple nodes. This comes under the purview of distributed programming and is not straightforward.
- Distributed training of deep learning systems on 8 GPUs (4/node) is about 16X slower than using 4 GPUs on the same node.
  - The above is a benchmark when you naively write code.
  - With good coding, you will still be 30% slower.
  - It also takes skills to handle training over multiple GPUs as well as distributed training.
- So, do not be over-adventurous. Do your homework before you run a code on GPUs.

Python usage

- Most of you, I expect will work with Python.
- The best approach is the following :
  - Install Anaconda in your home folder or elsewhere. I prefer to install it in /data/stars/user/uujjwal/collection-stars/anaconda3
  - For every library (e.g- PyTorch or TensorFlow), create a separate conda environment
    - An example : conda create -n pytorch python=3.6
  - Then you can invoke that environment with “source activate pytorch” and deactivate it by typing “source deactivate”
- The above approach, allows you to keep different libraries separate without any conflicts.

Pip installation Vs. compilation from sources

- Prefer compiling libraries from sources.
  - This increases speed.
  - This makes sure that the library is built for the GPUs you have.

Final Words of Advice on GPUs

- It does not hurt to ask.
  - Ask your peers and seniors (in terms of their duration of stay in the lab) , who can advise you about how to use GPUs.
- GPUs are costly resources. Use them judiciously.
- Do not leave GPUs in interactive mode when you leave for home.
- When doing non-critical GPU usage, book it in besteffort mode.
- Prefer submitting passive jobs over interactive jobs.
- Use the fewest possible number of GPUs for a job.
  - The more you use, greater are the chances of you making an error.
    - Such as, gradient combination, synchronization of GPUs etc.
    - Speedups are not linear. They decrease with the increase in number of GPUs.

Etiquettes about raising tickets

- Admins are there to assist and not serve.
- If we do not know how to conduct ourselves,
  - We do not have a right to raise a ticket to admins.
- Before, raising a ticket
  - Understand the problem.
  - Make sure you cannot resolve it by yourself.
  - Make sure that the problem is not created by yourself.
- Write in curt and decent language, your problem and wait for their response.
- Don’t argue with admins.