Barcelona Supercomputing Center
Centro Nacional de Supercomputación

User Best Practices and Results on MN5

Aitor Gonzalez-Agirre

Language Technologies Unit

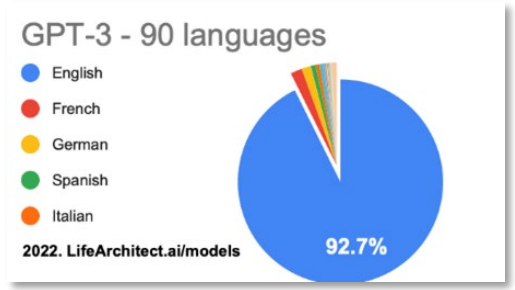
Language Technology Unit

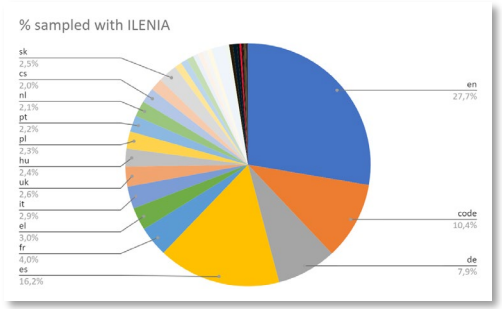
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Context and motivation

- Large Language Models (LLMs), such as OpenAI's GPT, Google's Gemini, and Meta's LLaMA, are trained on massive amounts of text data **primarily in English**, limiting their performance in other languages.
  - Performance of these LLMs for languages with a “weak” or “moderate” technology support, is always significantly weaker than in English.
- The field of generative AI is largely dominated by American technology giants.
  - Europe is far behind.
- Although most LLMs are 'weight open', they are far from being truly open source.
  - Often the code is not open and there is little information on training data.
- New European AI regulations impose levels of transparency and traceability of data that require an adequate approach.
- In response to this situation, three national projects (AINA, ILENIA and Modelos) were created to develop models and resources for the languages of Spain.

Towards truly multilingual EU LLMs





Barcelona Supercomputing Center
Centro Nacional de Supercomputación

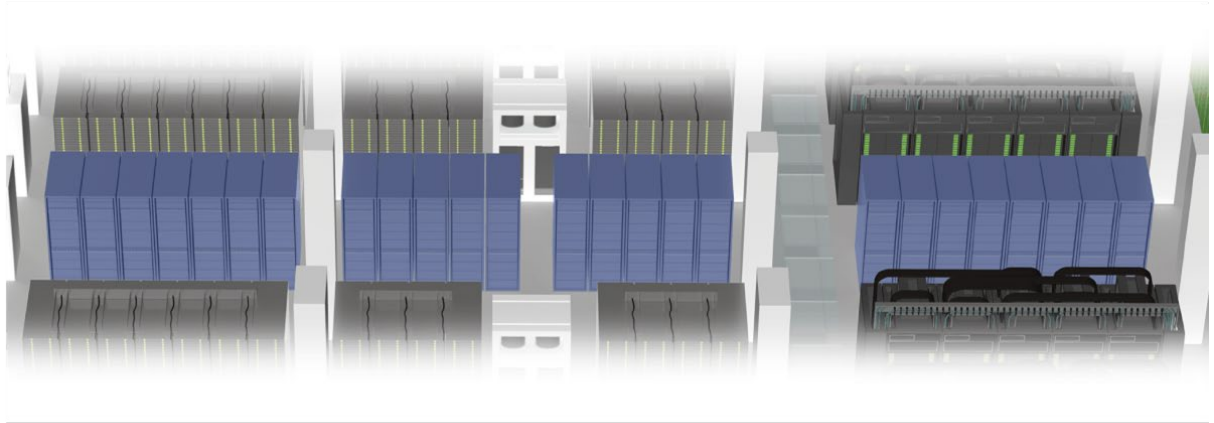
MareNostrum 5

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

MareNostrum 5: Storage

The results of the calculations carried out are stored in 25 racks, each containing 816 18-terabyte hard drives.

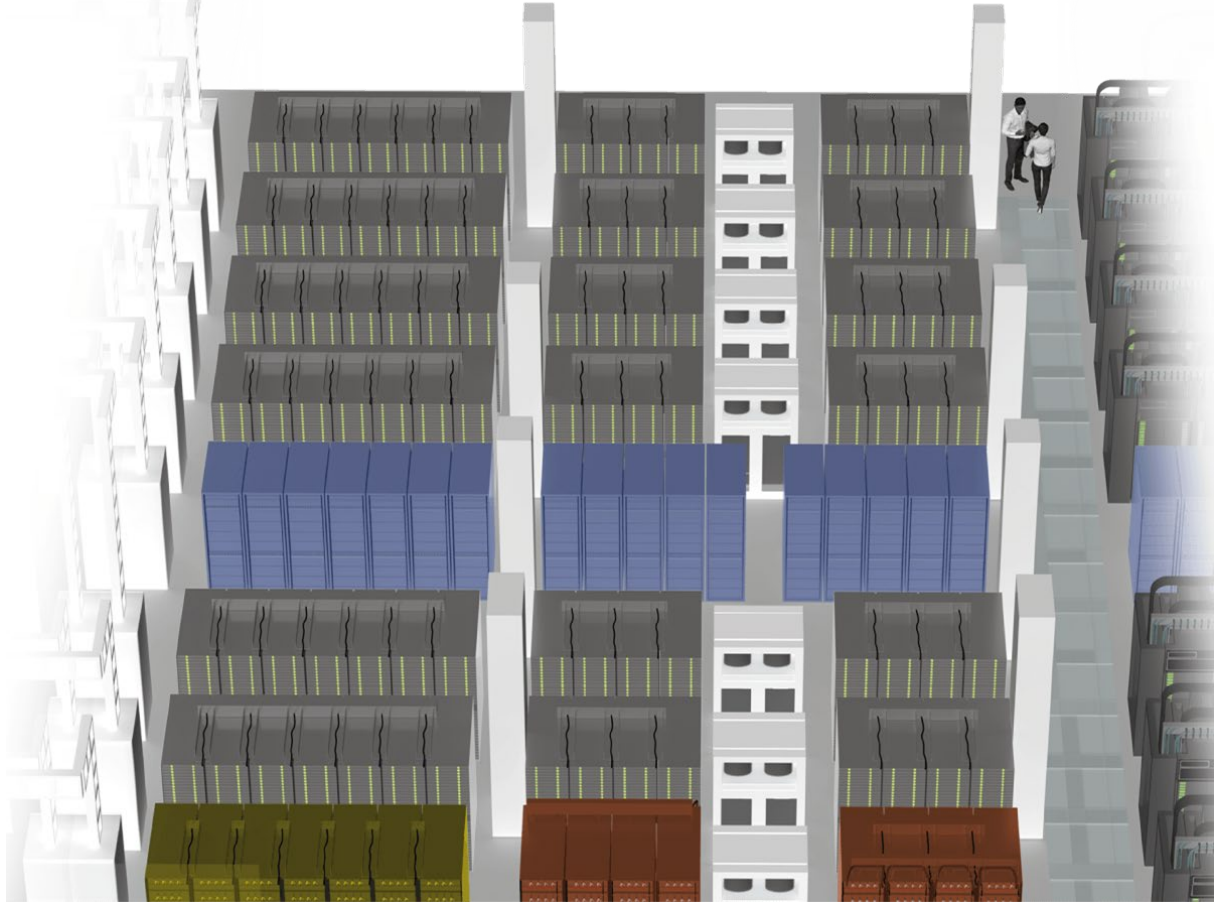
Total net space: 248,000 TB.



They could store 1,280 copies of every book catalogued throughout history.

MareNostrum 5: General Purpose Partition

To help solve major scientific problems. 90 racks, 6,480 nodes and 12,960 chips.

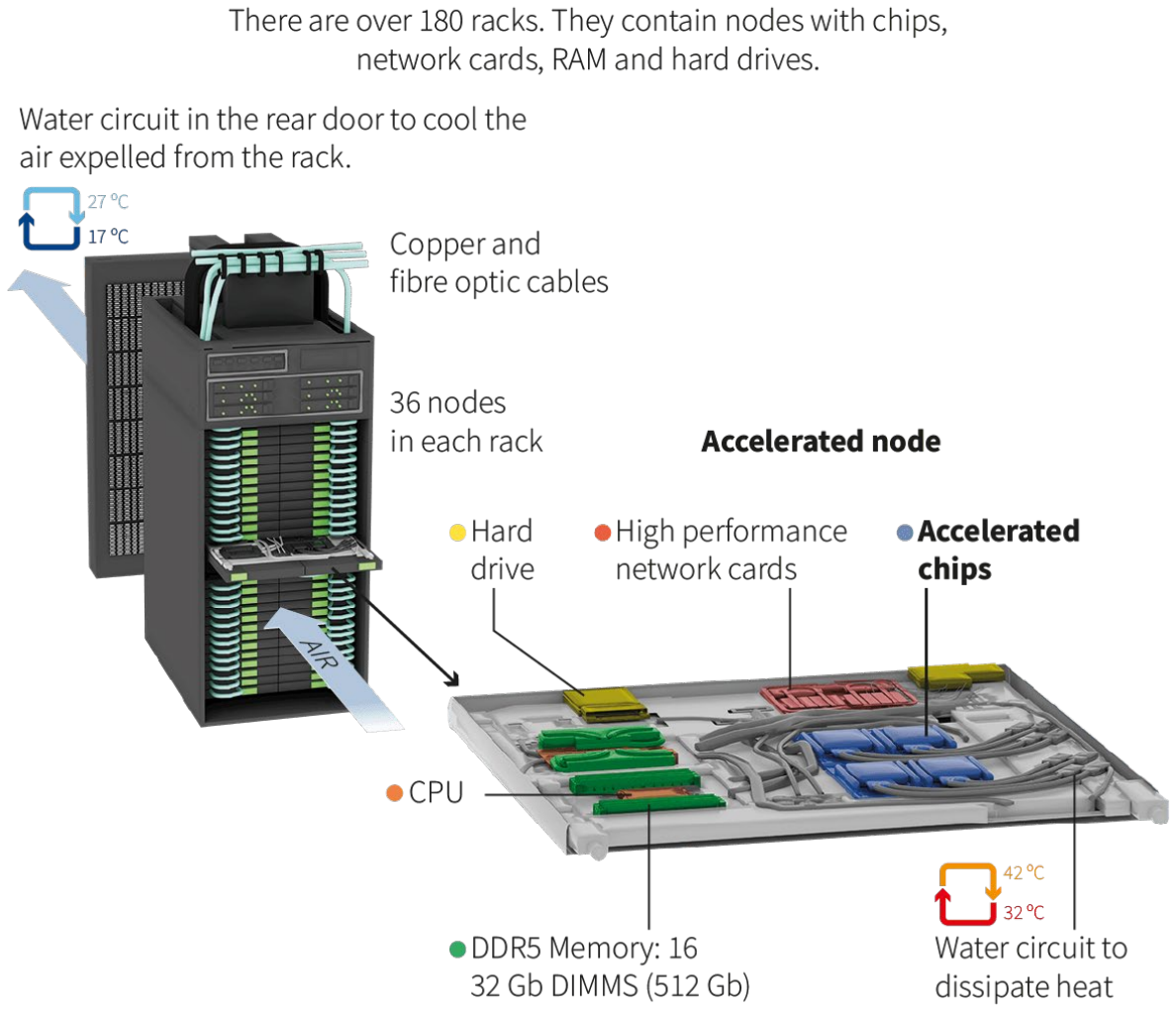


MareNostrum 5: Accelerated Partition

■ 35 racks, 1,120 nodes and 6,720 chips
(4,480 of them, accelerated).

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

MareNostrum 5: H100 64GB GPUs



What is 3D parallelism?

Objectives

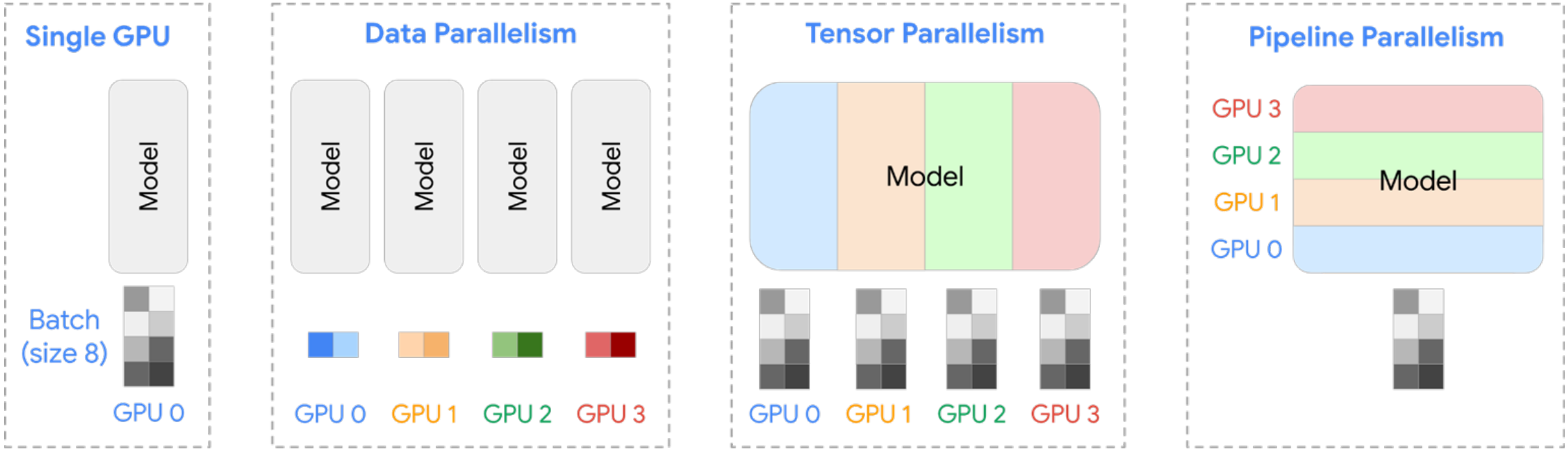
- Reduce the training time of large deep learning models.
- Allow for the training of larger models.
- Optimize hardware utilization.

Definition

3D parallelism training tackles these objectives with a combination of three methods:

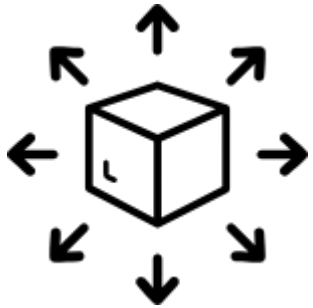
- Tensor Parallelism.
- Pipeline Parallelism.
- Data Parallelism.

What is 3D parallelism?



Typical Bottlenecks

- **Communication Latency.** The communication between nodes causes latency in the training that must be studied and optimized.
- **Coordination Complexity.** 3D parallelism involves intricate data partitioning and synchronization, making it challenging to implement and manage efficiently.
- **Framework Support.** Many deep learning frameworks and libraries lack robust support for 3D parallelism, limiting its usability.
- **Stability.** Larger models can exhibit spikes during pre-training, posing challenges for maintaining stable and consistent training progress.



Data, Pipeline and Tensor Parallelism

- Lower model parallelism does not always imply better performance.
  - Increased model parallelism reduce data parallelism, which in turn can improve the bottlenecks derived from the communication latency associated with it.

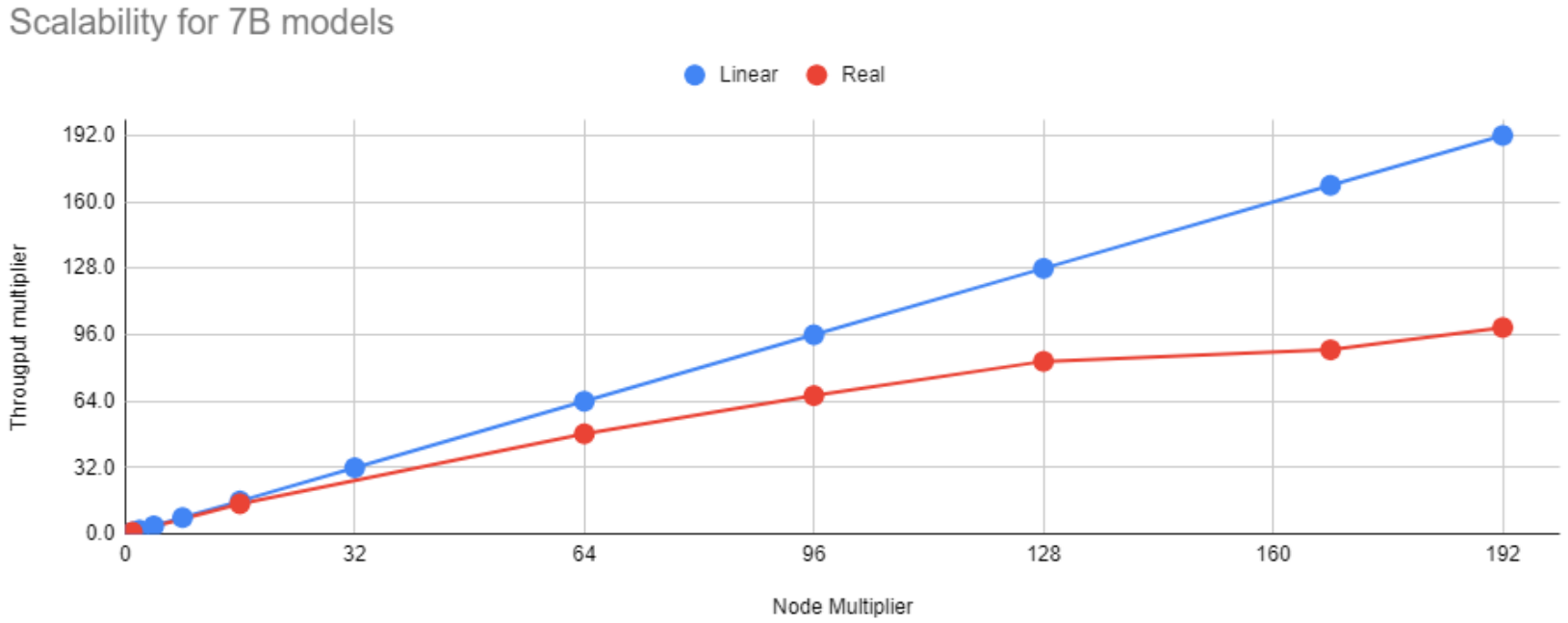
- We could not find a single run where FP8 was better than BF16 in performance, which indicates that the computation was not the main bottleneck.

- NeMo framework proved to be more optimal than GPT-NeoX.

jobid	NPARAMS(B)	world	gbs	mbs	tp	pp	dp	bits	TFlops/GPU
1561797	7220000000	512	1024	4	4	1	128	bf16	225.49
1549683	7200000000	512	1152	3	4	1	128	bf16	219.48
1549657	7200000000	512	1024	4	4	1	128	bf16	217.33
1542017	7200000000	512	1024	2	2	1	256	fp8	209.12
1561796	7220000000	512	1024	2	4	1	128	bf16	207.92
1561803	7220000000	512	1152	3	4	1	128	bf16	207.62
1549632	7200000000	512	1024	2	2	1	256	bf16	206.62

Scalability

- Scalability is not linear, and the performance per GPU decreases as we increase the number of nodes.
- The larger the model, the better it scales to a large number of nodes.
    - For the 7B model, going from 64 nodes to 128 nodes results in a throughput increase of 86%, but going from 64 nodes to 192 nodes, this increase is reduced to 68%.
    - For the 40B model, scaling from 256 nodes to 512 nodes still yields an increase of 73%.



Objectives & Roadmap

- **OBJECTIVE:** To generate a family of decoder models trained with a large volume of high quality, legal data, with a good multilingual composition and representation of different domains. We have collected a corpus of 10 Trillion tokens on 35 languages (+code), cleaned & deduplicated with a focus on Spanish:
  - **First stage: Train 2B, 7B & 40B models.**
  - Second stage: MoE and Multimodal variants.

- **ROADMAP:**
  1. Debugging phase of frameworks and corresponding bug fixing. **DONE**
  2. Ensure correct operation of critical functions such as: checkpoint copying; stopping and resuming of training; de-scaling of models (e.g. downscaling from 128 to 64 nodes, and still training correctly); log copying, ... **DONE**
  3. Performance testing, looking for the best hyper parameter settings to get the best performance. **DONE**
  4. Scalability tests, to check that the performance obtained in (3) is not lost by increasing nodes and/or model size. **DONE**
  5. Train the models! **DONE**/**In progress**
  6. Evaluate and release the models. **DONE**/**In progress**

Best Practices I

- Intended use of the Accelerated Partition (ACC) vs General Partition (GPP):
  - Bear in mind the specific use of each queue. If only CPUs are needed, DO NOT send to ACC.
- DO NOT run compute-intensive jobs in a login node to avoid overloading it.
- When sending a job to the queue, DO NOT ask for more resources than necessary.
  - This applies to both the number of nodes and the duration of the job.
- Use the debug queue for short jobs, very useful for rapid tests.
- With great power comes great responsibility: DO NOT abuse salloc.

Best Practices II

- Edit your ***.bashrc*** file to make sure that newly created files have the right permissions by default.
- To copy or move large files, consider using the ***dtcp***, ***dtmv***, and ***dtrsync*** commands.
- Use symbolic links to avoid duplicating unnecessary data.
- Increase your data loading speed by using the local storage in each node.
- Data management: The different **filesystems** and how to use each of them.
  - **Home**: Personal finals.
  - **Projects**: Projects data, environment, final results.
  - **Scratch**: Temporal files, such as checkpoint checkpoint or intermediate data.
  - **Tapes**: Long-term storage for data that is not expected to be processed in the short-medium term.

Tips for Users I

- Useful ***slurm*** commands:
  - sbatch: Submit a batch script to slurm.
  - squeue: View job information.
  - scancel: Cancel a slurm job.
  - salloc: Allocate nodes to work interactively. Use with caution, and DO NOT forget to release the nodes after finishing.
  - scontrol: View of modify slurm configuration.

- Useful ***bash*** commands:
  - htop: Process monitoring.
  - nvidia-smi: To monitor GPU usage.
  - df/du: Information about disk usage.
  - ps: Information about running processes.
  - chmod: Set and modify file permissions (set them correctly!).
  - screen/tmux: To use multiple shell sessions from a single ssh session.

Tips for Users II

- Useful ***MN5*** commands:
  - bsq_quota: Information about the available space on each file system.
    - Check regularly: If the system runs out of space (for example, due to creating checkpoints), execution will stop.
  - bsc_queues: Information about the queues available to the user.
    - Select the appropriate one, based on the number of nodes needed and the maximum duration of each queue.
    - For debugging, send shorter jobs to the debub to reduce the time in the queue.

- Create ***aliases*** in your *.bashrc* to save time when doing tasks you do frequently:
  - `alias goto_projects='cd /gpfs/projects/bsc88/'`
  - `alias myenv_load='source ~/venv/bin/activate/'`
  - `alias salloc_node='salloc -A bsc88 -q acc_bscls -n 1'`
  - `alias print_dir_structure="ls -R . | grep \":$\" | sed -e 's/:$//' -e 's/[^-][^\/]*\//--/g' -e 's/^/   /' -e 's/-/|/'"`

Good to know

- A daily backup is made from the support team:
  - Only for ***Projects*** filesystem.
  - If you accidentally delete something, you can contact them.

- Use ***alogin4*** for internet access:
  - Very useful to install libraries. Requires VPN.

- Visit the HPCPortal for job monitoring and machine stats:
  - You can share your jobs with other member of the BSC.

Tips for Scaling

- Everything that was working, suddenly, stopped working.
  - Increasing nodes does not improve performance.
  - Or worse, doubling nodes reduces performance.

- Identify what is causing the problem. Three usual suspects:
  - **GPUs**: Is your parallelization correct?
  - **CPUs**: Are some CPUs saturated?
    - If your software is not doing this correctly, you need to set the CPU affinity manually.
  - **Storage**: Can your filesystem load the necessary data fast enough?
    - Try moving the data to a faster disk: the compute nodes' local SSD disk, or even a virtual disk in RAM.
  - **Network**: Does your software manage InfiniBand correctly?
    - Each node has 4 InfinyBand lanes. Manually assign each lane to each of the GPUs (the closest one) to avoid all of them using the same.

Extra: Submitting jobs with dependency

- During the 7B training, I went on vacation for a week:
  - **You're blocking the queue.**

- Having a second job in the queue is useful, when the first fails the second will automatically resume the execution.
  - Problem 1: The second job was not dependent on the first.
  - Problem 2: They were 512 node jobs (MN5 has 1,120 nodes).
    - Since we were launching to a queue with the highest priority, slurm was trying to reserve another 512 nodes for the second job, and not letting any other jobs in.
  - Solution: Submit the jobs with dependency.

Best Practices III: Job dependencies.

- Slurm allows you to launch jobs with different dependencies on other jobs:
  - *sbatch --dependency=<type:job_id[:job_id][,type:job_id[:job_id]]> ...*

- Dependency types:
  - ***after***: Job begins once the dependency has started.
    - Useful to synchronize experiments.
  - ***afterany***: Job beings once the dependency has finished, regardless of what happened.
    - Useful for not using too many nodes at the same time with jobs that don't really depend on each other.
  - ***afternotok***: Job begins once the dependency has failed.
    - Useful for resuming jobs from a previous checkpoints.
    - Use this to submit your LLM training. Your second job will be in the queue without blocking other users.
  - **afterok**: Job begins once the dependency has completed without errors (exit code 0).
    - Useful to run a second step. For instance, to run a finetuning only after the pretraining has finished, or to run the evaluation after the finetuning.
    - Warning: Exit code 0 only indicates that there have been no errors in the execution. It does not ensure that there are no conceptual or design errors.

- Command example: *sbatch -d afternotok:3638431 launch_job_7b.sh*

One last tip

- READ the MN5 documentation:
  - https://www.bsc.es/supportkc/docs/MareNostrum5/intro/

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

User Best Practices and Results on MN5

Aitor Gonzalez-Agirre

Language Technologies Unit