

# Curso Pytorch Distribuido

Diego Andrade Canosa

Roberto López Castro



# Índice

- Introducción al curso
- Hardware para IA en FT3
- Herramientas de Profiling

# Introducción al curso

# Introducción al curso

- Curso de Introducción a Pytorch Distribuido
  - No introducción a Pytorch (conocimiento previo)
  - No introducción a Machine Learning
- Impartido por:
  - Diego Andrade Canosa (CITIC+UDC)
  - Roberto López Castro (CITIC+UDC)
  - Miembros de un grupo de arquitectura de computadores y HPC
  - Usamos Pytorch para investigación sobre la confluencia del HPC con ML
    - AutoML+HPC
    - Performance-aware pruning techniques and kernels

# Introducción al curso

- Duración (12 horas)
  - 17,18 y 19 de julio de 9.30 a 13.30
- Contenidos básicos:
  - Día 1:
    - Soporte nativo para entrenamiento distribuido en Pytorch
  - Día 2:
    - Herramientas avanzadas de entrenamiento distribuido en Pytorch
      - Lightning, DeepSpeed, otras...
  - Día 3:
    - Herramientas de AutoML, NAS y selección de modelos
      - Ray TUNE, AutoPytorch, otras...

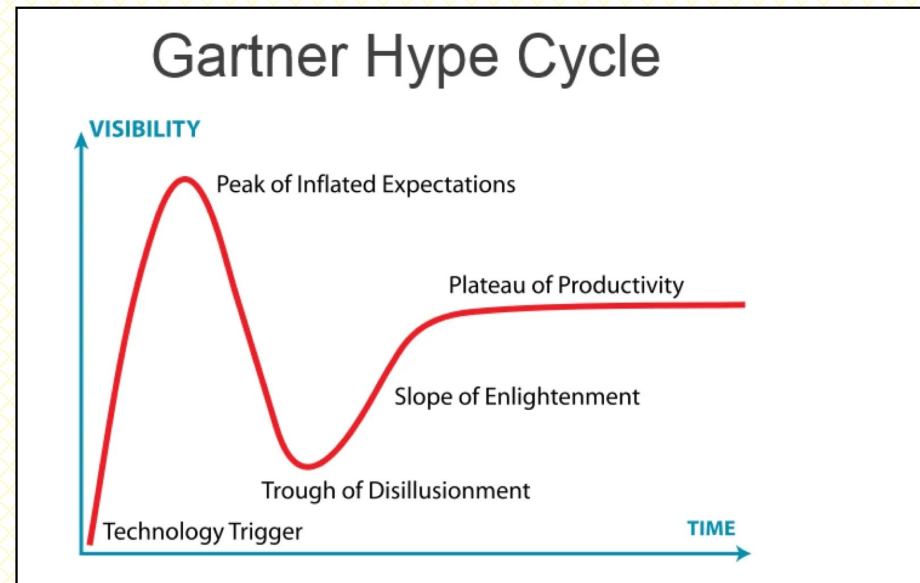
# Introducción al curso

- Soporte nativo de entrenamiento distribuido en Pytorch **DÍA 1**
  - 2.1 Distributed Data-Parallel Training (Diego)
  - 2.2 RPC-Based Distributed Training (Roberto)
- Herramientas avanzadas de entrenamiento distribuido en Pytorch **DÍA 2**
  - 3.1 Lightning (Diego)
  - 3.2 DeepSpeed (Roberto)
  - 3.3 Otras herramientas (Diego y Roberto)
- Herramientas y técnicas de AutoML, Network Architecture Search y Selección de Modelos **DÍA 3**
  - 4.1 Ray TUNE (Diego)
  - 4.2 AutoPytorch (Roberto)
  - 4.3 Otras herramientas (Diego y Roberto)

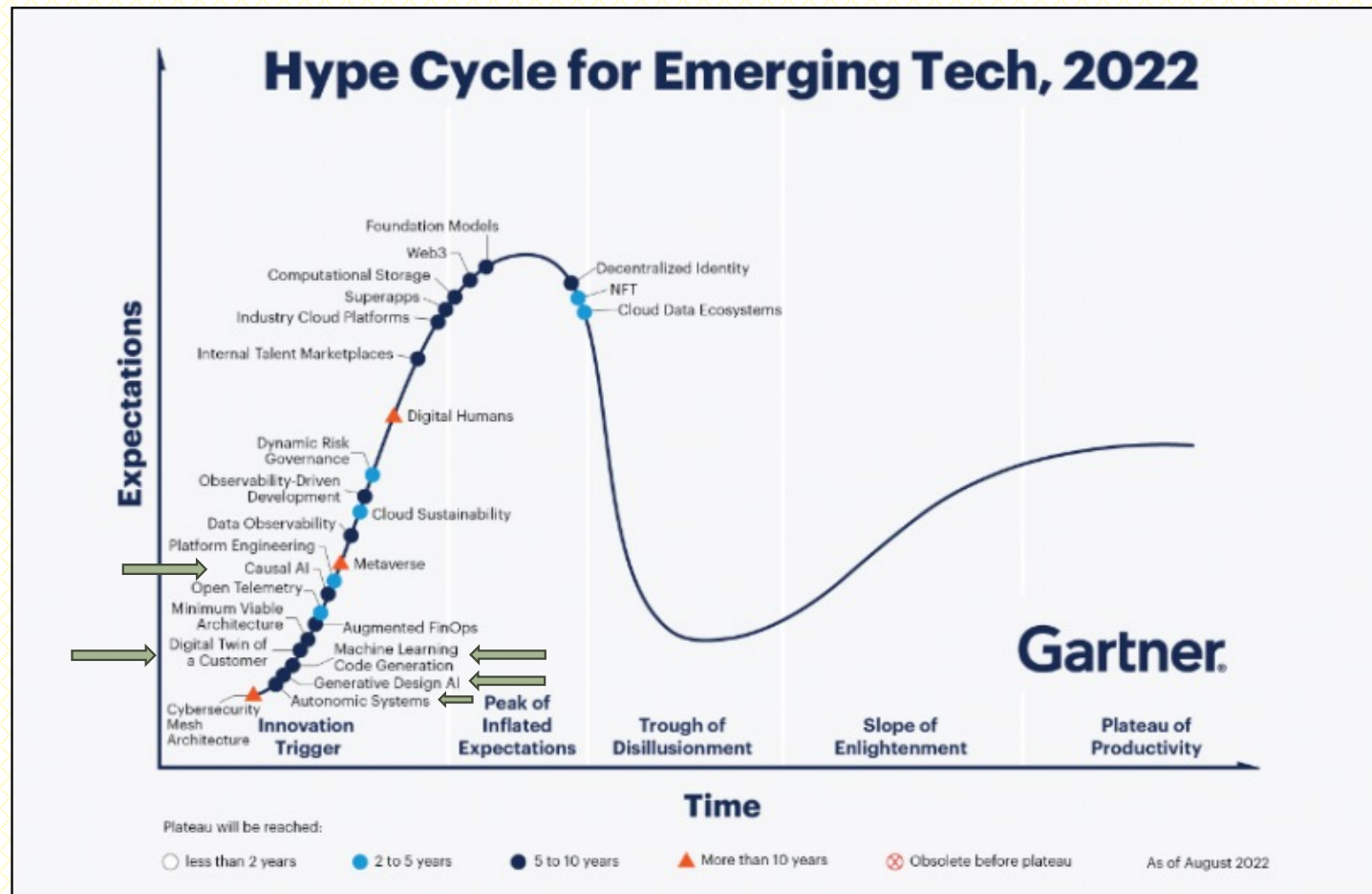
# Introducción al curso: **Motivación**

Why AI is the new electricity por Andrew Ng

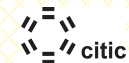
- **Motivo 1:** La Inteligencia Artificial (IA) juega y jugará un papel fundamental



# Introducción al curso: Motivación



Ciclo del Hype de Gartner

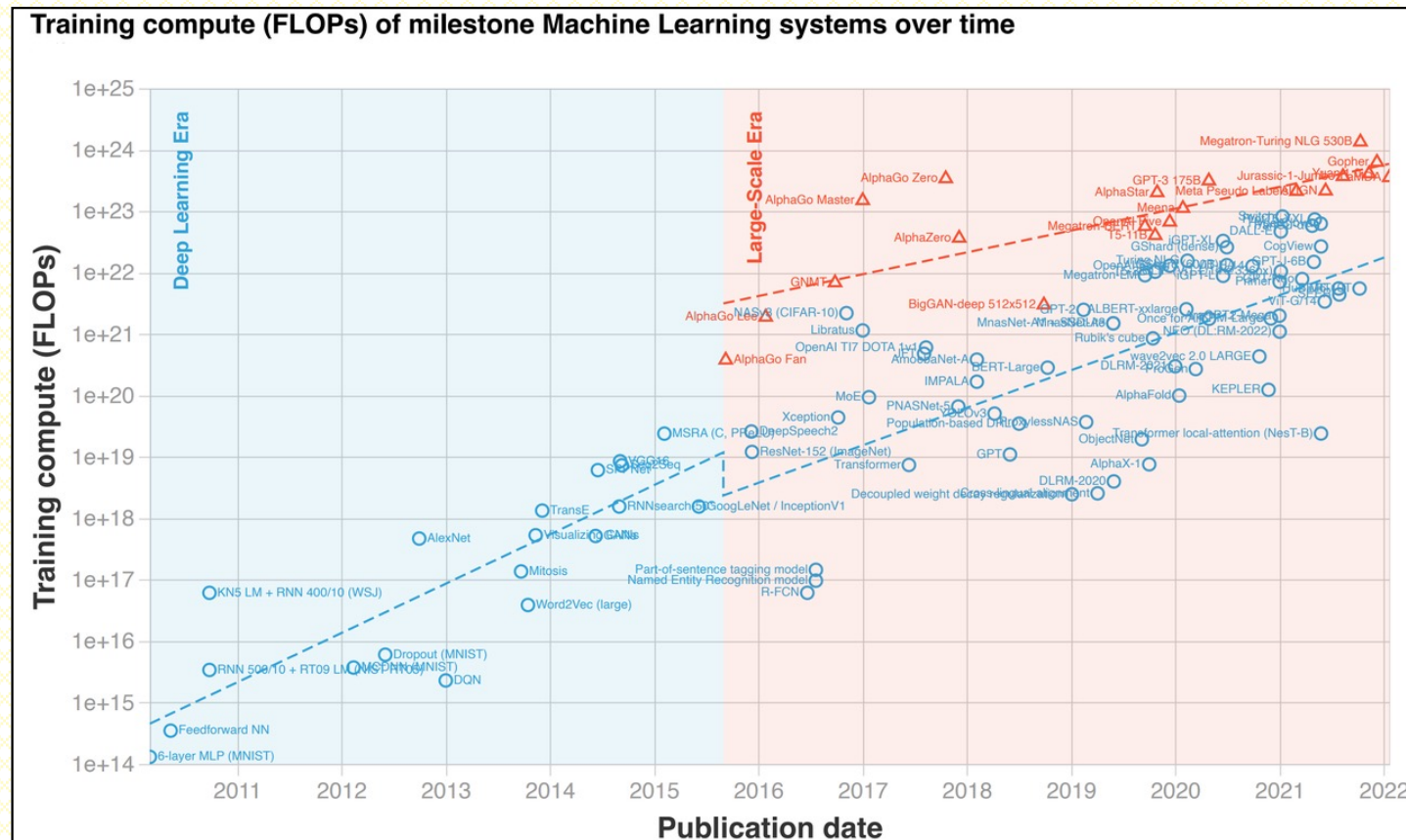




# Introducción al curso: **Motivación**

- **Motivo 2:** El coste de recursos (computo y memoria) de entrenar nuevos modelos está creciendo aceleradamente

# Introducción al curso: Motivación



Fuente: <https://www.marktechpost.com/2022/07/13/colossal-ai-a-unified-deep-learning-system-for-big-models-seamlessly-accelerates-large-models-at-low-costs-with-hugging-face/>



# Introducción al curso: Motivación

- **Motivo 3:** El entrenamiento de modelos requiere el uso de recursos computacionales heterogéneos

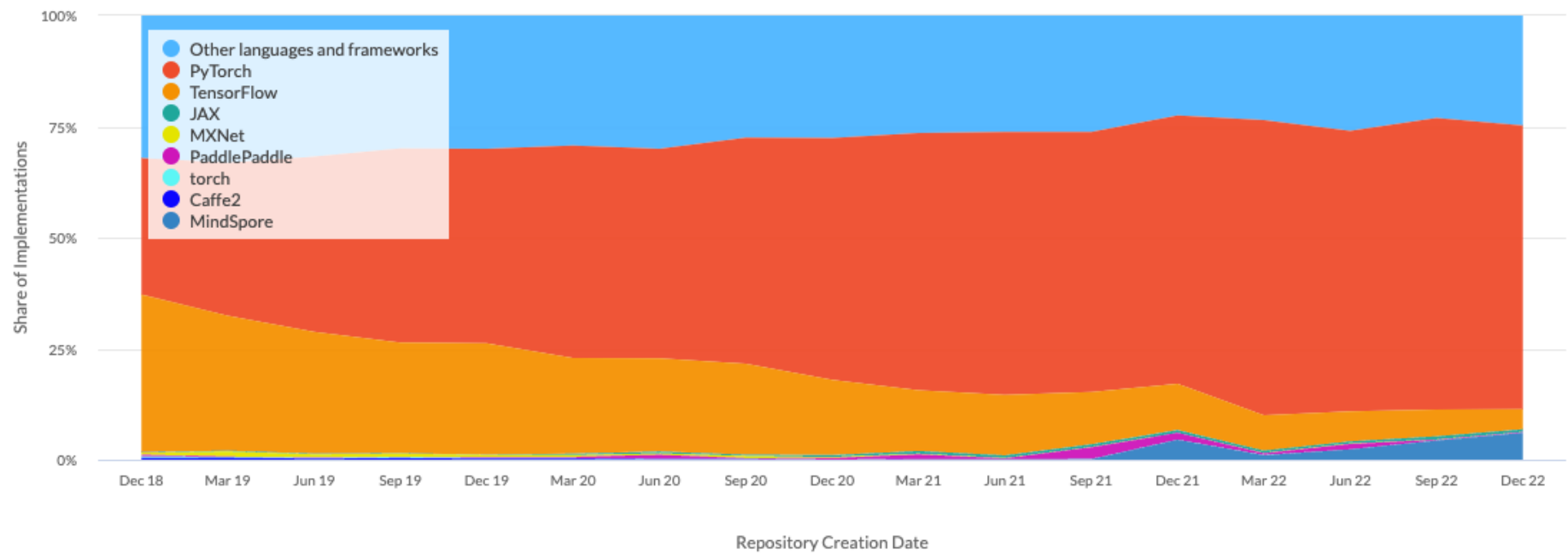
- Tarjetas gráficas: Graphic Processing Units (GPUs)
  - Conceptos asociados: GPGPU, CUDA
  - Tienen hw especializado para IA: Tensor Cores, Sparse Tensor Cores
    - Explotado por las librerías especializadas de Nvidia ([cuDNN](#), cuBLAS, cuSparseLT, cutlass), usadas a su vez por los frameworks más conocidos de IA (TF, Pytorch)
- CPUs multinúcleo: Multicores
  - Más lentas para IA que las GPU
  - Tienen hw especializado para IA
    - Principalmente extensiones del juego de instrucciones vectoriales o multimedia del procesador
      - Explotado por librerías especializadas de los fabricantes ([Intel OneAPI AI Analytics](#))
- Hardware de propósito específico
  - Ejemplo: Tensor Processing Units (TPUs) de Google
- Computadores (o aceleradores) cuánticos (*en desarrollo*)



# Pytorch domina la innovación en IA

## Motivo 4: Pytorch domina el desarrollo innovador en ML

Paper Implementations grouped by framework



# tldr;

- *El soporte nativo de Pytorch para entornos distribuidos es suficiente en el 90% de los casos*
- *Para todo lo demás ¡Usa Pytorch Lightning (o Fabric)!*
  - *Tecnologías como DeepSpeed o Accelerate exploran los límites del Hardware*
- *Ray es una navaja suiza*

# Modelos escalables vs no escalables

- Hay arquitecturas de red que son difícilmente paralelizables
  - Ejemplo RNNs
    - Al ser recurrentes se reduce la posibilidad de paralelizarlas
      - No son paralelizables
        - Tiene dependencias en la dimensión temporal que lo impiden
    - Las redes de atención (*attention networks*) están concebidas para implementar RNNs paralelizables
      - Permiten procesar diferentes partes de la entrada, en paralelo, con distintas cabezas (*heads*)
      - Cada *head* utiliza una máscara diferente

| Scheme                           | Number of parameters (billion) | Model-parallel size | Batch size | Number of GPUs | Microbatch size | Achieved teraFLOP/s per GPU | Training time for 300B tokens (days) |
|----------------------------------|--------------------------------|---------------------|------------|----------------|-----------------|-----------------------------|--------------------------------------|
| ZeRO-3 without Model Parallelism | 174.6                          | 1                   | 1536       | 384            | 4               | 144                         | 90                                   |
|                                  |                                |                     |            | 768            | 2               | 88                          | 74                                   |
|                                  |                                |                     |            | 1536           | 1               | 44                          | 74                                   |
|                                  | 529.6                          | 1                   | 2560*      | 640            | 4               | 138                         | 169                                  |
|                                  |                                |                     | 2240       | 1120           | 2               | 98                          | 137                                  |
|                                  |                                |                     |            | 2240           | 1               | 48                          | 140                                  |
| PTD Parallelism                  | 174.6                          | 96                  | 1536       | 384            | 1               | 153                         | 84                                   |
|                                  |                                |                     |            | 768            | 1               | 149                         | 43                                   |
|                                  |                                |                     |            | 1536           | 1               | 141                         | 23                                   |
|                                  | 529.6                          | 280                 | 2240       | 560            | 1               | 171                         | 156                                  |
|                                  |                                |                     |            | 1120           | 1               | 167                         | 80                                   |
|                                  |                                |                     |            | 2240           | 1               | 159                         | 42                                   |

Pipeline-Tensor-Data Parallelism vs Zero3 (sin MP)

Fuente: <https://arxiv.org/pdf/2104.04473.pdf>



# Modelos que requieren gran capacidad de cómputo

- Los modelos de Deep Learning (DL) actuales requieren cada vez más recursos:
  - Memoria
  - Cómputo
    - Entrenamiento
      - Inicial + Fine-tuning
    - Inferencia



# Ejemplos de de modelos grandes: Bert

|                        | BERT  | RoBERTa   | DistilBERT   | XLNet   |
|------------------------|---|---|--|---|
| <b>Size (millions)</b> | Base: 110<br>Large: 340   | Base: 110<br>Large: 340                               | Base: 66   | Base: ~110<br>Large: ~340   |
| <b>Training Time</b>   | Base: 8 x V100 x 12 days*<br>Large: 64 TPU Chips x 4 days (or 280 x V100 x 1 days*) | Large: 1024 x V100 x 1 day; 4-5 times more than BERT. | Base: 8 x V100 x 3.5 days; 4 times less than BERT. | Large: 512 TPU Chips x 2.5 days; 5 times more than BERT.  |
| <b>Performance</b>     | Outperforms state-of-the-art in Oct 2018  | 2-20% improvement over BERT                           | 3% degradation from BERT                           | 2-15% improvement over BERT   |
| <b>Data</b>            | 16 GB BERT data (Books Corpus + Wikipedia).<br>3.3 Billion words.                   | 160 GB (16 GB BERT data + 144 GB additional)          | 16 GB BERT data.<br>3.3 Billion words.             | Base: 16 GB BERT data<br>Large: 113 GB (16 GB BERT data + 97 GB additional).<br>33 Billion words. |
| <b>Method</b>          | BERT (Bidirectional Transformer with MLM and NSP)                                   | BERT without NSP**                                    | BERT Distillation                                  | Bidirectional Transformer with Permutation based modeling   |

[Fuente: BERT, RoBERTa, DistilBERT, XLNet — which one to use? | by Suleiman Khan, Ph.D. | Towards Data Science](#)



# Ejemplos de de modelos grandes: GPT-3

- Llevaría 355 años entrenar este modelo en una sola Tesla V100
  - 4.6 millones de dólares de coste en un proveedor de cloud [[fuente](#)]

# Entrada/Salida

| # | Lab        | Dataset          | Size (TB)     | Tokens (trillion) | Notes                                 |
|---|------------|------------------|---------------|-------------------|---------------------------------------|
| 1 | Google ▾   | Piper monorepo   | <b>86TB</b>   | <i>37.9T</i>      | DIDACT, code only. From 2016 paper.   |
| 2 | OpenAI ▾   | GPT-4            | <i>40TB</i>   | <i>20T</i>        | 1T model ∴ 20T tokens. gdb said 40TB. |
| 3 | TTI        | RefinedWeb       | <i>23.2TB</i> | <b>5.0T</b>       | CC-only dataset prepared by UAE.      |
| 4 | DeepMind ▾ | MassiveText (ml) | <i>20TB</i>   | <b>5.0T</b>       | From Retro paper.                     |
| 5 | Google ▾   | PaLM 2           | <i>13TB</i>   | <b>3.6T</b>       | From PaLM 2 CNBC report.              |
| 6 | Google ▾   | Infiniset        | <i>12.6TB</i> | <b>2.8T</b>       | From LaMDA paper.                     |

**Table. 2023 largest dataset estimates to Jun/2023.** Rounded. Disclosed in **bold**. Determined in *italics*. For similar models, see my *What's in my AI* paper.



# Hardware para IA en FT3

# Descripción avanzada de Finisterrae III (FT3)

|             | Thin nodes                                    | GPU nodes | Vis nodes    | Fat nodes           | Transfer nodes     | QLM node   |
|-------------|---|-----------|--------------|---------------------|--------------------|--|
| Cantidad    | 256   | 64        | 16           | 16                  | 2                  | Simulador<br>de circuitos<br>cuánticos<br>Atos<br>30 qbits |
| Proc.       | 2x Intel Xeon Ice Lake 8352Y 32 cores 2.2 GHz |           |              |                     |                    |  |
| Cores       | 64  |           |              |                     |                    |  |
| Memoria     | 256 GB  |           |              | 2 TB                | 256 GB             |  |
| Disco local | 960 GB SSD NVMe                               |           |              | 1920 GB<br>SSD NVMe | 960 GB SSD<br>NVMe |  |
| GPU         | 2x Nvidia<br>A100                             |           | Nvidia<br>T4 |                     |                    |  |

Fuente: [CESGA - Portal de usuarios](#)

# Backends de Pytorch

- Habilitan el uso eficiente del hardware disponible habilitando el uso de los kernels numéricos de los fabricantes

```
torch.backends.cuda
```

```
torch.backends.cudnn
```

```
torch.backends.mps
```

```
torch.backends.mk1
```

```
torch.backends.mkldnn
```

```
torch.backends.openmp
```

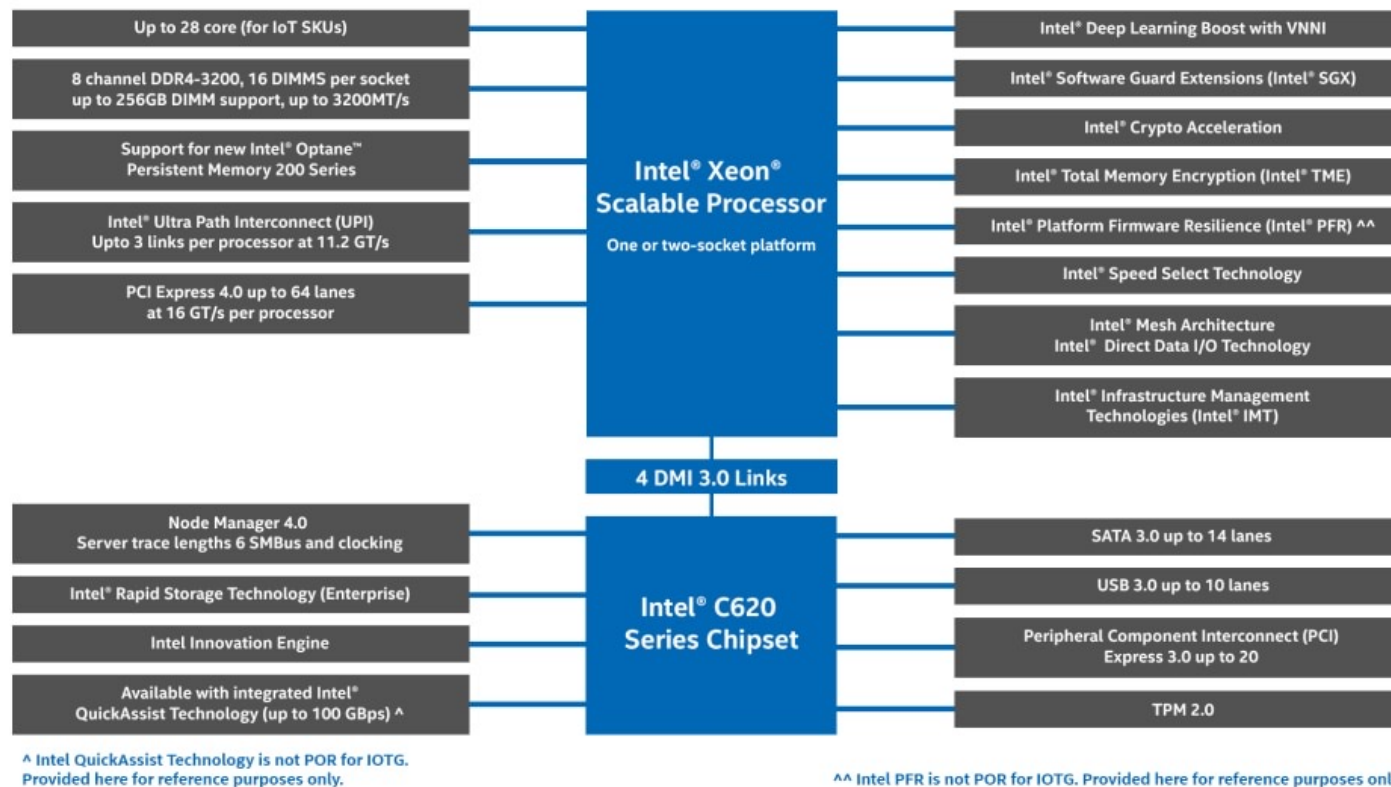
```
torch.backends.opt_einsum
```

```
torch.backends.xeon
```



# Descripción avanzada de Finisterrae III (FT3)

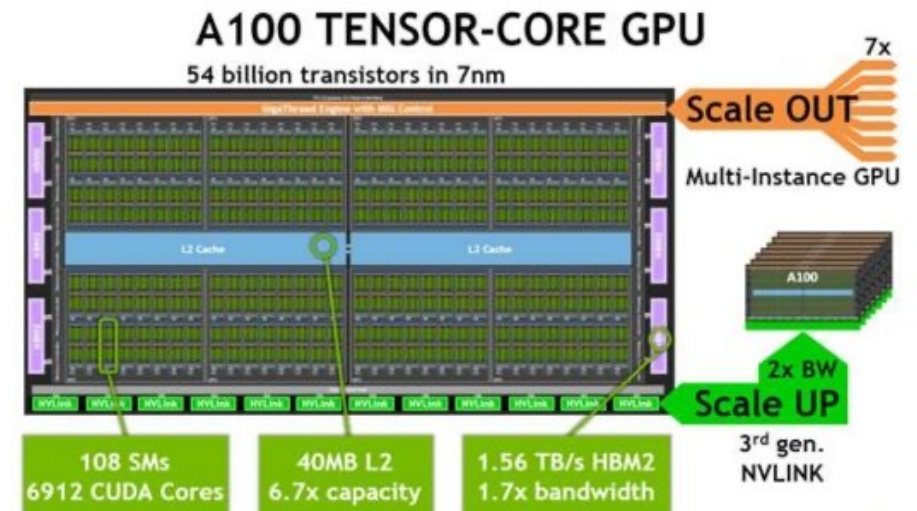
Cada nodo tiene 2 procesadores Intel Xeon 8352Y de 32 cores cada uno -> 64 cores por nodo



Fuente: [Procesador Intel® Xeon® Platinum 8352Y](#) y [Xeon Platinum - Intel WikiChip](#)

# Descripción avanzada de Finisterrae III (FT3)

- Varios modelos de GPU:
  - Tesla T4. Arquitectura Turing.
  - Tesla A100. Arquitectura A100.





# Arquitectura de la A100

- 40 GB de memoria HBM2 (1555 GB/s)
- 40MB de cache L2 (nivel 2). Topología crossbar
- Con la tecnología MIG (Multi-Instance GPU) se puede particionar una A100 en hasta 7 particiones

The **NVIDIA A100 Tensor Core GPU** implementation of the GA100 GPU includes the following units:

- 7 GPCs, 7 or 8 TPCs/GPC, 2 SMs/TPC, up to 16 SMs/GPC, 108 SMs
- 64 FP32 CUDA Cores/SM, 6912 FP32 CUDA Cores per GPU
- 4 Third-generation Tensor Cores/SM, 432 Third-generation Tensor Cores per GPU
- 5 HBM2 stacks, 10 512-bit Memory Controllers



# Descripción avanzada de Finisterrae III (FT3)

- A100 Streaming Multiprocessor:
- 4 bloques de procesamiento cada uno con
  - 1 x L1 cache
  - 1 x Warp scheduler
  - 16 x INT32 CUDA cores
  - 16 x FP32 CUDA cores
  - 8 x FP64 CUDA cores
  - 8 x Load/Store cores
  - 1 x Tensor core for matrix multiplication
  - 1 x 16K 32-bit register file
- 32 thread blocks por SM

Fuente: [NVIDIA A100](#) | [NVIDIA](#)



# Rendimiento pico

|                                    |                                      |
|------------------------------------|--------------------------------------|
| Peak FP64 <sup>1</sup>             | 9.7 TFLOPS                           |
| Peak FP64 Tensor Core <sup>1</sup> | 19.5 TFLOPS                          |
| Peak FP32 <sup>1</sup>             | 19.5 TFLOPS                          |
| Peak FP16 <sup>1</sup>             | 78 TFLOPS                            |
| Peak BF16 <sup>1</sup>             | 39 TFLOPS                            |
| Peak TF32 Tensor Core <sup>1</sup> | 156 TFLOPS   312 TFLOPS <sup>2</sup> |
| Peak FP16 Tensor Core <sup>1</sup> | 312 TFLOPS   624 TFLOPS <sup>2</sup> |
| Peak BF16 Tensor Core <sup>1</sup> | 312 TFLOPS   624 TFLOPS <sup>2</sup> |
| Peak INT8 Tensor Core <sup>1</sup> | 624 TOPS   1,248 TOPS <sup>2</sup>   |
| Peak INT4 Tensor Core <sup>1</sup> | 1,248 TOPS   2,496 TOPS <sup>2</sup> |

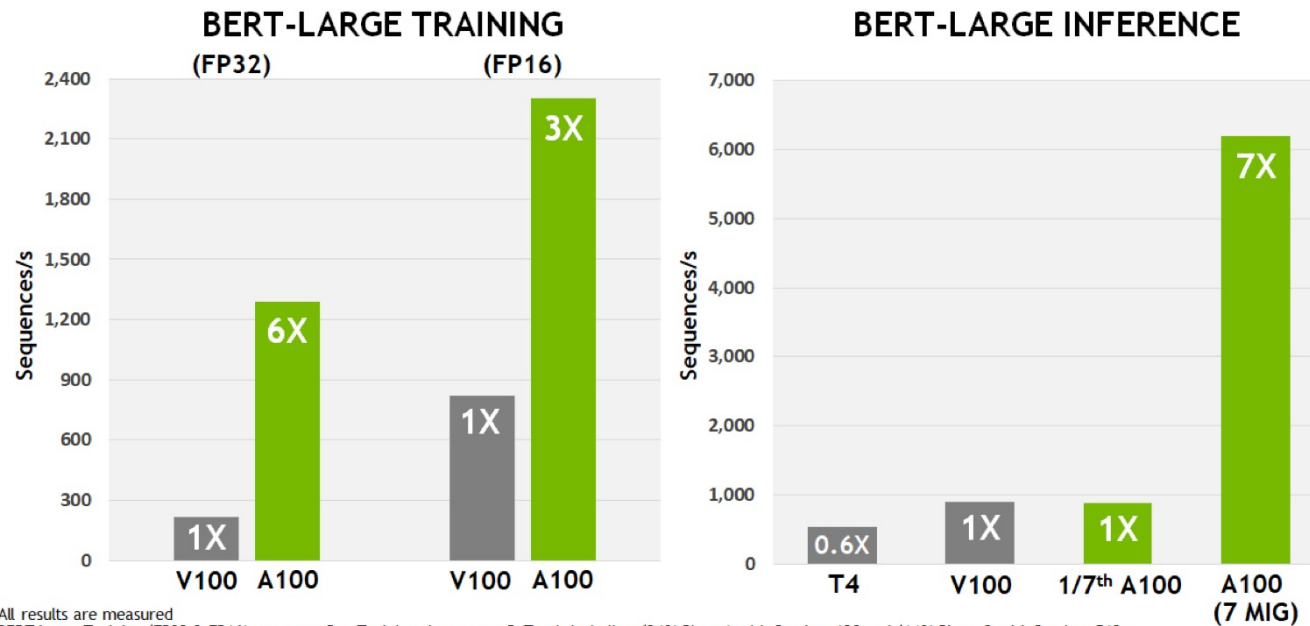
1 - Peak rates are based on GPU Boost Clock.

2 - Effective TFLOPS / TOPS using the new Sparsity feature



# A100 vs V100

## UNIFIED AI ACCELERATION



All results are measured  
BERT Large Training (FP32 & FP16) measures Pre-Training phase, uses PyTorch including (2/3) Phase1 with Seq Len 128 and (1/3) Phase 2 with Seq Len 512,  
V100 is DGX1 Server with 8xV100, A100 is DGX A100 Server with 8xA100, A100 uses TF32 Tensor Core for FP32 training  
BERT Large Inference uses TRT 7.1 for T4/V100, with INT8/FP16 at batch size 256. Pre-production TRT for A100, uses batch size 94 and INT8 with sparsity



# Descripción avanzada de Finisterrae III (FT3)

Nvidia Tensor Cores

## TENSOR CORE 4X4X4 MATRIX-MULTIPLY ACC

$$\mathbf{D} = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

FP16 or FP32      FP16      FP16 or FP32

8 NVIDIA

Fuente: [Tensor Cores: versatilidad para HPC e IA | NVIDIA](#)

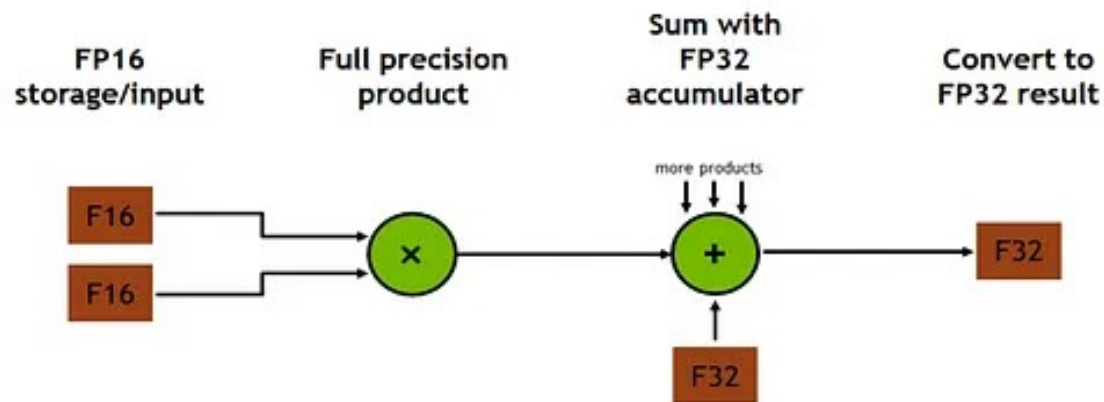


# Nvidia Tensor Cores (Ampere ver.)

- 1 x A100 Tensor Core ejecuta:
  - 256 x operaciones FMA (fused multiply-add) por ciclo en precisión FP16
    - $8 \times 4 \times 8$  mixed-precision MatMul por ciclo

# Descripción avanzada de Finisterrae III (FT3)

Nvidia Tensor Cores

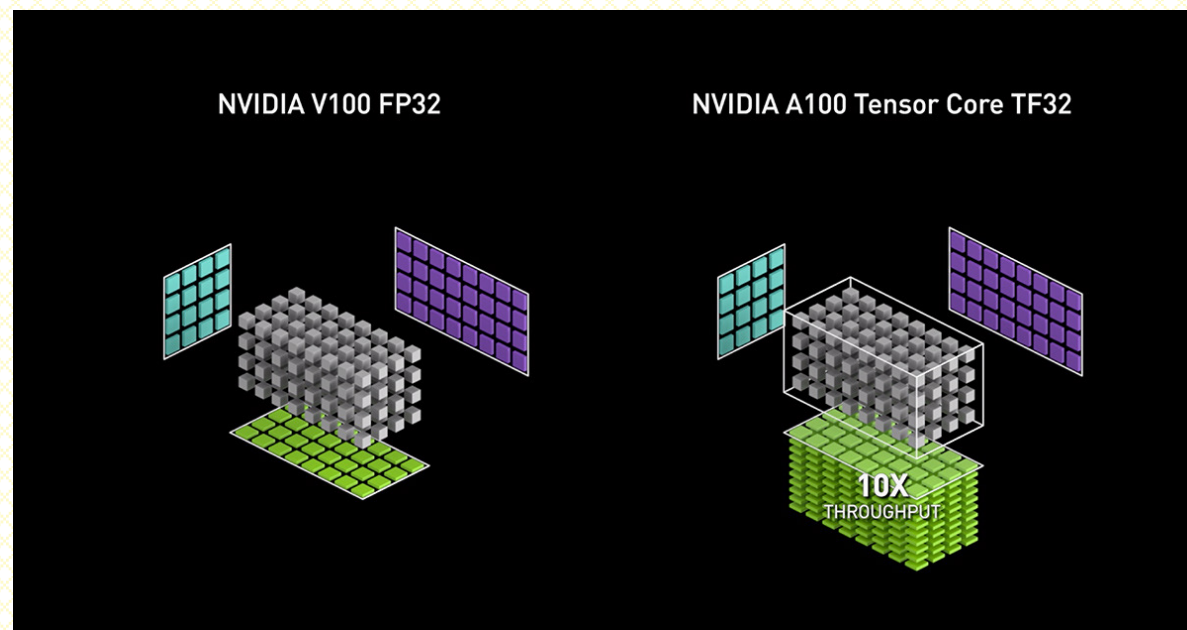


Fuente: [Tensor Cores: versatilidad para HPC e IA | NVIDIA](#)



# Descripción avanzada de Finisterrae III (FT3)

Nvidia Tensor Cores



Fuente: [Tensor Cores: versatilidad para HPC e IA | NVIDIA](#)





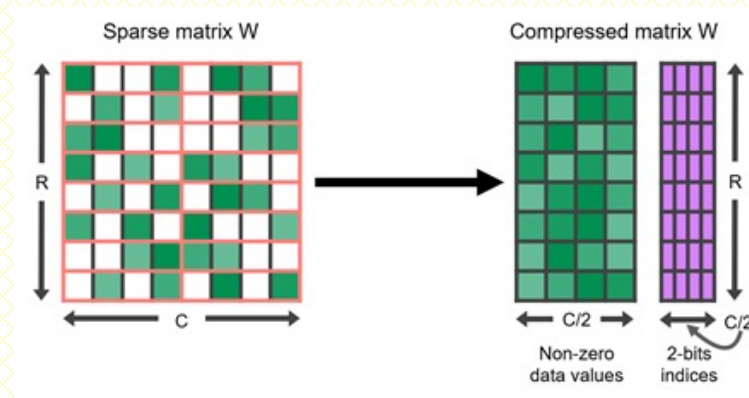
# Tensor Cores de 3ª generación

- Aceleración para todos los tipos de datos FP16, BF16, TF32, FP64, INT8, INT4 y Binary
  - TF32 es un tipo de dato que emula FP32 con longitud 16
    - Mismo rango -> menor precisión
- Cómputo sparse para dispersión estructurada (2:4)
  - Dobla el rendimiento de cómputo de TC sin dispersión
- Mixed precision (FP16/FP32) es 2.5x más rápido que Tesla (5x con dispersión)



# Descripción avanzada de Finisterrae III (FT3)

Nvidia **Sparse** Tensor Cores

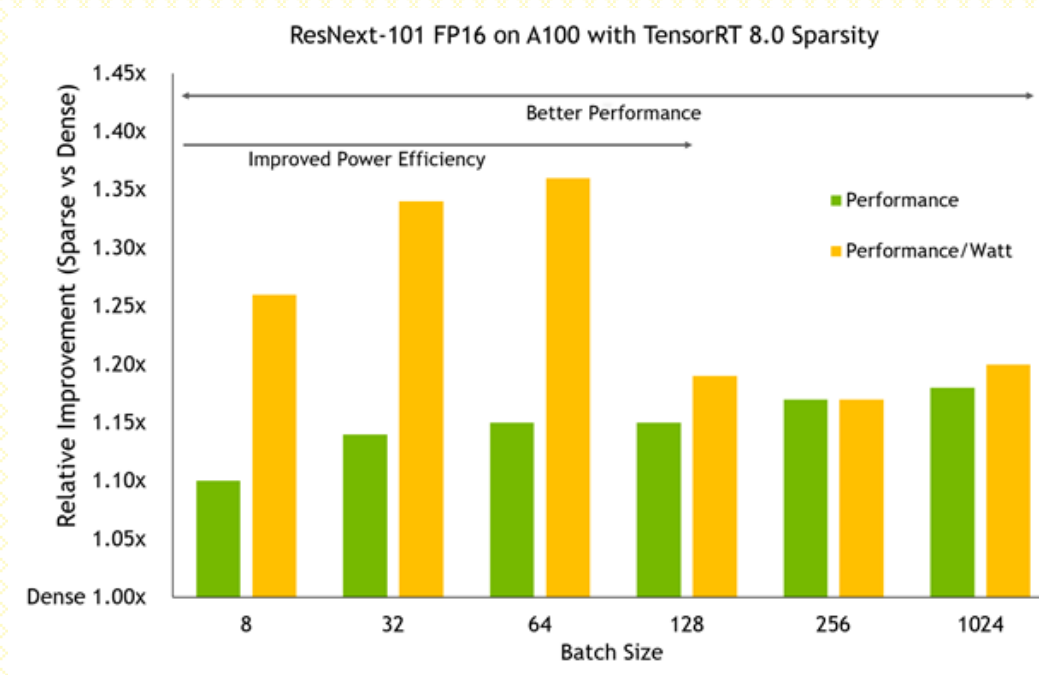


Fuente: [Working with sparse tensors | TensorFlow Core y Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT | NVIDIA Technical Blog](#)



# Descripción avanzada de Finisterrae III (FT3)

Nvidia **Sparse** Tensor Cores



Fuente: [Working with sparse tensors | TensorFlow Core y Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT | NVIDIA Technical Blog](#)



# Nvidia A100

- Explicación detallada: <https://jonathan-hui.medium.com/ai-chips-a100-gpu-with-nvidia-ampere-architecture-3034ed685e6e>

# Descripción avanzada de Finisterrae III (FT3)

- Almacenamiento compartido: Home y Store
- Accesible desde todas las infraestructuras de CESGA
- Localización cableada en las variables de entorno \$HOME y \$STORE
- Límites:
  - HOME: 10 GB y 100.000 ficheros
  - STORE: 500 GB y 300.000 ficheros
- Sólo se hace backup de Home

| Directory | Use   | User limits            | Backup |
|-----------|---|------------------------|--------|
| \$HOME    | Store code files<br>Low speed access                | 10GB<br>100.000 files  | Yes    |
| \$STORE   | Store simulations final results<br>Low speed access | 500GB<br>300.000 files | No     |
| \$LUSTRE  | Simulation runs<br>High speed access                | 3TB<br>200.000 files   | No     |

```
[ulcesdac@login211-1 dac]$ cd $LUSTRE
[ulcesdac@login211-1 dac]$ pwd
/mnt/lustre/scratch/nlsas/home/ulc/es/dac
[ulcesdac@login211-1 dac]$ cd $HOME
[ulcesdac@login211-1 ~]$ pwd
/home/ulc/es/dac
[ulcesdac@login211-1 ~]$ cd $STORE
[ulcesdac@login211-1 dac]$ pwd
/mnt/netapp2/Store_uni/home/ulc/es/dac
[ulcesdac@login211-1 dac]$ |
```

Fuente: [CESGA - Portal de usuarios](#)



# Descripción avanzada de Finisterrae III (FT3)

- Sistema paralelo de almacenamiento compartido basado en Lustre
- Dividido en 2 pools: NVMe y NLSAS
- Accesible a través de
  - \$LUSTRE: Permanente. Pool NLSAS
  - \$LUSTRE\_SCRATCH: Disponible durante la ejecución de los trabajos. Pool NVMe.
- Límites: Hasta 3 TB por usuario y 200.000 ficheros

```
[ulcesdac@login211-1 dac]$ cd $LUSTRE
[ulcesdac@login211-1 dac]$ pwd
/mnt/lustre/scratch/nlsas/home/ulc/es/dac
[ulcesdac@login211-1 dac]$ cd $HOME
[ulcesdac@login211-1 ~]$ pwd
/home/ulc/es/dac
[ulcesdac@login211-1 ~]$ cd $STORE
[ulcesdac@login211-1 dac]$ pwd
/mnt/netapp2/Store_uni/home/ulc/es/dac
[ulcesdac@login211-1 dac]$ |
```

Fuente: [CESGA - Portal de usuarios](#)



# Descripción avanzada de Finisterrae III (FT3)

- Usamos el comando myquota para conocer el espacio disponible

```
[ulcesdac@login211-1 ~]$ myquota

- HOME and Store filesystems:
-----
Filesystem      space  quota  limit  files  quota  limit
10.117.49.201:/Home_FT2  7966M  10005M  10240M  48959  100k   101k
10.117.49.201:/Store_uni 96361M 499G   500G   280k   300k   301k
10.117.49.101:/Home_BD   3008M  800G   1024G   60     4295m  4295m

- LUSTRE filesystem:
-----
Filesystem  used  quota  limit  grace  files  quota  limit  grace
/mnt/lustre/scratch
           12.6G   3T    3.5T    -    86665  200000  240000  -
```

Fuente: [CESGA - Portal de usuarios](#)



# Pruebas de rendimiento: archivos grandes

```
[ulcesdac@c206-1 dac]$ cd $HOME
[ulcesdac@c206-1 ~]$ dd if=/dev/zero of=./test1.img bs=1G count=10 oflag=dsync
dd: error writing './test1.img': Disk quota exceeded
4+0 records in
3+0 records out
3221225472 bytes (3.2 GB, 3.0 GiB) copied, 9.29581 s, 347 MB/s
[ulcesdac@c206-1 ~]$ cd $STORE
[ulcesdac@c206-1 dac]$ dd if=/dev/zero of=./test1.img bs=1G count=10 oflag=dsync
10+0 records in
10+0 records out
10737418240 bytes (11 GB, 10 GiB) copied, 21.4888 s, 500 MB/s
[ulcesdac@c206-1 dac]$ cd $LUSTRE
[ulcesdac@c206-1 dac]$ dd if=/dev/zero of=./test1.img bs=1G count=10 oflag=dsync
10+0 records in
10+0 records out
10737418240 bytes (11 GB, 10 GiB) copied, 20.0192 s, 536 MB/s
[ulcesdac@c206-1 dac]$ cd $LUSTRE_SCRATCH
[ulcesdac@c206-1 3483424]$ dd if=/dev/zero of=./test1.img bs=1G count=10 oflag=dsync
10+0 records in
10+0 records out
10737418240 bytes (11 GB, 10 GiB) copied, 12.2554 s, 876 MB/s
```



# Pruebas de rendimiento: archivos pequeños

```
[ulcesdac@c206-1 dac]$ cd $HOME
[ulcesdac@c206-1 ~]$ dd if=/dev/zero of=./test1.img bs=512 count=50000 oflag=dsync
50000+0 records in
50000+0 records out
25600000 bytes (26 MB, 24 MiB) copied, 9.04545 s, 2.8 MB/s
[ulcesdac@c206-1 ~]$ cd $STORE
[ulcesdac@c206-1 dac]$ dd if=/dev/zero of=./test1.img bs=512 count=50000 oflag=dsync
50000+0 records in
50000+0 records out
25600000 bytes (26 MB, 24 MiB) copied, 8.89278 s, 2.9 MB/s
[ulcesdac@c206-1 dac]$ cd $LUSTRE
[ulcesdac@c206-1 dac]$ dd if=/dev/zero of=./test1.img bs=512 count=50000 oflag=dsync
50000+0 records in
50000+0 records out
25600000 bytes (26 MB, 24 MiB) copied, 26.607 s, 962 kB/s
[ulcesdac@c206-1 dac]$ cd $LUSTRE_SCRATCH
[ulcesdac@c206-1 ~]$ dd if=/dev/zero of=./test1.img bs=512 count=50000 oflag=dsync
50000+0 records in
50000+0 records out
25600000 bytes (26 MB, 24 MiB) copied, 8.74333 s, 2.9 MB/s
```

# Descripción avanzada de Finisterrae III (FT3)

- Módulos relevantes:
  - Módulos disponibles consultables con el comando: *module avail*
  - Cargables con el comando: *module load nombre\_del\_modulo*
  - Módulos relevantes:
    - **intel:** Diversas herramientas de intel, algunas relacionadas con IA
    - **tensorflow:** Framework de google para IA
    - **transformers:** Varias herramientas de IA relacionadas con NLP (procesamiento del lenguaje natural), incluido TensorFlow y Pytorch
    - **pytorch:** Popular framework de IA
    - **torchvision:** Conjuntos de datos, y modelos para vision por computador
    - **scikit-learn:** Librería que implementa diversos algoritmos de *Machine Learning*, implementados con numpy, scipy y matplotlib
    - **r-keras:** API en R para TensorFlow keras



# Retos de entrenamientos distribuidos en FT3

- Utilizar el conjunto de herramientas adecuado/deseado
  - Módulos disponibles en el FT3
  - Entornos conda
    - `module load cesga/system miniconda3/22.11`
- Utilizar el sistema de ficheros más adecuado para almacenar cada componente de un entrenamiento distribuido
  - Entornos CONDA: \$STORE
  - Ficheros de dataset: \$STORE o \$LUSTRE según características
- Evitar pasarnos de la cuota disponible en cada sistema de ficheros
  - Visualizable con el comando `myquota`
  - Elementos que ocupan espacio
    - Entornos conda (espacio + nº ficheros)
    - Conjuntos de datos (espacio + nº ficheros)
    - Puntos intermedios del modelo (parámetros en un punto de entrenamiento)
    - Información de profiling y checkpointing de los modelos
      - Herramientas como Ray generan una gran cantidad de información en cada ejecución

# Retos de entrenamientos distribuidos en FT3

- Utilizar el mejor hardware para los entrenamientos
  - CPUs multinúcleo de Intel
    - Reservar un número adecuado de GPUs
  - Suficiente cantidad de memoria RAM
    - Tener en cuenta el tamaño del modelo + tamaño de batch
  - Utilizar las aceleradores A100 disponibles en el FT3
    - Dos por nodo
      - 40GB por GPU
    - Varios nodos
  - Configurar una estrategia de entrenamiento distribuido adecuada
    - Paralelismo de datos
    - Paralelismo de modelo
    - Paralelismo de tensor
    - Paralelismo de pipeline
    - Estrategias híbridas
    - Estrategias avanzadas

# Retos de entrenamientos distribuidos en FT3

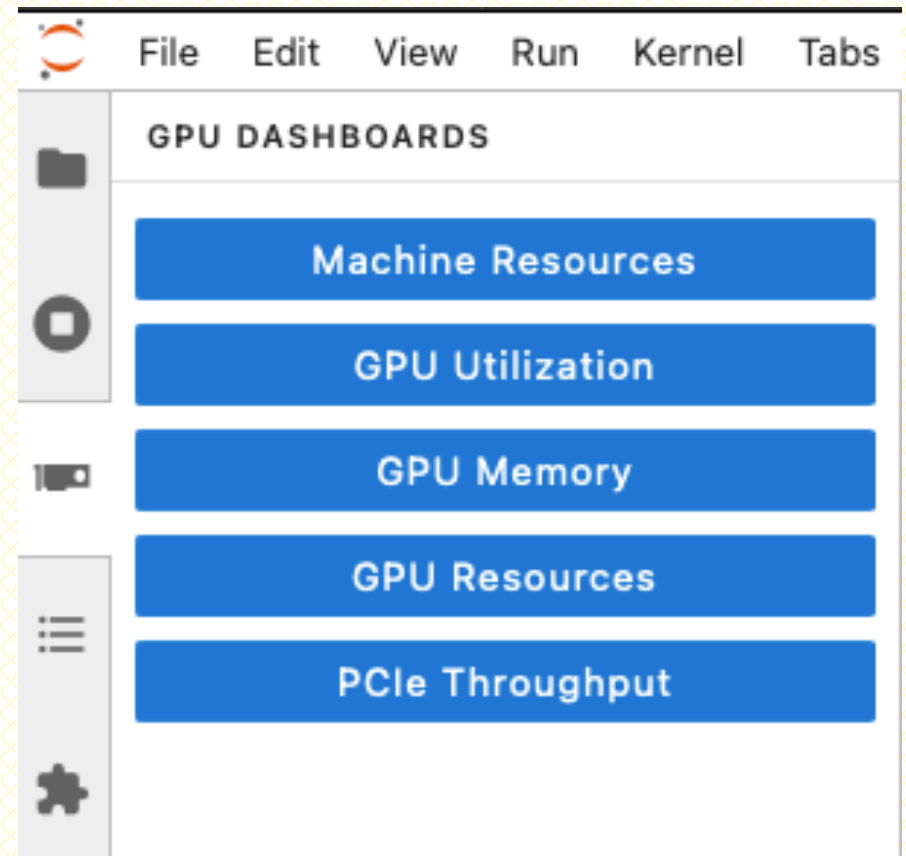
- Cargar los datos de forma colaborativa utilizando varios *workers* (trabajadores)
  - Existen formas estándar de hacer disponibles en las herramientas que se verán en este curso
- Asegurarnos de utilizar la versión más eficiente de los kernels computacionales
  - oneDNN (Intel)
  - cuDNN (CUDA)
    - Mixed-precision para activar los Tensor Cores (automatic mixed precision)



# Herramientas de Profiling

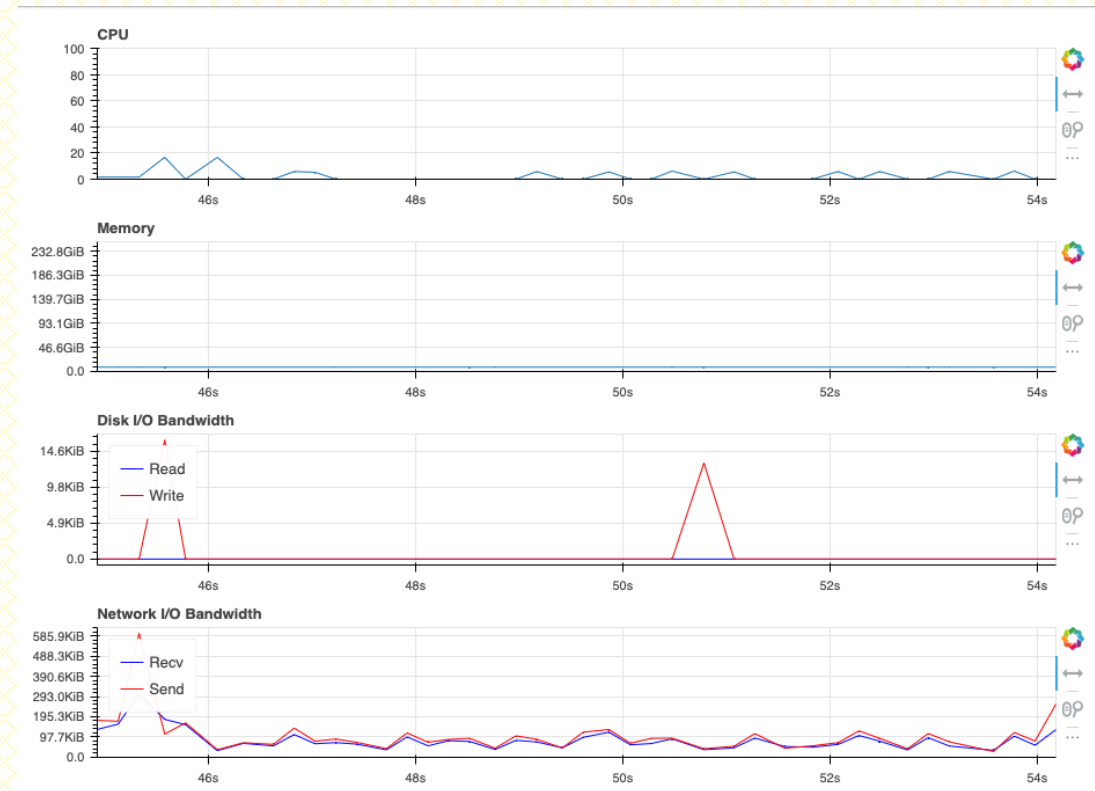
# Herramientas de profiling del uso de recursos

- NVBoard: Es una extensión de jupyterlab que nos permite observar en tiempo real la ocupación de los recursos de la máquina durante la ejecución de un código
  - CPU
  - Memoria
  - I/O
    - Memoria
    - Red
  - GPU
    - Utilización
    - Memoria
    - PCIe throughput



# Herramientas de profiling del uso de recursos: nvboard

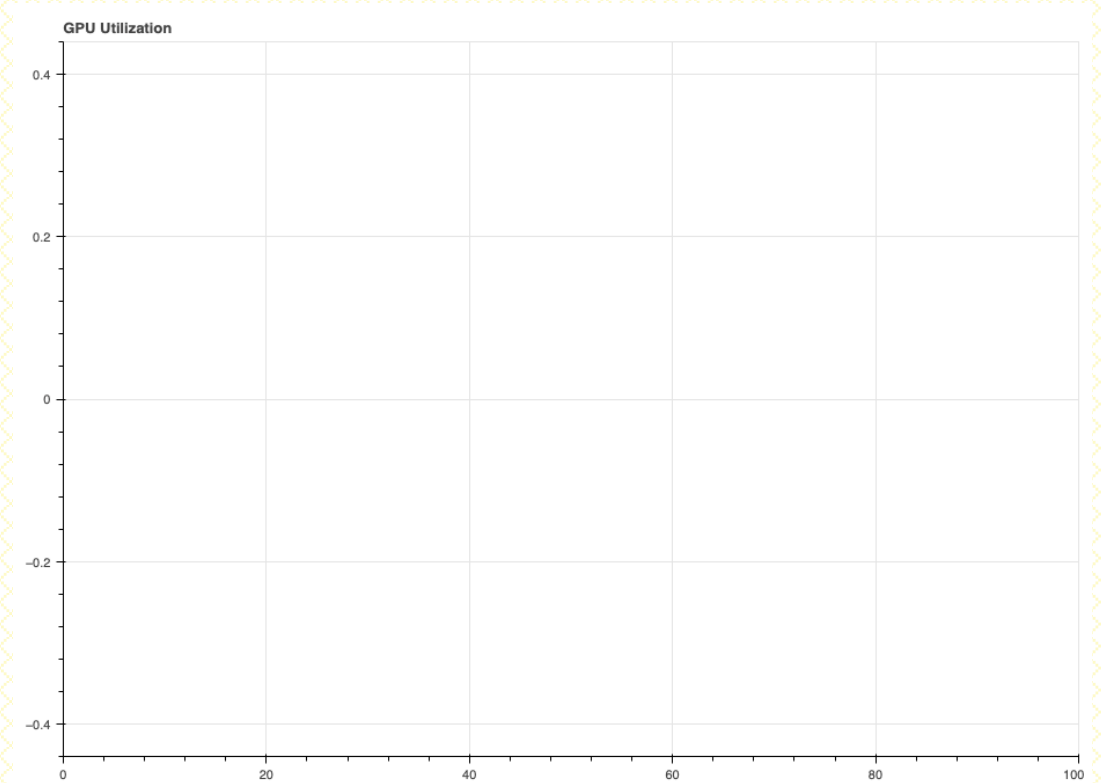
- Vista “machine resources”





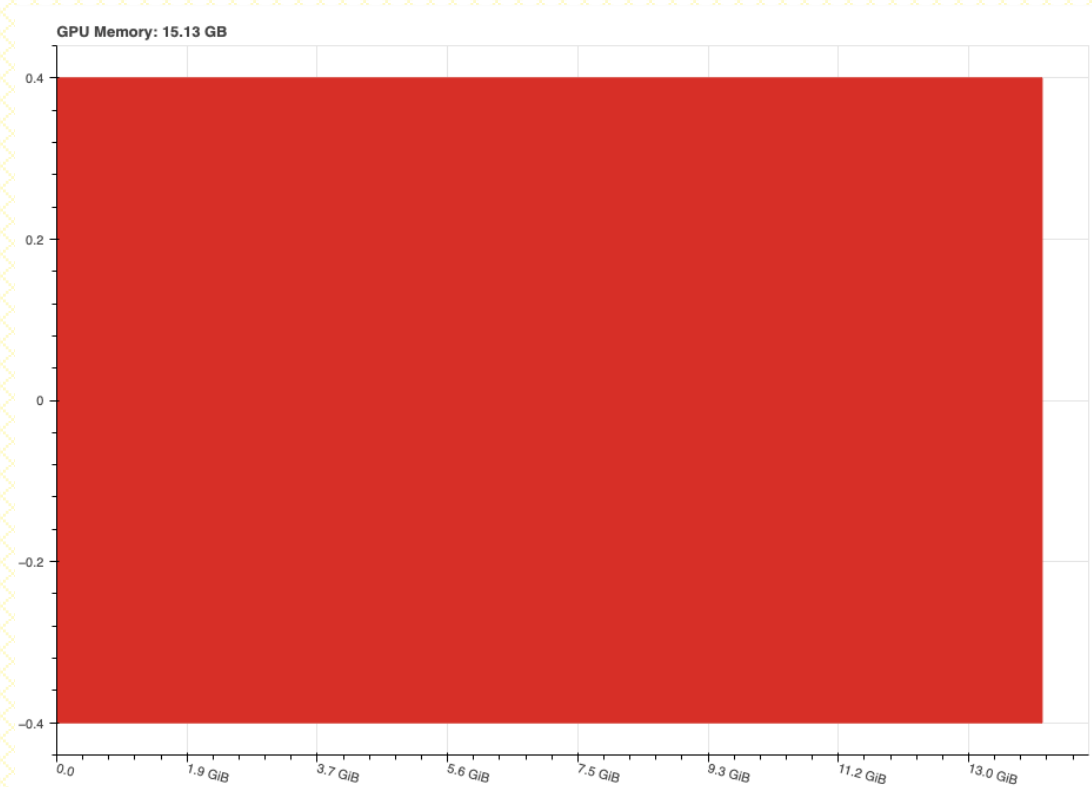
# Herramientas de profiling del uso de recursos: nvboard

- Vista “GPU utilization”



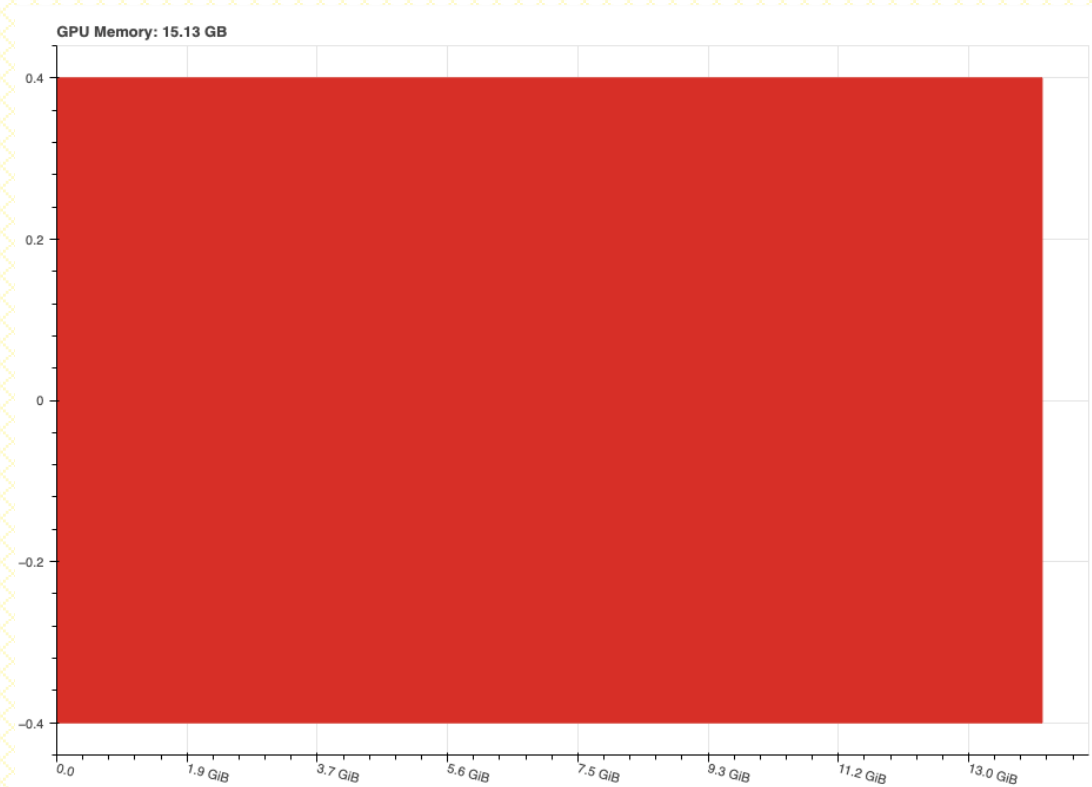
# Herramientas de profiling del uso de recursos: nvboard

- Vista “GPU memory”



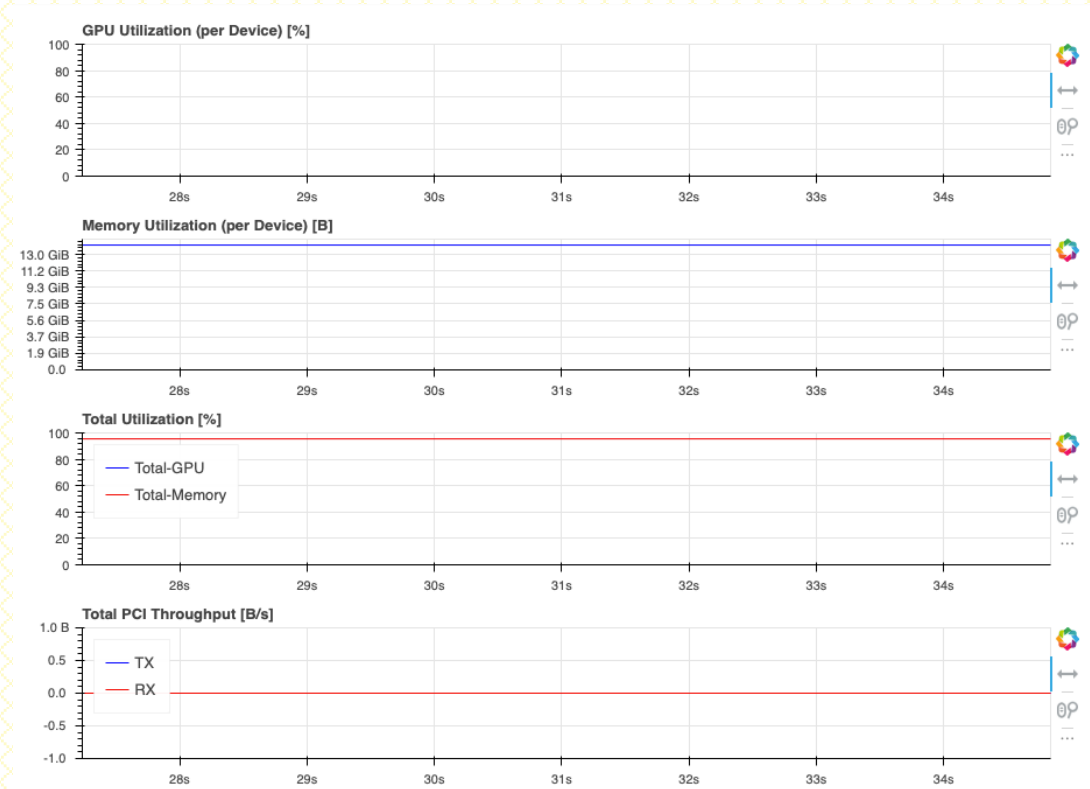
# Herramientas de profiling del uso de recursos: nvboard

- Vista “GPU memory”



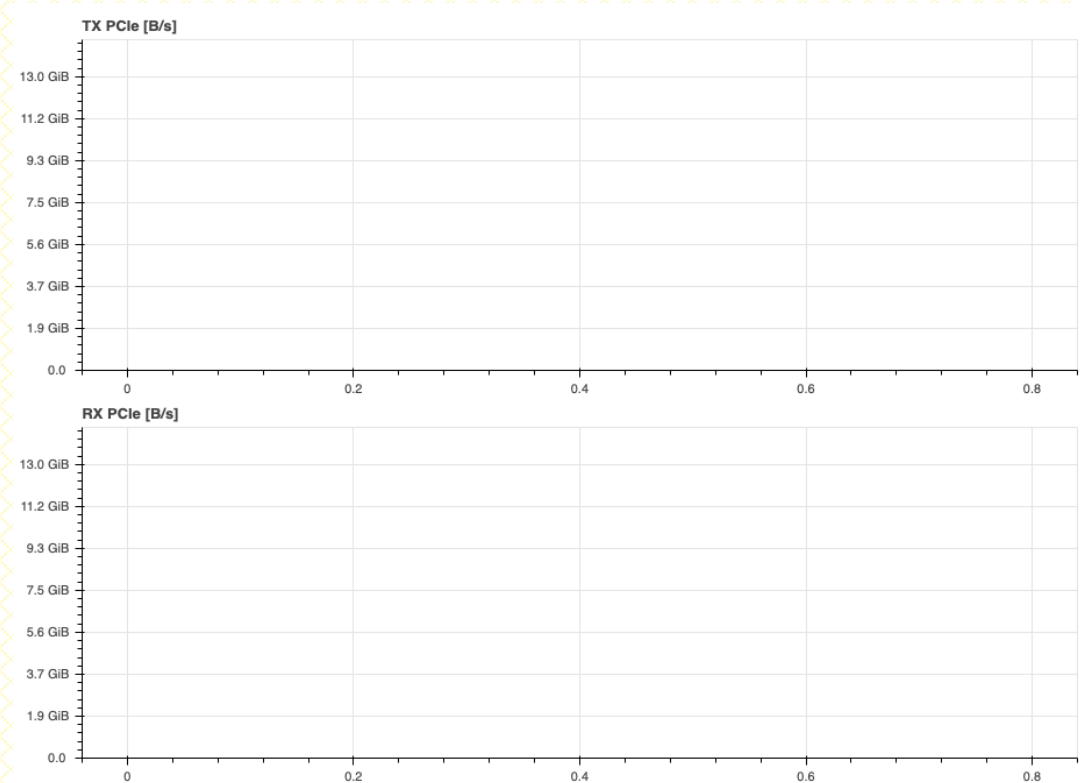
# Herramientas de profiling del uso de recursos: nvboard

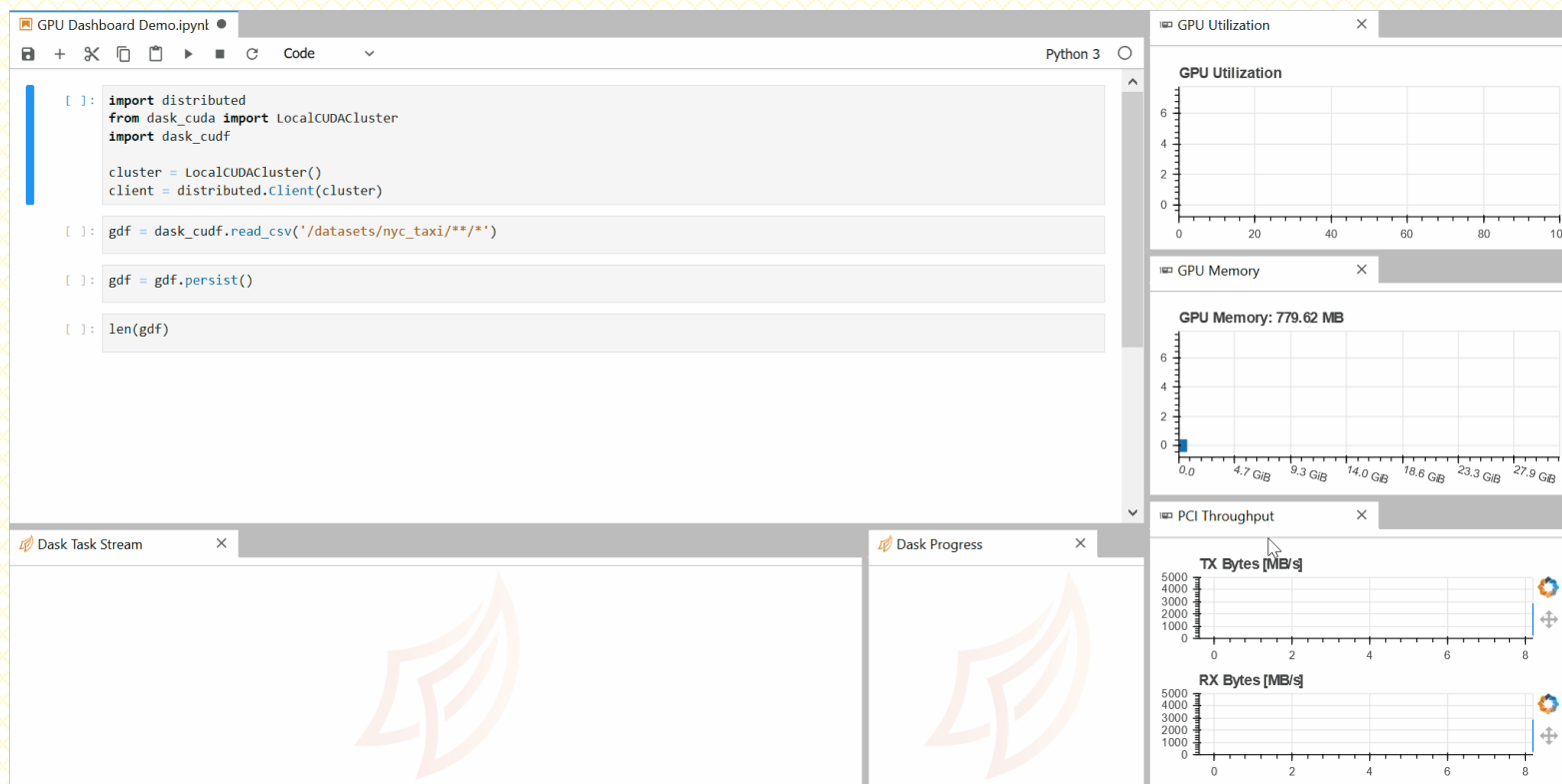
- Vista “GPU resources”



# Herramientas de profiling del uso de recursos: nvboard

- Vista “PCIe resources”





Fuente: <https://github.com/rapidsai/jupyterlab-nvdashboard>



# Herramientas de profiling del uso de recursos: TensorBoard

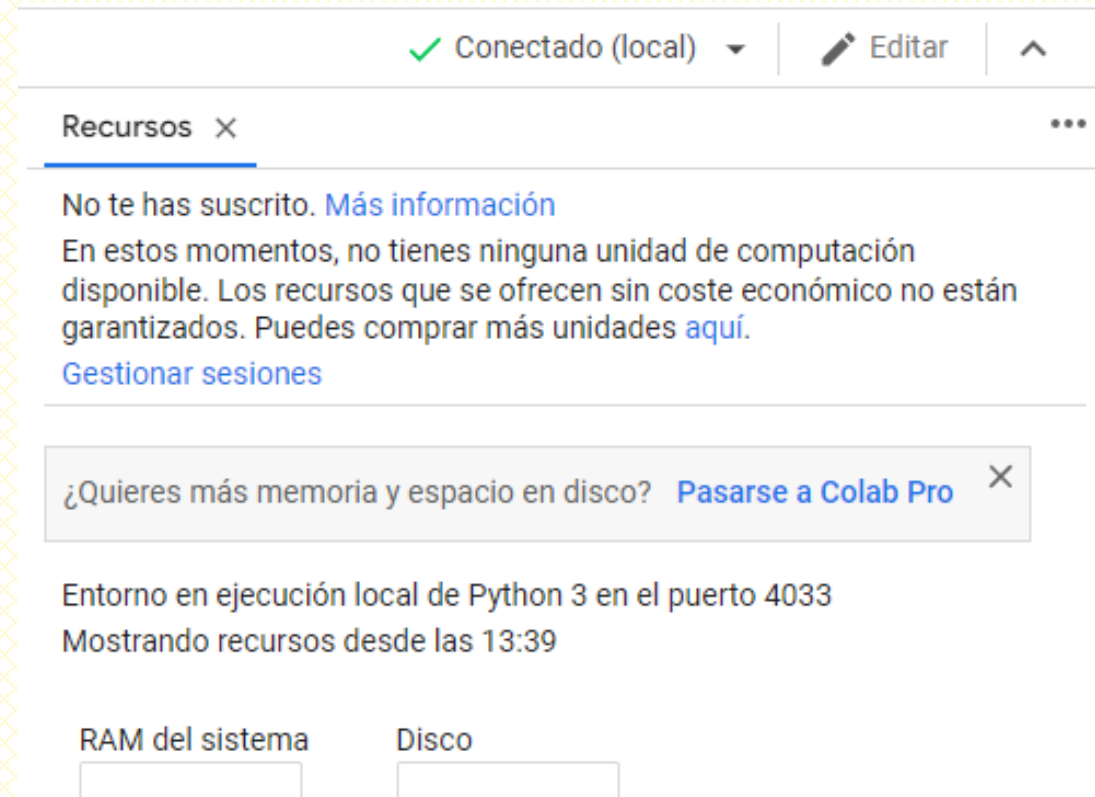
- TensorBoard es un conjunto de herramientas de visualización para ML
  - Soporte para TF y Pytorch
  - Visualización de la evolución de métricas como: loss y accuracy
  - Visualización del grafo del modelo
  - Visualización de histogramas de pesos, bias y otros tensores mientras evolucionan en el tiempo
  - ...
  - **Profiling del rendimiento del proceso de entrenamiento**

Fuente: [Get started with TensorBoard | TensorFlow](#)



## Herramientas de profiling del uso de recursos: TensorBoard

- Probar la conexión “local” de un notebook alojado en Google Colab
- Elegir Conectarse a un entorno de ejecución local
- Seguir las instrucciones para conectarse al Jupyter en ejecución en el FT3



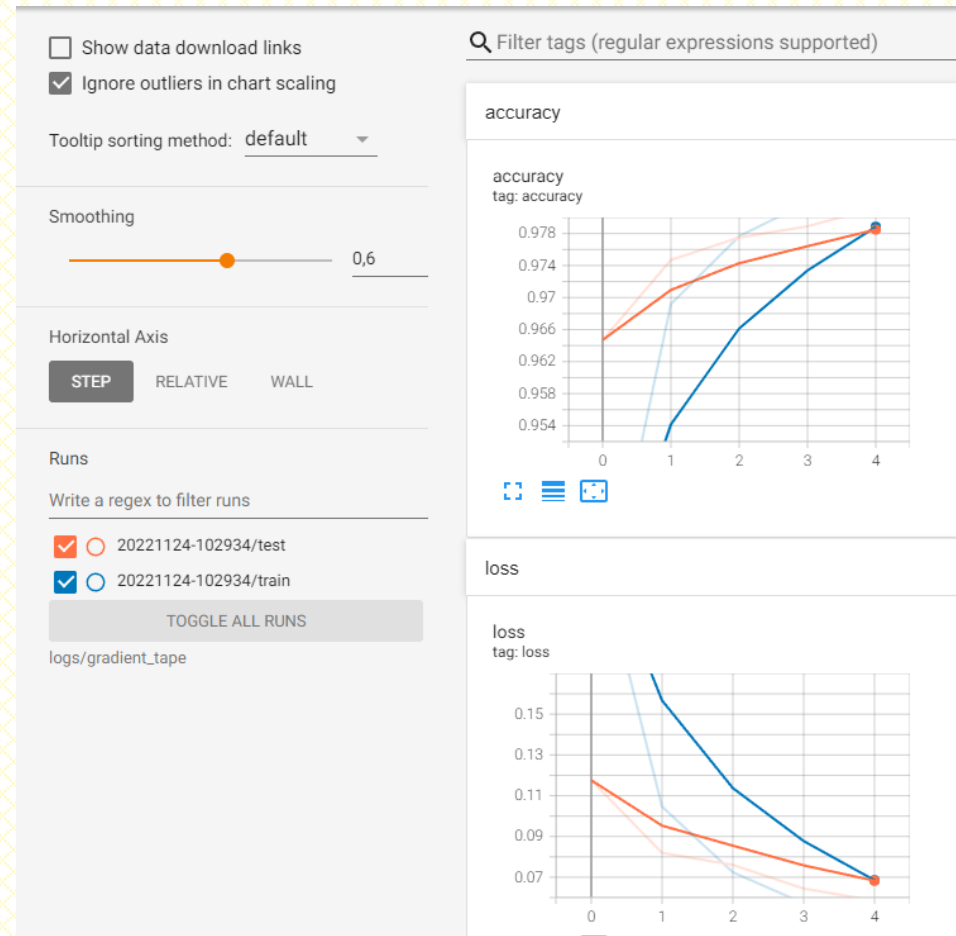
The screenshot displays the Google Colab interface. At the top, a status bar indicates 'Conectado (local)' with a green checkmark, alongside an 'Editar' button and an upward arrow. Below this, a tab labeled 'Recursos' is active. The main content area shows a message: 'No te has suscrito. Más información'. It explains that no computational units are currently available and that free resources are not guaranteed, with a link to 'aquí' for purchasing more units. A 'Gestionar sesiones' link is also present. A grey notification box asks '¿Quieres más memoria y espacio en disco?' with a 'Pasarse a Colab Pro' link and a close button. Below the notification, it states 'Entorno en ejecución local de Python 3 en el puerto 4033' and 'Mostrando recursos desde las 13:39'. At the bottom, there are two progress bars: 'RAM del sistema' and 'Disco', both currently showing 0% usage.



# Herramientas de profiling del uso de recursos: TensorBoard

- TensorBoard es un conjunto de herramientas de visualización para ML
  - Visualización de la evolución de métricas como: loss y accuracy

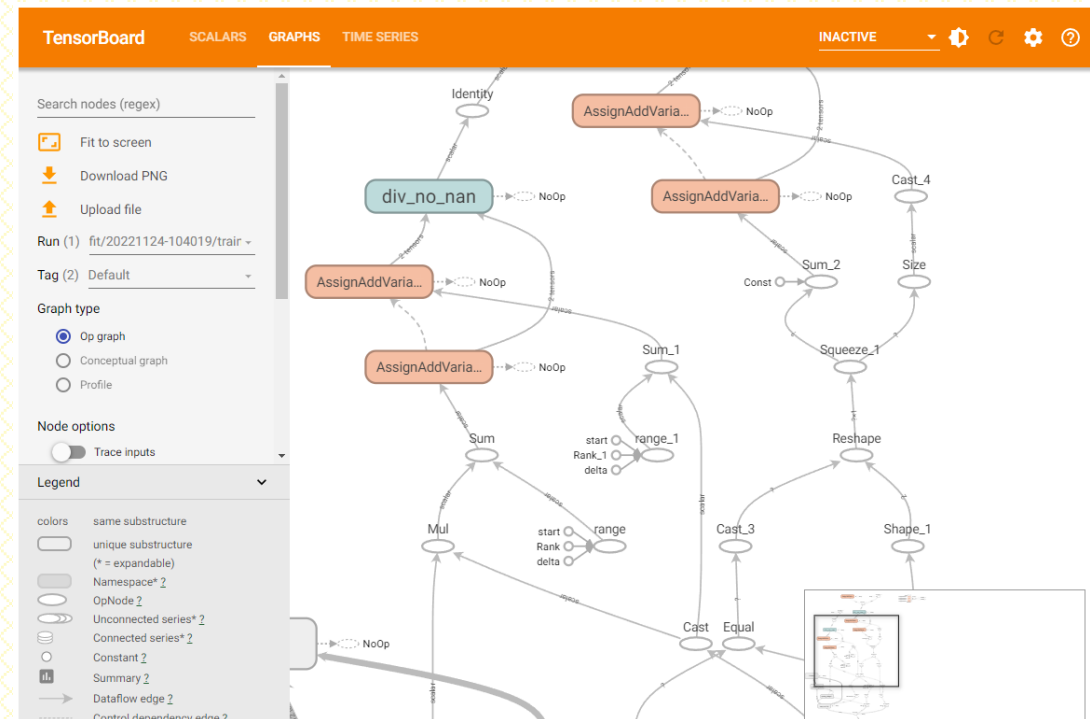
*Fuente:* [TensorBoard Scalars: Logging training metrics in Keras](#) | [TensorFlow](#)



# Herramientas de profiling del uso de recursos: TensorBoard

- TensorBoard es un conjunto de herramientas de visualización para ML
  - Visualización del grafo del modelo

*Fuente:* [Examining the TensorFlow Graph | TensorBoard](#)

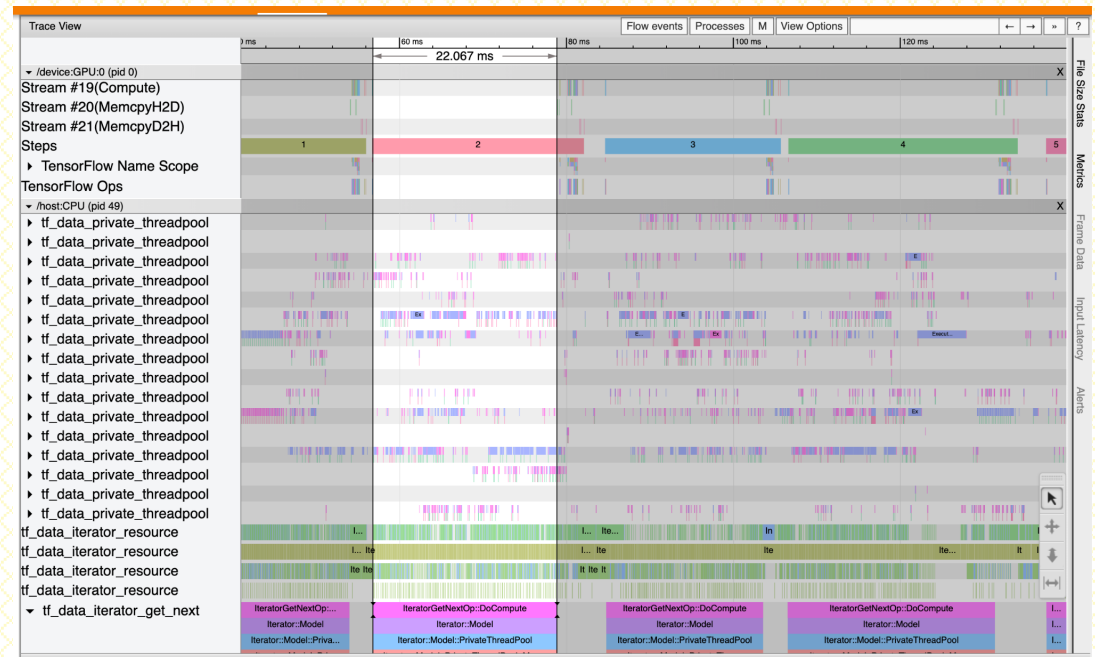


# Herramientas de profiling del uso de recursos: TensorBoard

- TensorBoard es un conjunto de herramientas de visualización para ML
  - **Profiling del rendimiento del proceso de entrenamiento**

*Fuente:*

[tensorboard\\_profiling\\_keras.ipynb](https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/tensorboard/tensorboard_profiling_keras.ipynb) - Colaboratory (google.com)



# Entorno

- Instalación del entorno. En ft3 con conda unpack
  - Conectarse a ft3
  - Conectarse a login1 (ssh login1)
  - Cd \$STORE
  - git clone <https://github.com/diegoandradecanosa/Cesga2023Courses>
  - cd Cesga2023Courses/pytorch\_dist/scripts
  - ./setupConda.sh
- Podemos usar el siguiente script para activar el entorno  
source activateconda.sh