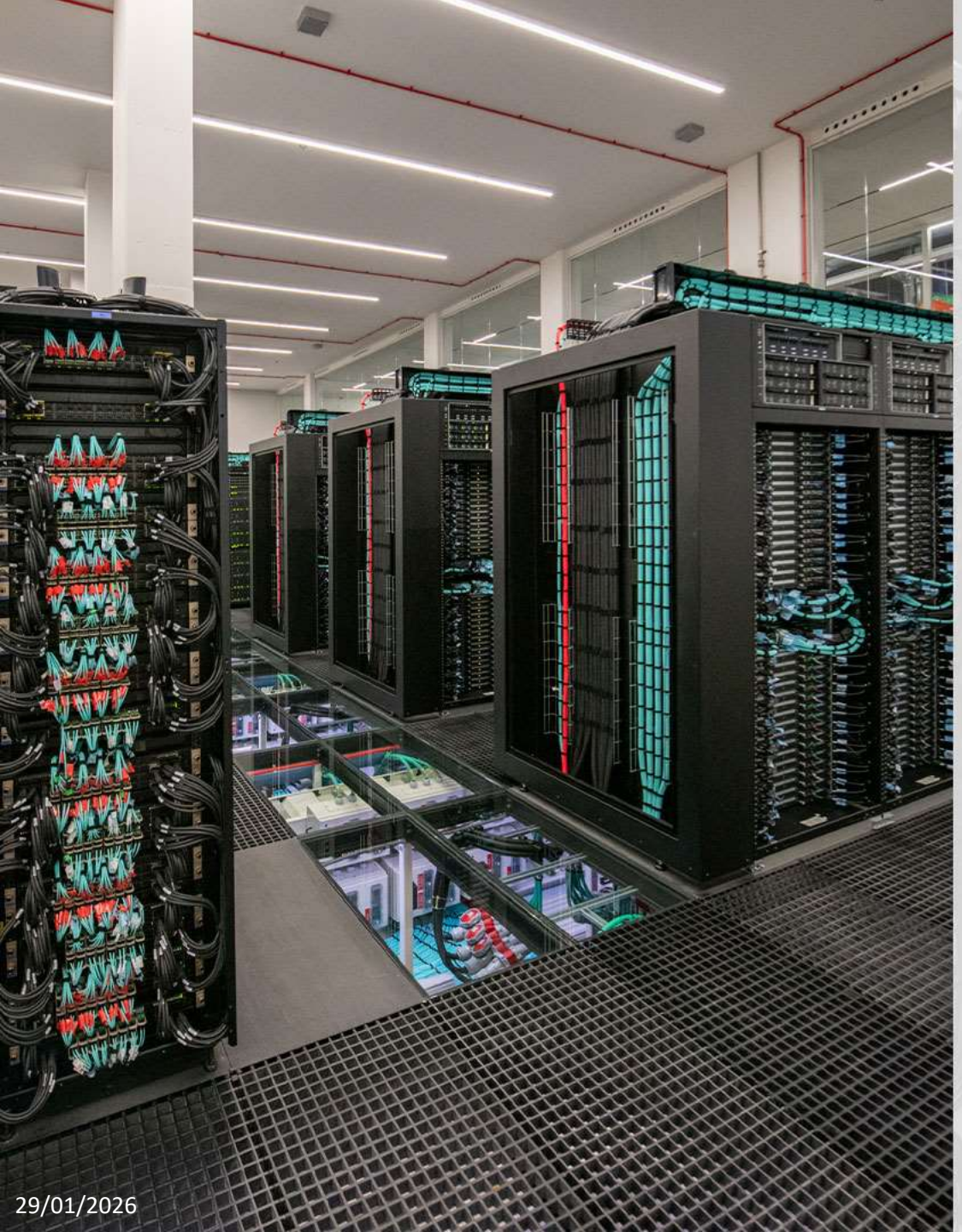


BSC AI Factory

MareNostrum 5 Yapısı ve Kullanımı: MN5 101

Bahar Görgün

bahar.gorgun@tubitak.gov.tr

TÜBİTAK ULAKBİM, Network Technologies Department,

TRUBA User Support Team

BSC AI Factory Partners

Affiliated Entities

İÇERİK

- Marenostrum 5 Giriş
- Maresnostrum GPP ve ACC Nodeları
- HPC İş Akışı
- Marenostrum 5’te İş Gönderme

BSC AI Factory

TÜBİTAK

PRE-EXASCALE SİSTEM - MARENOSTRUM 5

- 50% EuroHPC JU – 50% katılımcı ülkeler (İspanya %35.13, Türkiye %9.87, Portekiz %5)

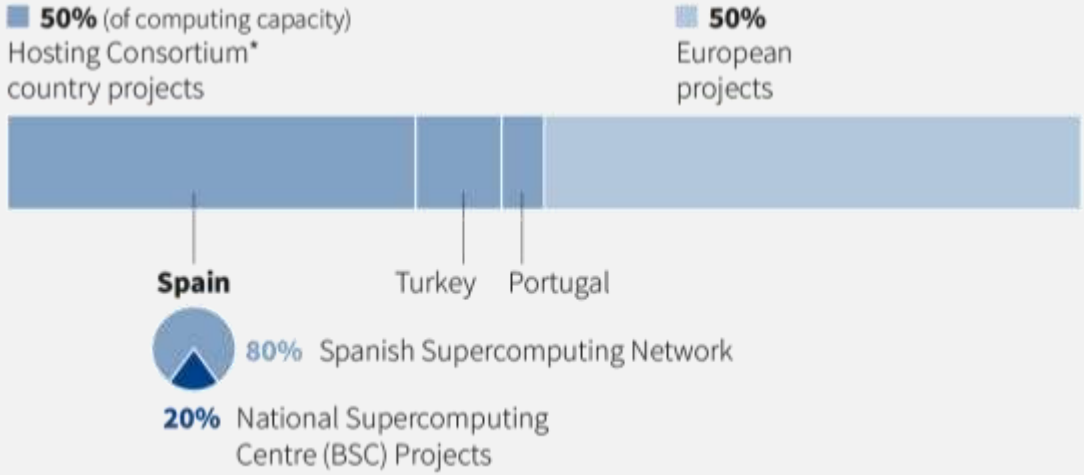
Investors:

Hosting Consortium* Spain Portugal Türkiye

European Commission

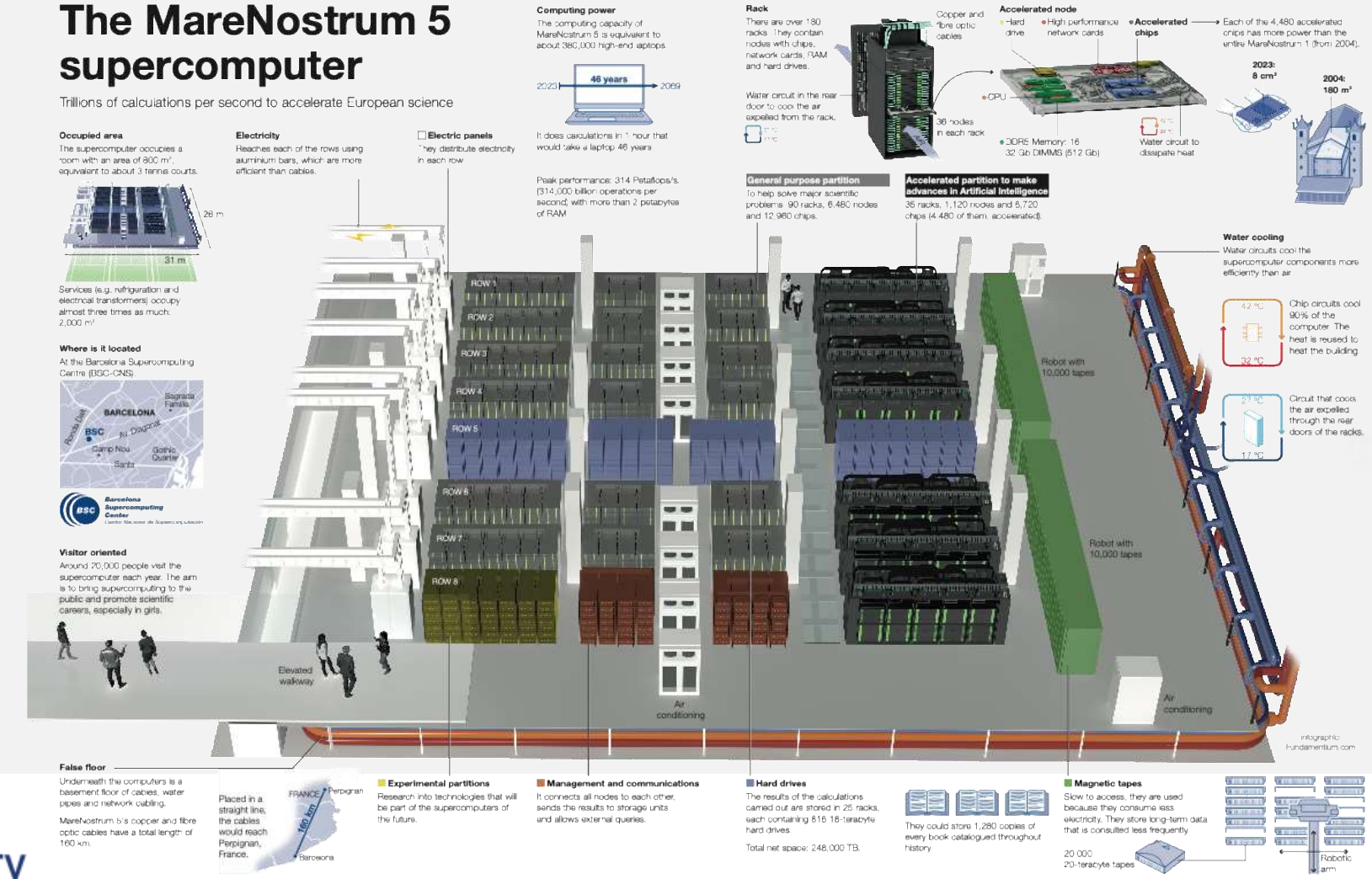
It is the largest European investment in scientific infrastructure in Spain

Free access to the most valuable research projects for society, chosen by external committees.



TeraFLOPS: Saniyede bir trilyon (10^{12}) kayan nokta (floating-point) işlemi gerçekleştirmek demektir. (Sıradan bir dizüstü bilgisayar yaklaşık 1 teraFLOPS'a kadar işlem yapabilir.)
Petascale: Saniyede en az 10^{15} kayan nokta işlemi yapabilme kapasitesidir (1 petaFLOPS).
Pre-exascale: 100 petaFLOPS'tan fazla ve 1 exaFLOPS'tan az işlem yapabilme yeteneğidir.
Exascale: Saniyede en az 10^{18} kayan nokta işlemi yapabilme kapasitesidir (1 exaFLOPS).

MN5 Teknik Özellikler



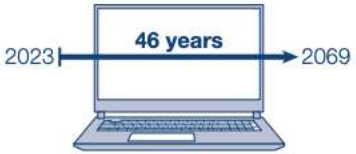
BSC AI Factory

TÜBİTAK

MN5 Teknik Özellikler-GPP

Computing power

The computing capacity of MareNostrum 5 is equivalent to about 380,000 high-end laptops.

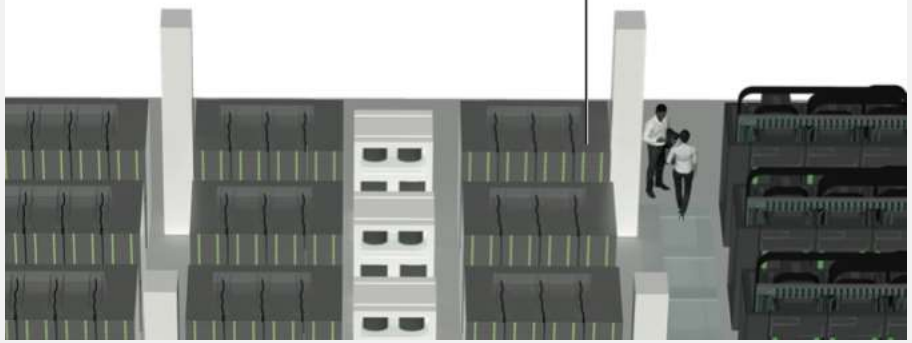


It does calculations in 1 hour that would take a laptop 46 years.

Peak performance: 314 Petaflops/s. (314,000 billion operations per second) with more than 2 petabytes of RAM.

General purpose partition

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



BSC AI Factory

TÜBİTAK

GPP Nodeları

Node type	Node count	Cores per node	Main memory per node	Usable memory per core
GPP	6,192	112	256 GiB	2 GB
GPP-HighMem	216	112	1,024 GiB	9 GB
GPP-Data	10	112	2,048 GiB	18 GB
GPP-HBM	72	112	128 GiB	1 GB

BSC AI Factory

TÜBİTAK

Marenostrum 5 (MN5) HPC Sistemine Erişim

Marenostrum 5 kaynaklarına erişim, tüm veri trafiğinin şifrelendiği güvenli bir protokol olan **SSH** üzerinden sağlanır.

Linux ve macOS

- Sistemle birlikte gelen yerleşik **OpenSSH** istemcisi kullanılır.
- Erişim için **Terminal** arayüzü tercih edilir.

Windows

- **PuTTY:** Yaygın olarak kullanılan açık kaynaklı SSH istemcisi.
- **MobaXterm:** X11 yönlendirme ve dosya transferi desteği sunan gelişmiş alternatif.
- **Windows PowerShell / CMD:** Güncel sürümlerde yer alan OpenSSH desteği ile doğrudan komut satırı erişimi.

Marenostrum 5 (MN5) HPC Sistemine Erişim

```
$ ssh {kullanici_adi}@glogin1.bsc.es
```

GPP Nodelarına erişim

```
$ ssh {kullanici_adi}@alogin1.bsc.es
```

ACC Nodelarına erişim

```
$ ssh {kullanici_adi}@transfer1.bsc.es
```

Storage ve veri transferi Nodelarına erişim

BSC AI Factory

TÜBİTAK

MN5 Veri Transferi

⚠ Önemli Kısıtlamalar

•**CPU Zaman Sınırı:** Login (giriş) düğümlerinde CPU kullanım limiti **5 dakikadır**. Uzun süreli işlemler için hesaplama düğümleri kullanılmalıdır.

•**Ağ Erişimi:** MN5 üzerinden dış dünyaya doğrudan giden bağlantılara izin verilmez.

🔄 Veri Aktarımı (Data Transfer)

Dosya transfer işlemleri için özel aktarım düğümü kullanılmalıdır: **transfer1.bsc.es**

```
$ scp localfile username@transfer1.bsc.es:/path/remote/dir
$ scp username@transfer1.bsc.es:/path/remote/file localdir
```

```
$ rsync -av /path/localdir username@transfer1.bsc.es:/path/remotedir
```

BSC AI Factory

TÜBİTAK

Dosya Sistemi

```
$ bsc_quota
```

Filesystem	Type	Usage	Quota	Limit	In doubt	Grace	Files	In doubt
gpfs_home	USR	4.42 MB	80.00 GB	84.00 GB	0.00 KB	None	23	0
gpfs_projects	GRP	154.75 GB	1000.00 GB	1.03 TB	0.00 KB	None	32411	0
gpfs_scratch	GRP	144.00 KB	1000.00 GB	1.03 TB	0.00 KB	None	11	0

For information regarding /gpfs/tapes, run this command from a Storage 5 node (transfer[1..4].bsc.es).

General Parallel Filesystem (GPFS)

BSC AI Factory

TÜBİTAK

/gpfs/apps

```
$ ls /gpfs/apps/MN5/ACC

ACCELERATE        DAGSTER       HDF5VIEW      NANO              ROSETTAFOLD
ADIOS2            DARSHAN       HECUBA        NCCL              SCALA
AEC               DATACLAY      HOOMD-BLUE    NCVIEW            SCALAPACK
ALPHAFILL         DATE          HOPR          NEDIT             SCALASCA
ALPHAFOLD         DEEPMD-GNN    HPCVIEWER     NETCDF            SCOREP
ALYA              DEEPMD-KIT    HPCX          NEXTFLOW          SIESTA
ALYA-MPIO-TOOLS   DMR           HYPRE         NGCCLI            SINGULARITY
AMBER             DORADO        I-PI          NIM               SLEPC
AMGX              DRISHTI-IO    JQ            NVBANDWIDTH       SOWING
ANACONDA          DSSP          JULIA         NVIDIA-HPC-SDK    SPACK
ASC               DUALSPHYSICS  KOKKOS        NVSHMEM           SPGLIB
BAZEL             DXT-EXPLORER  LAMMPS        OCTOPUS           SQLITE3
BERKELEYGW        EASYBUILD     LAPACK        OLLAMA            SRC
BIGDFT            ECCODES       LIBAIO        ONEAPI            SUNDIALS
BIN_UTILS         EIGEN         LIBBEEF       OPENBLAS          SUPPORT
BOOST             ELPA          LIBCIFPP      OPENCV            TENSORFLOW
BSCTOOLS          EMBOSS        LIBCURL       OPENFE            TENSORRT
BSC_AFFINITY      EVOBIND       LIBFABRIC     OPENMPI           TESSERACT
BTOP              FALL3D        LIBFYAML      OPENVDB           TINKER-HP
CAIROSVG          FFMPEG        LIBJANSSON    ORCA              TORCHFORT
                                                                  ....
```

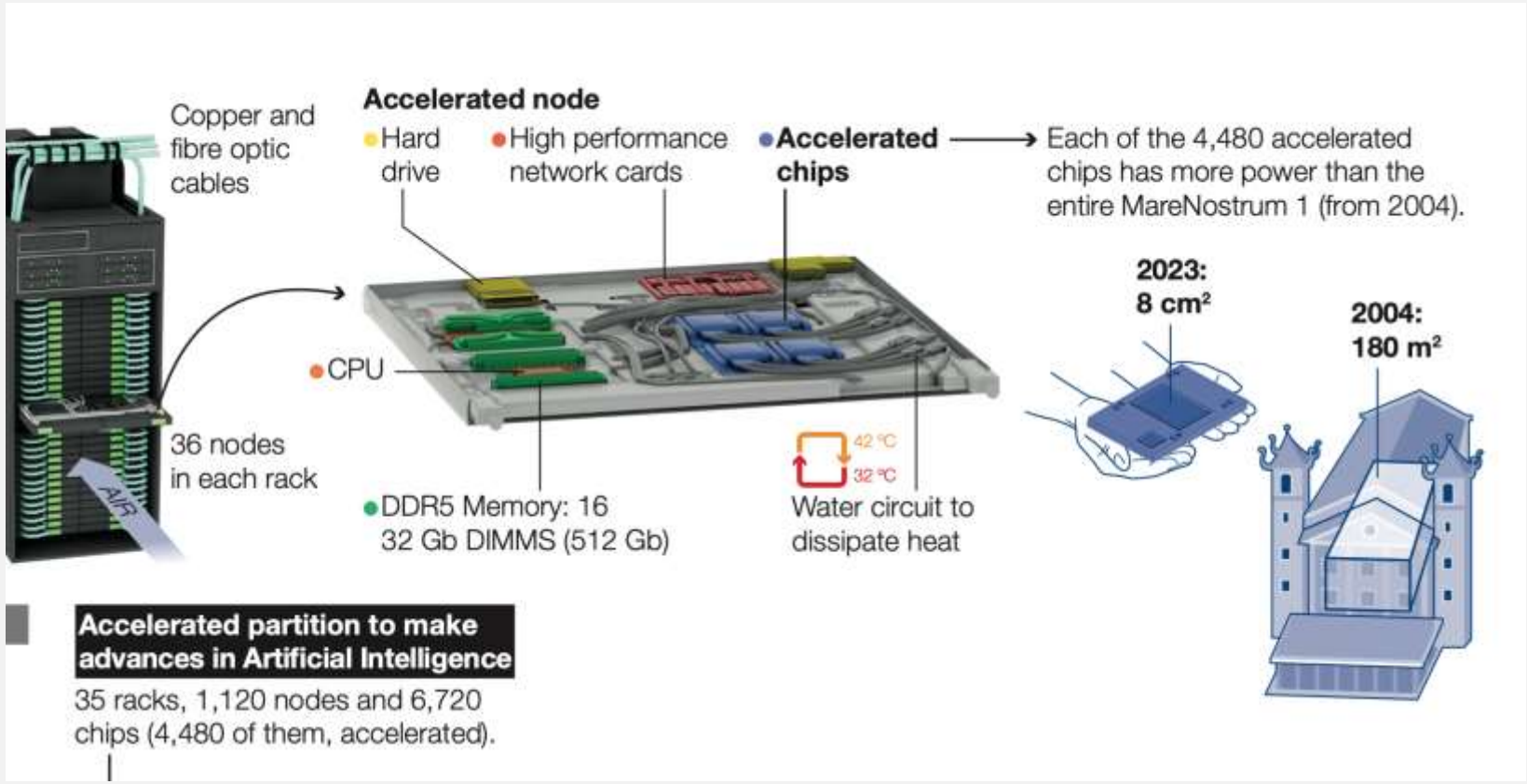
BSC AI Factory

TÜBİTAK

MN5 Yüklü Modüller

Komut	Kullanım Örneği	Açıklama
avail	module avail [program]	Mevcut tüm modülleri listeler
list	module list	Halihazırda yüklü olan modülleri gösterir
load	module load gcc/5.1.0	Belirli bir modülü sisteme yükler
unload	module unload gcc	Yüklü olan bir modülü kaldırır
keyword	module keyword pytorch	Modül isminde geçenlere göre arama yapar.
purge	module purge	Tüm yüklü modülleri temizler

MN5 Teknik Özellikler - ACC



BSC AI Factory

TÜBİTAK

ACC Nodeları

- **Toplam Düğüm Sayısı:** 1.120 Adet
- **Toplam Hesaplama Birimi:** 680.960 (CPU + GPU toplamı)
- **Mimari:** Intel Xeon Sapphire Rapids ve NVIDIA Hopper GPU entegrasyonu

Bileşen	Detaylı Özellikler
İşlemci (CPU)	2x Intel Xeon Platinum 8460Y+ (40C, 2GHz) — **80 Çekirdek/Düğüm**
Grafik İşlemci (GPU)	4x NVIDIA Hopper H100 64GB HBM2
Bellek (RAM)	512GB DDR5 4800MHz (16x 32GB DIMM)
Depolama	480GB NVMe Yerel Depolama
Ağ Bağlantısı	4x ConnectX-7 NDR200 InfiniBand (**800Gb/s Bandwidth**)

BSC AI Factory

TÜBİTAK

Marenostrum 5 Yazılım Ortamı

Software type	MN5
Operating system	Red Hat Enterprise Linux
Compiler Suite	Intel OneAPI HPC Toolkit Nvidia SDK (PGI)
Numerical libraries	Intel MKL Nvidia SDK
Debugging/profiler tools	BSC Performance tools ARM DDT Nvidia SDK Intel OneAPI HPC Toolkit (vtune, ...)
Resource and workload manager	SLURM Only one Slurm cluster, with different partitions
Energy Efficiency and Power Management	EAR

İş Gönderme

1. **İş Betiği Hazırlama:** Kullanıcı, kaynak gereksinimlerini (CPU, GPU sayısı, sunucu sayısı, zaman…) ve çalıştırılacak komutları içeren bir .sh dosyası hazırlar.
2. **Kuyruğa Gönderim:** sbatch komutu ile dosya sisteme iletilir.
3. **Planlama (Scheduling):** SLURM, uygun kaynaklar boşaldığında işi en uygun düğümlere atar.
4. **Yürütme ve İzleme:** İş çalışırken standart çıktılar (stdout) ve hatalar (stderr) dosyalara kaydedilir.
5. **Tamamlanma:** İş bitiminde kaynaklar serbest bırakılır.

İş Gönderme

•**Kaynak Belirleme:** İhtiyaç duyulan CPU, bellek (RAM) ve zaman limitlerini (Wall clock time) önceden tanımlamanızı sağlar.

•**Hassas Kontrol:** İşin hangi kuyrukta (partition) ve kaç çekirdek ile çalışacağını netleştirir.

Verimlilik ve Düzen

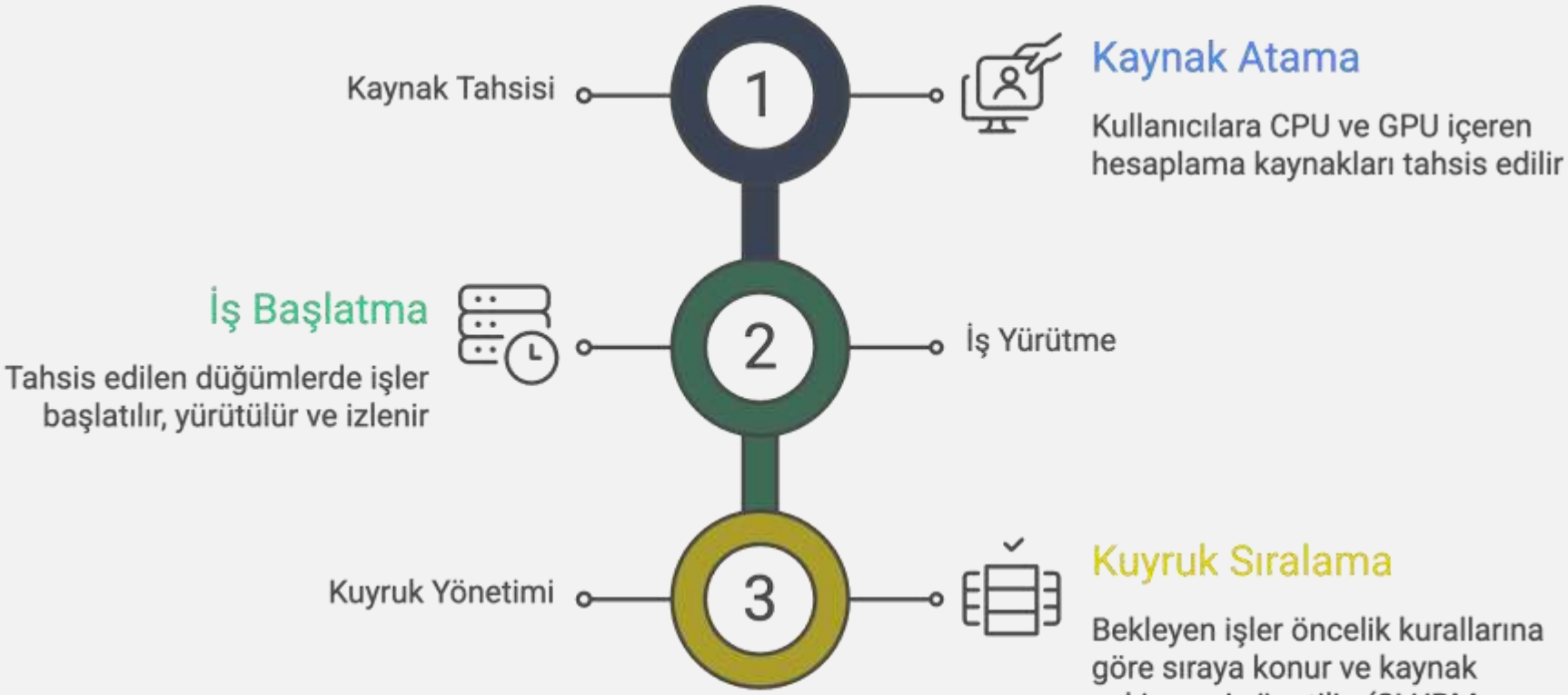
•**Tekrar Kullanılabilirlik:** Bir kez yazılan betik, parametreler değiştirilerek farklı veri setleri için defalarca kullanılabilir.

💻 **Teknik Yapı**

•**Esnek Kabuk Betikleri:** İş betikleri aslında standart **Shell Script** dosyalarıdır.

•**Özel İşaretleyiciler (Directives):** İşlemciye (Scheduler) talimat gönderen özel #SBATCH (veya ilgili sistemin) işaretçilerini içerir.

İş Gönderme


Kaynak Tahsisi
1
Kaynak Atama
Kullanıcılara CPU ve GPU içeren hesaplama kaynakları tahsis edilir
İş Başlatma
Tahsis edilen düğümlerde işler başlatılır, yürütülür ve izlenir
2
İş Yürütme
Kuyruk Yönetimi
3
Kuyruk Sıralama
Bekleyen işler öncelik kurallarına göre sıraya konur ve kaynak çekişmesi yönetilir. (SLURM Tarafından)

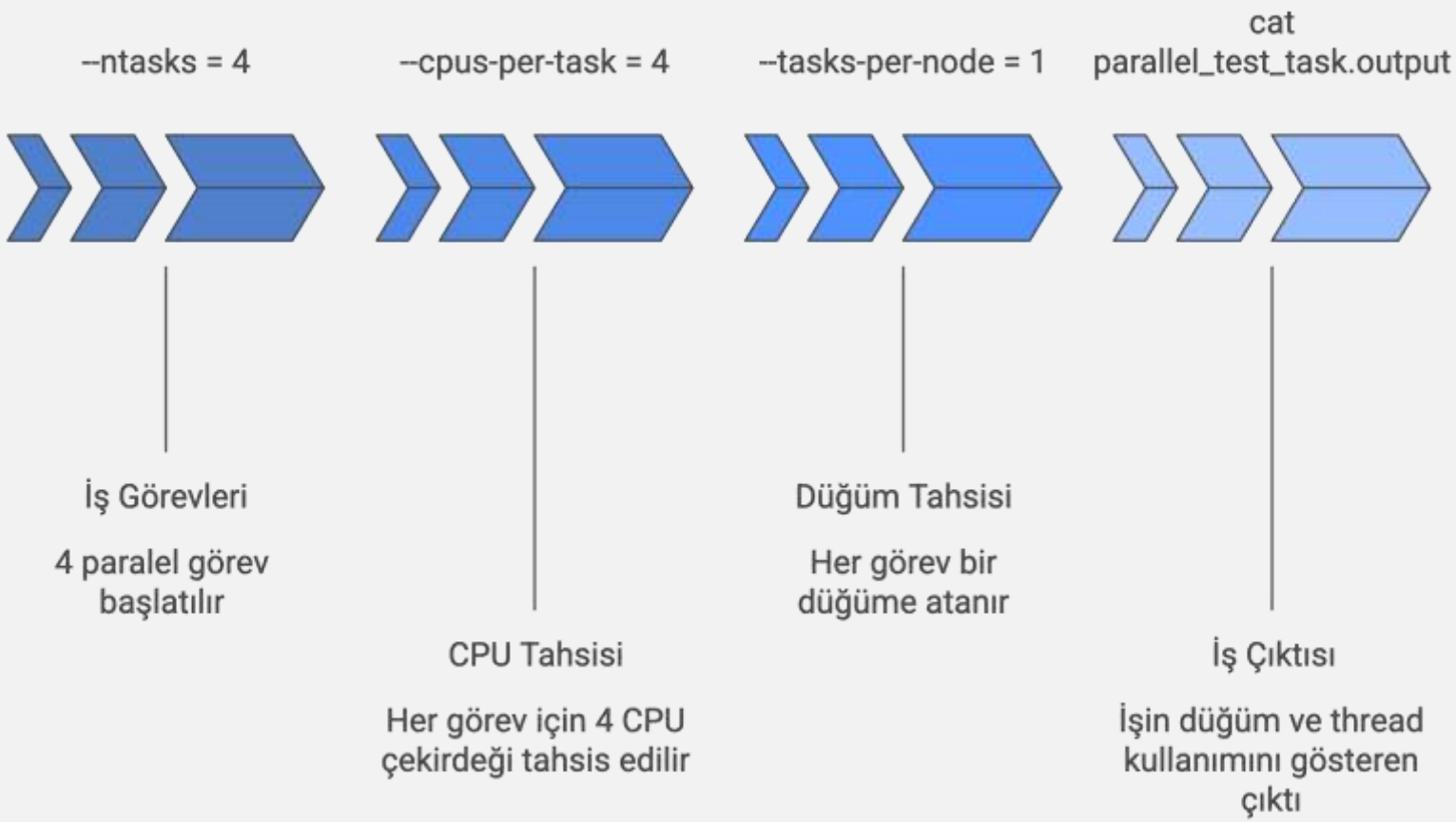
SLURM Scripti

```
#!/bin/bash
#SBATCH --time=00:02:00                  ---→ işin çalışacağı süre
#SBATCH --qos=gp_debug                   ---→ gönderileceği kuyruk
#SBATCH --account=etur71                 ---→ kullanıcı isminiz
#SBATCH --job-name=test_parallel         ---→ işin sistemde gözükeceği ismi
#SBATCH --output=mpi_%j.out
#SBATCH --error=mpi_%j.err
#SBATCH --ntasks=4                       --→ Paralel çalışacak iş sayısı
#SBATCH --cpus-per-task=4                --→ Her bir task için gereken CPU
#SBATCH --tasks-per-node=1

# İşlemcileri doğru hizalamak için (Pinning)
export SRUN_CPUS_PER_TASK=$SLURM_CPUS_PER_TASK

srun ./parallel_binary > parallel_test_task.output
```

SLURM İş Akışı



parallel_test_task.output

```
$cat parallel_test_task.output

[Rank 1 / 4] Node: gs04r3b40 -> 4 thread kullanıyor.
[Rank 0 / 4] Node: gs04r3b31 -> 4 thread kullanıyor.
[Rank 3 / 4] Node: gs04r3b58 -> 4 thread kullanıyor.
[Rank 2 / 4] Node: gs04r3b43 -> 4 thread kullanıyor.
```

ddp_or_fsdp.py

```python
import os
import torch
import torch.nn as nn
import torch.distributed as dist
import torch.multiprocessing as mp
from torch.nn.parallel import DistributedDataParallel as DDP
from torch.distributed.fsdp import FullyShardedDataParallel as FSDP

def setup(rank, world_size):
    os.environ["MASTER_ADDR"] = "localhost"
    os.environ["MASTER_PORT"] = "12355"
    dist.init_process_group("nccl", rank=rank, world_size=world_size)
    torch.cuda.set_device(rank)

def cleanup():
    dist.destroy_process_group()

def make_big_model():
    return nn.Sequential(
        nn.Linear(8192, 8192),
        nn.ReLU(),
        nn.Linear(8192, 8192),
        nn.ReLU(),
        nn.Linear(8192, 8192),
        nn.ReLU(),
        nn.Linear(8192, 1000),
    )
```

```python
# ---------------------------------------------------------------------
def run_ddp(rank, world_size, results):
    setup(rank, world_size)
    torch.cuda.reset_peak_memory_stats(rank)

    model = make_big_model().to(rank)
    mem_model = mb(rank)

    ddp_model = DDP(model, device_ids=[rank])
    optimizer = torch.optim.Adam(ddp_model.parameters(), lr=0.001)

    # 1 step eğitim
    x = torch.randn(32, 8192, device=rank)
    out = ddp_model(x)
    out.sum().backward()
    optimizer.step()

    torch.cuda.synchronize()
    mem_peak = torch.cuda.max_memory_allocated(rank) / 1024**2

    results[rank] = {"mem_model": mem_model, "mem_peak": mem_peak}
    cleanup()
# ---------------------------------------------------------------------
# FSDP testi
# ---------------------------------------------------------------------
def run_fsdp(rank, world_size, results):
    setup(rank, world_size)
    torch.cuda.reset_peak_memory_stats(rank)

    model = make_big_model().to(rank)
    fsdp_model = FSDP(model, device_id=rank)
    optimizer = torch.optim.Adam(fsdp_model.parameters(), lr=0.001)

    mem_after_wrap = mb(rank)

    x = torch.randn(32, 8192, device=rank)
    out = fsdp_model(x)
    out.sum().backward()
    optimizer.step()

    torch.cuda.synchronize()
    mem_peak = torch.cuda.max_memory_allocated(rank) / 1024**2

    results[rank] = {"mem_after_wrap": mem_after_wrap, "mem_peak": mem_peak}
    cleanup()
```

BSC AI Factory

TÜBİTAK

ddp_or_fsdp.slurm

```
#!/bin/bash
#############################
#SBATCH -J BSCAI
#SBATCH -n 1
#SBATCH --cpus-per-task=80
#SBATCH --error=bt-bscai-%j.err
#SBATCH --output=bt-bscai-%j.out
#SBATCH -D .
#SBATCH -t 00:10:00
#SBATCH --gres=gpu:4
#SBATCH --qos=acc_debug
#SBATCH --account=etur71
#SBATCH --exclusive
#SBATCH --hint=nomultithread

export SLURM_CPU_BIND=none
export PY_SCRIPT=./ddp_or_fsdp.py
export IMAGE=/apps/ACC/SINGULARITY/images/pytorch_22.09-py3.sif

module purge
module load singularity

export CUDA_VISIBLE_DEVICES="0,1,2,3"

time srun singularity exec --nv --bind $TMPDIR:$TMPDIR --pwd $PWD $IMAGE python ${PY_SCRIPT} 2>&1
```

ddp_or_fsdp.out

```
============================================================
DDP  - Her GPU'da modelin TAMAMI var
============================================================
  GPU 0: model=800 MB  peak=4514 MB
  GPU 1: model=800 MB  peak=4514 MB
  GPU 2: model=800 MB  peak=4514 MB
  GPU 3: model=800 MB  peak=4514 MB

============================================================
FSDP - Model 4 GPU'ya BOLUNMUS
============================================================
  GPU 0: shard=200 MB  peak=2601 MB
  GPU 1: shard=200 MB  peak=2601 MB
  GPU 2: shard=200 MB  peak=2601 MB
  GPU 3: shard=200 MB  peak=2601 MB

============================================================
Toplam bellek:  DDP=18054 MB  FSDP=10405 MB
Tasarruf: %42
============================================================
```

BSC AI Factory

Teşekkürler

Contact Form

aifactory@tubitak.gov.tr

bsc-ie@bsc.es

Social Media

Web Page

Co-funded by the European Union

EuroHPC Joint Undertaking

TÜBİTAK

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101234399 and Spain, Portugal, Romania and Türkiye. The JU receives support from the European Union's Horizon Europe Programme.

BSC AI Factory Partners

20 BSC YEARS

fct Fundação para a Ciência e a Tecnologia

TÜBİTAK

ICI

Affiliated entities

UPC UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

CNCA Centro Nacional de Computação Avançada