Sağlıkta Yapay Zeka İçin HPC Altyapılarının Kullanımı

MareNostrum5 Üzerinde Büyük Ölçekli Model Eğitimi

Sektörel YZ Kümelenmesi

Sağlık

15:15 – 16:00

Kemal ALEÇAKIR & Hüseyin ALEÇAKIR

Co-founders | Logethra

Gündem

01 **Neden HPC?**

Sağlıkta AI'nin hesaplama zorlukları

02 **MareNostrum5**

Mimari, kapasite ve GPU altyapısı

03 **BSC AI Factory**

Servisler ve erişim süreci

04 **Environment Kurulumu**

Modüller, konteynerler, SLURM

05 **Model Eğitimi**

Dağıtık eğitim ve optimizasyon

06 **Sağlıkta Senaryolar**

Gerçek dünya uygulama alanları

07 **Logethra Yaklaşımı**

Framework ve pratik örnek

Neden HPC? — Sağlıkta AI'nin Hesaplama Zorlukları

10^9+

Parametre

Modern sağlık modelleri (genomik, radyoloji)

TB

Tıbbi Veri

Tek bir hastane yıllık görüntüleme verisi

Aylarca

Eğitim Süresi

Tek GPU ile büyük model eğitimi

Saniyeler

İnferans İhtiyacı

Klinik uygulamalarda gerçek zamanlı karar

Tek GPU ile Yapılamaz

- BioGPT / Med-PaLM: 3B–540B parametre eğitimi
- 3D MRI segmentasyon: tek görüntü ~1 GB VRAM

Dağıtık AI ile Yapılabilir (HPC)

- Yüzlerce GPU ile paralel eğitim (aylar → günler)
- Shared high-speed storage: Petabyte ölçeği
- Interconnect (InfiniBand): GPU arası hız < 1 µs
- BSC MareNostrum5: 4.480 NVIDIA H100 GPU

MareNostrum5 — Teknik Mimari

Genel Amaçlı Bölüm (GPP)

- 6.408 Intel Sapphire Rapids düğümü (+72 HBM)
- Toplam 726.880 çekirdek (112/düğüm)
- 256 GB DDR5 RAM / düğüm
- InfiniBand NDR200 ağ yapısı
- GPP Peak: ~45 PFlop/s

Hızlandırıcı Bölüm (ACC / AI)

- 1.120 NVIDIA H100 GPU düğümü
- Düğüm başı 4× H100 (64 GB HBM2e)
- Toplam 4.480 H100 GPU
- ACC Rpeak: 200.79 PFlop/s
- InfiniBand NDR, 3-katmanlı fat-tree

Depolama & Ağ

- Toplam depo: 248 PB net kapasite (SSD/HDD)
- Arşiv: 402 PB teyp depolama
- Bant genişliği: 1.2 TB/s yazma, 1.6 TB/s okuma
- SLURM iş yöneticisi
- Singularity konteynerleri desteklenir

★ *AI Fabrikası kapsamında GPP + ACC bölümlerine ayrı kotalar atanmaktadır.*

MareNostrum5 — SSH ile Bağlantı ve Veri Transferi

Referans

GPP Giriş Düğümü (Genel Amaçlı)

```
ssh {kullanici_adi}@glogin1.bsc.es
ssh {kullanici_adi}@glogin2.bsc.es
```

ACC Giriş Düğümü (GPU / AI İşleri)

```
ssh {kullanici_adi}@alogin1.bsc.es
ssh {kullanici_adi}@alogin2.bsc.es
```

Veri Transferi (transfer node'ları kullan)

```
scp dosya.tar.gz {kullanici_adi}@transfer1.bsc.es:/gpfs/...
rsync -azP ./klasor/ {kullanici_adi}@transfer2.bsc.es:/gpfs/...
```

Kaynak: docs.truba.gov.tr/1-kaynaklar/mn5/mn5_baglanti.html

GPP Login Nodes

- glogin1.bsc.es
- glogin2.bsc.es

ACC Login Nodes

- alogin1.bsc.es
- alogin2.bsc.es

Transfer Nodes — Büyük veri aktarımı için

- transfer1.bsc.es
- transfer2.bsc.es
- transfer3.bsc.es
- transfer4.bsc.es

Önemli Notlar

- Login node'lardan büyük veri transferi yapma — transfer node kullan
- SSH key ile bağlantı kurulması şiddetle tavsiye edilir
- glogin → GPP iş kuyruğu, alogin → ACC (GPU) iş kuyruğu

Environment Kurulumu - BSC Önerisi

Uygulama Katmanı

PyTorch / TensorFlow / JAX / Hugging Face

Framework Katmanı

CUDA 12, cuDNN, NCCL, MPI

Ortam Katmanı

Singularity Container / Python venv / Conda

Kaynak Katmanı

SLURM İş Yöneticisi + Module System

Donanım Katmanı

NVIDIA H100 · InfiniBand · NVMe Storage

★ *Conda/venv de desteklenir; saf Singularity container önerilir (taşınabilirlik).*

Temel Komutlar

```
# Modül listele
module avail pytorch

# Modül yükle
module load pytorch/2.1-cuda12.1

# Singularity ile çalıştır
singularity exec --nv \
  myenv.sif python train.py

# SLURM iş gönder
sbatch train_job.slurm

# GPU durumu izle
squeue -u $USER
```

Environment Kurulumu — Conda Ortamı Oluşturma

Ne zaman kullanılır?

- Yerel makinede çalışan ortamı birebir HPC'ye taşı
- PyPI/conda-forge erişimi gerektirmez (offline/kapalı ağ)
- Aynı OS + CPU mimarisi şart (MN5: Linux x86_64)
- CUDA gerektiren paketler için host NVIDIA driver uyumlu olmalı

⚠ Editable install'lar (-e) pakete dahil edilmez; hedef makinede yeniden yüklenmeli.

💡 Kısa hedef path kullan:
~/envs/myenv (prefix uzunluğu sorunu önler)

conda-unpack → path'leri düzeltir; her zaman çalıştır.

0 Aracı yükle (kaynak makine — bir kez)

```
conda install -y -c conda-forge conda-pack
```

1 Cache temizle (isteğe bağlı)

```
conda clean --all -y
```

2 Aktif ortamı tarball'a paketle

```
ENV_NAME="$(basename "$CONDA_PREFIX")"
conda-pack -p "$CONDA_PREFIX" -o "${ENV_NAME}.tar.gz" --n-threads 8
```

3 HPC'ye rsync ile aktar

```
REMOTE="username@transfer1.bsc.es"
rsync -azP "${ENV_NAME}.tar.gz" "$REMOTE:~/environments"
```

4 Uzaktan aç, aktive et, path'leri düzelt

```
ssh "$REMOTE"
 mkdir -p ~/environments/${ENV_NAME}
 tar -xzf ~/environments/${ENV_NAME}.tar.gz \
      -C ~/environments/${ENV_NAME}
```

Environment Kurulumu — Ek Python Paketi Kurulumu

Wheel Store

Neden Wheel Store?

- MN5 dış ağa kapalı — PyPI'a direkt erişim yok
- Conda ortamı kurulduktan sonra ek paketler için
- Tüm .whl dosyaları merkezi bir klasörde toplanır
- pip.conf sayesinde her pip install otomatik oradan çeker
- Ekip üyeleri aynı wheel store'u paylaşır

~/.pip/pip.conf (önceden yapılandırılmış)

```
[global]
no-index = true
find-links = /gpfs/projects/eturXX/python/wheelstore
```

⚠ Paketleri her zaman /gpfs/projects/eturXX/python/wheelstore yoluna gönder.

1 — Wheel'ı yerel makinenden indir

```
pip download SimpleITK==2.4.0 \
    --python-version 3.10 \
    --platform manylinux2014_x86_64 \
    --only-binary=:all: \
    -d ./simpleitk_pkg
```

2 — HPC Wheel Store'a transfer et (scp)

```
scp ./simpleitk_pkg/\
  SimpleITK-2.4.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl \
  username@transfer1.bsc.es:/gpfs/projects/eturXX/python/wheelstore
```

3 — HPC üzerinde pip install (pip.conf otomatik çeker)

```
pip install SimpleITK
# → PyPI'a gitmez, wheelstore'dan yükler
```

Aynı wheel store tüm ekip üyeleri tarafından kullanılabilir. Bir kez yükle, herkes faydalansın.

Environment — SLURM ile GPU İşi Gönderme

Bölüm Seçimi

acc_ehpc / gp_ehpc → AI işleri için acc bölümü

GPU Sayısı

--gres=gpu:4 → düğüm başına 4× H100

Bellek

--mem=0 → tüm düğüm belleğini kullan

Zaman Limiti

--time=24:00:00 → iş başına maks. süre (kotaya bağlı)

Output Yönlendirme

--output=%j.log → job-id bazlı log dosyası

★ *'scontrol show partition' komutuyla mevcut bölümler listelenebilir.*

train_job.slurm

```
#!/bin/bash
#SBATCH --job-name=health_ai
#SBATCH --partition=acc_ehpc
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=4
#SBATCH --gres=gpu:4
#SBATCH --time=12:00:00
#SBATCH --output=%j.log

module load pytorch/2.1-cuda12.1
module load openmpi/4.1.5

srun --mpi=pmix \
  singularity exec --nv \
  myenv.sif python train.py \
  --nodes 4 --gpus-per-node 4
```

SLURM — İnteraktif Debug Oturumu Başlatma

Referans

⊘ Login Node Kuralı

Login node'ları (glogin / alogin) yalnızca iş göndermek için kullanılmalıdır. Kod çalıştırma, derleme veya test yapılmamalıdır.

✓ Debug Node Ne Zaman Kullanılır?

- Ortam kurulumunu test etme
- Kodun ilk çalışıp çalışmadığını doğrulama
- Küçük ölçekli interaktif geliştirme

Maksimum süre: 2 saat (acc_debug QoS)

Tipik Geliştirme Akışı

1. SSH ile alogin'e bağlan
2. srun ile debug node al
3. Ortamı aktive et, kodu test et
4. Hazırsa sbatch ile tam iş gönder

srun — Interactive Debug Session

```
srun --partition=acc \
    --gres=gpu:1 \
    --ntasks=1 \
    --cpus-per-task=20 \
    --account=eturXX \
    --qos=acc_debug \
    --pty bash
```

Parametre Açıklamaları

--partition=acc	ACC (GPU) bölümü — H100 düğümleri
--gres=gpu:1	1 adet GPU talep et
--ntasks=1	Tek görev (interactive shell için yeterli)
--cpus-per-task=20	20 CPU çekirdeği
--account=eturXX	Proje hesabı (kota buradan kesilir)
--qos=acc_debug	Debug kuyruğu — maks. 2 saat
--pty bash	İnteraktif terminal aç

SLURM — Job Array ile Paralel Preprocessing

Neden Job Array?

Aynı pipeline'ı çok sayıda veri chunk'ı üzerinde paralel çalıştırır

Tipik Kullanım

10–20 TB NIfTI veriyi CPU node'larında chunk'lara bölerek işleriz

Bu Örnekte

--array=0-51 → aynı anda 52 paralel job

Task Mantığı

Her task kendi chunk_${SLURM_ARRAY_TASK_ID}.json dosyasını işler

Kazanç

Günler sürecek preprocessing işini saatler mertebesine indirir

★ *52 task farklı veri chunk'larını işler; memmap çıktısı hızlı random access sağlar.*

memmap_array.slurm

```
#!/usr/bin/env bash
#SBATCH --job-name=memmap_array
#SBATCH --partition=cpu
#SBATCH --cpus-per-task=4
#SBATCH --array=0-51
#SBATCH --time=12:00:00
#SBATCH --output=logs/memmap-%A_%a.out



CID="${SLURM_ARRAY_TASK_ID}"



python mem_tool.py \
  --chunks_dir /path/to/chunks \
  --chunk_idx "$CID" \
  --output_dir /path/to/output
```

Büyük Ölçekli Model Eğitimi — Dağıtık Stratejiler

Data Parallelism — DP / DDP

Aynı model her GPU'ya kopyalanır; veri bölünür.

Kullanım: Orta büyüklükte modeller, çok veri

Araç: **PyTorch DDP / Hugging Face Accelerate**

Tensor Parallelism — TP

Model katmanları GPU'lar arasında yatay bölünür.

Kullanım: Tek katmanın GPU'ya sığmadığı durumlar

Araç: **Megatron-LM / DeepSpeed**

Pipeline Parallelism — PP

Model katmanları dikey olarak sıralanır (stage).

Kullanım: Çok derin ağlar, çok sayıda GPU

Araç: **GPipe / DeepSpeed PipelineEngine**

Fully Sharded (ZeRO) — FSDP / ZeRO

Parametre, gradient, optimizer parçalanır.

Kullanım: Milyarlarca parametreli modeller

Araç: **DeepSpeed ZeRO-3 / PyTorch FSDP**

Model Eğitimi — Optimizasyon İpuçları

Mixed Precision (FP16/BF16)

- torch.cuda.amp.autocast() ile otomatik
- BF16: H100 GPU'da daha stabil (overflow yok)
- Tipik hızlanma: 1.5–3× ; bellek: ~%50 azalır

Gradient Checkpointing

- Aktivasyonlar forward'da yeniden hesaplanır
- Bellek kullanımı ~%40 azalır (zaman bedeli ~%20)
- model.gradient_checkpointing_enable()

Gradient Accumulation

- Küçük batch × N adım = büyük effective batch
- GPU belleği kısıtlıyken global batch boyutunu korur
- accumulation_steps = effective_batch // local_batch

FlashAttention-2

- Transformer dikkat hesabını bellek-verimli yapar
- H100 ile 5–8× attention hızlanması
- pip install flash-attn --no-build-isolation

Profiling & Tuning

- torch.profiler ile darboğaz tespiti
- NVIDIA Nsight Systems: GPU utilizasyonu
- DataLoader: num_workers=8, pin_memory=True

Checkpoint Stratejisi

- Her N step'te save_pretrained() ile kayıt
- Async checkpoint: eğitimi durdurmaz
- SLURM checkpointing: job sona ermeden önce

İş İzleme ve Loglama — Monitoring Araçları

Weights & Biases (W&B)

Offline Mode

HPC ortamında offline olarak loglar toplanır, sonra W&B'ye aktarılır.

Kullanım: Loss, metrik, GPU kullanımı takibi

Araç: **wandb sync ./wandb/offline-run-***

MLflow

Alternatif

Açık kaynak experiment tracking; self-hosted sunucuyla çalışır.

Kullanım: Hyperparameter ve artifact yönetimi

Araç: **mlflow server + mlflow.log_metric()**

TensorBoard

Görselleştirme

Eğitim eğrileri ve model grafiğini görselleştirir.

Kullanım: Loss/accuracy eğrileri, embedding projeksiyonu

Araç: **tensorboard --logdir=./runs --port=6006**

SLURM Loglama

Sistem İzleme

SLURM çıktı dosyaları ve sacct ile iş performansı takibi.

Kullanım: GPU utilizasyonu, bellek, iş süresi analizi

Araç: **sacct -j $JOBID --format=Elapsed,MaxRSS**

Sağlıkta AI — HPC Gerektiren Kullanım Senaryoları

Tıbbi Görüntüleme

Model: 3D U-Net / SegFormer / SAM-Med

Veri: CT, MRI, PET taramaları (100+ GB/hasta)

HPC İhtiyacı:

Multi-GPU segmentasyon; 3D işlem için >64 GB VRAM

Genomik & Proteomik

Model: AlphaFold2 / ESMFold / DNABERT

Veri: WGS verileri (~100 GB/örnek); milyonlarca dizi

HPC İhtiyacı:

Protein folding: 1 yapı → 10-200 GPU-saat

Klinik NLP

Model: BioBERT / Med-PaLM / ClinicalBERT

Veri: EHR kayıtları, klinik notlar, ICD-10 etiketleri

HPC İhtiyacı:

Fine-tuning: 7B–70B model; çoklu GPU zorunlu

İlaç Keşfi

Model: MolBERT / DiffSBDD / EquiBind

Veri: ZINC DB: 1.5 milyar molekül

HPC İhtiyacı:

Sanal tarama: trilyon ölçeği hesaplama

Epidemiyoloji

Model: SEIR + Transformer hibrid / GNN

Veri: Nüfus hareketi, semptom, çevre verisi

HPC İhtiyacı:

Büyük ölçekli simülasyon + eğitim döngüsü

Radyoloji Raporlama

Model: LLaVA-Med / BioViL-T / MedVLP

Veri: Görüntü + metin çiftleri (multimodal)

HPC İhtiyacı:

Multimodal pre-training: petaFLOP ölçeği

Logethra — HPC Destekli Sağlık AI Geliştirme Yaklaşımı

logethra.ai

1. Problem Tanımı

Klinik ihtiyaç → ML görevi eşlemesi
(segmentasyon, sınıflama, üretim, öneri)

2. Veri Mimarisi

DICOM / FHIR / HL7 → HPC-uyumlu
format dönüşümü ve depolama stratejisi

3. Altyapı Seçimi

BSC MN5 başvurusu; kota planlaması

4. Eğitim Pipeline'ı

Hugging Face + DeepSpeed ZeRO;
multi-node SLURM iş akışı

5. Değerlendirme

Klinik metrikler (AUC, Dice, BLEU-C);
regülasyon uyumluluğu (GDPR, KVKK)

6. Dağıtım

ONNX/TensorRT dönüşümü;
hastane sistemine entegrasyon

HPC Kullanım Senaryosu

CT Verisi Üzerinde SSL Eğitimi

- Büyük hacimli CT verisi üzerinde SSL eğitimi (yaklaşık 1 milyar parametreli)
- Tek GPU ile süre aylar düzeyindedir
- Pilot ölçümler H100 üzerinde yapılmıştır
- 32 GPU ölçeğinde eğitim süresi yaklaşık 8–10 güne iner
- HPC altyapısı bu çalışmayı uygulanabilir hale getirir
- Tek koşu yaklaşık 7.700 GPU-saat gerektirir
- Sonuç: HPC, eğitimi yapılabilir hale getirir

Case Study

Teşekkürler

Soru & Cevap

Kemal ALEÇAKIR

kemal@logethra.ai