EuroHPC Development Access Proposal

Project Title: Scalable Multi-Node Distributed Inference for Large Language Models
Applicants: Boris Gans, Yousef Amirghofran, Anze Zgonc, Léa Abou Jaoudé, Matthew Porteous, Alp Baghirov
Duration Requested: 6 months
Compute Requested: ~4,000 GPU node-hours on the Leonardo Booster Module

Abstract & Objectives

The increasing computational and memory demands of large language models have turned even inference into a distributed systems challenge. Existing GPU nodes on many EuroHPC systems provide high-bandwidth interconnects, multi-GPU topologies, and large aggregate memory, enabling advanced parallelization strategies beyond what is possible on smaller, single-GPU nodes. This project aims to develop a robust, scalable, and portable distributed inference pipeline for LLMs using pipeline parallelism, activation offloading, and optimized inter-node communication.

Building on prior exploratory work in a constrained single-GPU-per-node environment, where pipeline partitioning, offload strategies, and Slurm-orchestrated distributed execution were prototyped, this project now seeks to **(1) resolve framework-level device-mapping limitations, (2) integrate GPU-accelerated profiling, (3) implement deeper multi-stage pipelines, and (4) scale to larger model sizes** using EuroHPC's multi-GPU nodes. The expected outcome is a validated, production-grade distributed inference workflow, performance-characterized across GPU types, node counts, batch sizes, and pipeline depths, suitable for future Benchmark and Regular Access proposals.

EuroHPC resources are essential due to the project's need for high-bandwidth GPU interconnects, large VRAM capacity, and profiling tools not available on the constrained cluster used in the initial study. The development period will deliver a refined, portable inference stack, a reproducible containerized environment, and performance results enabling confident scaling toward multi-billion-parameter LLM inference.

State of the Art

Distributed inference for LLMs remains a technically challenging task due to model memory footprints exceeding a single GPU, communication overhead during inter-stage activation transfer, and inefficiencies of standard framework loading paths when models are partitioned across devices. Pipeline parallelism—dividing sequential transformer layers across accelerators—is a well-established approach (Shoeybi et al., 2019; Narayanan et al., 2021), but most frameworks assume either multi-GPU nodes or stable interconnects such as NVLink. In constrained environments with one GPU per node and PCIe-only communication, significant bottlenecks appear.

Our study demonstrated several of these limitations and their subsequent affects in practice:

- One GPU per node (NVIDIA T4, 16 GB)
- No NVLink, sole reliance on PCIe for NCCL communication
- Failed attempts to load model partitions across nodes using the Hugging Face `from_pretrained` loading path

State-of-the-art inference frameworks (DeepSpeed-Inference, TensorRT-LLM, and Hugging Face Accelerate) often assume homogeneous multi-GPU nodes, large memory pools, and easy installation of profiling libraries. They perform suboptimally or fail entirely when deployed in minimal or legacy cluster setups.

This proposal addresses these gaps by building an end-to-end distributed inference pipeline that explicitly manages device placement, supports multi-node disjoint partitions, remains functional under limited-memory conditions, and leverages modern EuroHPC hardware where such restrictions are lifted.

Current Code, Architecture & TRL Assessment

3.1 Codebase Overview

Our distributed-inference system includes the following:

- **Python/PyTorch implementation** for transformer inference
- **Manual pipeline partitioning**, with ranks hosting different layer subsets
- **Slurm-orchestrated distributed execution**, using NCCL
- **Apptainer container execution**, binding in model directories and Python environments at runtime
- **Disk-based offloading**, storing non-local layers on shared scratch
- **Simple decode loop** using greedy sampling, optimized for correctness rather than quality
- **Logging infrastructure** with timing, throughput, and NCCL diagnostics

We utilize this software stack to perform distributed inference on the OpenLLaMa 3B v2 model.

Current Limitations

During development our application encountered several critical blockers:

- Hugging Face loading APIs fail to place partitions on remote devices:
  ```
  model.layers.0.self_attn.q_proj.weight doesn’t have any device set
  model.embed_tokens.weight doesn’t have any device set
  ```
- T4 memory (16GB) insufficient for 7B weights; even 3B weights fail under disjoint partitioning
- Limited to one parallelism strategy with two pipeline stages because nodes provide only *one* GPU

Technology Readiness Level (TRL)

- **TRL 3–4:** Proof-of-concept components implemented and distributed initialization complete, but our inference loop is not yet functional.

Gaps preventing TRL 5 readiness include stable multi-node weight loading, container reproducibility, and profiling support.

Target EuroHPC Machine & Software Stack

The project will use the Leonardo Booster Module (CINECA) with the following specifications:

- **GPU architecture:** NVIDIA A100 40GB (SXM4)
- **Interconnect:** NVLink (intra-node) + HDR InfiniBand (inter-node)
- **Node configuration:** 4× A100 GPUs per node
- **Software environment:** CUDA 11/12, NCCL, NVHPC/GCC toolchains, Nsight profiling suite, Apptainer/Singularity support

The Leonardo Booster partition is the most technically appropriate EuroHPC system for this project's objectives, which center on **pipeline parallelism, multi-node distributed inference, and GPU-bound performance profiling**. This system provides architectural capabilities that directly address the limitations observed in the initial study and that are essential for debugging, optimizing, and scaling transformer-based LLM inference.

Required Hardware Features

- Multi-GPU nodes (4–8 GPUs/node) to allow local multi-stage pipelines
- High-bandwidth interconnect (NVLink + InfiniBand)
- ≥40 GB VRAM per GPU (A100 or MI250X)
- Large memory nodes for hosting CPU offload
- Sufficient I/O bandwidth for large model loading

Software Stack

- **PyTorch 2.x** (CUDA or ROCm depending on machine)
- **NCCL / RCCL** for multi-GPU and multi-node communication
- **DeepSpeed** (inference mode only)
- **Transformers**, **Accelerate** (Hugging Face)
- **Nsight Systems / Nsight Compute** or ROCm Profiling Suite
- **Apptainer/Singularity** for portable builds
- **HDF5**, **BLAS**, **MPI** libraries provided by the system

Container Environment

We will construct a lean production-ready container, unlike the minimal PyTorch image in our initial study that required runtime installation of dependencies and lacked CUDA driver bindings . This upgraded container will include:

- Proper CUDA/NVHPC or ROCm stack
- Preinstalled transformer libraries
- Fixed pip wheels to avoid network-unavailable installation
- Preloaded model weights using bind-mounts or overlay FS

Work Plan, Tasks, Milestones, Risks

Duration: 6 months

WP1 – Foundational Infrastructure & Environment Stabilization (Month 1–2)

Goals

This initial work package focuses on establishing a reliable execution foundation on Leonardo Booster so that later profiling, scaling, and optimization work can proceed without the blockers encountered in the initial cluster environment. In particular, it addresses:

- Ensuring **correct multi-GPU model loading**, resolving the Hugging Face / DeepSpeed device-mapping failures observed previously (e.g., unmapped parameters, stage misplacement).
- Building a **deterministic, profiling-enabled container**, avoiding the runtime dependency installation and CUDA visibility issues encountered earlier (e.g., missing driver bindings, broken LD_LIBRARY_PATH)
- Preparing a **stable NCCL environment** across the Leonardo Booster nodes with NVLink + InfiniBand.

Methods

- Replace `from_pretrained` with **tensor-slice loading** or custom partitioned-weight loaders.
- Pre-shard weights offline into **per-stage device bundles**.
- Construct a **fully self-contained Apptainer image** with PyTorch, DeepSpeed, CUDA/NCCL, and HF libs preinstalled.
- Validate correctness using a **minimal 2-GPU pipeline** before expanding.

Milestone 1:

A fully functioning multi-GPU environment on Leonardo, with correct model loading and a production-ready container image validated on at least two nodes.

WP2 – Expanded Pipeline Parallelism & Model Scaling (Month 2–4)

Goals

This work package extends pipeline parallelism to the full capabilities of Leonardo Booster's multi-GPU nodes, enabling deeper transformer partitioning and setting the stage for scaling experiments and high-resolution profiling.

Key objectives:

- Expand from **2-stage** to **4–8 stage pipelines**, exploiting intra-node NVLink bandwidth.
- Evaluate **pipeline depth vs. activation recomputation** trade-offs.
- Integrate **activation checkpointing**, quantized weights (FP8/int4), and optimized KV-cache strategies.
- Test **larger models** that were infeasible on 16 GB T4 GPUs (e.g., 7B–13B class).

Methods

- Use heuristic layer balancing to avoid stage bottlenecks.
- Implement automated **pipeline bubble detection** using lightweight timing instrumentation.
- Test multiple parallelization mixes: *pipeline-only*, *pipeline* + *tensor*, and *pipeline* + *quantization*.
- Continuously monitor memory headroom enabled by A100 40GB HBM.

Milestone 2:

Stable multi-stage distributed inference on ≥4 GPUs with demonstrated correctness, balanced stage times, and capacity to support larger model sizes.

WP3 – Profiling, Performance Optimization & Multi-Node Scaling (Month 3–5)

Goals

This work package performs in-depth performance characterization using the profiling tools unavailable in the prior cluster (Nsight Systems, Nsight Compute). It focuses on quantifying and optimizing multi-node scaling behavior, communication overheads, and GPU utilization.

Experiments & Activities

1. Kernel-Level Profiling (Nsight Systems / Nsight Compute)

- Capture GPU kernel traces, occupancy, warp efficiency.
- Identify memory-bound or communication-bound transformer components.
- Analyze NVLink vs. InfiniBand communication patterns.

2. Strong Scaling

- Fixed prompt set, increasing GPU counts: 4 → 8 → 16 → 32 GPUs.
- Measure deviations from ideal scaling.
- Diagnose pipeline bubbles, synchronization stalls, and activation-transfer bottlenecks.

3. Weak Scaling

- Proportionally increase the number of prompts or sequences with the number of GPUs.
- Evaluate per-GPU throughput stability.
- Identify communication-to-computation ratios across node counts.

4. Batch-Size Sensitivity Studies

- Systematically sweep microbatch sizes.
- Map throughput, latency, and memory consumption curves.

5. Optimization

Based on profiling findings:

- Tune NCCL/RDMA settings for Booster’s Infiniband fabric.
- Refine pipeline boundary placement using data-driven rebalancing.
- Introduce fused attention kernels and quantized weight formats (FP8/int4).

Milestone 3:

Comprehensive performance report including scaling curves, bottleneck analysis, kernel-level profiling traces, and an optimized multi-stage pipeline with measurable speedup.

WP4 – Reproducibility, FAIR Packaging & Public Release (Month 5–6)

Goals

The final work package consolidates the project outputs into production-grade artifacts for community uptake and future EuroHPC Regular Access proposals.

Activities

- Release a **container image** and **modular inference toolkit** (weight-sharding scripts, launcher utilities).
- Provide **canonical Slurm run scripts** for single-node and multi-node pipeline execution.
- Publish an **open dataset of profiling traces**, stripped of sensitive paths.
- Produce full documentation:
  - Installation and environment setup
  - Pipeline configuration
  - Scaling guidelines
  - Troubleshooting and device-mapping advice
- Archive results to Zenodo with digital object identifiers (DOIs).

Milestone 4:

Public release (open-source) of the entire framework, including containers, scripts, documentation, and profiling outputs, meeting FAIR principles and ready for extension in a Regular Access proposal.

Risks & Mitigation

Risk	Mitigation
Model sizes exceed GPU memory	Activation checkpointing, FP8/INT4 quantization
Pipeline imbalance	Automated layer-to-GPU mapping heuristics
NCCL/RCCL topology issues	Use system profilers + topology-aware mapping

Profiling tool unavailability	Fall back to in-application timers (as is done currently) but EuroHPC systems almost always support Nsight

Resource Justification

The project focuses on development, profiling, and modest-scale benchmark runs. Approximate requirements:

Compute Requirements

Task	Node type	Node-hours
WP1 Model loading & debugging	2–4 A100 nodes	600
WP2 Container development	1 GPU node	200
WP3 Multi-stage pipelines	4–8 GPU nodes	1,200
WP4 Scaling experiments	8–16 GPU nodes	2,000
WP5 Documentation & reproducibility	1 node	100

Total Request: ~4,000 GPU node-hours

Justification:

- The initial environment lacked multi-GPU nodes; EuroHPC's topology is essential.
- Strong/weak scaling experiments require multi-node jobs, especially for 4–16 GPU configurations.
- Profiling runs require repeated execution under different batch sizes and pipeline depths.
- Extensive debugging is expected during WP1–WP2 due to framework complexity.

Data Management, FAIR Plan & Ethics

- **Input Data:** Only model weights and text prompts. No personal data.
- **Output Data:**
  - Logs, traces, throughput tables
  - Profiling files (100 MB–50 GB depending on run)
- **FAIR Compliance:**
  - **Findable:** Git repository + Zenodo archive of container & scripts
  - **Accessible:** Public artifacts, open licenses
  - **Interoperable:** Standard formats (JSON, CSV, HDF5)
  - **Reusable:** Clear documentation, versioned releases

All transformer weights used will be from models with permissive inference licenses (e.g., OpenLLaMA variants permitting research use). No GDPR-sensitive or proprietary data will be processed.

Expected Impact

This project delivers both **technical advancements** and **practical value** to the EuroHPC community:

Technical Impact

- First fully validated, portable, multi-node inference stack extending the prototype developed in your initial study—which could not proceed due to environment limitations .
- Deep profiling of communication paths across EuroHPC interconnects.
- Evaluation of memory-saving techniques for hosting multi-billion-parameter LLMs on A100/MI250X GPUs.

Community Impact

- A high-quality reference implementation for pipeline parallel inference.
- Reusable, open-source container and Slurm launcher templates.
- Benchmarks useful to other teams preparing Regular Access proposals involving LLM inference or model serving.

Strategic EuroHPC Impact

- Demonstrates the value of EuroHPC's GPU accelerators for AI workloads.
- Encourages broader adoption of LUMI/Leonardo/MeluXina for LLM research.
- Prepares the groundwork for a future **EuroHPC Regular Access** proposal targeting multi-hundred-GPU scaling.

References

- OpenLLaMA: An Open Reproduction of LLaMA
- Open_llama_3b_v2
- Shoeybi et al., "Megatron-LM: Training Multi-Billion Parameter Models Using Model Parallelism," NeurIPS Systems 2019
- Narayanan et al., "GPT-3 Training: Scaling Laws and Infrastructure," DeepSpeed Engineering Notes (2021)