A ***Control Plane*** for Intelligent AI GPU Interconnects

Problem Statement

Managing distributed AI GPU systems presents major challenges due to their high cost, extreme performance requirements, and mission-critical workloads. Effective orchestration must account for both physical GPU interconnect topologies and workload communication patterns to ensure optimal performance and resource utilization.

Core Concept

A progressive web app or even a native app can be developed with a database of all the GPUs and all the Network IDs (Architecturally, this will be used to identify the network connecting various GPUs).

Topology-aware scheduling :

Each network once established would be assigned a unique topology as well so that intelligent dynamic topology interchange can occur.

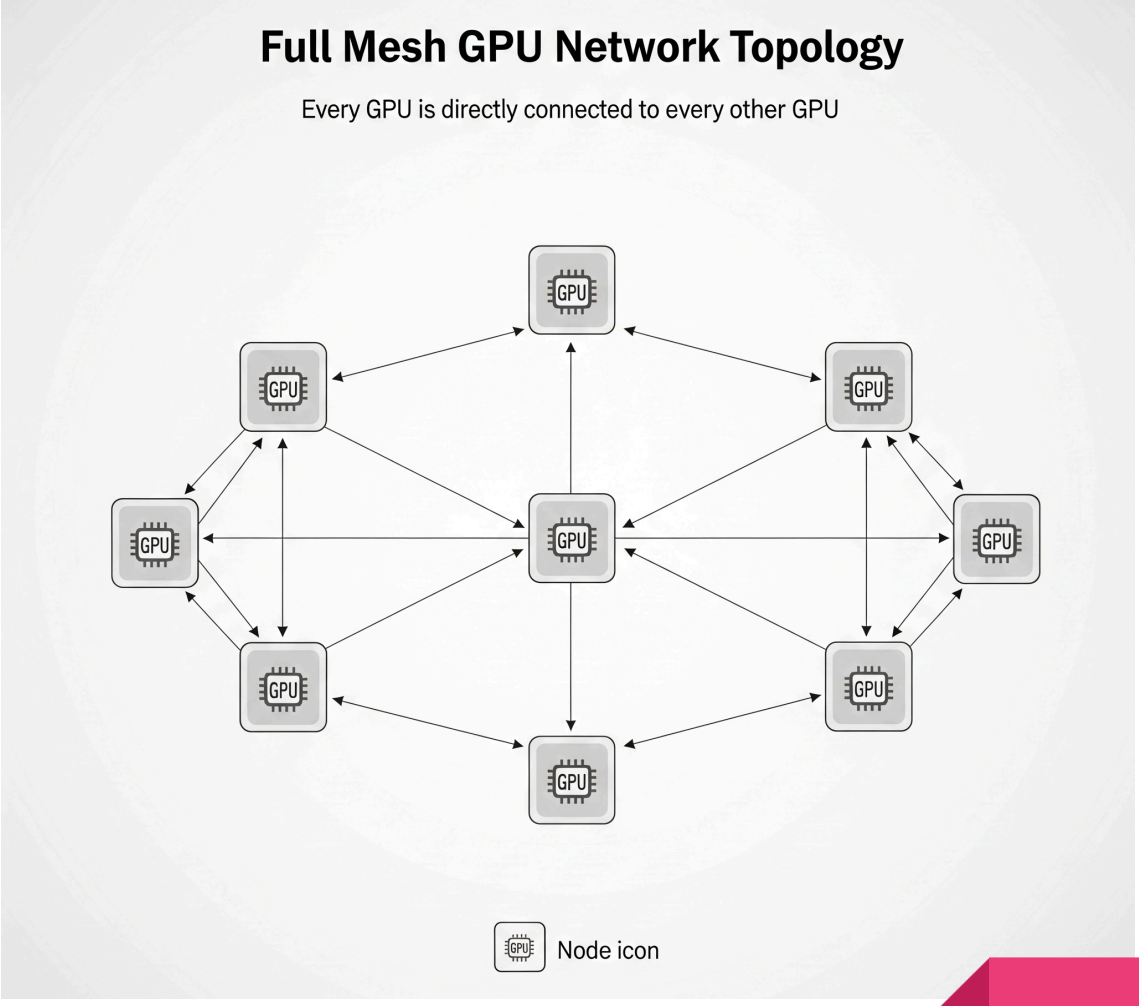
Topology Discovery: The control plane's monitoring component uses tools (like nvidia-smi topo -m or similar APIs) to discover the physical connections between all the GPUs in the cluster. It builds a map of the network, identifying which GPUs are on the same server, which are in the same rack, and which are farther apart.

Workload Analysis (using AI): Your AI techniques would analyze the incoming workload's communication requirements.

- ***High-performance workload:*** A distributed training job that requires frequent, high-volume communication between all GPUs (e.g., using a collective communication algorithm). This kind of job is sensitive to latency.
- **Low-performance workload:** A batch inference job where each GPU works independently with minimal communication between them. This job is less sensitive to network topology.

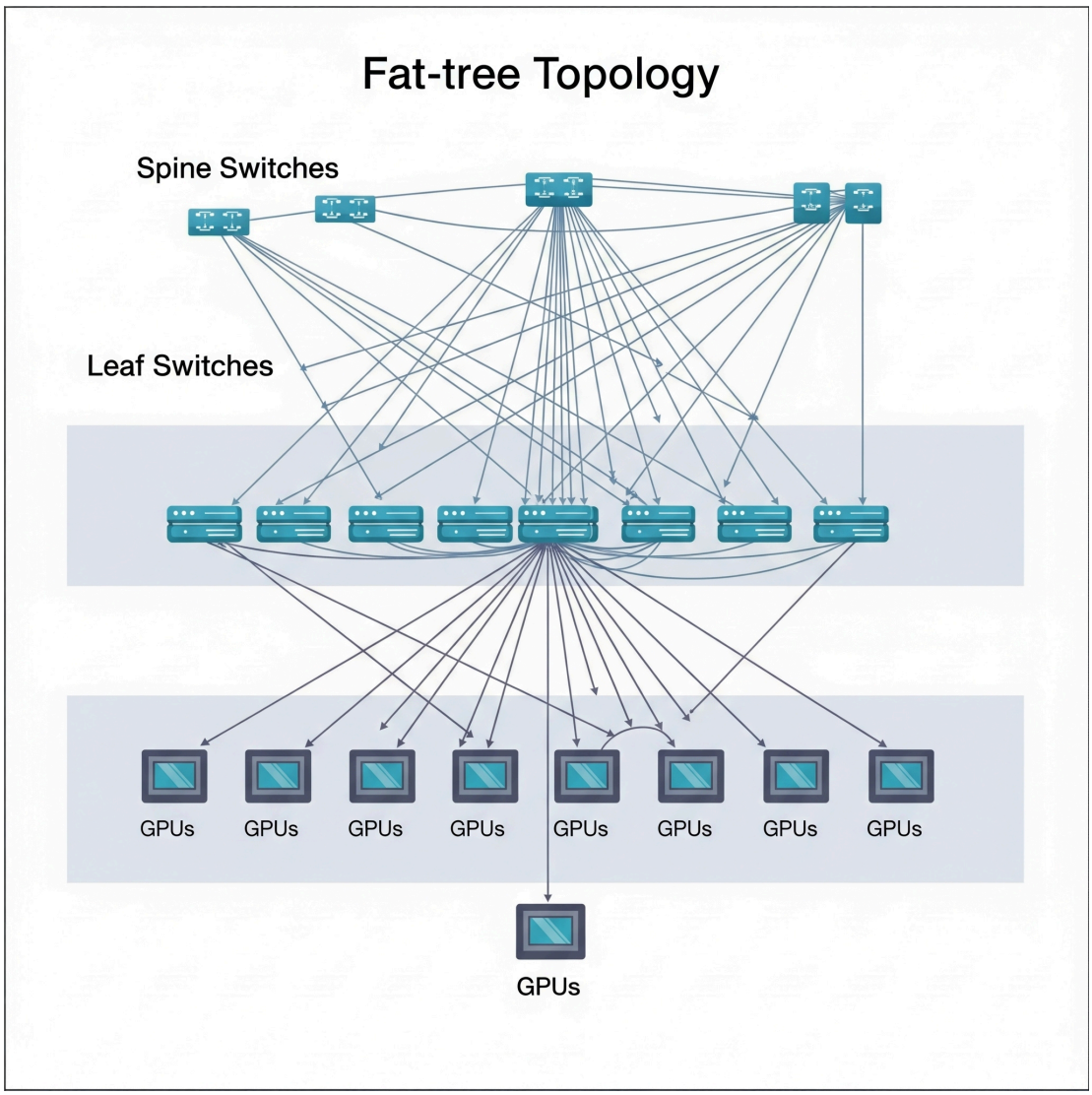
***Intelligent Scheduling*:** Based on the workload's needs and the physical topology map, the control plane schedules the job to the most appropriate set of GPUs.

- For the **high-performance workload**, it would try to allocate all the GPUs on a single server, where they can communicate over the ultra-fast NVLink full-mesh. If



a single server isn't enough, it would try to place the job on GPUs within the same rack to leverage the high-speed Infini Band network.

- For the **low-performance workload**, the scheduler can afford to be "topology-agnostic." It can place the GPUs anywhere in the cluster, freeing up the high-bandwidth, intra-server resources for other, more demanding jobs.



Methodologies & Technologies:

We will use the following methodologies-:

SystemC → High-level simulation and verification during GPU network architecture design.

gem5 → System-level performance analysis of CPU–GPU interactions.

Python → Backend development, workload scheduling logic, and GPU parameter visualization.

HTML → Graphical user interface for topology visualization and control.

Features:

1. **Separate Network Handling**
   Manage entirely different GPU networks through a simple selection interface.

2. **Network Integration**
   Merge two distinct networks into a single large one, using a bridging server capable of interfacing between different topologies.

3. **Network Splitting**
   Divide a network into multiple smaller ones to isolate workloads and improve efficiency.

4. **Fault & Thermal Management**

   Automatically detect GPU faults or temperature thresholds. Reallocate tasks to optimal GPUs until the issue is resolved, then revert.

5. **Interactive Topology Visualization**

   Display GPUs as nodes in a topology graph. Users can select nodes to monitor and control individual GPUs without memorizing their physical connections.

Assumptions

- Communication protocol specifics are abstracted, ensuring platform independence.
- Topologies conform to known, structured categories rather than arbitrary random layouts.

Conclusion:

This control plane enables intelligent**,** topology-aware GPU workload scheduling for distributed AI systems, improving performance, fault tolerance, and resource efficiency. By combining real-time topology mapping, AI-driven workload analysis, and interactive user controls, it bridges the gap between hardware-level complexity and intuitive operational management.