HyperCluster: Decentralized Large Language Model Inference over Peer-to-Peer Wireless Networks

Vyoman Jain, Samarth P, Akepati Ramya Sri, Sanjiv Raghunandan, Richa Sharma
Department of Computer Science
PES University
Bengaluru, India
{PES2UG22CS672, PES2UG22CS495, PES2UG22CS045, PES2UG22CS504}@pesu.pes.edu
richasharma@pes.edu

Abstract

The substantial memory and computational requirements of large language models (LLMs) hinder their deployment on individual resource-constrained wireless devices. This paper introduces HyperCluster, a framework for fully decentralized collaborative inference over wireless networks. Our system presents two core innovations: first, a ring-based pipelined inference protocol where nodes deterministically self-organize into a computational ring based on device capabilities, passing intermediate states directly between peers by leveraging content-addressed networking and document synchronization primitives. Second, we introduce a generalizable model sharding methodology built upon the Hugging Face Transformers library, which automatically partitions any dense LLM across heterogeneous devices according to their available compute and memory. We provide a practical validation of this architecture, demonstrating successful distributed inference of models up to a billion parameters on a network of consumer-grade devices.

Introduction

The proliferation of large language models (LLMs) has created an unprecedented demand for computational resources, with state-of-the-art models requiring hundreds of gigabytes of memory that far exceed the capacity of typical consumer and edge devices. While cloud based services provide access to these models, they introduce privacy concerns and increase reliance on centralized infrastructure, which are particularly acute in wireless and edge computing scenarios. An alternative paradigm is collaborative inference, where multiple devices in a peer-to-peer (P2P) network pool their resources to collectively execute a model that is too large for any single device.

However, realizing this vision in a decentralized wireless environment presents significant challenges. Frameworks designed for distributed training in data centers, such as Megatron-LM (Shoeybi et al. 2019) and Deepspeed (Rasley et al. 2020), depend on high-bandwidth, low-latency interconnects and stable cluster memberships, assumptions that do not hold in dynamic P2P networks. Existing work on distributed inference has often been limited to local networks or has required significant, model-specific modifications, hindering compatibility with the vast ecosystem of publicly available LLMs.

This paper introduces HyperCluster, a prototype framework for fully decentralized collaborative inference designed specifically for heterogeneous devices over P2P networks across the internet. Our work is motivated by the need for a system that can operate over internet-scale connections and maintain broad compatibility with standard model implementations. To this end, HyperCluster makes the following contributions:

1. **Decentralized Ring-Pipeline Protocol:** We introduce a coordinator-free inference protocol where nodes self-organize into a deterministic computational ring based on their advertised capabilities. Intermediate activations and critical states are passed directly between peers, enabling correct autoregressive generation in a fully decentralized manner.
2. **Generalizable Transformers-Based Sharding Engine:** We implement a dynamic model sharding methodology that integrates directly with the Hugging Face Transformers library (Wolf et al. 2020). This allows dense, off-the-shelf LLMs to be automatically partitioned across heterogeneous devices based on their available memory, without requiring any modification to the original model architecture.

We provide a practical validation of this architecture, demonstrating successful distributed inference of models up to a billion parameters on a network of consumer-grade devices. Our results establish HyperCluster as a viable testbed for exploring the foundations of collaborative AI in decentralized wireless systems.

Background and Related Work

This research builds upon two primary domains: (1) large-scale parallel systems designed for centralized data centers and (2) emerging frameworks for decentralized, peer-to-peer (P2P) computation. Our work also incorporates specific network topologies to optimize communication for distributed inference.

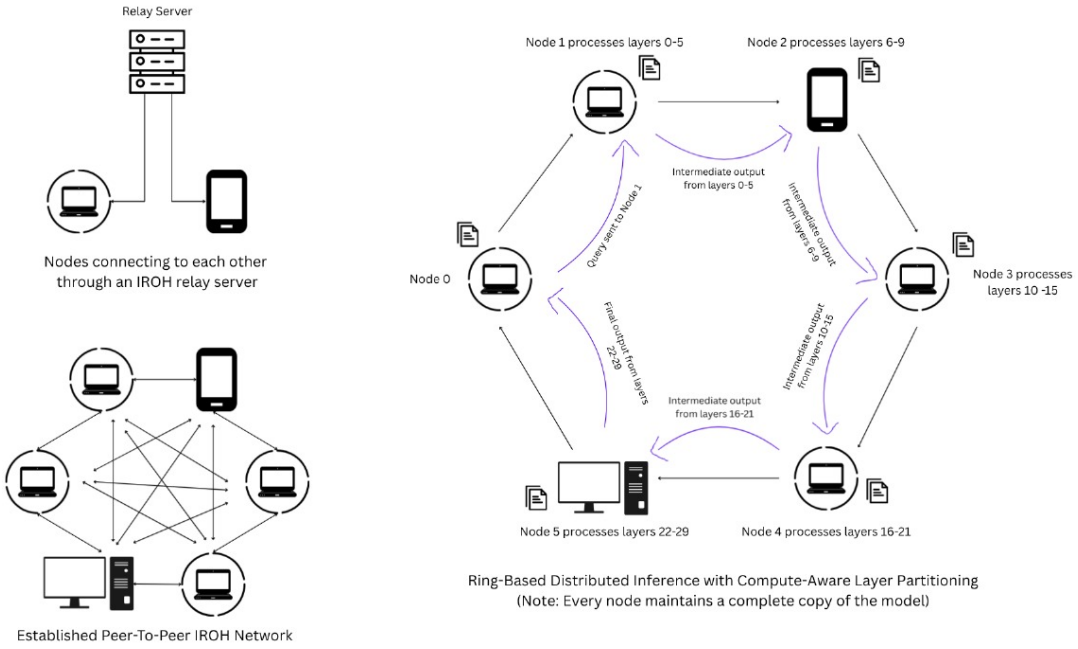

Figure 1: Working of HyperCluster

Model Parallelism in Centralized Environments

The challenge of partitioning large neural models has been thoroughly investigated within the high-trust, homogeneous environments of data centers. Seminal systems like Megatron-LM (Shoeybi et al. 2019) and DeepSpeed (Rasley et al. 2020) introduced foundational techniques for distributing models across multiple GPUs. More recent frameworks such as Alpa (Zheng et al. 2022) have further refined these methods. However, these systems are architected for environments with stable hardware and high-bandwidth, low-latency interconnects (e.g., NVLink, InfiniBand). These foundational assumptions render them unsuitable for the dynamic, bandwidth-constrained, and heterogeneous P2P networks that HyperCluster is designed to operate in.

Decentralized and Peer-to-Peer Inference

Recent work has begun to explore the distribution of model inference workloads over less reliable, internet-scale networks. Petals (Borzunov et al. 2023) demonstrated a collaborative inference and fine-tuning system where users contribute GPU resources to a shared network. However, Petals relies on a central authority for peer discovery and assumes high-bandwidth, stable connections, which limits its applicability in dynamic wireless environments.

Other systems have provided key insights into decentralized inference on consumer-grade hardware. Exo (ExoLabs 2024) is a framework that enables distributed inference by sharding models and distributing them across a ”home cluster” of everyday devices. It uses a peer-to-peer architecture and employs a ”ring memory weighted partitioning” strategy to dynamically split models based on device capabilities. While Exo’s focus on heterogeneous, local networks influenced HyperCluster’s partitioning strategy, it is primarily designed for devices on the same local network, does not address the challenges of NAT traversal for internet-scale collaboration and has inference support for only a handful of model families.

The foundational idea of a general-purpose P2P network for AI is explored by Hyperspace, which is designed as a decentralized network for distributed model inference using Distributed Hash Tables (DHTs) and cryptographic protocols for security. Lattica (Yang et al. 2025b) provides a universal, cross-NAT communication framework for decentralized AI, focusing on establishing a globally addressable P2P mesh that can operate across firewalls. It serves as a foundational layer for moving data in distributed AI systems. HyperCluster builds upon the principles demonstrated by Hyperspace and Lattica by integrating P2P networking primitives directly into a purpose-built, application-level protocol for collaborative inference.

Network Topologies for Communication

The ring all-reduce algorithm (Patarasuk and Yuan 2009) is a well-established communication pattern for distributed training, and its principles have recently been adapted for inference. Prima.cpp (Li et al. 2025) demonstrated that a pipelined ring architecture can be highly effective for splitting LLMs across a static ”home cluster” of heterogeneous devices, even those with insufficient RAM, by overlapping disk I/O with computation and communication.

HyperCluster extends this concept by adapting the ring topology for a fully dynamic, geographically distributed wireless environment. By implementing our protocol on a modern P2P networking stack like Iroh (Iroh, Inc. 2024),

which provides robust NAT traversal and connection management, HyperCluster is designed to handle the fault tolerance and connectivity challenges inherent in decentralized wireless networks. This allows nodes to self-organize and maintain a computational ring without centralized coordination, representing a key step towards practical, internet-scale collaborative AI.

Methodology and System Architecture

The HyperCluster architecture is engineered for decentralized, collaborative inference over dynamic wireless networks. It comprises three core, interconnected subsystems: a **Network Layer** responsible for P2P communication and topology management, a **Partitioning Layer** that performs dynamic model sharding, and an **Execution Layer** that orchestrates the distributed inference process.

Network Layer: P2P Communication via Iroh

To meet the demands of a fully decentralized system, we build our network layer upon **Iroh** (Iroh, Inc. 2024), a content-addressed networking library that provides essential primitives for P2P environments: NAT traversal for establishing direct peer connections, gossip protocols for decentralized peer discovery, and document synchronization for maintaining consistent distributed state.

Each node advertises its capabilities (available memory and compute) through Iroh's document synchronization mechanism. The `TopologyManager` maintains a real-time view of all active peers and their resources, enabling dynamic cluster formation without centralized coordination. Node identities are content hashes, providing cryptographic addressing and deduplication guarantees.

Partitioning Layer: Memory-Weighted Sharding

To distribute a model with L layers across N heterogeneous nodes with memory capacities $M = \{m_1, m_2, ..., m_n\}$, we employ a **memory-proportional partitioning strategy**. Nodes are first sorted by available memory in descending order, establishing the ring topology. Layer assignments are then computed proportionally based on each node's memory fraction of the total cluster memory.

For modern Transformer architectures, layers have approximately uniform memory requirements, making layer count a reliable proxy for resource allocation. The partitioning algorithm is detailed in Algorithm 1.

Execution Layer: Ring-Pipeline Inference

Following layer assignment, nodes form a deterministic logical ring $N_0 \to N_1 \to \cdots \to N_{n-1} \to N_0$, where rank corresponds to the established memory-based ordering. Node N_0 (rank 0) holds the initial layers including token embeddings and serves as the entry point for inference requests.

Pipeline Operation: Inference proceeds in two distinct phases:

1. **Prefill Phase:** The initial prompt tokens are processed in a single forward pass through the ring. Each node executes its assigned layers sequentially, passing hidden states to its successor.

Algorithm 1: Memory-Weighted Layer Partitioning

Input: Total layers L, Sorted nodes N
Output: Layer assignments $W = \{w_1, ..., w_n\}$

Compute total memory: $M_{\text{total}} = \sum_{i=1}^{n} m_i$
Initialize layer counter: $L_{\text{current}} = 0$
for each node i from 1 to n **do**
  Compute memory fraction: $f_i = m_i / M_{\text{total}}$
  if $i = n$ **then**
    $\ell_i \leftarrow L - L_{\text{current}}$ {Last node gets remaining layers}
  else
    $\ell_i \leftarrow \max(1, \lfloor f_i \times L \rfloor)$ {Ensure at least 1 layer}
  end if
  $w_i \leftarrow \text{LayerWindow}(L_{\text{current}}, L_{\text{current}} + \ell_i - 1)$
  $L_{\text{current}} \leftarrow L_{\text{current}} + \ell_i$
end for
return W

2. **Autoregressive Generation Phase:** For each new token generation step (up to `max_tokens`), the single-token input must traverse the entire ring. The final node (N_{n-1}) computes output logits, samples the next token using nucleus sampling, and broadcasts it back to all nodes to synchronize state for the subsequent generation step.

The complete autoregressive generation loop for a 3-node cluster is illustrated below.

Listing 1: Ring Pipeline Autoregressive Generation (3 nodes, 24 layers).

```
# PREFILL PHASE (prompt="Hello")
Cycle 0: Node 0 [L0-7]:   Tokens -> H_0
Cycle 1: Node 1 [L8-15]:  H_0 -> H_1
Cycle 2: Node 2 [L16-23]: H_1 -> Logits
    -> Sample token_1

# GENERATION PHASE (token-by-token)

# Generation step 1: Process token_1
Cycle 0: Node 0 [L0-7]: token_1 -> H_0'
Cycle 1: Node 1 [L8-15]: H_0' -> H_1'
Cycle 2: Node 2 [L16-23]: H_1' -> Logits
    ' -> Sample token_2

# Generation step 2: Process token_2
Cycle 0: Node 0 [L0-7]: token_2 -> H_0''
...
# Repeat for max_tokens iterations
```

Transformer Layer Sharding via Dynamic Wrapping

Our sharding engine implements runtime layer extraction through the `TransformersShard` wrapper class, which dynamically isolates assigned layers from any Hugging Face `AutoModelForCausalLM` instance. The sharding mechanism extracts layer references directly from the base model's module hierarchy. The wrapper categorizes shards by their role:

- **Initial Shard (N_0):** Executes token embedding (`embed_tokens`) followed by layers $[0, k_0]$.

- **Intermediate Shards (N_i, $0 < i < n-1$):** Process layers $[k_{i-1}+1, k_i]$ on received hidden states.
- **Final Shard (N_{n-1}):** Executes layers $[k_{n-2}+1, L-1]$, applies final layer normalization (`norm`), and projects to vocabulary via the language model head (`lm_head`).

Distributed State Management:

For correct autoregressive generation, the inference state is synchronized across the ring. Each forward pass transmits the hidden states along with metadata like position IDs and the attention mask. The KV cache is not transmitted between nodes; instead, each node maintains and updates a local cache only for its assigned layers. This design ensures that only the hidden states flow through the ring, while the much larger KV cache remains node-local.

Communication Protocol

Tensor transmission between ring nodes uses Iroh's blob storage mechanism. Large tensors are stored as content-addressed blobs in Iroh's document store, which automatically synchronizes to peers. A lightweight metadata message containing the blob hash and tensor metadata is sent via Iroh's gossip layer.

Evaluation and Results

Experimental Setup

The hardware configuration for our experiments consists of three heterogeneous nodes: a MacBook Air M2 with 16GB RAM, a Mac Mini equipped with 8GB RAM, and a CPU-only Linux laptop with an Intel i5 processor and 8GB RAM. These nodes were located in three separate networks communicated over the Iroh network. The models tested were Llama-3.2-1B-Instruct (Grattafiori et al. 2024) and Qwen3-0.6B (Yang et al. 2025a).

Performance Metrics

In Table 1, we show metrics for throughput and latency, including **Tokens per second** (throughput), **Time to first token** (TTFT), and **Average time per token** (ATPT). We vary the number of nodes (1-3) and model size (0.5B, 1B) to measure weak and strong scaling.

Table 1: Detailed latency breakdown averaged across 32, 64 and 128 token sequence generation

Setup	TTFT (ms)	ATPT (ms)	Tokens/sec
1-Node			
Llama 3.2 1B	942.04	130.31	6.96
Qwen3 0.6B	661.15	113.44	8.06
2-Node			
Llama 3.2 1B	10505.98	816.73	0.99
Qwen3 0.6B	2096.35	558.53	1.70
3-Node			
Llama 3.2 1B	10507.73	953.35	0.89
Qwen3 0.6B	3028.68	669.27	1.36

Discussion

Insights and Observations

Our performance analysis reveals that scaling from a single to a multi-node setup introduces a severe communication bottleneck, degrading performance. As shown in the data, moving from one to two nodes causes throughput to plummet, while TTFT increases due to the initial setup and synchronization overhead required to distribute the model state across the nodes before the first token can be generated.

Limitations

1. Performance is sensitive to network latency, as inter-token generation speed is limited by the round-trip time through the ring.
2. While the system handles node discovery, it currently requires an inference restart on node failure; seamless migration of ongoing tasks is an area for future work.

Future Work

1. **Accelerating Inference Speed:** Integrate model quantization techniques to reduce computational and memory overhead, thereby increasing overall inference speed, enabling execution on truly resource-constrained edge devices.
2. **Optimizing Communication:**Implement optimized protocols for tensor communication, with a specific focus on leveraging peer-to-peer frameworks like Iroh to minimize latency.
3. **Expanding Model Support:** Support a wider range of architectures, including vision models for multimodal applications and Mixture of Experts (MoE) models to enable efficient scaling to larger parameter counts.

Conclusion

We present HyperCluster, a decentralized system enabling large language model inference across heterogeneous wireless networks. By combining P2P networking, memory-weighted sharding, and ring-based pipelined execution, our prototype demonstrates that collaborative AI inference is possible on resource-constrained devices. Our key contributions include a fully decentralized architecture, dynamic sharding that adapts to heterogeneous device capabilities, and a ring pipeline for efficient execution. This work opens new possibilities for AI-native wireless applications, reducing dependence on cloud infrastructure while enabling privacy-preserving AI at the edge.

Acknowledgments

This work builds upon ideas from the Exo project for model sharding, Prima.cpp for ring pipeline architectures, and IROH for P2P networking primitives. We thank the open-source community for these foundational contributions.

References

Borzunov, A.; Baranchuk, D.; Dettmers, T.; Riabinin, M.; Belkada, Y.; Chumachenko, A.; Samygin, P.; and Raffel, C. 2023. Petals: Collaborative inference and fine-tuning of large models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, 558–568.

ExoLabs. 2024. exo: Run your own AI cluster at home with everyday devices. https://github.com/exo-explore/exo. Accessed 2025-11-08.

Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. 2024. The llama 3 herd of models. *arXiv:2407.21783*.

Iroh, Inc. 2024. Iroh: P2P for everyone. https://www.iroh.computer. Accessed 2025-11-08.

Li, Z.; Li, T.; Feng, W.; Guizani, M.; and Yu, H. 2025. PRIMA. CPP: Speeding Up 70B-Scale LLM Inference on Low-Resource Everyday Home Clusters. *arXiv:2504.08791*.

Patarasuk, P.; and Yuan, X. 2009. Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing*, 69(2): 117–124.

Rasley, J.; Rajbhandari, S.; Ruwase, O.; and He, Y. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 3505–3506.

Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; and Catanzaro, B. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv:1909.08053*.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Le Scao, T.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. 2020. Transformers: State-of-the-Art Natural Language Processing. In Liu, Q.; and Schlangen, D., eds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Online: Association for Computational Linguistics.

Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; et al. 2025a. Qwen3 technical report. *arXiv:2505.09388*.

Yang, W.; Liu, J.; Wang, S.; Song, X.; Ai, L.; Yang, E.; and Shi, T. 2025b. Lattica: A Decentralized Cross-NAT Communication Framework for Scalable AI Inference and Training. *arXiv:2510.00183*.

Zheng, L.; Li, Z.; Zhang, H.; Zhuang, Y.; Chen, Z.; Huang, Y.; Wang, Y.; Xu, Y.; Zhuo, D.; Xing, E. P.; et al. 2022. Alpa: Automating inter-and {Intra-Operator} parallelism for distributed deep learning. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, 559–578.