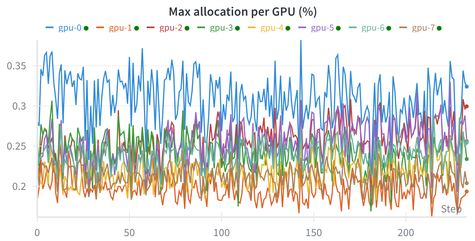
Least-Loaded Expert Parallelism: Load Balancing an Imbalanced Mixture-of-Experts

Xuan-Phi Nguyen · Shrey Pandit · Austin Xu · Caiming Xiong · Shafiq Joty | Salesforce AI Research

Problem: MoE Routing Is Intrinsically Imbalanced

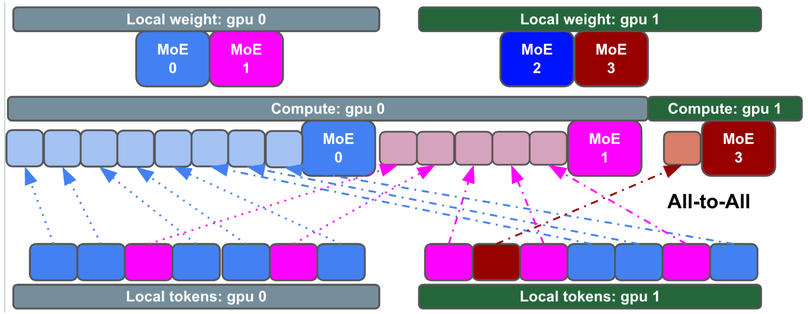
Even well-trained MoE models route tokens **unevenly** — a **desirable** sign of expert specialization. Profiling **gpt-oss-20b** under 8-way EP, 1 GPU persistently absorbs more than twice its fair share of tokens.

Worse, the imbalance is large and shifts **unpredictably** from batch to batch — no single expert is reliably hot — so it cannot be fixed by any *static* expert-to-GPU placement.



Max load per GPU vs. step (gpt-oss-20b, 8-way EP). GPU-0 stays at 30–35% vs. the 12.5% balanced share.

Standard Expert Parallelism (EP) Suffers Under Imbalance



Experts are sharded across GPUs; tokens are dispatched (All-to-All) to the device holding their expert, computed, then combined back. **Imbalanced routing overloads GPU 0 while GPU 1 idles.**

- Collective latency is bound by the **slowest** device → spiking latency under imbalance.
- Peak memory on hot GPUs grows up to **4×** → out-of-memory (OOM) crashes.
- The gap **widens** with more GPUs — every device stalls waiting on the single straggler.
- Post-training & inference **cannot** use load-balancing losses — they alter the pretrained model.
- Naive fixes (small batches, redundant experts) cost throughput or memory and still fail in the worst case.

Cost Model: Why Imbalance Hurts

Per-GPU latency and peak memory both scale with the tokens *Bi* routed to each local expert:

$$T_{gpu} \approx \sum_i \left(T_{launch} + B_i \cdot t(B_i, D, H) \right)$$

$$M_{gpu} \approx \sum_i \left(B_i D + DH + B_i H \right)$$

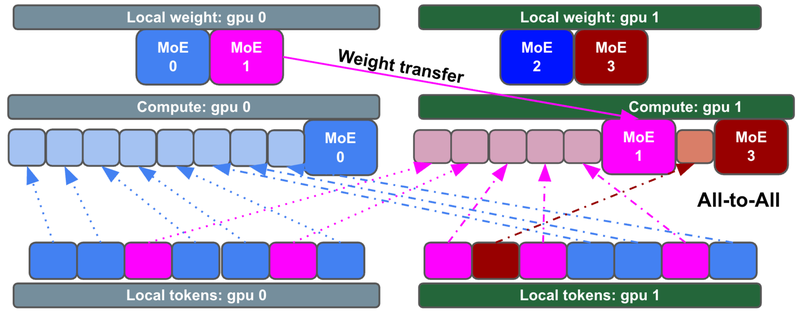
Under EP, a hot GPU's *Bi* can approach the **global** batch size ⇒ latency spikes and memory blows up. LLEP caps *Bi* near the balanced mean on every device, freeing memory to run **larger batches at higher throughput**.

Why Existing Fixes Fall Short

- **Load-balancing losses** change *which* expert a token picks — they retrain the model and cannot be applied at post-training or inference.
- **Expert replication (EPLB)** statically duplicates hot experts: extra memory, and it lags the routing that drifts every batch.
- **Capacity dropping** discards overflow tokens, degrading quality.

LLEP keeps the routing exact and instead rebalances the *physical* compute — at training, post-training, and inference alike.

LLEP Least-Loaded Expert Parallelism OURS



When a GPU exceeds its capacity threshold, LLEP spills the excess tokens — and the corresponding expert weights (P2P) — to the least-loaded GPUs, so all devices finish within the minimum collective latency under a memory budget.

- **Least-Loaded Assignment:** greedily place each expert's load, prioritizing native GPUs and spilling overflow to the least-loaded device.
- **Cost-aware:** a transfer fires only when moving tokens beats computing locally (balances compute, memory & comms).
- **Adaptive:** near-balanced routing (ratio < λ) falls back to standard EP at near-zero overhead.
- **Exact & trainable:** mathematically identical to EP, with full backward-pass support — training **and** inference.

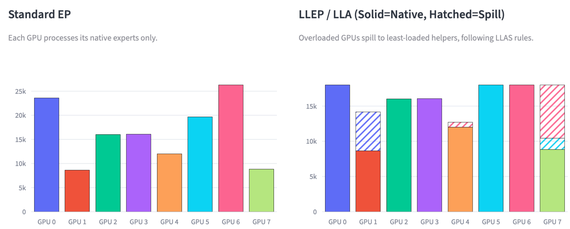
Least-Loaded Assignment (LLA)

```
# per-expert loads l, capacity mα, min-chunk m
sort experts by load (descending)
for each expert e (largest → smallest):
  fit as much of e on its native GPU as mα allows
  spill overflow to the least-loaded GPUs
  # skip a spill if its chunk < m
emit a P2P weight transfer per spilled expert

# each step:
if max(l)/mean(l) < λ  →  run plain EP
else → dispatch · P2P weights · grouped-GEMM · combine
```

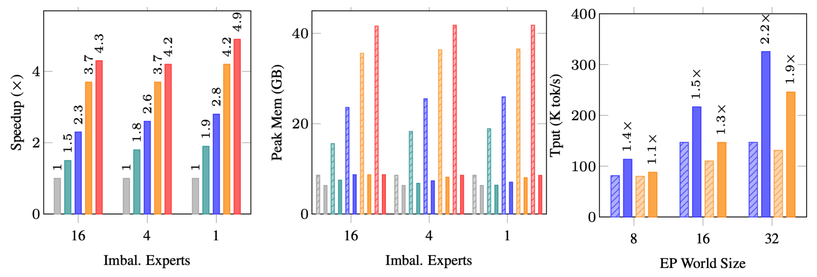
LLEP in Action

1. GPU Load Comparison



EP overloads GPU 6 (~26K tokens) while others idle; **LLEP spills the excess** (hatched) so the busiest GPU holds just **~18K tokens**.

Headline Results



(a) Speedup: LLEP vs. EP. (b) Peak memory (**lower is better**) (c) Full-model throughput

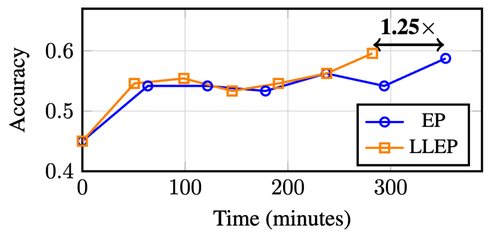
Imbal: Bal 30% 50% 80% 95% **gpt-oss:** 20b 120b **Method:** EP LLEP

Up to 5× faster and more memory-efficient than EP under imbalance — with 2.2× full-model throughput.

5× faster MoE layer

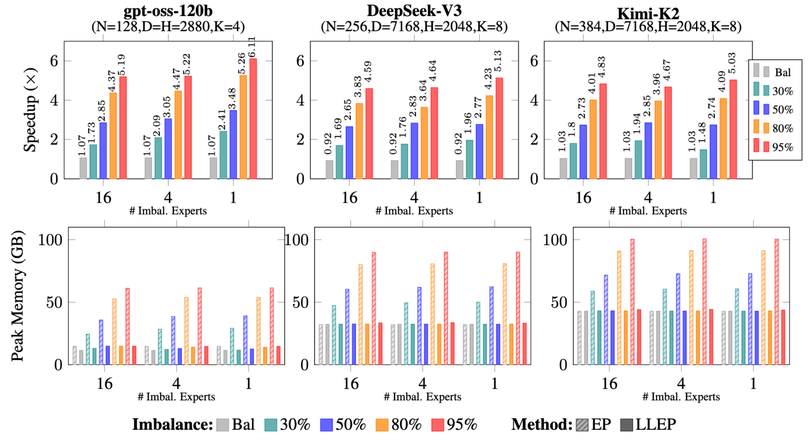
4× lower peak memory

1.9× faster gpt-oss-120b



Real gpt-oss-20b training (ZeRO-3 + CPU offload): same AIME'25 accuracy, reached **1.25× faster**.

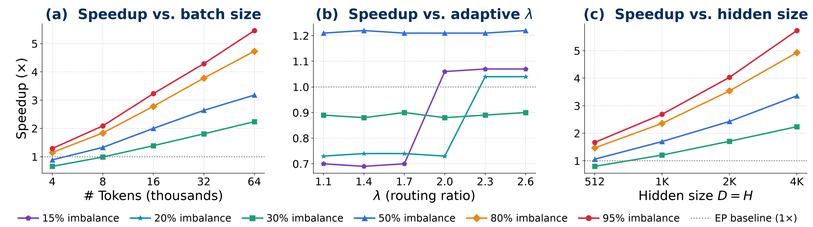
Generalizes Across Architectures And Scale



Imbalance: Bal 30% 50% 80% 95% **Method:** EP LLEP

Speedup (top, up to 6.1×) and peak memory (bottom) for gpt-oss-120b, DeepSeek-V3 and Kimi-K2 across imbalance levels. Higher imbalance → larger speedup; LLEP memory stays near-constant while EP grows up to 5×.

Ablation: When LLEP Wins More



Speedup grows with (a) batch size and (c) hidden size as compute saturates GPUs and dominates comms; **adaptive λ (b)** reverts to plain EP when near-balanced to avoid transfer overhead — up to **5.7×**.

Takeaways

- **Embrace imbalance** — preserve learned expert specialization instead of fighting it.
- **Exact computation**, no approximation — the first load balancer for **training and inference** alike.
- **Support both Training and Inference** without weight modifications or extra memory overhead.
- Tunable knobs (**α, m, λ**) enable principled, hardware-specific optimization.

Paper & code →