# Leveraging Expert Usage to Speed up LLM Inference with Expert Parallelism

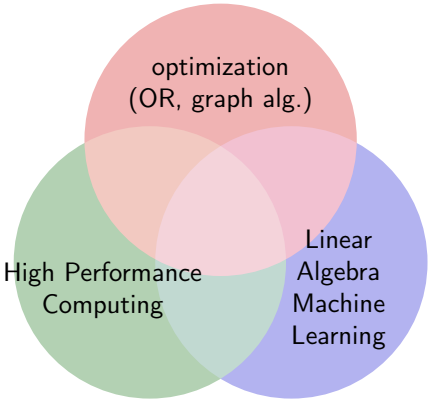
**Loris Marchal**
(CNRS & ÉTS Montréal)

***April 2025***



# Who am I?

- Senior researcher from CNRS
- Actually at ILLS, in Montréal
  International Laboratory on Learning Systems
  *ETS Montréal, McGill, MILA, CNRS, Univ. Paris Saclay*



# Outline

Inference of Mixture-Of-Experts LLMs

Opportunities for parallelism and optimization

Experimental evaluation

Extension to larger subsets of experts

Other research projects for LLM inference
- Paging for Mixture-Of-Experts Models
- Design of parallel LLMs

# Outline

Inference of Mixture-Of-Experts LLMs

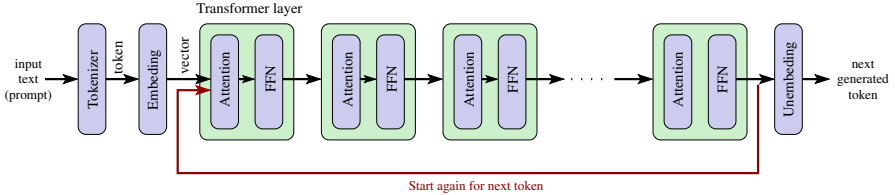
Opportunities for parallelism and optimization

Experimental evaluation

Extension to larger subsets of experts

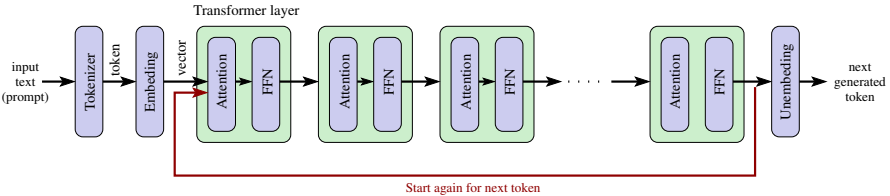
Other research projects for LLM inference
- Paging for Mixture-Of-Experts Models
- Design of parallel LLMs

# Large Language Models (Transformer-based)




- Input sequence transformed into tokens (tokenization)
- Tokens encoded are vectors (embedding)
- Network made of a succession of layers, each layer contains:
  - Attention mechanism (relation with other tokens)
  - Feed Forward Network (MultiLayered Perceptron)
- Resulting vector converted into token (un-embedding)

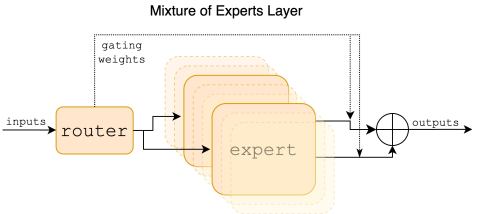
# Challenge for high inference throughput




- Producing one token: going through the whole network, one layer after the other
- Token $i$ needed to produce token $i + 1$
- LLMs have reached huge sizes: several high-end GPUs needed just to store the model weights
- Or resort to costly I/O (load weights when needed)

# LLM with Mixture-Of-Experts

- MoE proposed to reduce computation at inference time




- Replace the FFN block in each transformer with expert module

- Gating function (router) selects $k$ experts out of $n$ (Mixtral: 2 out of 8, Llama-7B: 4 / 16, DeepSeek-R1: 6 / 64)
- Large set of weights for training
- Smaller set used for inference

Rest of talk: we consider pairs of experts (e.g. Mixtral) but can be extended for any subset

# Outline

Inference of Mixture-Of-Experts LLMs

Opportunities for parallelism and optimization

Experimental evaluation

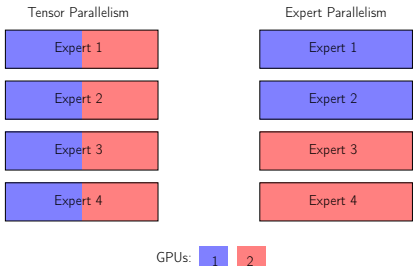
Extension to larger subsets of experts

Other research projects for LLM inference
- Paging for Mixture-Of-Experts Models
- Design of parallel LLMs

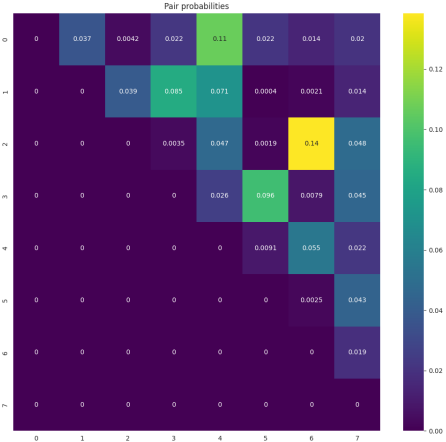
# Distributing experts on multiple GPUs

- Need for serving large language models fast
- Even on commodity machines
- Usually requires several GPUs to store the model
- How to distribute model weights?

1. Tensor parallelism
   (may not be possible
   with complex experts☹)
2. Expert parallelism
   (selected experts may lie on the
   same GPU, leaving other idle☹)



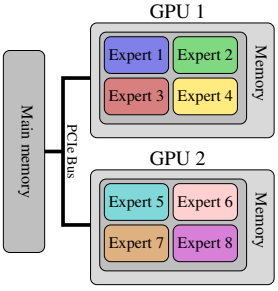
# Probability of expert usage for a given layer

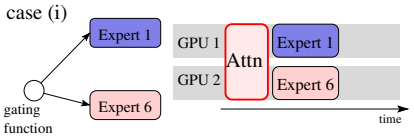



- By design: same probability for each expert to be used
- But: different probabilities for each pairs of experts

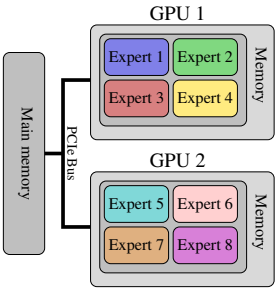
Key idea: map high-probability pairs of experts on distinct GPUs
$\Rightarrow$ allow for parallel inference

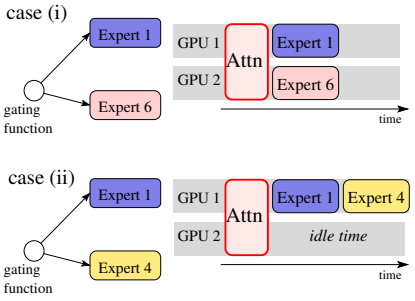
# Problem 1: static allocation, no replication



# Problem 1: static allocation, no replication

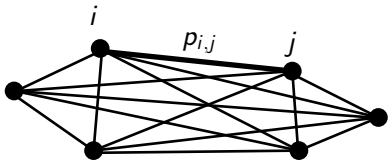



Objective:
map experts to GPU so that frequent pairs are processed in parallel.

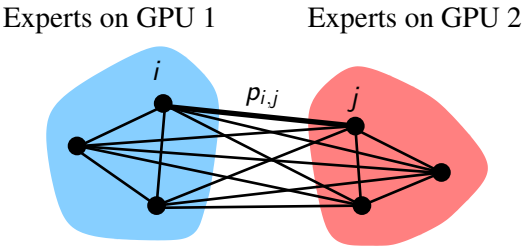
# Problem complexity of static allocation

**Input:** usage probability of expert pairs
**Output:** mapping of experts to the 2 GPUs that minimizes expected processing time



# Problem complexity of static allocation

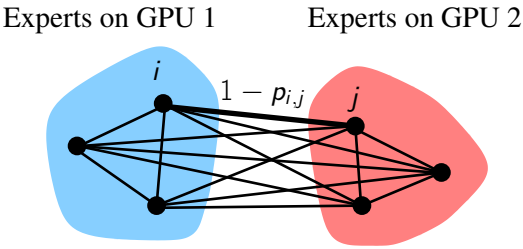
**Input:** usage probability of expert pairs
**Output:** mapping of experts to the 2 GPUs that minimizes expected processing time




Search for graph bisection with maximal cut

# Problem complexity of static allocation

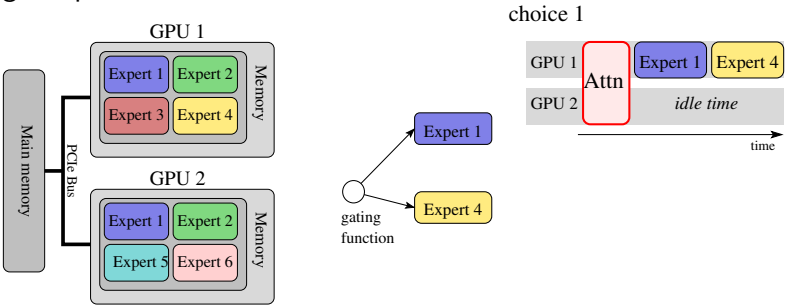
**Input:** usage probability of expert pairs
**Output:** mapping of experts to the 2 GPUs that minimizes expected processing time




Search for graph bisection with ~~maximal~~ minimal cut
$\Rightarrow$ NP-complete problem

# Problem 2: dynamic allocation with replication

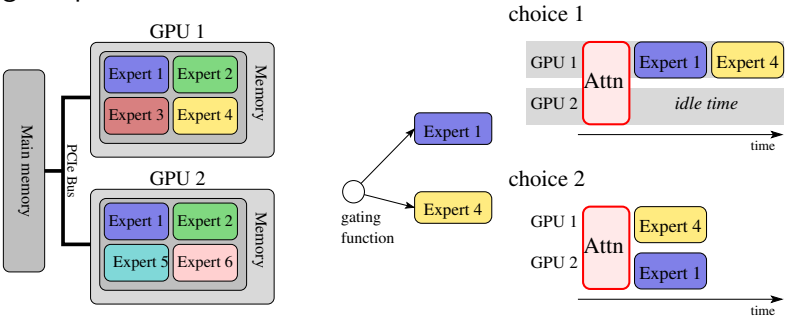
Mapping experts→GPU is fixed, choose which replica is used to serve a given pair.




Replica choice (=dynamic allocation) influences total processing time

# Problem 2: dynamic allocation with replication

Mapping experts→GPU is fixed, choose which replica is used to serve a given pair.

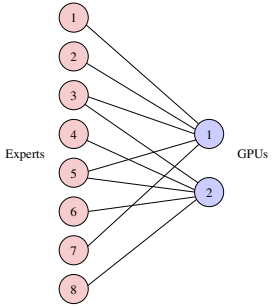



Replica choice (=dynamic allocation) influences total processing time

How to compute optimal dynamic allocation with minimal processing time?

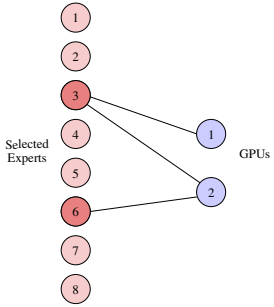
# Solving the dynamic allocation problem

Build bipartite graph corresponding to expert mapping:



# Solving the dynamic allocation problem

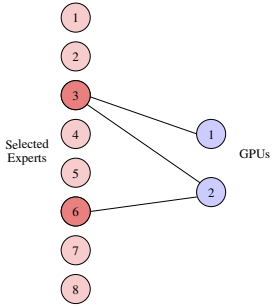
Build bipartite graph corresponding to expert mapping:




- For each expert pair (subset), consider graph restricted to the subset

# Solving the dynamic allocation problem

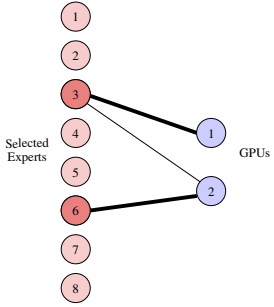
Build bipartite graph corresponding to expert mapping:




- For each expert pair (subset), consider graph restricted to the subset
- Is there a matching covering both experts ?

# Solving the dynamic allocation problem

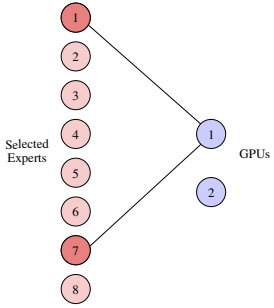
Build bipartite graph corresponding to expert mapping:




- For each expert pair (subset), consider graph restricted to the subset
- Is there a matching covering both experts ?
  - Yes ? Processing time = 1

# Solving the dynamic allocation problem

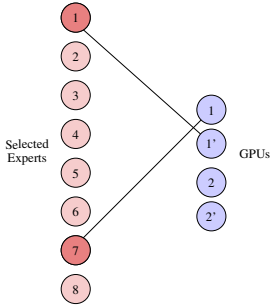
Build bipartite graph corresponding to expert mapping:




- For each expert pair (subset), consider graph restricted to the subset
- Is there a matching covering both experts ?
  - Yes ? Processing time = 1
  - No ?

# Solving the dynamic allocation problem

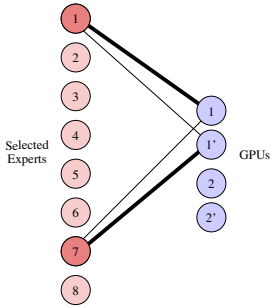
Build bipartite graph corresponding to expert mapping:




- For each expert pair (subset), consider graph restricted to the subset
- Is there a matching covering both experts ?
  - Yes ? Processing time = 1
  - No ? Duplicate GPUs (corresponding to 2 times slots on each GPUS)

# Solving the dynamic allocation problem

Build bipartite graph corresponding to expert mapping:




- For each expert pair (subset), consider graph restricted to the subset
- Is there a matching covering both experts ?
  - Yes ? Processing time = 1
  - No ? Duplicate GPUs (corresponding to 2 times slots on each GPUS)
  - There exists a matching covering both experts $\Rightarrow$ Processing time = 2

# How to solve the static allocation problem?

Two strategies that solves both problems at once:

1. Linear programming solution:
   $x_{i,k} = 1$ iff expert $i$ mapped on GPU $k$
   $y_{i,j,k} = 1$ iff expert $i$ on GPU $k$ is used for pair $i, j$

2. Simple greedy heuristic
   - Start with empty mapping
   - Map new copy of expert $i$ on GPU $k$ with maximal gain on cost
   - Stop when memory full
   - Compute which copy to use with graph matching algorithm

# Outline

Inference of Mixture-Of-Experts LLMs

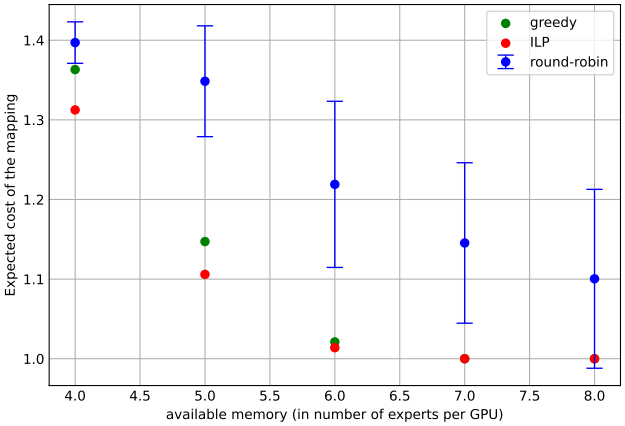
Opportunities for parallelism and optimization

Experimental evaluation

Extension to larger subsets of experts

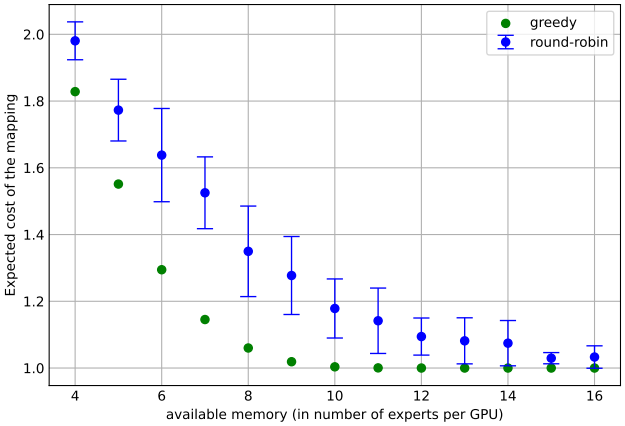
Other research projects for LLM inference
- Paging for Mixture-Of-Experts Models
- Design of parallel LLMs

# Potential gain on expected processing time 1/2




Simulations with Mixtral usage (pairs of experts, 2 GPUs)

# Potential gain on expected processing time 2/2



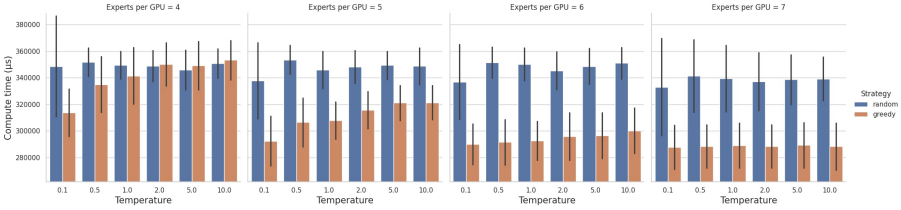

Simulations with DRBX usage (subset of 4 experts, 4 GPUs)

# Preliminary experimental evaluation

- Mixtral model on HuggingFace
  (using only one layer for now)
- Added support for expert parallelism
- Real + Synthetic distributions of expert probabilities
  $\Rightarrow$ test a range of variance
- 2 GPUs, range of memory constraint:

4 experts/GPU (no duplication) $\rightarrow$ 8 experts/GPU (complete duplication)

# Experiments with Mixtral on two GPUs




X: high variance in pair probability (left) – low variance (right)

Y: inference time (truncated)

# Outline

Inference of Mixture-Of-Experts LLMs

Opportunities for parallelism and optimization

Experimental evaluation

Extension to larger subsets of experts

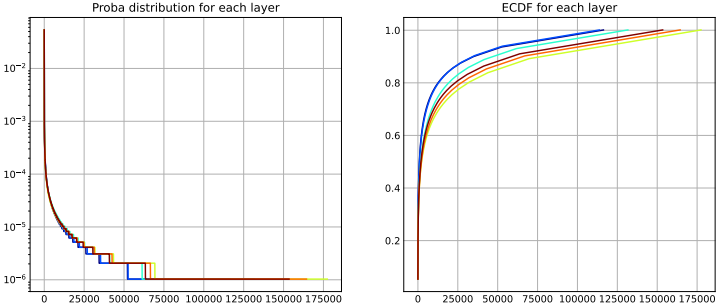
Other research projects for LLM inference
- Paging for Mixture-Of-Experts Models
- Design of parallel LLMs

# Extension to larger subsets of experts

- Theory: Algorithms/Linear Program easily extended to any subset size
- Practice: DeepSeek-R1: choose 6 experts among 64 experts
  $\rightarrow$ 74.974.368 subsets
- Up to 6 matching computations per subset, just to evaluate the cost of a solution!
- Many more for the greedy algorithm
- Impractical computation times ☹
- Forget about optimization?☹

# Simplifying the problem

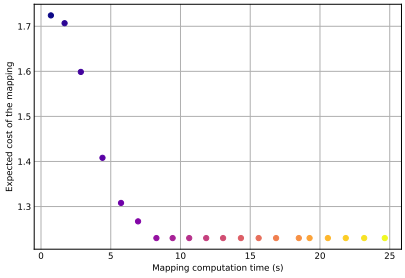
Not all expert subsets are created equal:

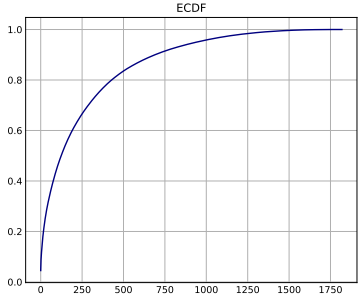



subset usage for DeepSeek-R1, first 6 layers

- 0.2 % of the subsets account for 99% of the subset usage
- We can safely consider only 150k subsets for DeepDeek
- 0.03 % of the subsets account for 85% of the subset usage
- Consider 25k subsets should be enough for sufficient accuracy

# Preliminary experiments




Simulations with the DRBX model and greedy mapping heuristic
Using from 90 (left) to 1820 (right) subsets

Using only subsets that cover 80% of the cases is enough to get optimal performance

# Conclusions/Perspectives

- Proof-of-concept for expert parallelism at inference
- Take advantage of usage statistics for better performance
- What if memory is too limited ? How to efficiently use expert pair popularities?

More at `https://hal.science/hal-04994839`

# Outline

Inference of Mixture-Of-Experts LLMs

Opportunities for parallelism and optimization

Experimental evaluation

Extension to larger subsets of experts

Other research projects for LLM inference
- Paging for Mixture-Of-Experts Models
- Design of parallel LLMs

# Paging for Mixture-Of-Experts Models

What to do if not enough memory to store all experts?

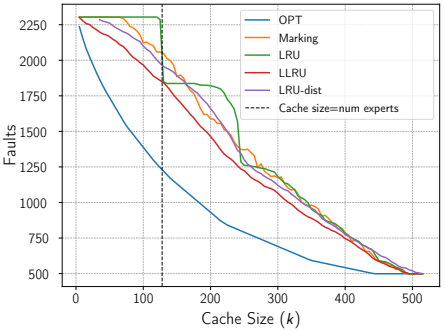
- Sequence of experts (a few experts per layer)
- Some experts may reappear soon, some of them later
- Special case of cache management to avoid costly expert loading

# Paging for Mixture-Of-Experts Models

What to do if not enough memory to store all experts?

- Sequence of experts (a few experts per layer)
- Some experts may reappear soon, some of them later
- Special case of cache management to avoid costly expert loading

- Definition of the layered paging problem
- New lower bounds on the cache misses, both for deterministic and randomized algorithms
- Layer-aware LRU eviction policy to avoid worse-case scenarios
- Use predictors to improve performance?



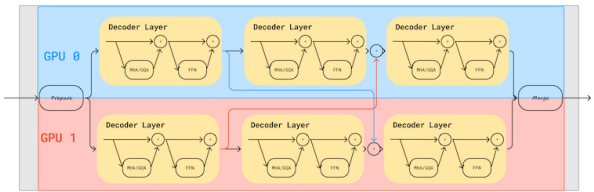

https://hal.science/hal-04961621

# Design of parallel LLMs

- Inference throughput limited by the sequentiality between layers
- Model parallelism:
  - ☺ Good to training (batches processed independently, may be pipelined)
  - ☹ Useless for inference, except for distributing the model
- Tensor parallelism:
  - ☹ Costly synchronizations (all-reduce)

# Design of parallel LLMs

- Inference throughput limited by the sequentiality between layers
- Model parallelism:
  - ☺ Good to training (batches processed independently, may be pipelined)
  - ☹ Useless for inference, except for distributing the model
- Tensor parallelism:
  - ☹ Costly synchronizations (all-reduce)




- Run sub-models in parallel, one on each GPU
- Synchronizations every few layers
- Non-blocking synchronizations

`https://hal.science/hal-04920049`