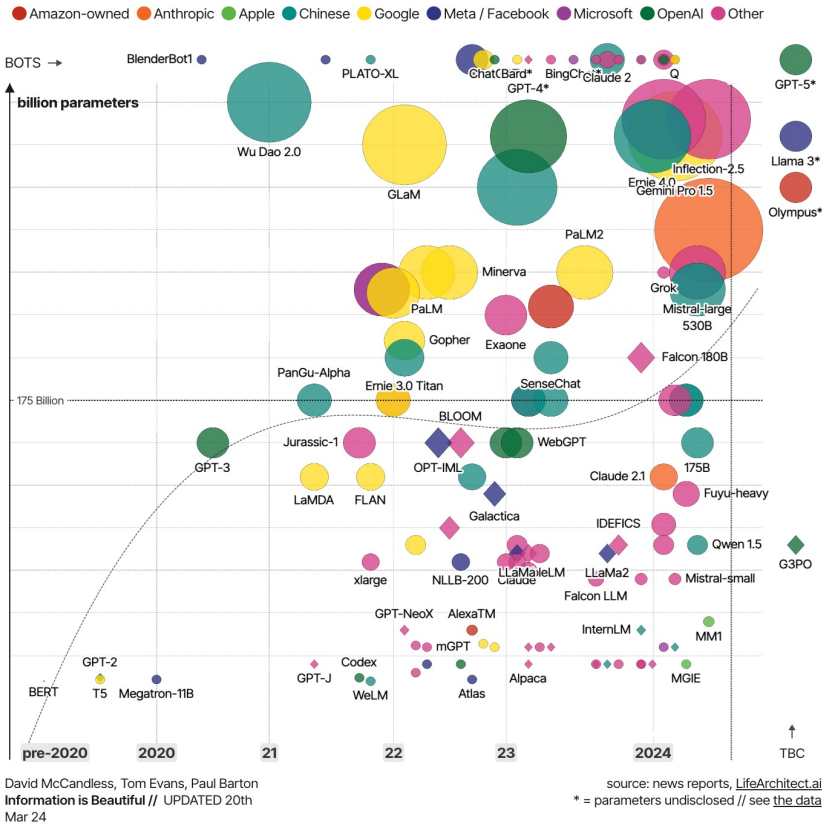
PowerInfer: Fast Large Language Model Serving with a Consumer-grade GPU

Yixin Song, Zeyu Mi, Haotong Xie, Haibo Chen

Shanghai Jiao Tong University

The Era of LLMs



LLMs: A Flourishing Ecosystem

OpenAI

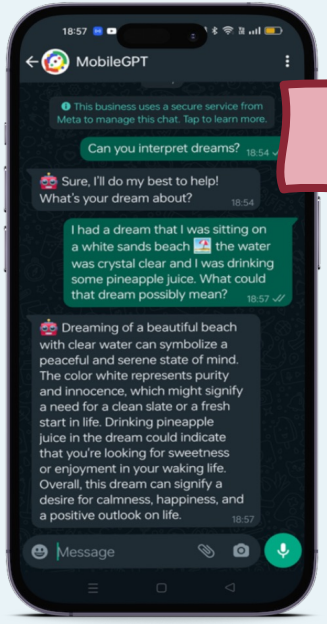
LLaMA

Gemini 1.5 Pro

The Tech Giants' Race in LLM Development

Local LLMs Empower Billions Local Devices

AI Phone



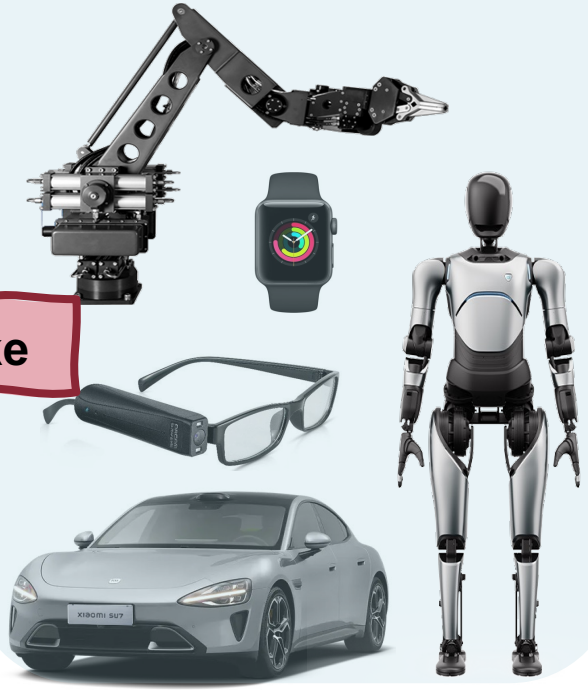
Smarter

AI PC

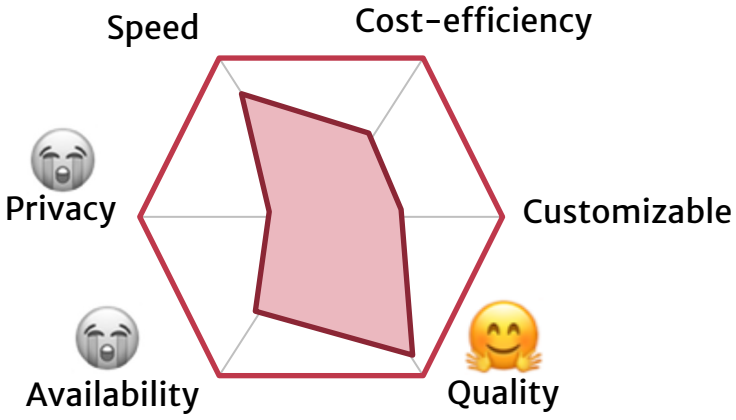
Human-like

Our Focus

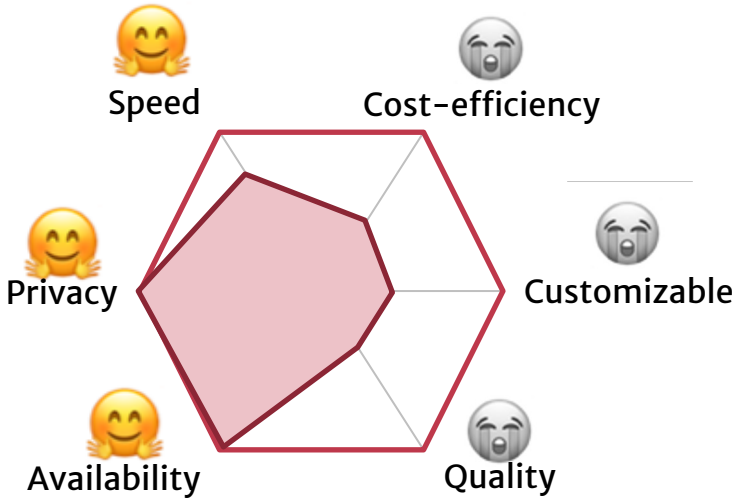
Embodied AI



Cloud LLM v.s. Local SLM

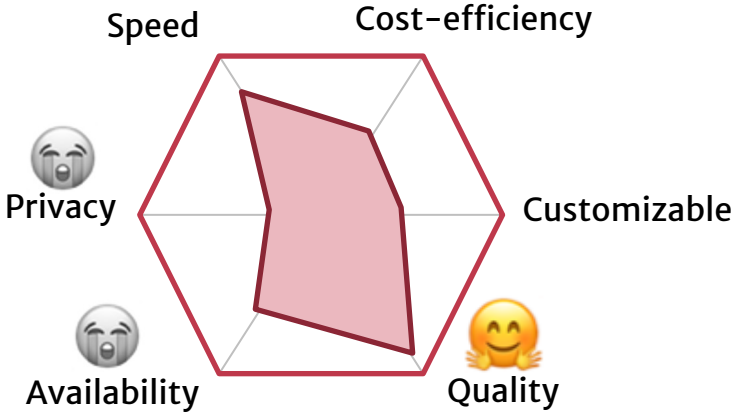


Cloud LLM

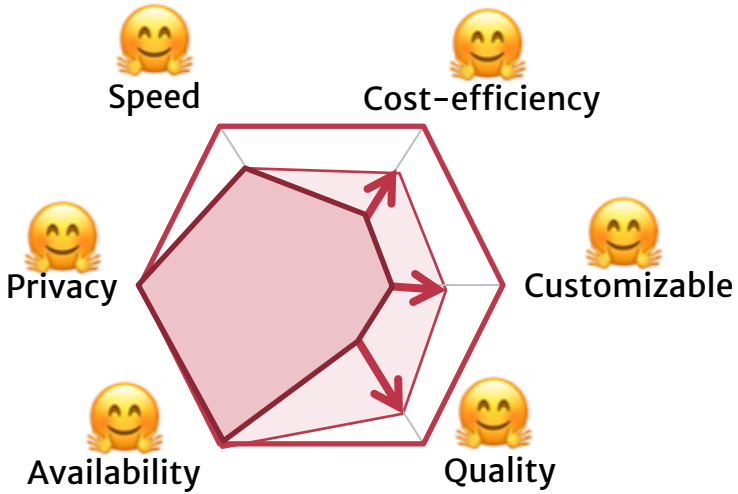


Local SLM

Cloud LLM v.s. Local SLM



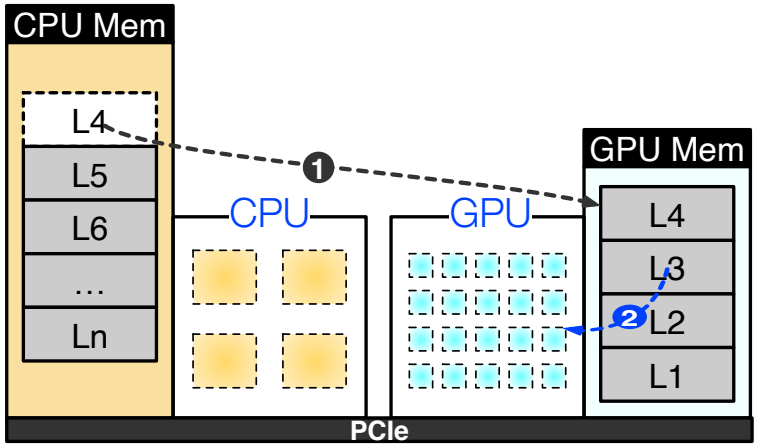
Cloud LLM



Local SLM

Can we achieve better by deploying Larger LM Locally?

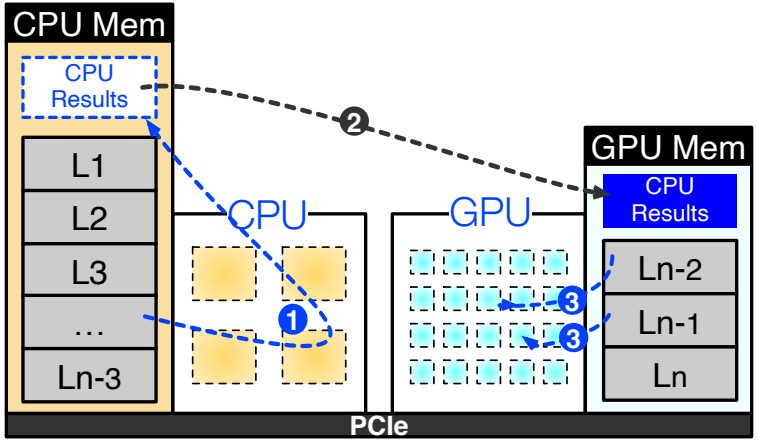
Prior Approaches: Offloading



GPU-Centric Offloading

FlexGen (ICML 2023)

- Computation : GPU
- Memory : CPU DRAM & GPU VRAM



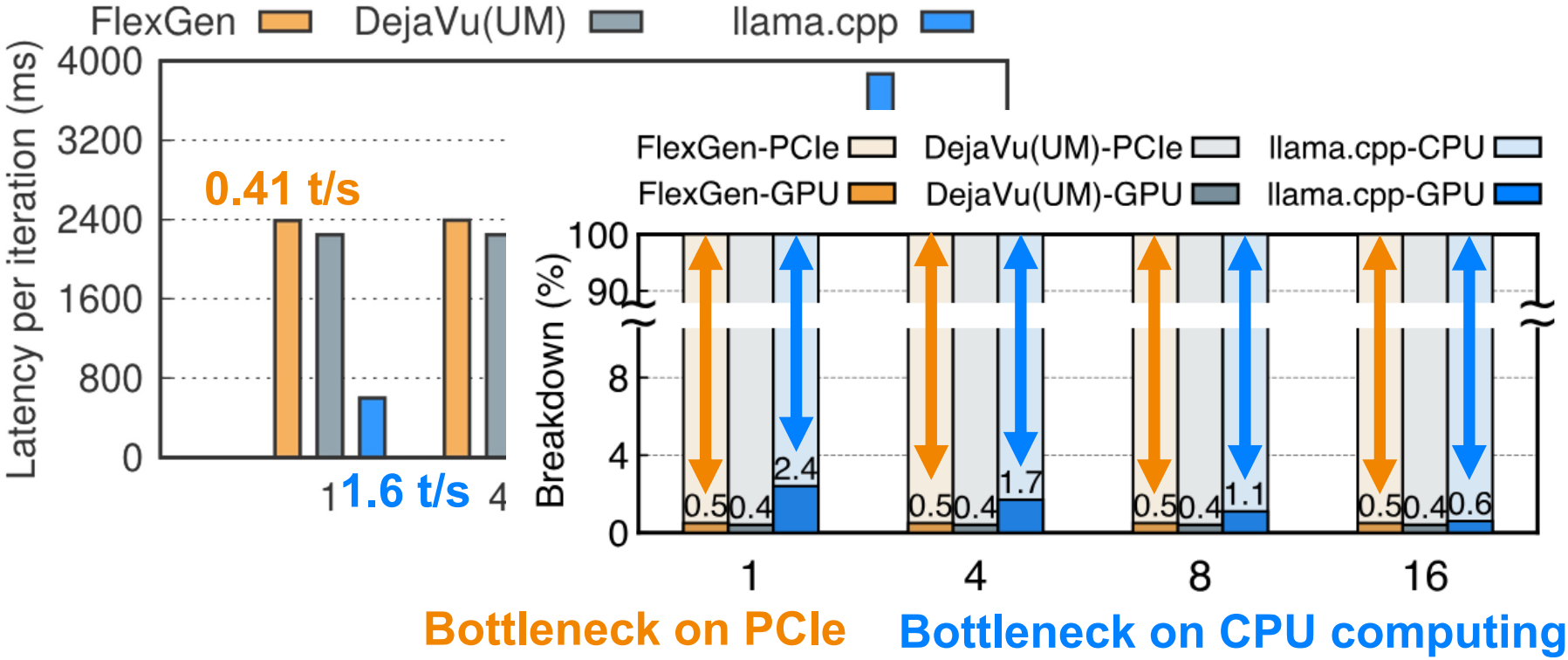
GPU-CPU Hybrid Offloading

llama.cpp

- Computation : CPU and GPU
- Memory : CPU DRAM & GPU VRAM

Poor Generation Speed

Both offloading methods bring significant slow latency

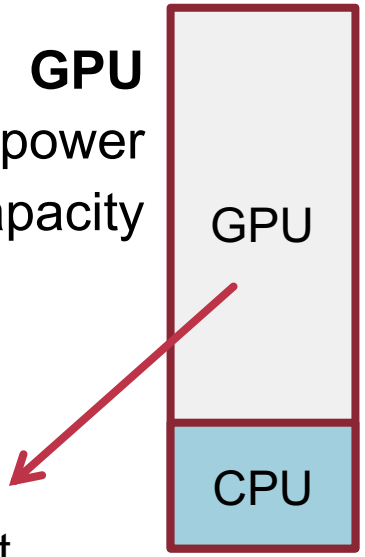


Locality Mismatch

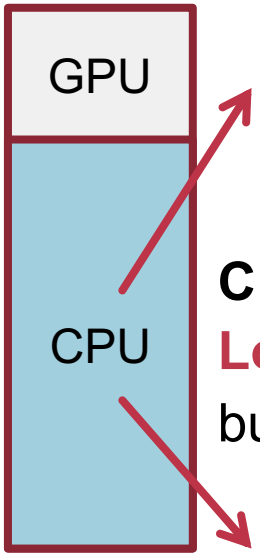
GPU

High computational power but **low** memory capacity

Small, frequently accessed working set



Ideal compute utilization



Existing compute utilization

Large, less frequently accessed working set

CPU

Low computational power but **large** memory capacity

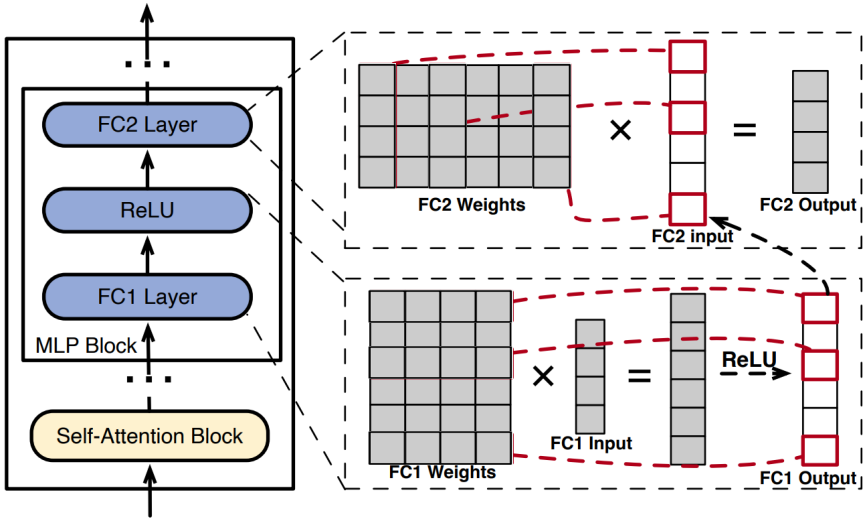
CPU handles heavy workload

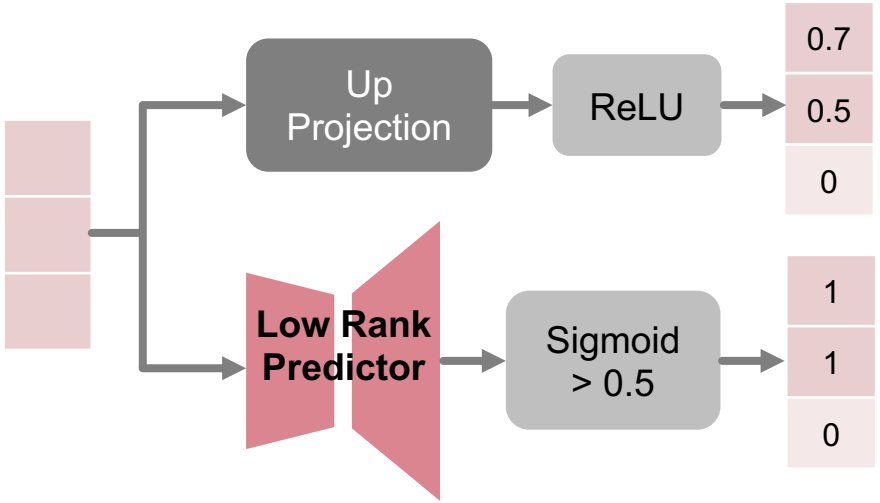
Full-Parameter Activation Leads to Severe Locality Mismatch

Opportunity: Predictable Sparsity

LLM exhibit predictable sparse activation

- A subset of neurons contributes to the output for a single token
- The activated neurons can be predicted with a lightweight predictor

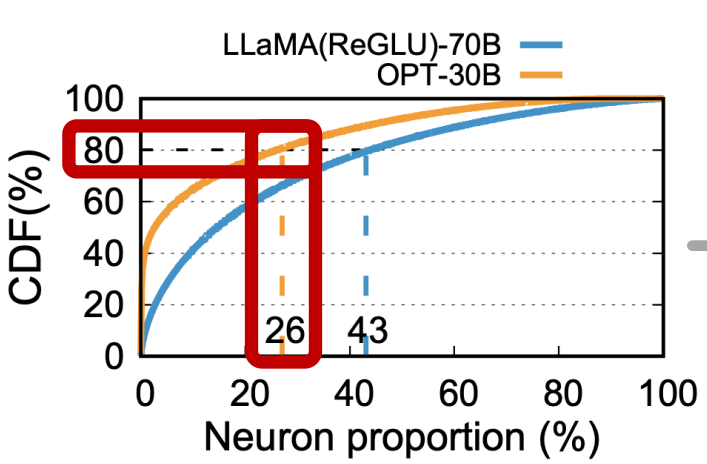


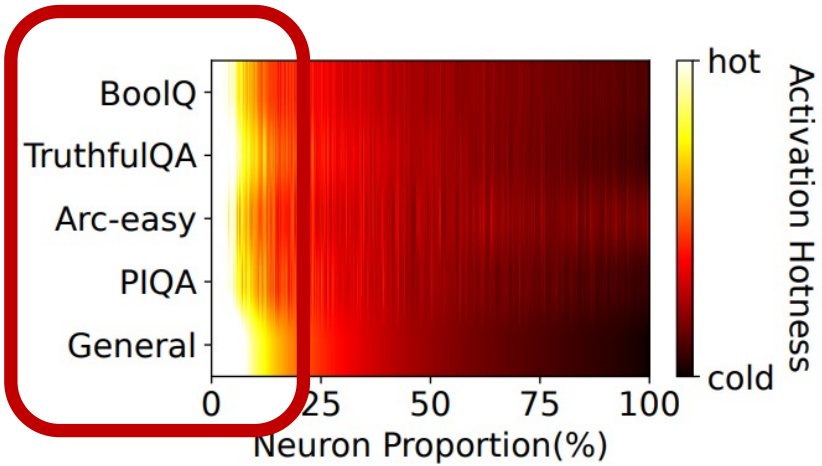


Does neuron activation show a skewed distribution?

Insight-1: Power-law Activation

Hot Neurons vs. Cold Neurons



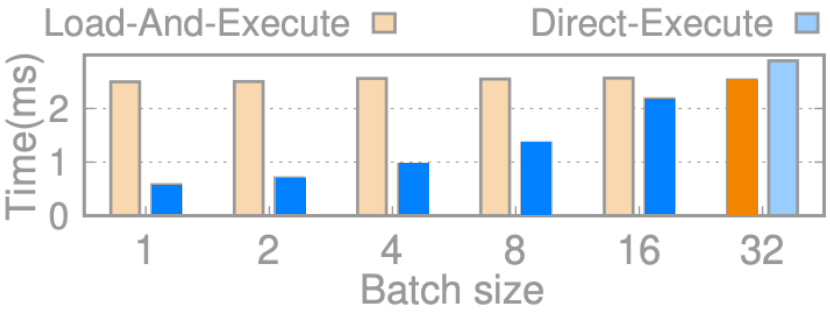


26% neurons contributes to **80%** computation

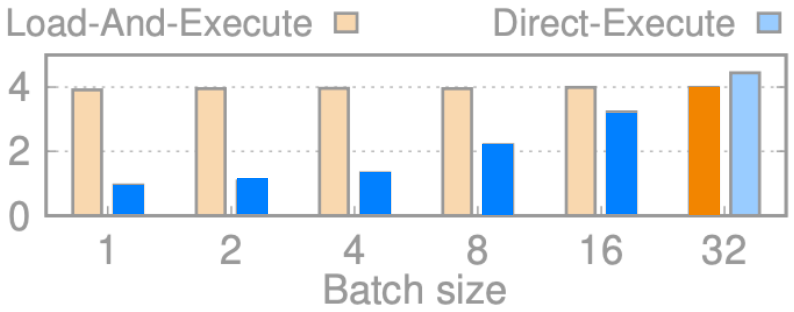
Hot neurons **consistently** frequently activated across different tasks

Hot neurons should be placed on **GPU** due to their high locality

Insight-2: Fast In-CPU Computation



(a) MLP block



(b) Attention block

When cold neurons are offloaded on DRAM, which offloading method is better for latency?

Load the weight to GPU VRAM then compute using **GPU** (prefill)

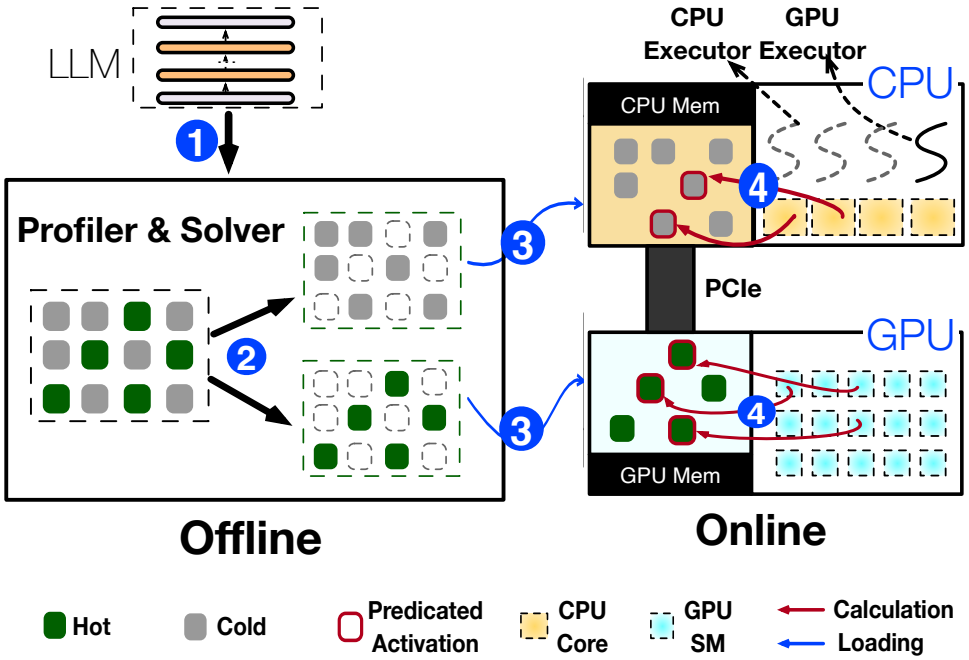
Direct execute computation on **CPU** (decode)

PowerInfer: Combine Model Sparsity & System Locality

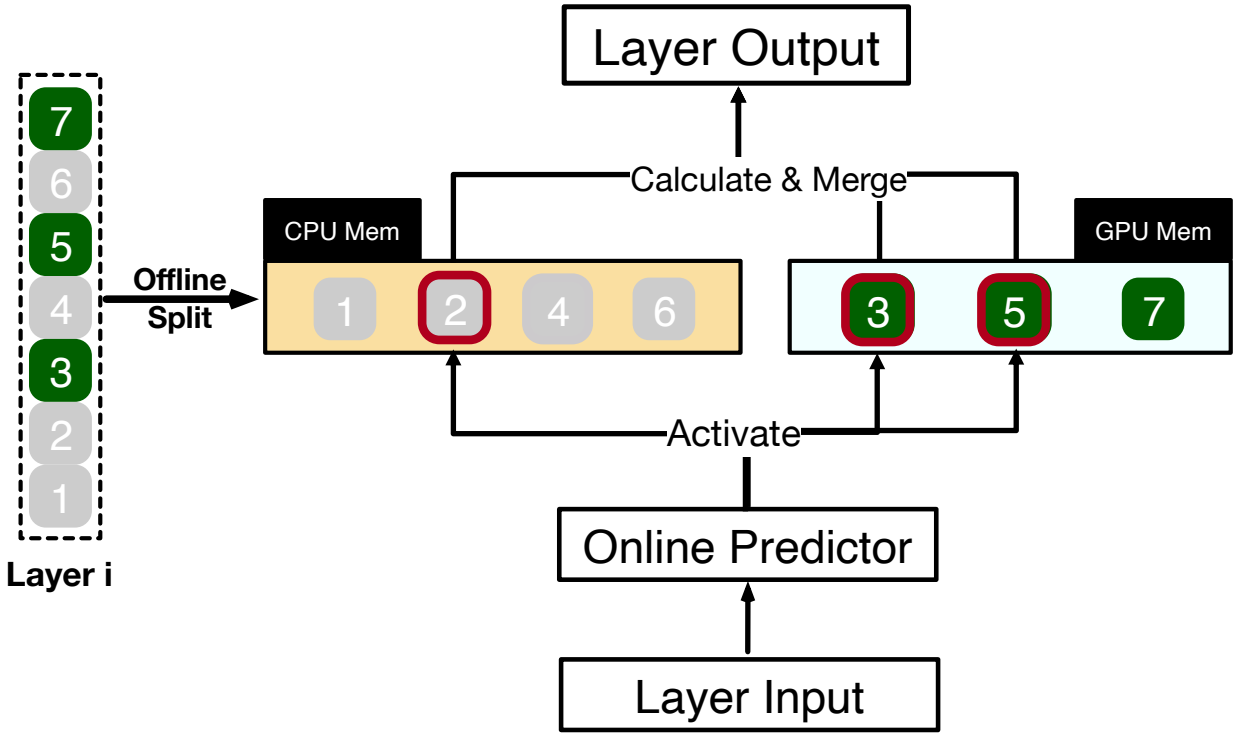
Design

GPU
load and compute
small hot neurons

CPU
load and compute
large cold neurons



Concrete Example



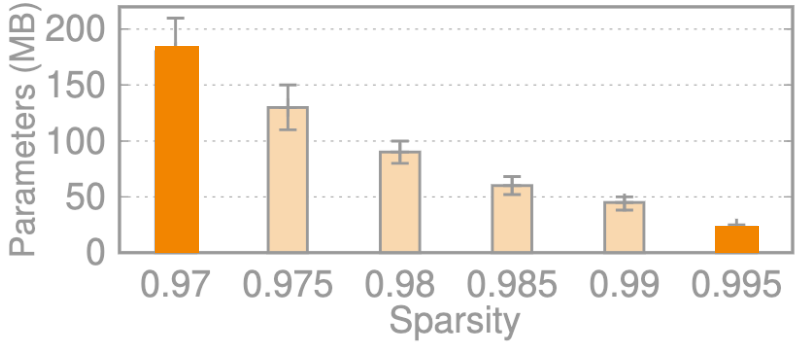
Adaptive Predictor

Predictors occupy a considerable GPU memory

- Predictors needs another 7GB VRAM for 30B LLMs.
  (Occupy over 30% VRAM Capacity on NVIDIA RTX 4090)

Not every layer need the same size predictor

- Adaptive training method for predictors
- Save 50% VRAM



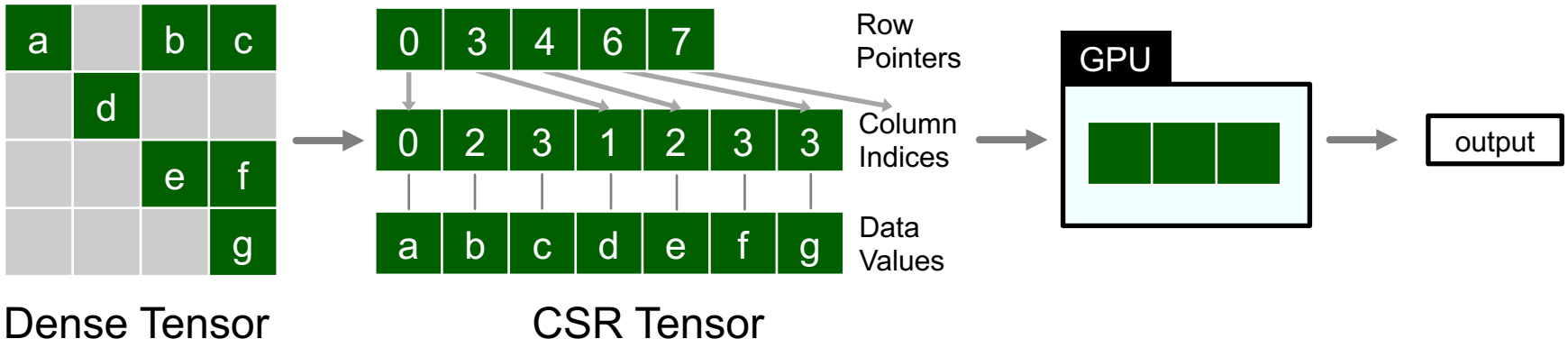
High sparsity => small predictor

Low sparsity => large predictor

Neuron-aware Operator

Traditional Sparse Operator

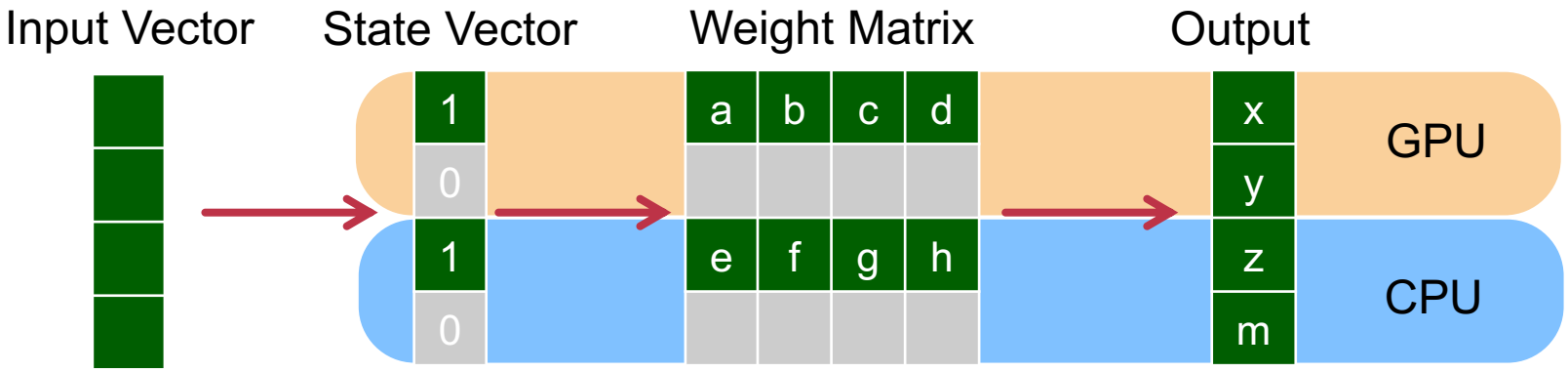
- Significant **overhead** for **converting** input to sparse format
- Lack of support for CPU/GPU **hybrid computation**



Neuron-aware Operator

The activation sparsity is rows structured

- **Neuron granular** computation without converting overhead
- Enables efficient **hybrid computation** across CPU and GPU for distinct neuron subsets following the neuron table



Implementation & Experiments Setting

Implementation:

build based on llama.cpp

4200 LoC: C++ & CUDA & Python

Modify the loading、computing process and operator

Evaluation Setting:

PC-High: Intel i9-13900K (5.4GHz, 8 cores), 192GB Host Memory, NVIDIA RTX 4090 (24GB)

PC-Low: Intel i7-12700K (4.9GHz, 8 cores), 64GB Host Memory, NVIDIA RTX 2080Ti (11GB)

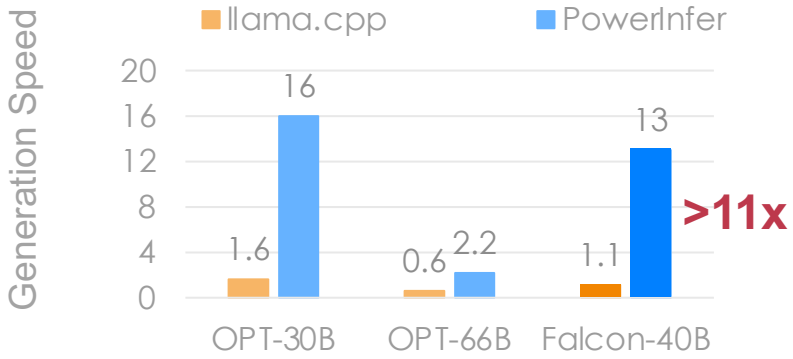
Models: OPT(7B-175B), ReLUFalcon-40B, LLaMA2(13B-70B), Bamboo-7B

Workloads: Chatbot arena prompt, ChatGPT prompts, Alpaca

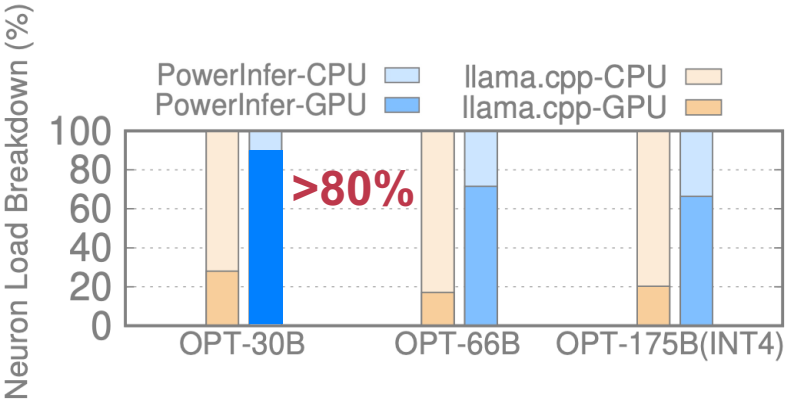
Baseline: llama.cpp, SpecInfer

Metrics: Generation Speed (tokens/s)

End-to-end Performance

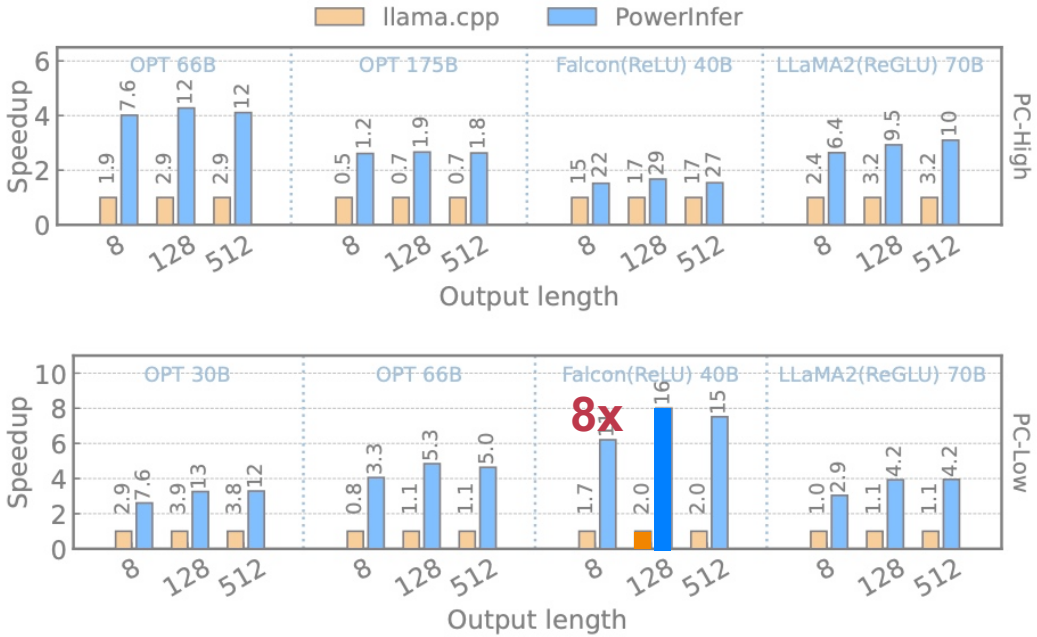


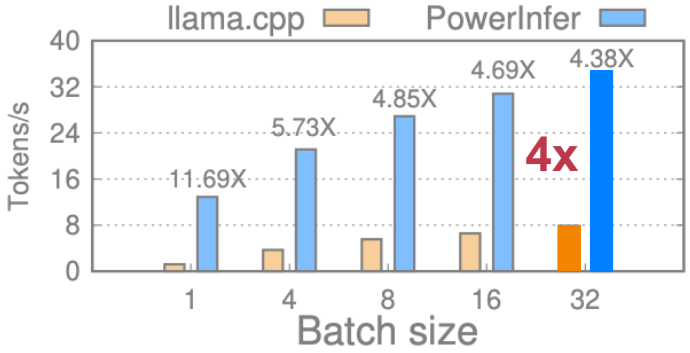
Compared to llama.cpp, PowerInfer achieves an average speedup of 7.23x and reaches up to 11.69x faster performance.



PowerInfer makes over 80% computation happens on GPU, significantly mitigating the locality mismatch problem

End-to-end Performance for Quantization and Batch





4x throughput

Speedup: **average 2.89x / Up to 8.00x**

Accuracy

	PIQA	Winogrande	Arc-Challenge	MMLU	GSM8K	Average
OPT-7B	75.78%	65.19%	30.63%	24.95%	1.90%	39.69%
OPT-7B-PowerInfer	75.67%	65.51%	30.63%	24.73%	1.82%	39.67%
OPT-13B	76.01%	64.96%	32.94%	25.02%	2.12%	40.21%
OPT-13B-PowerInfer	76.28%	65.98%	33.19%	24.76%	2.20%	40.48%
LLaMA(ReGLU)-13B	76.44%	70.09%	36.52%	50.21%	25.40%	51.73%
LLaMA(ReGLU)-13B-PowerInfer	74.06%	69.93%	36.60%	49.47%	23.90%	50.79%
Falcon-40B	81.23%	75.45%	50.68%	51.78%	21.99%	56.23%
Falcon-40B-PowerInfer	81.01%	75.92%	50.68%	51.68%	20.45%	55.95%
LLaMA(ReGLU))-70B	82.01%	75.93%	52.39%	62.30%	62.30%	66.99%
LLAMA(ReGLU)-70B-PowerInfer	82.05%	75.53%	51.45%	61.90%	61.90%	66.57%

PowerInfer successfully maintains models' accuracy

LLM v.s. SLM

PowerInfer keeps SLM's speed, but get LLM's accuracy

Table 9. Performance comparison between SLM (Qwen-1.5-4B) and PowerInfer (Bamboo-7B).

Model	TBT(ms)	Average(%)	MMLU(%)	GSM8K(%)	ARC-C(%)
Qwen1.5-4B	10.83	52.23	55.26	53.9	47.53
Bamboo-7B-PowerInfer	11.85	65.65	62.26	70.54	64.16
Bamboo-7B-dense	18.54	65.49	62.46	70.28	63.74

Similar **speed** with 4B

Similar **performance** with 7B

More Details in Our Paper

- Long Sequences
- Performance breakdown
- Comparison with A100
- Sensitvity Study to prompts
- Perdictors' Overhead
- Operators' evaluation
- Others...

Table 4. End-to-end latency comparison for different models with 1.5K input length and 256 output length on PC-High.

LLMs	llama.cpp (ms)	PowerInfer (ms)	Speedup
LLaMA(ReGLU)-13B-FP16	49.91	14.38	3.47×
Falcon(ReLU)-40B-FP16	321.63	56.48	5.69×
LLaMA(ReGLU)-70B-FP16	92.76	37.17	2.50×

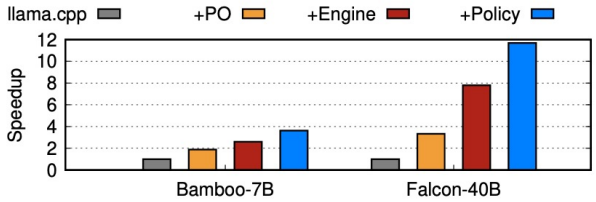
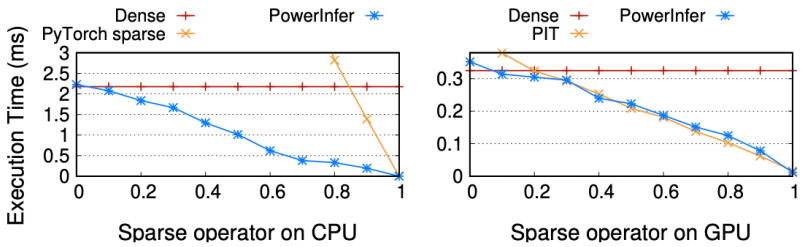


Figure 15. *Performance breakdown for each component of PowerInfer. The Falcon-40B is running on PC-High and Bamboo-7B is running on PC-Low.*



Conclusion

PowerInfer: Explore new possibilities for deploying large language models on personal computers

Mechanism: Locality-aware CPU-GPU hybrid computation

Significantly improves the inference speed: Up to 11.69x speedup!

GitHub https://github.com/SJTU-IPADS/PowerInfer

Hugging Face

Thanks!