AssyLLM: Efficient Federated Fine-tuning of LLMs via Assembling Pre-trained Blocks

Shichen Zhan, Li Li*, and Chengzhong Xu

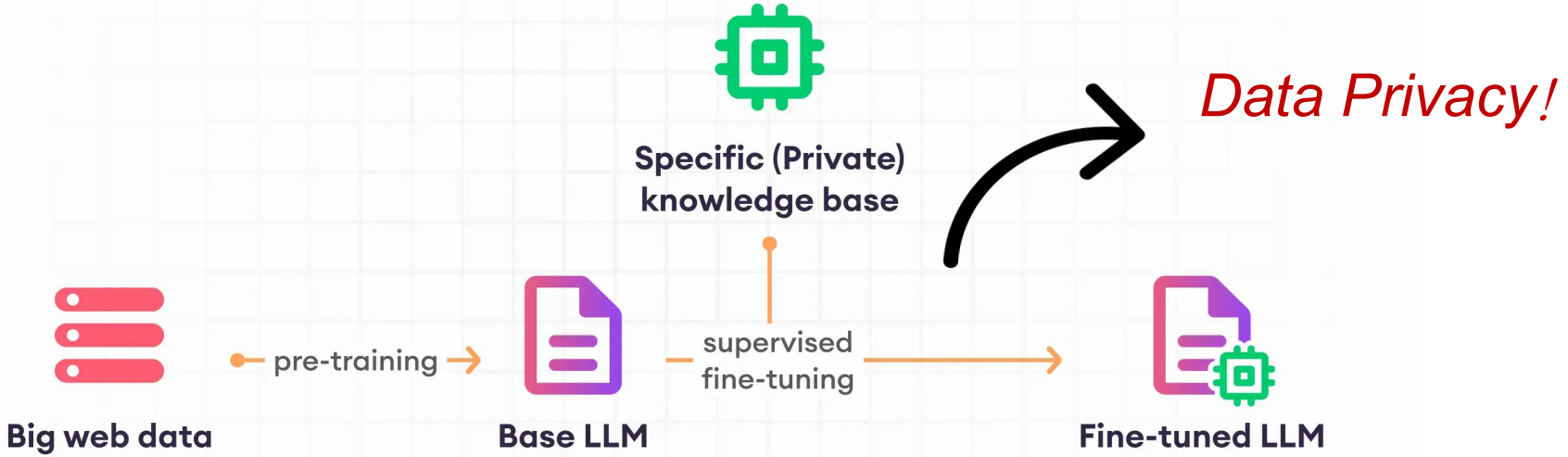
State Key Laboratory of Internet of Things for Smart City, University of Macau

澳門大學
UNIVERSIDADE DE MACAU
UNIVERSITY OF MACAU

SKL-IOTSC
智慧城市物聯網國家重點實驗室(澳門大學)
Laboratório de Referência do Estado de Internet das Coisas para a Cidade Inteligente (Universidade de Macau)
State Key Laboratory of Internet of Things for Smart City (University of Macau)

Data Scarcity and Privacy in Edge Fine-tuning

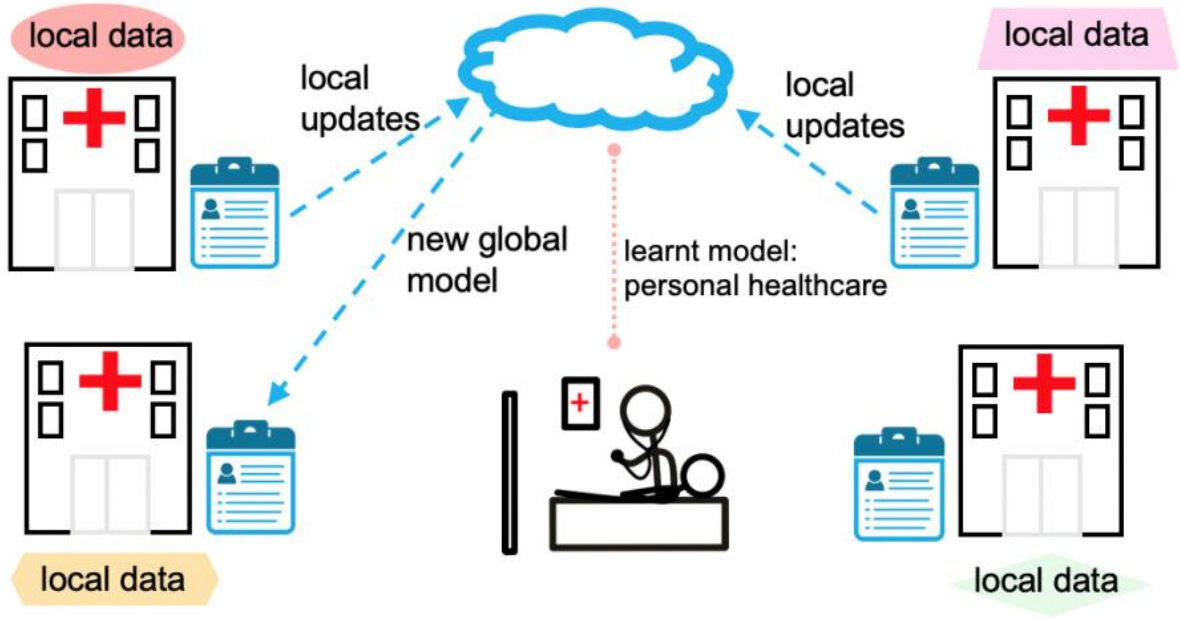
- Fine-tuning LLMs improves downstream performance on edge.
  - Single-device data is often insufficient.
  - Centralizing data from multiple devices poses serious privacy risks.



Data Privacy!

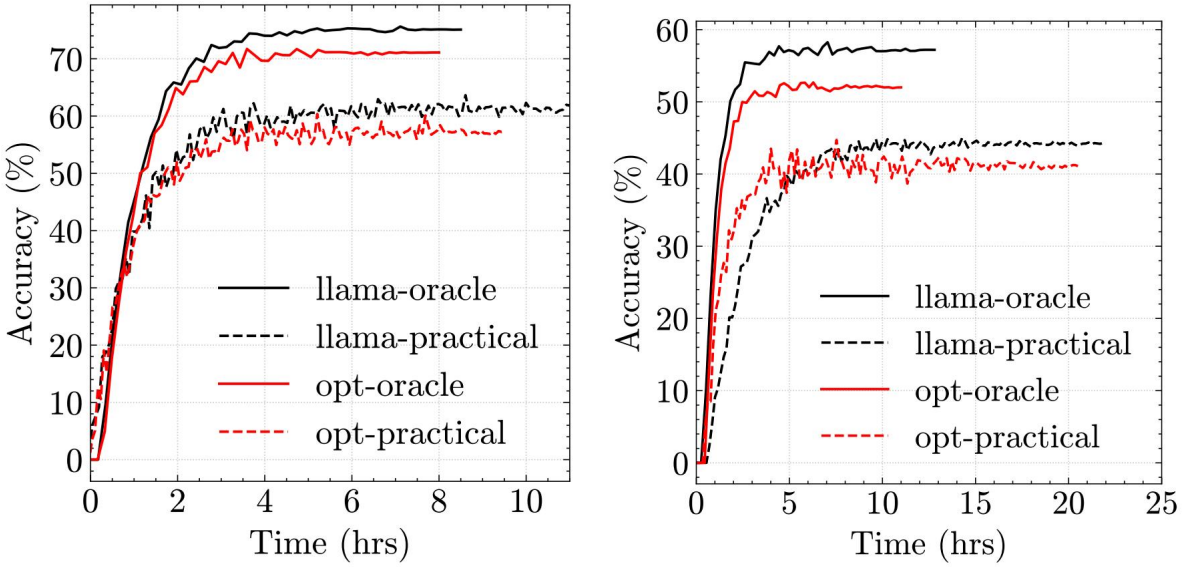
On-device FedLLM

- Federated Learning (FL) is a decentralized machine learning approach where models are trained across multiple edge devices or clients without sharing raw data.
  - Data remains on the device — only model updates are shared.
  - FL is widely applied to LLM fine-tuning (FedLLM).



Memory Wall: A Barrier to Edge Participation

- High memory overhead in fine-tuning.
  - e.g., fine-tuning Llama-7B with batch size=16 needs >40GB memory.
- Edge resource limitations.
  - 4~16GB RAM for most edge devices.
  - Many clients are excluded (90% for llama and 75% for opt), leading to reduced data diversity and degraded performance.

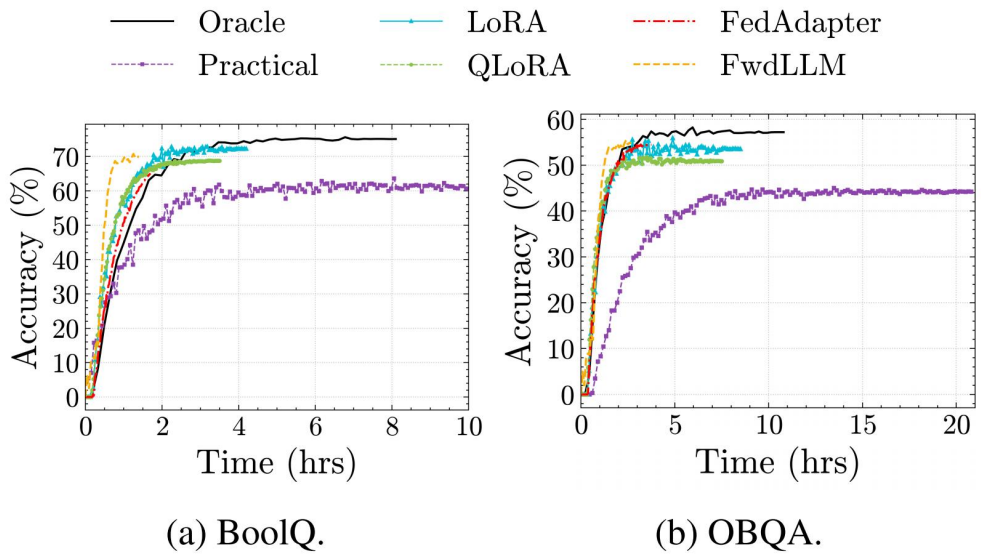


(a) BoolQ. (b) OBQA.

Figure 1: The memory wall in FedLLM; llama denotes Llama-7B, opt denotes OPT-6.7B. Datasets: BoolQ and OBQA. Participants settings: 10% (64GB), 15% (32GB),15% (16GB), 30% (8GB), and 30% (4GB).

Current Work Limitation

- Parameter-efficient fine-tuning (PEFT).
  - LoRA, QLoRA, Adapters.
  - *Accuracy loss with higher compression or partial fine-tuning.*
- Backpropagation(BP)-Free.
  - Zeroth-order or forward gradient.
  - *Unstable performance due to gradient estimation.*



(a) BoolQ. (b) OBQA.

Figure 2: Performance comparison for existing works. Model: Llama-7B.

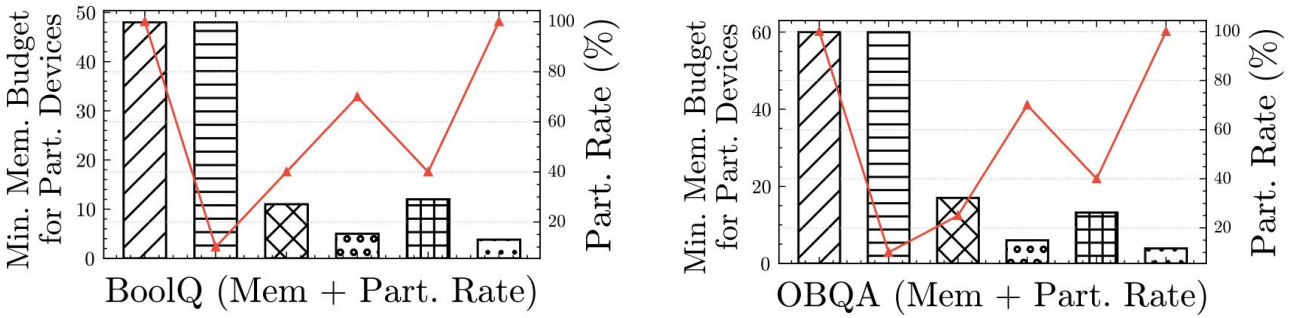
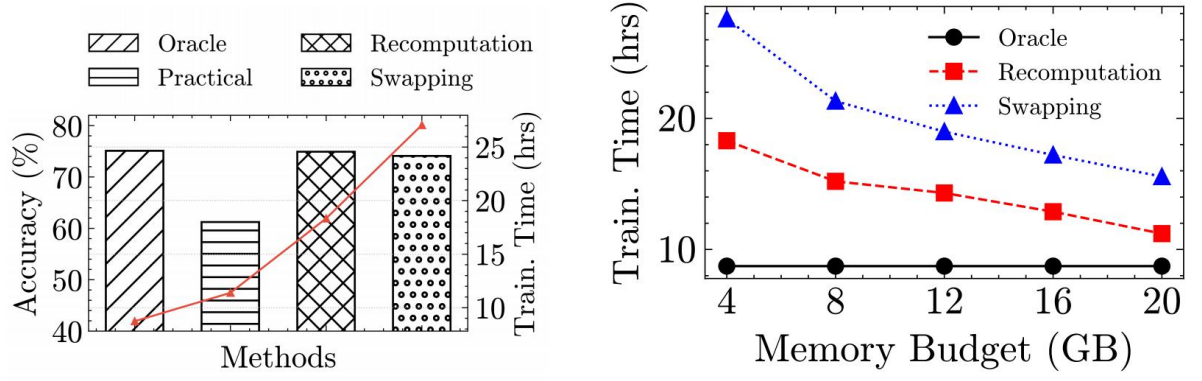


Figure 3: Minimum memory budget for participation devices and participation rate.

Current Work Limitation

- System-level solutions.
  - Gradient recomputation, swapping.
  - *Higher training time & I/O overhead.*



(a) Performance comparison with memory budget 4GB.

(b) Fine-tuning time under different memory budget.

Figure 4: System-level memory saving techniques analysis. Model: Llama-7B; Dataset: BoolQ.

A new fine-tuning paradigm guaranteeing performance and efficiency is urgently required in real-world cases.

Block Assembling: A New Fine-tuning Paradigm

Instead of single model, we utilize multiple pre-trained models for better generalization ability.
Instead of training, we generate global models by just splitting and assembling.

1. Initialization: split multiple pre-trained models into blocks.
2. Local block selection:
   - Assembling different blocks into assembled networks.
   - Performing inference computation for assembled networks on local data.
   - Based on the computation results, obtaining the compatibility scores of blocks and selecting.
3. Uploading selected block.
4. Server block aggregation and assembling.

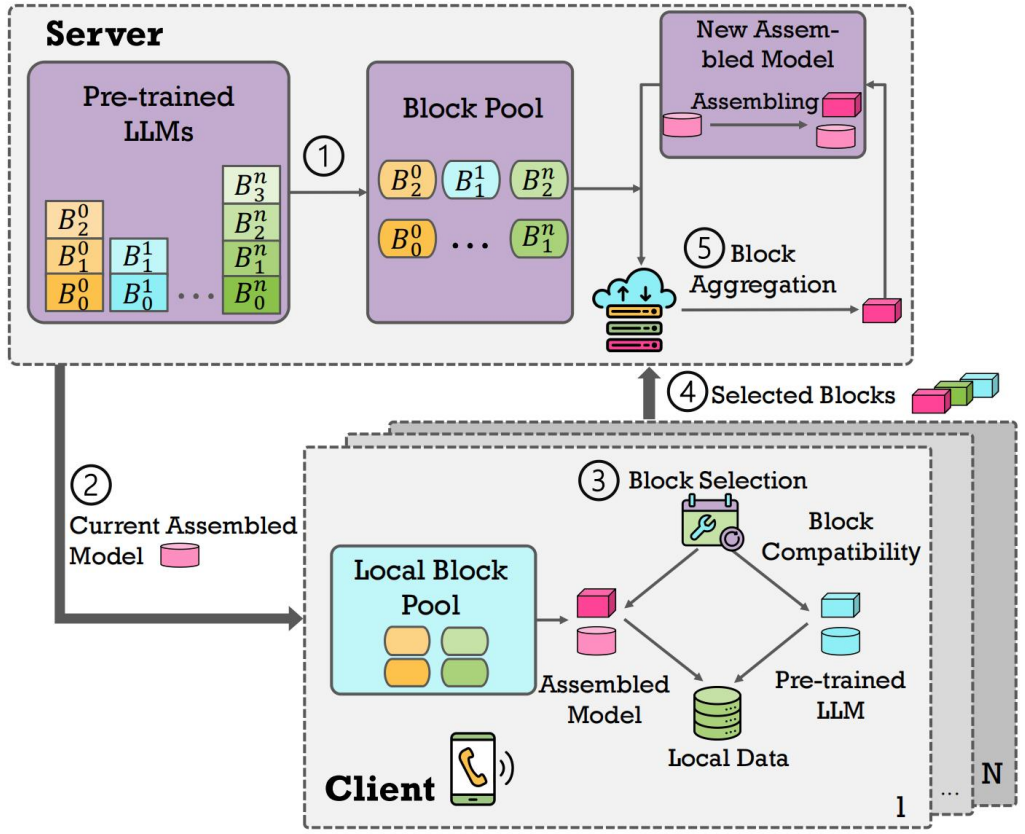


Figure 5: Workflow of assembled model generation.

Block Assembling: A New Fine-tuning Paradigm

- Key insights:
  - Pretrained models as efficient building blocks.
  - Dynamic block assembling for flexibility.
- Requiring only forward passes.
  - Activatioin-related Computation and Memory Savings.

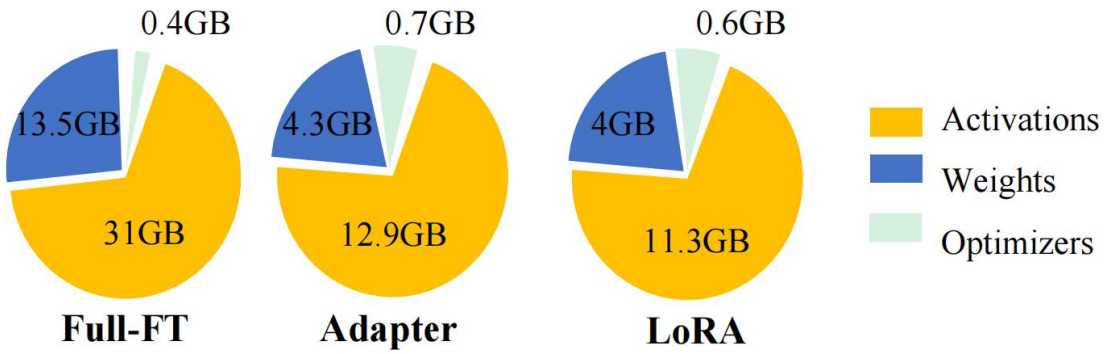


Figure 6: Memory breakdown. Model: Llama-7B. FT denotes fine-tuning.

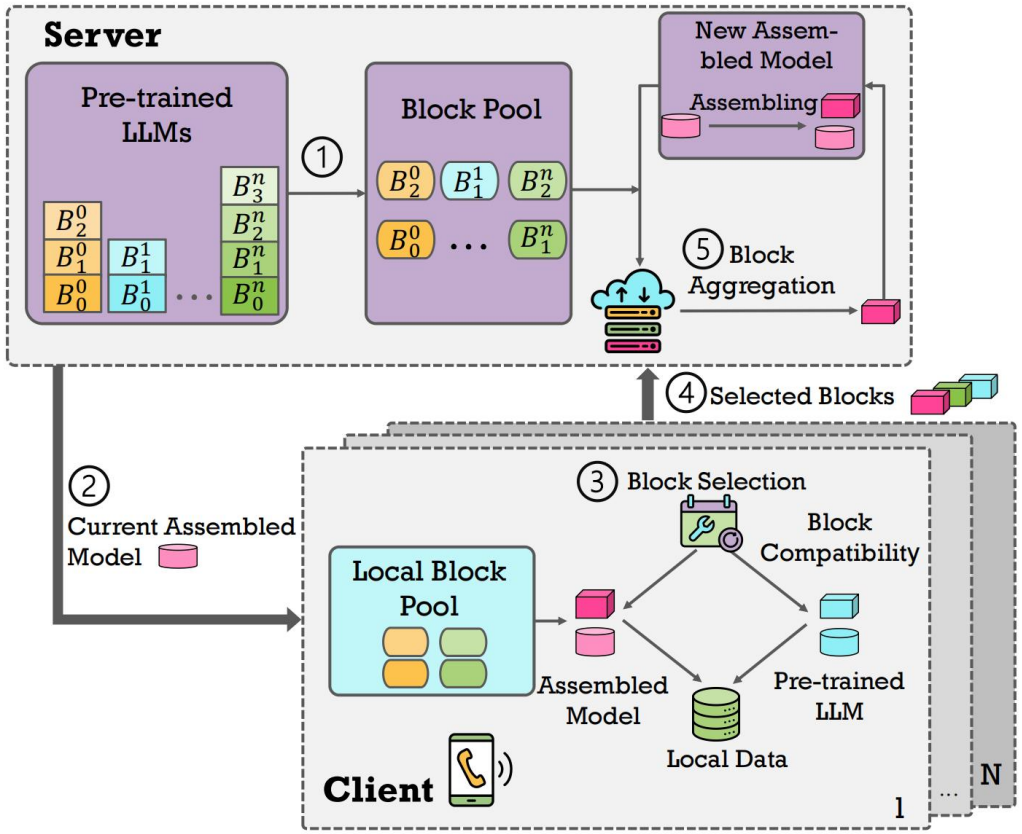


Figure 7: Workflow of assembled model generation.

Key Challenges and AssyLLM

Q1: Block Compatibility

- How to identify the most compatible blocks for assembling?

Q2: Block Integration

- How to effectively assemble blocks with structural and semantic differences?

Q3: Memory Overhead

- How to manage the large memory footprint from multiple pre-trained LLMs?

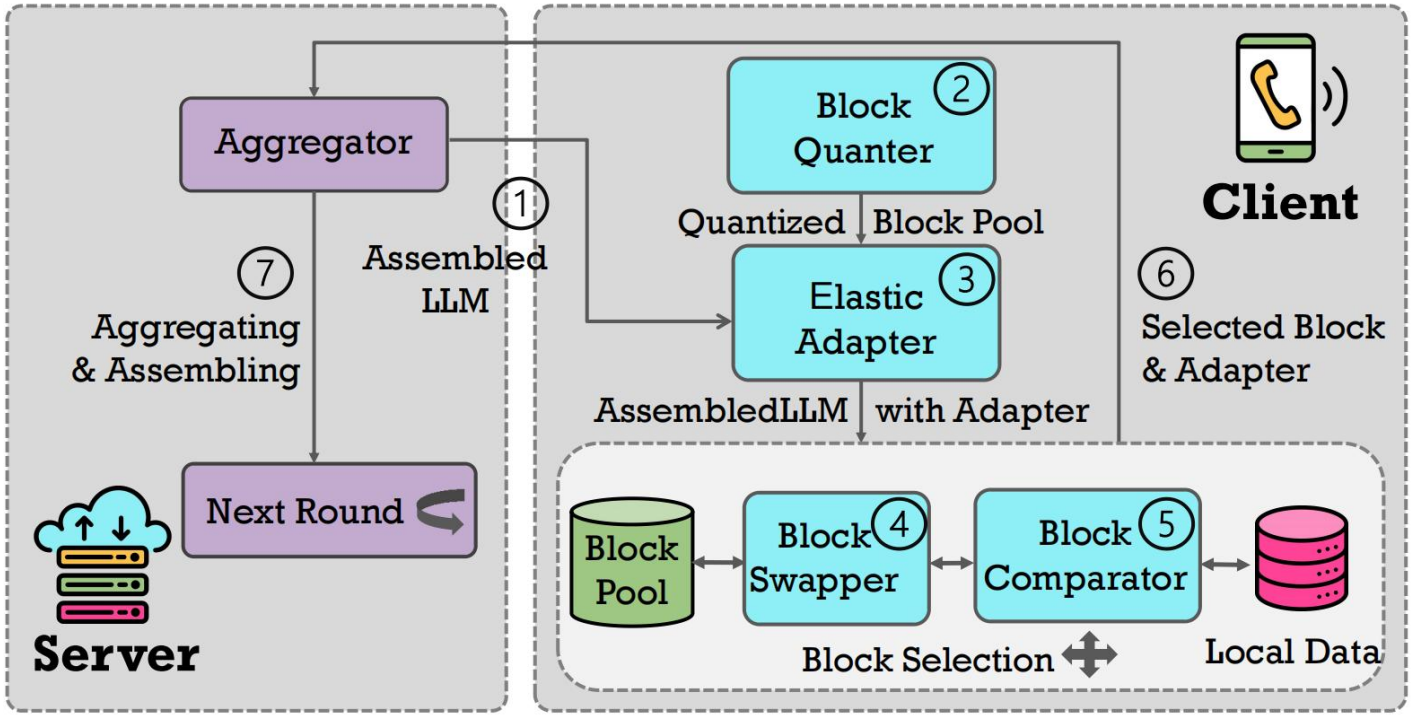
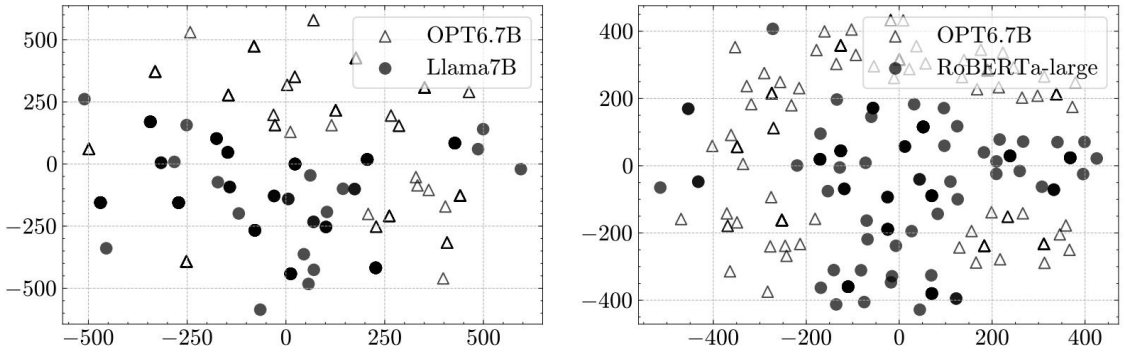


Figure 8: AssyLLM architecture.

We introduce AssyLLM, an FL system for deploying LLMs that optimizes memory and performance.

Q1: Measure the Compatibility between Blocks

- The higher the compatibility, the better the performance of the assembled model.
- Blocks show structural and feature differences.
  - Different network architectures produce significantly different feature distributions across layers.



(a) OPT-6.7B vs Llama-7B. (b) OPT6.7B vs RoBERT-large.

Figure 9 : Feature differences visualization using t-SNE.

Q1: Measure the Compatibility between Blocks

- Centered Kernel Alignment (CKA).
  - HSIC is Hilbert-Schmidt Independence Criterion.

$$CKA(K, L) = \frac{\mathbf{HSIC}(K, L)}{\sqrt{\mathbf{HSIC}(K, K)\mathbf{HSIC}(L, L)}} \qquad \mathbf{HSIC}(K, L) = ||\mathrm{cov}(X^T X, Y^T Y)||_F^2$$

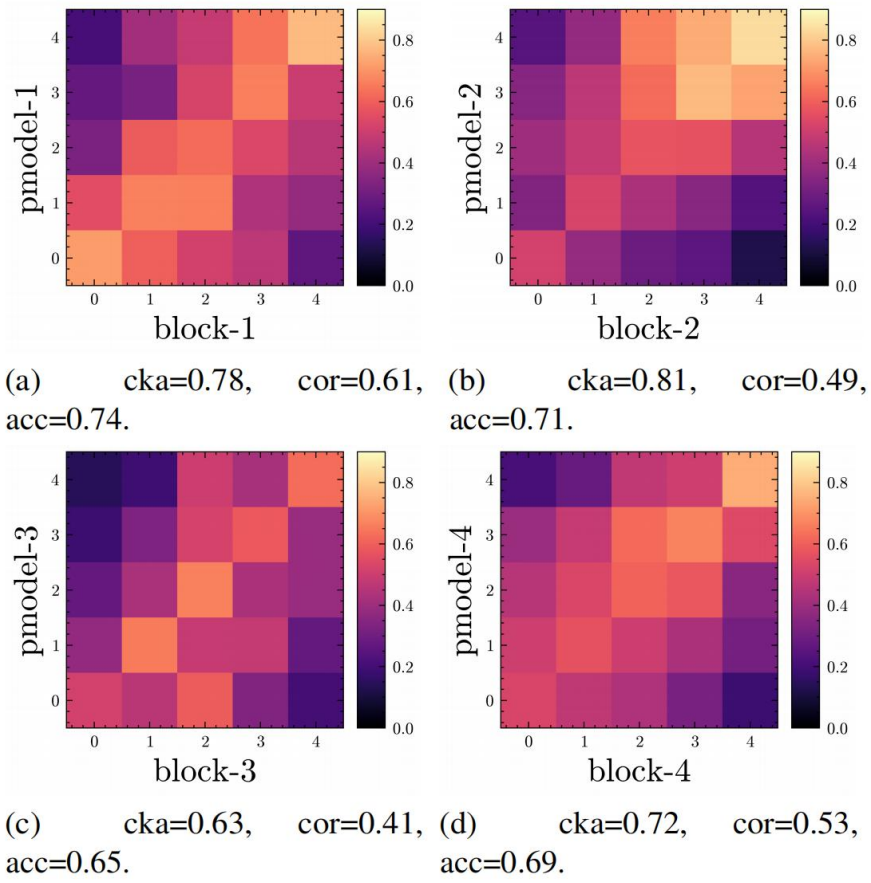
- Layer-correlation (COR).
  - Comparing activation distribution similarities between layers using Kullback-Leibler (KL) divergence.

$$D_{KL}(P \parallel Q) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right)$$

$P(i)$: the probability of activation i in the first distribution; $Q(i)$: the probability in the second distribution.

Q1: Measure the Compatibility between Blocks

- We combine CKA and COR in a weighted manner to form a more reliable metric, block compatibility, enabling more accurate block selection.



(a) cka=0.78, cor=0.61, acc=0.74. (b) cka=0.81, cor=0.49, acc=0.71.

(c) cka=0.63, cor=0.41, acc=0.65. (d) cka=0.72, cor=0.53, acc=0.69.

Figure 10 : CKA Heatmaps for four various candidate blocks; *pmodel* denotes pre-trained LLM where the block originates from; acc denotes final assembled model performance; Dataset: BoolQ.

Q2: Assemble Blocks Together

- Challenge:
  - Blocks from different LLMs exhibit dimensional misalignment, semantic inconsistency, and attention mechanism differences.
- Elastic Adapter.
  - Projects outputs to aligned dimensions.
  - Applies cross-attention to match semantics.
  - Adapts to different attention types (e.g., multi-head vs single-head).
- Lightweight fine-tuning.

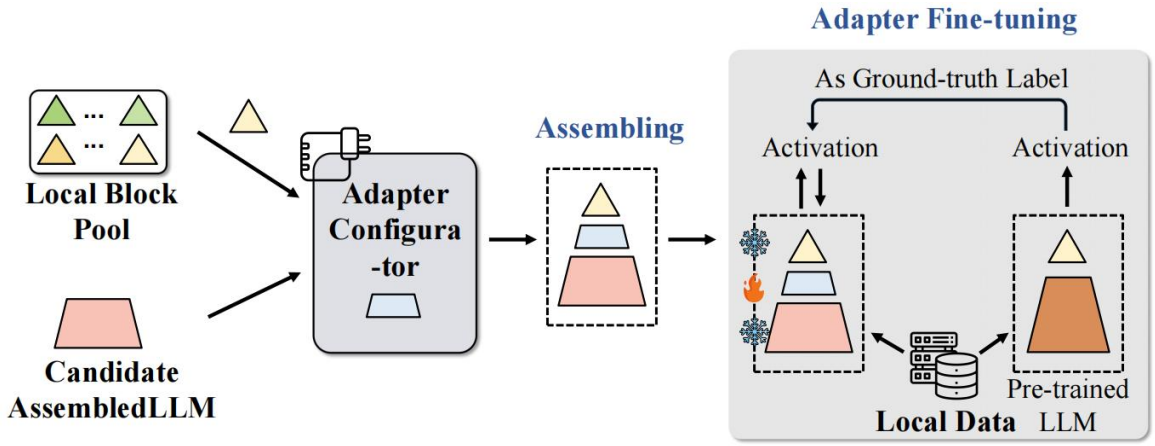


Figure 11: Elastic Adapter fine-tuning.

Q3: The Extra Memory Overhead of Block Pool

- Challenge: Block pool = multiple LLMs → high memory usage.

- Block Quanter: Efficient Mixed-Precision Quantization.
  - Analyze weight sparsity & activation sensitivity.
  - Allocate high precision to critical weights.

Table 1: Assembled model performance and memory overhead of the block pool with different quantization methods comparison. BP denotes the block pool. Dataset: BoolQ. Block pool: Llama-7B, OPT-6.7B, Vicuna-7B, BERT-base, and RoBERT-large.

Precision	**BP Overhead**	**Model Accuracy (%)**
FP16	42.2GB	75.8 ± 2.5
INT8	21.1GB	73.1 ± 4.7
INT4	11.5GB	67.3 ± 8.9

Q3: System-Level Optimization via Block Swapper

- Even with Block Quanter, the memory overhead remains substantial.
- Block Swapper.
  - Swaps blocks between memory and storage.
  - Combines LRU + block correlation for swap-out.
  - Uses pre-loading & pre-swapping to reduce I/O delays.
  - Enables large block pools on memory-constrained devices.

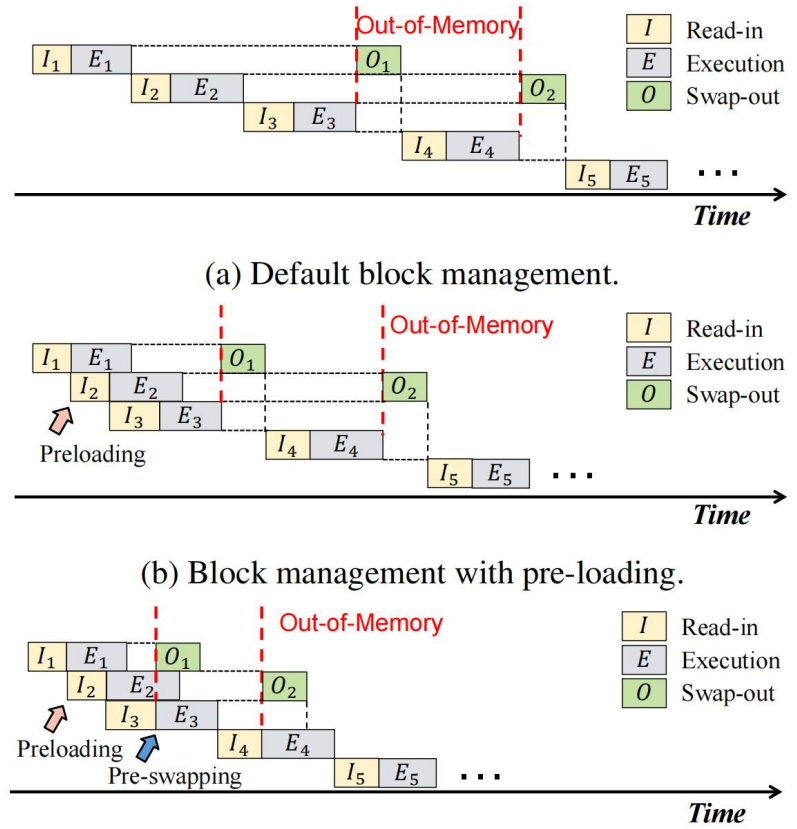


Figure 12: Different pipeline strategies for block swapping.

AssyLLM

- Block Comparator.
- Elastic Adapter.
- Block Quanter.
- Block Swapper.

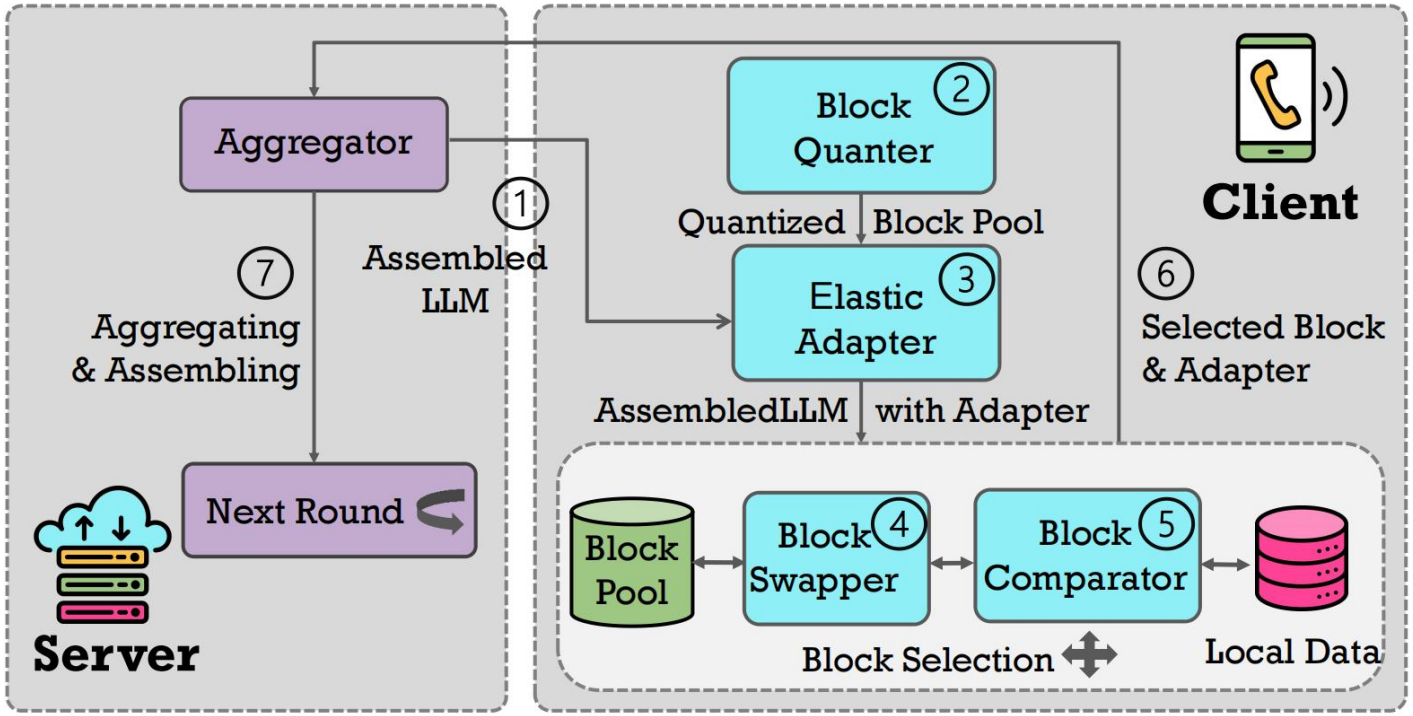


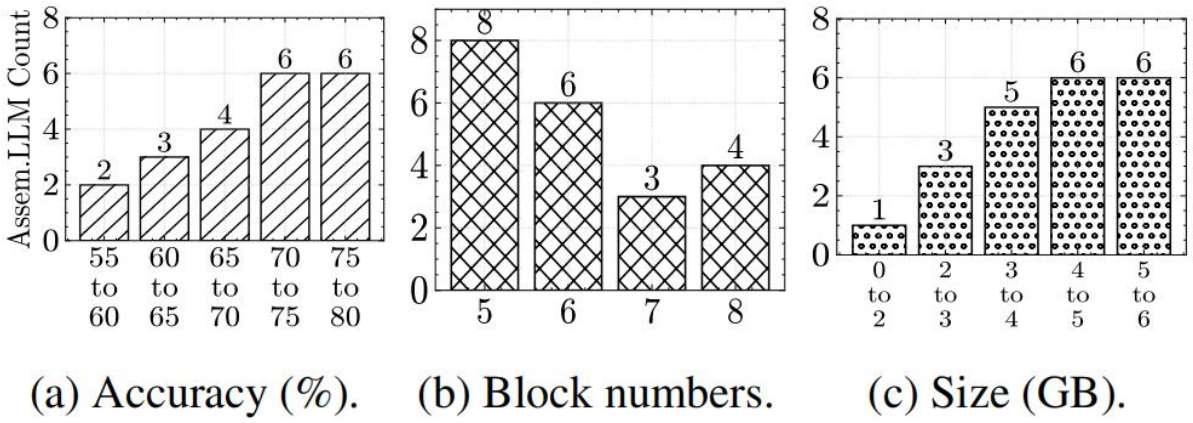
Figure 8: AssyLLM architecture.

Experimental Settings

- LLMs pool: Llama-7B, OPT-6.7B, BERT-base, Vicuna-7B, RoBERTa-large.
- Datasets: BoolQ (reading), PIQA (physical), OBQA (commonsense).
- Baselines: LoRA, QLoRA, FT-oracle/practical, FedAdapter, FwdLLM, Recomputation, Swapping.
- Platforms: Nvidia A100 (simulation) & Jetson TX2 (real devices).
- FL settings: 200 users, 20 each round.
  - Non-IID: apply Dirichlet distribution with a concentration parameter $\alpha = 1$.

Table 2: Model size (FP16).

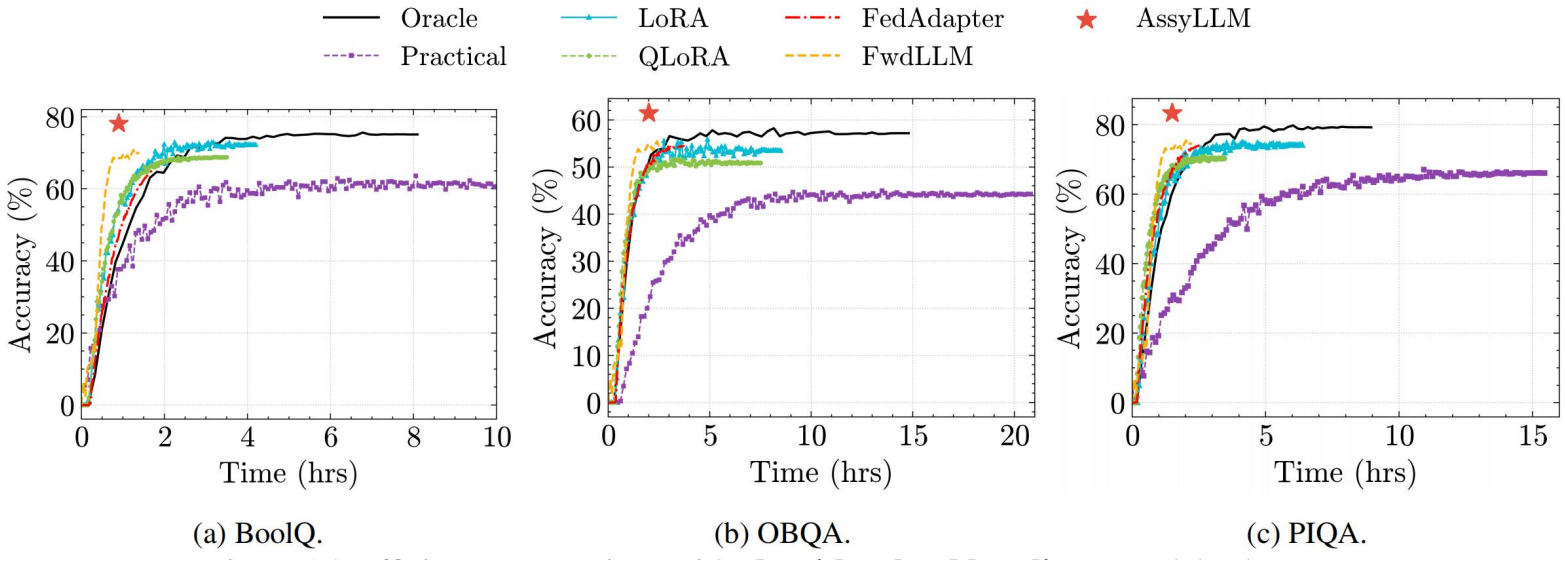
Model	Size
Llama-7B	14GB
OPT-6.7B	13.4GB
BERT-base	220MB
Vicuna-7B	14GB
RoBERT-large	710MB



(a) Accuracy (%). (b) Block numbers. (c) Size (GB).

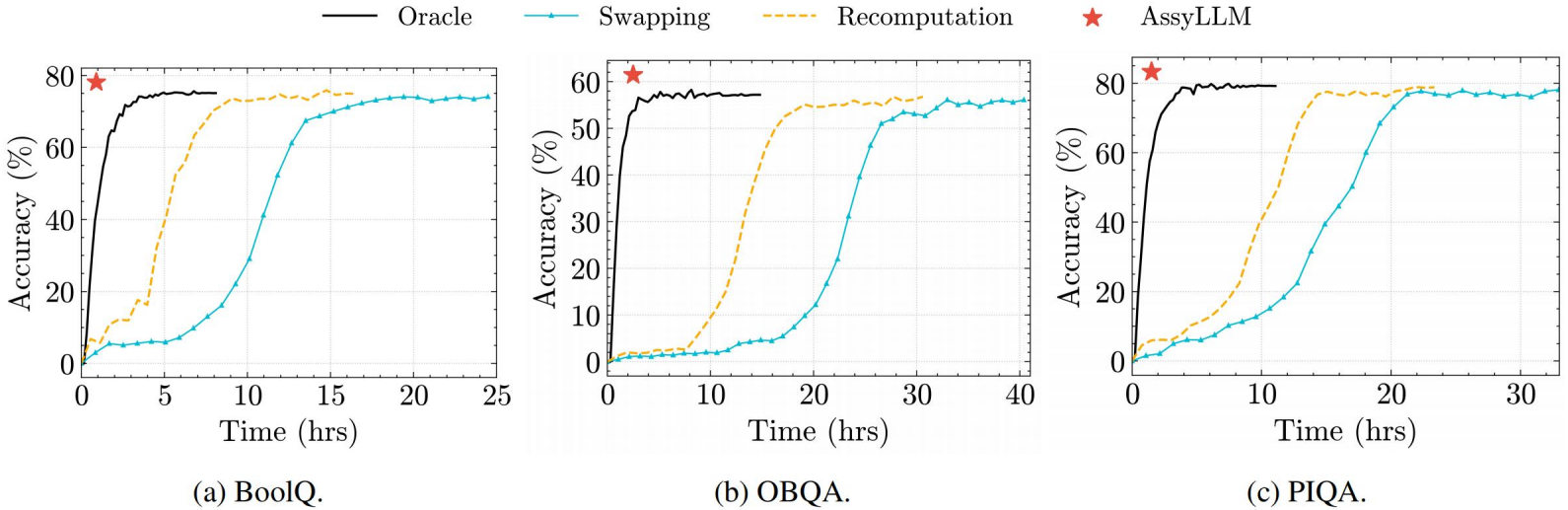
Figure 13: Statistics for generated assembled models.

Evaluation: Accuracy & Efficiency



(a) BoolQ. (b) OBQA. (c) PIQA.

Figure 14: Efficiency comparison with algorithm-level baselines. Model: Llama-7B.



(a) BoolQ. (b) OBQA. (c) PIQA.

Figure 15: Efficiency comparison with system-level baselines. Model: Llama-7B.

AssyLLM achieves higher accuracy and faster training.

Memory, Energy & Communication

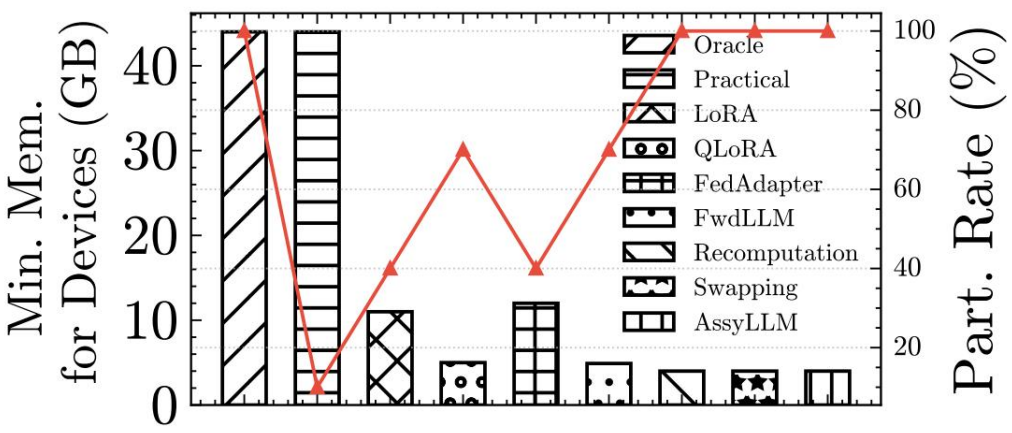
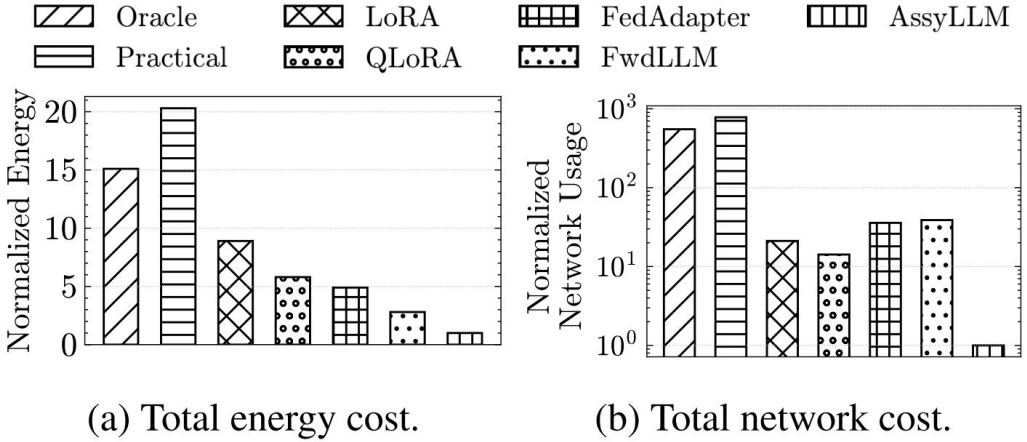


Figure 16: Peak memory footprint and client participation rate comparison on BoolQ. Min. Mem. for Devices denotes the minimum memory required for the participant to apply the corresponding methods.



(a) Total energy cost. (b) Total network cost.

Figure 17: Resource cost of AssyLLM and baselines. Dataset:BoolQ.

AssyLLM is lighter, greener, and more efficient.

Discussion

- Criteria for selecting pre-trained models.
  - Choosing diverse pre-trained models improves generalization but requires careful trade-offs.
- Why block-level assembly works better?
  - Block-level assembly enables broader participation and leverages redundancy for robust performance.

More possibilities to explore ahead!

Thanks for your attention!

Questions are welcome

Shichen Zhan, *Li Li, and Chengzhong Xu*
Contact: shichenzhan84@gmail.com