LLMPrism: Black-box Performance Diagnosis for Production LLM Training Platforms

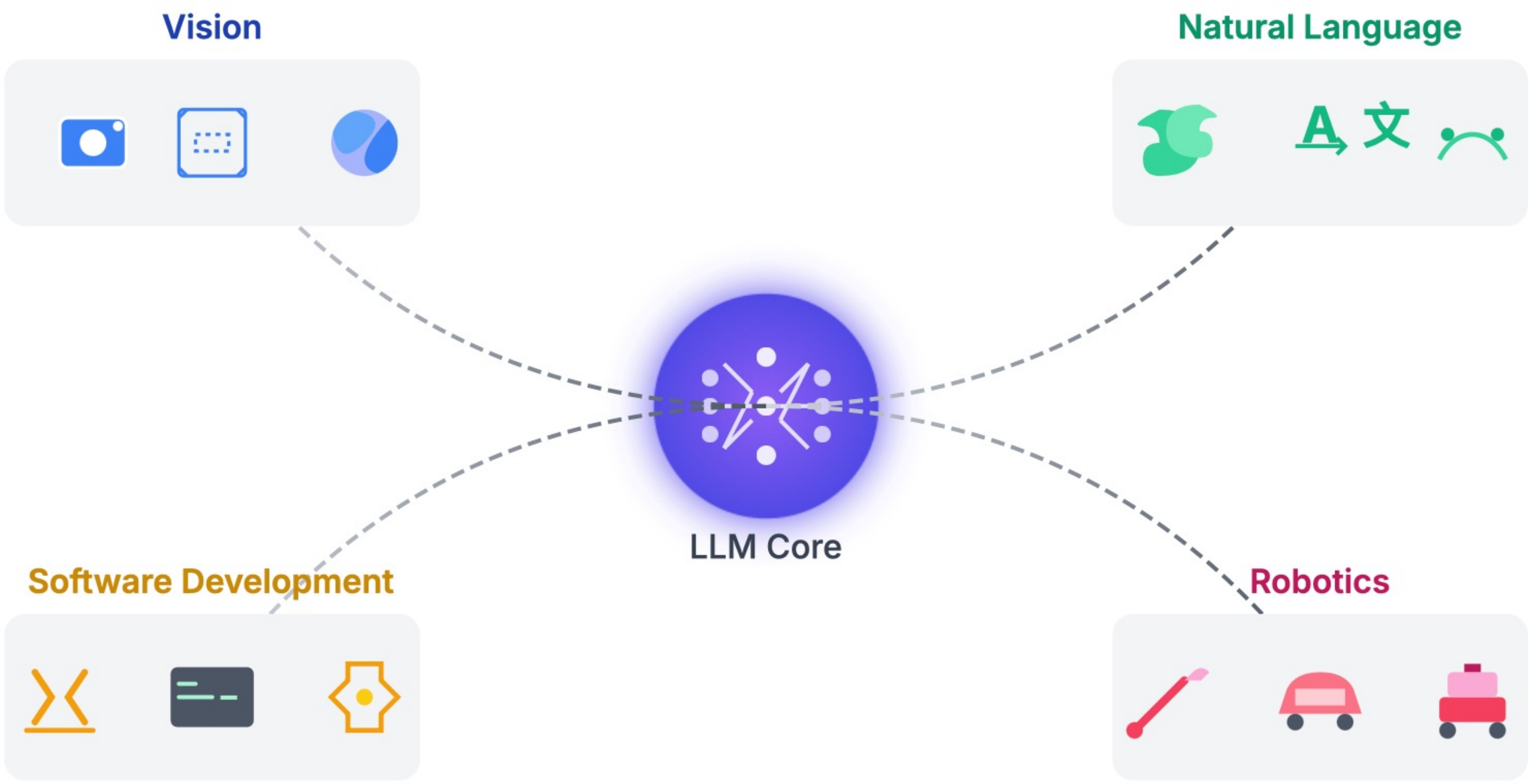
Zhihan Jiang[1], Rui Ren[2], Guangba Yu[1], Yulun Wu[1], Wenwei Gu[1], Yichen Li[1] , Yujie Huang[1]
Cong Feng[2], Yongqiang Yang[2], Zengyin Yang[2] and Michael R. Lyu[1]

[1]*The Chinese University of Hong Kong,*
[2]*Huawei Cloud*

ARISE
Automated Reliable Intelligent Software Engineering
HUAWEI

Background

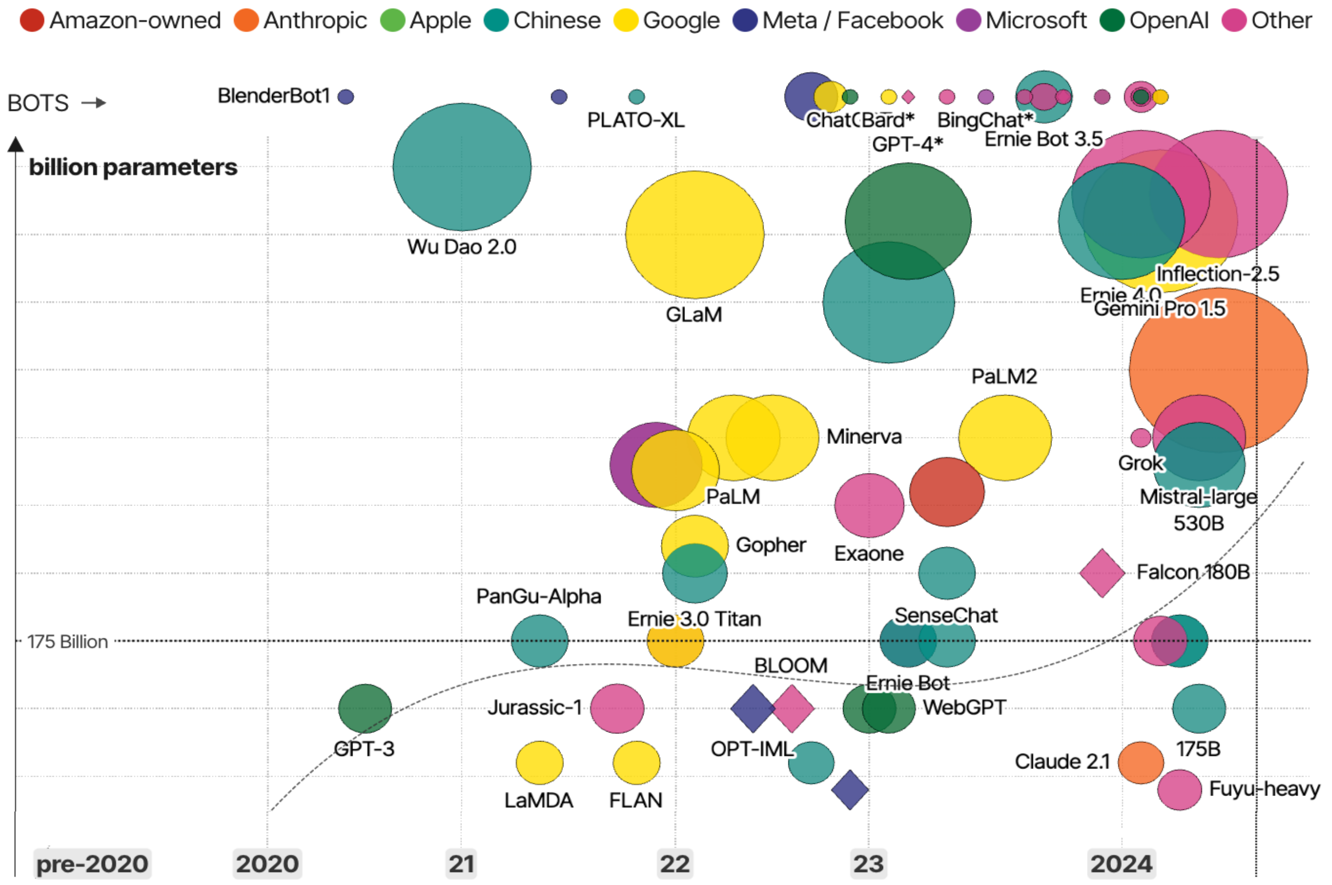
Large Language Models (LLMs) are everywhere for everyone



Scaling Law in LLMs

LLMs have grown rapidly in scale, e.g., >100B parameters

- Mistral Large 123B
- GPT-3 175B
- LLaMA3.1 405B
- DeepSeek-R1 671B

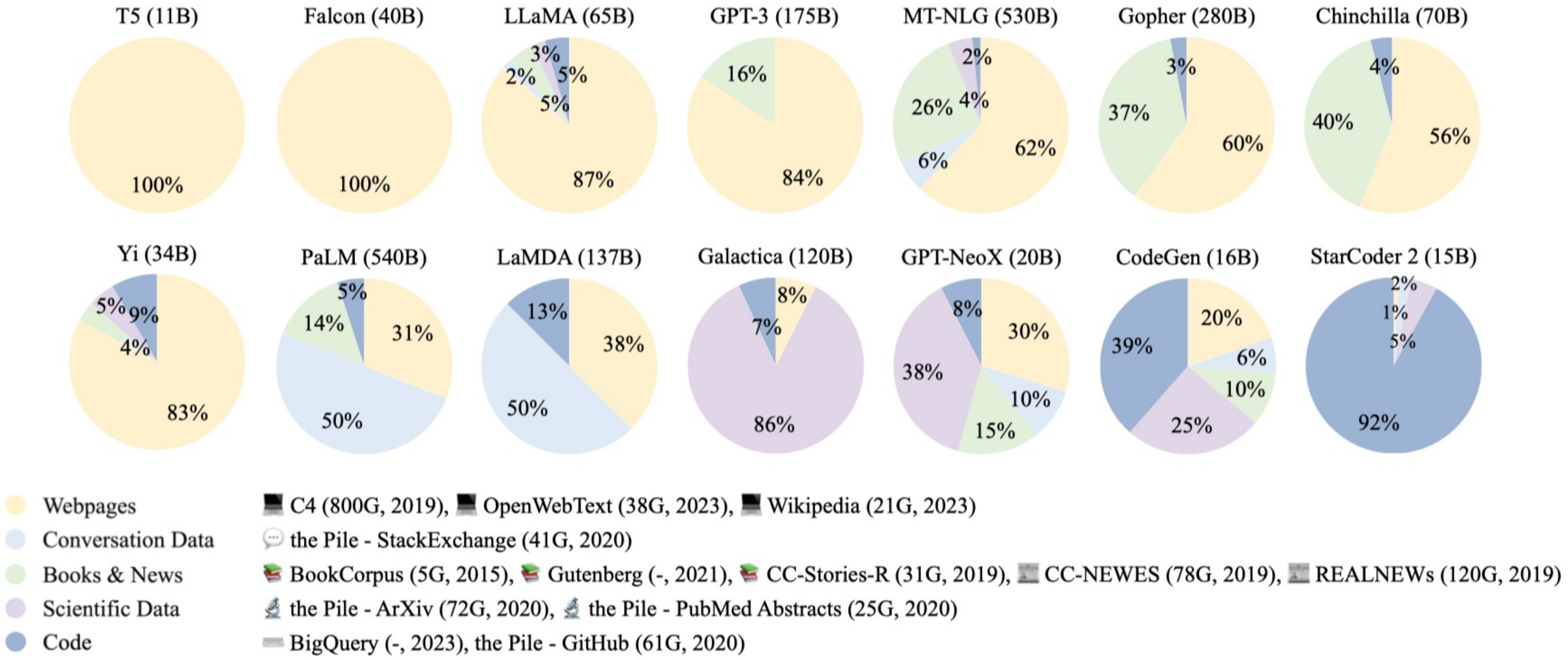


https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgpt/

Scaling Law in LLMs

The training datasets are also increasingly diverse and massive

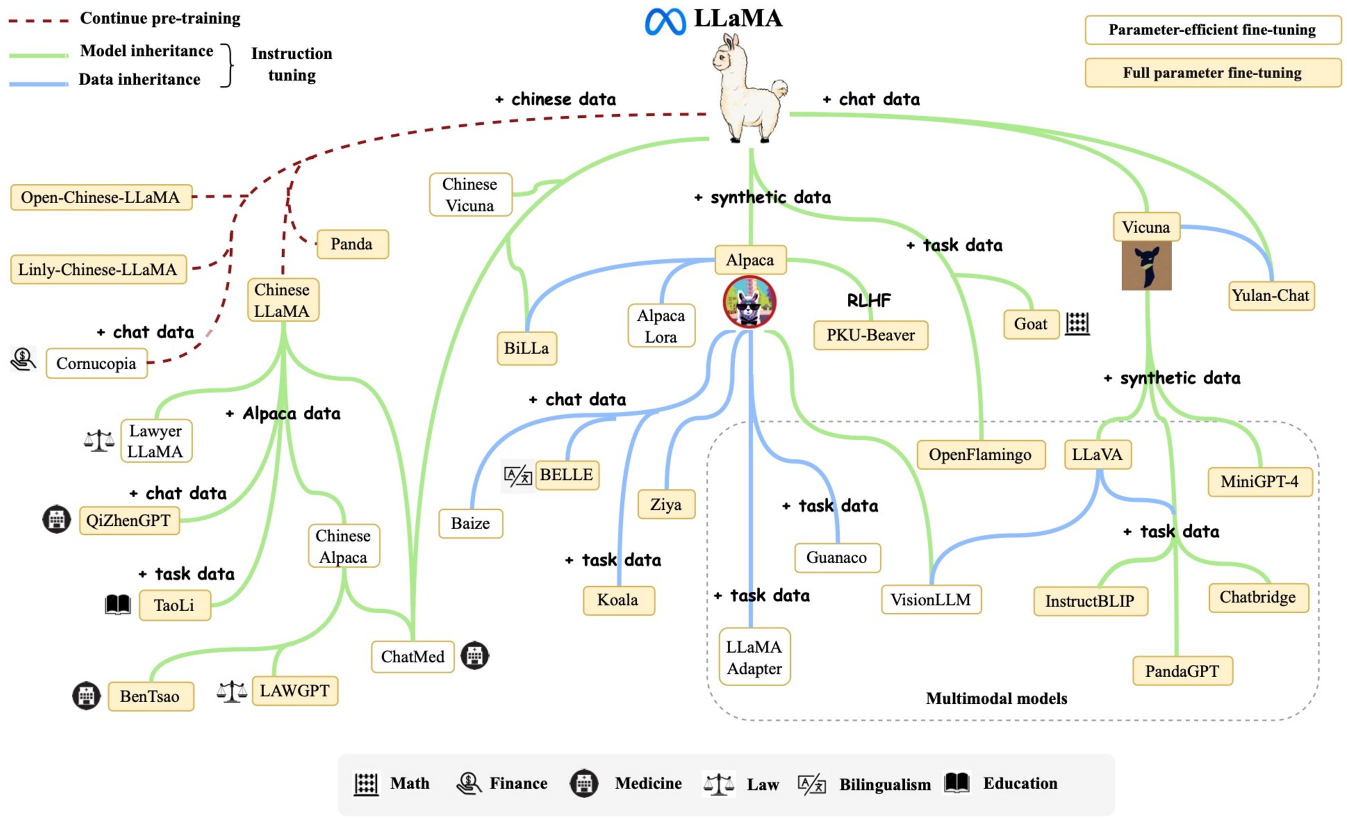
- **LAION-2B-en** datasets: over 360B tokens
- **RedPajama** datasets: over 30T tokens



A survey of large language models[J]. arXiv preprint arXiv:2303.18223, 2023.

LLM Training and Tuning

Fine-tuning LLMs on domain-specific datasets is essential for achieving SOTA performance in specialized fields



A survey of large language models[J]. arXiv preprint arXiv:2303.18223, 2023.

LLM Training and Tuning

Training or tuning LLMs require substantial computing resources

Full-training

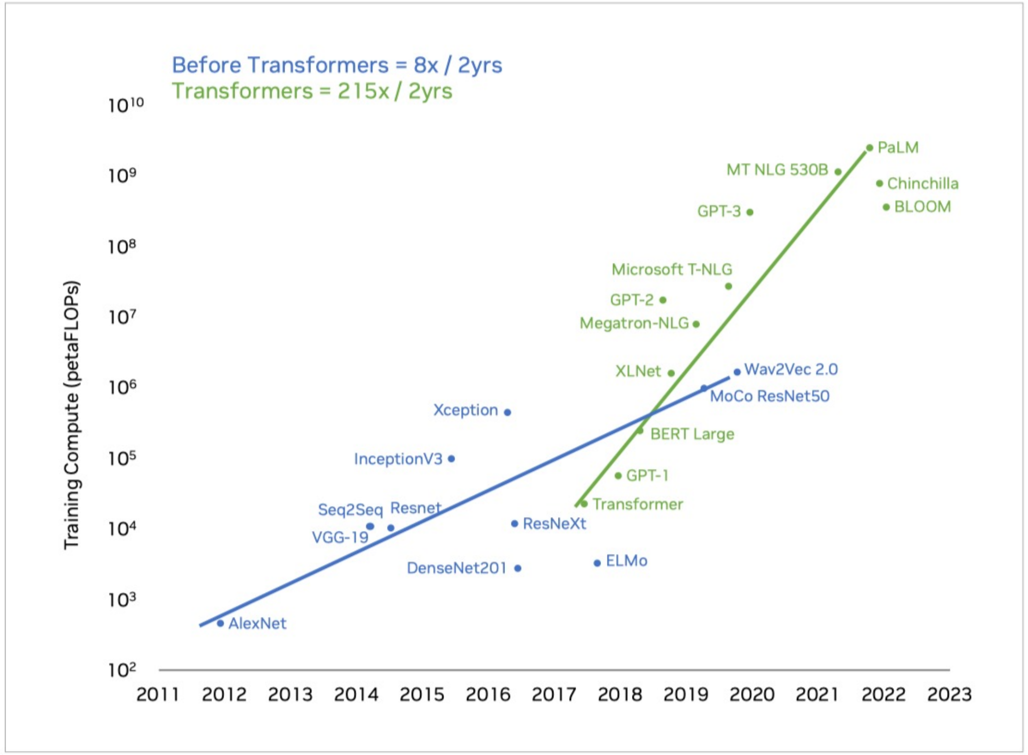
BLOOM-176B: 384 A100 GPUs for 3.5 months

LLaMA 3.1 405B: 16,384 H100 GPUs for 54 days

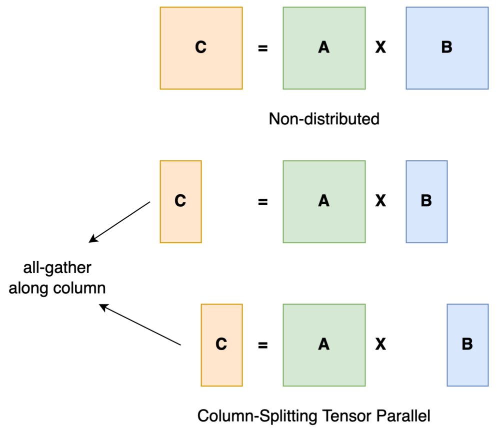
DeepSeek-V3-671B: 2048 H800 GPUs for 56+ days (2.788M GPU Hours)

Post-training (Fine-tuning)

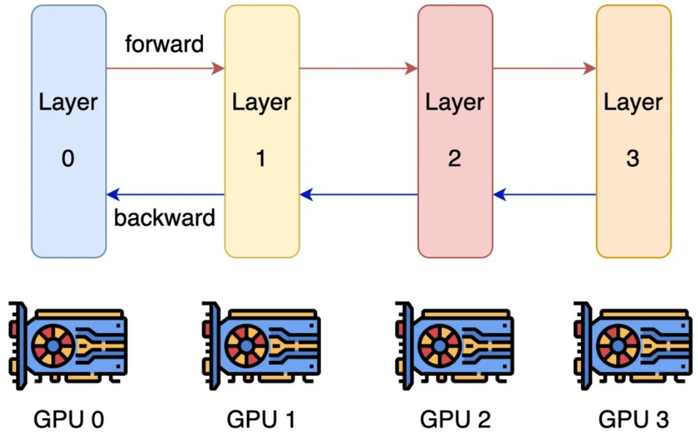
DeepSeek-V3-671B: 64 H800 GPUs for 3+ days (5K GPU Hours)



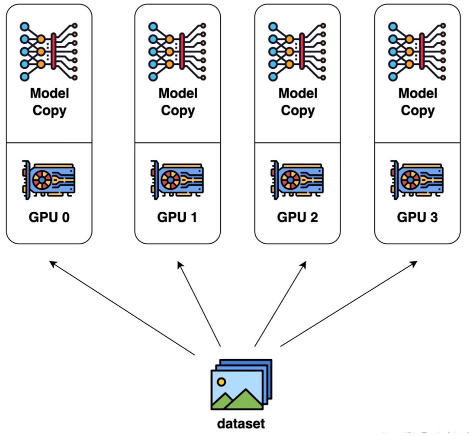
LLM Training Parallelism



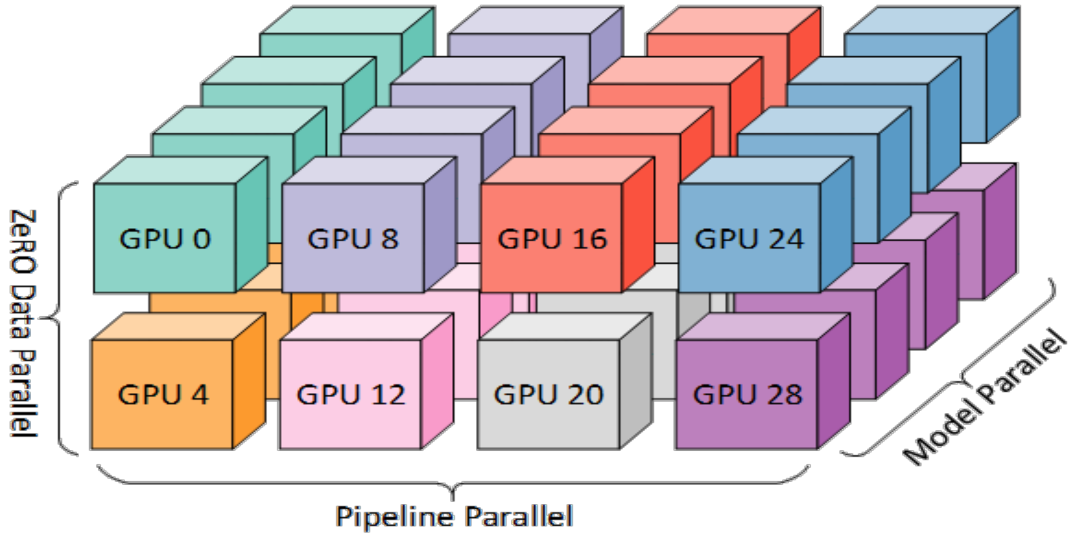
Tensor Parallelism



Pipeline Parallelism



Data Parallelism



3D Parallelism

LLM Training Platforms

LLM training platforms: simplify the LLM training and tuning process

Amazon SageMaker

Google Vertex AI

Databricks

Huawei ModelArts

LLM Training Job Slowdown

Performance degration is frequent due to:

- The scale and complexity of LLM training jobs
- Strong synchronization requirements at training-step boundaries

Potential Reasons

network congestion

Resource contention

garbage collection

thermal throttling

LLM Training Profiling

LLM training profiling and performance monitoring is important!

Existing LLM training profiler:

- PyTorch Profiler
- xpu-timer
- Meta HTA
- Meta Strobelight

Limitation:

- Intrusiveness and Compatibility
- Performance Overhead
- Visibility Constraints

LLM Training Network Monitoring

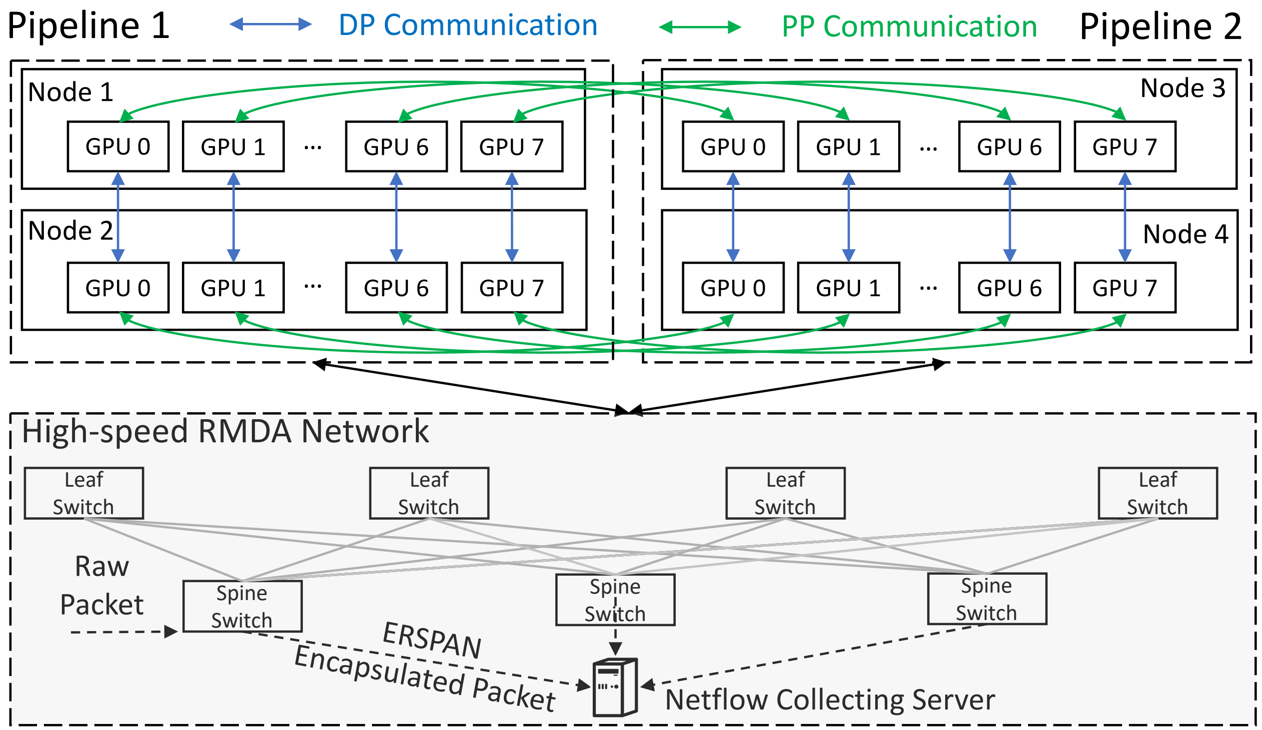
What can we leverage to design a non-intrusive profiling solution with minimal performance impact?

Network traffic monitoring

- non-intrusive and independent approach
- minimal interference with the network and associated workloads
- e.g., Encapsulated Remote Switch Port Analyzer (ERSPAN)

Data format

flow start time, source address, destination address, involved switches, flow size, and flow durations.



LLM Training Network Monitoring

Three observed distinct patterns inherent in LLM training procedure:

Spatial Communication Patterns

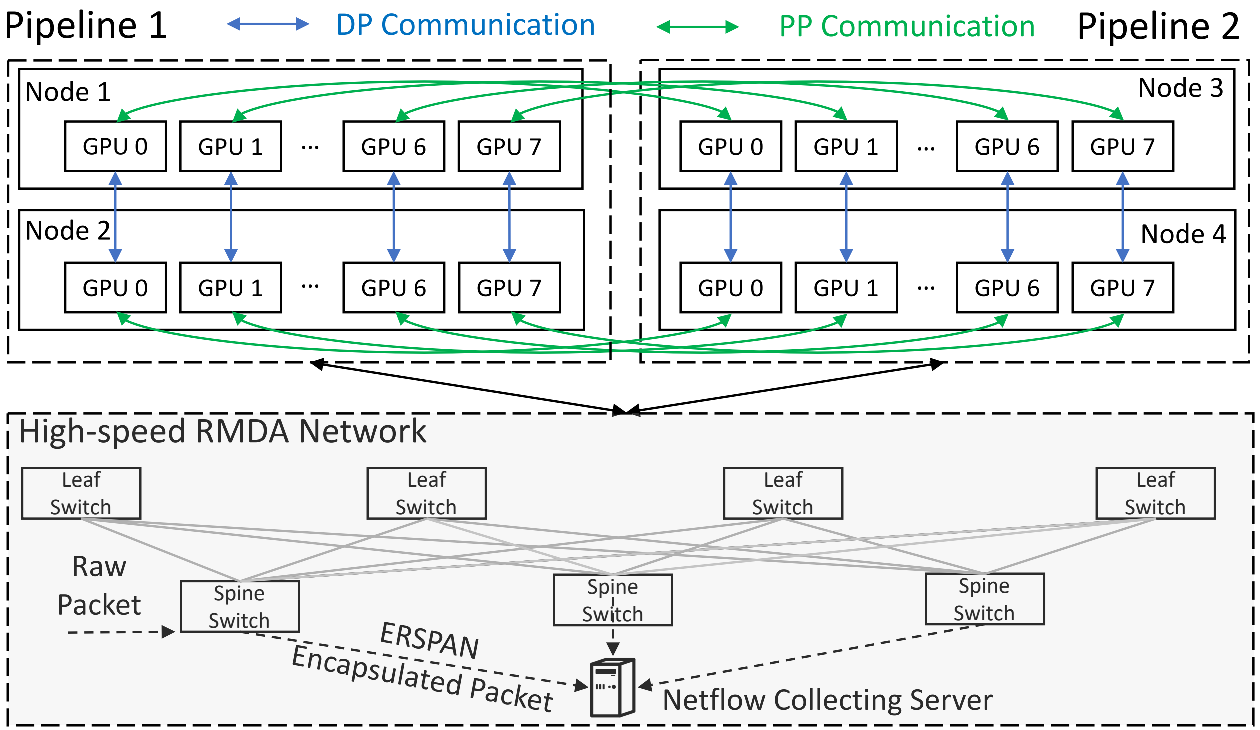
- Communications during LLM training exhibit spatial stability, with interactions strictly confined to GPUs within the same DP or PP groups.

Temporal Communication Patterns

- Communications traffics demonstrate clear temporal periodicity, as the training steps keep cycling and the workload remains stable.

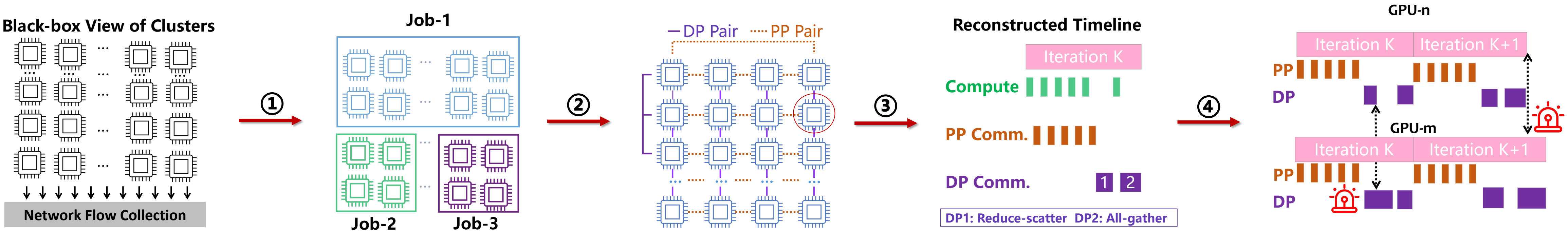
Distinctive Parallelism Communication

- Different communication modes (DP and PP) display uniquely identifiable characteristics, enabling differentiation between them.



Method: LLMPrism

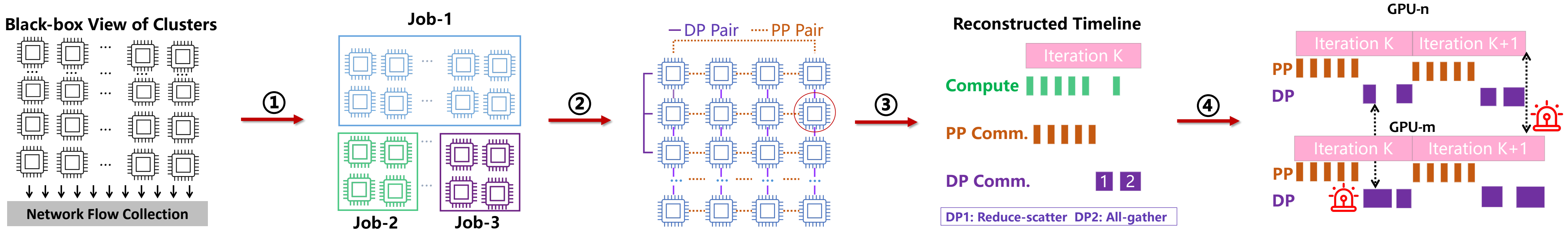
Goal: non-intrusively monitor and diagnose the performance of LLM training jobs



① **LLM Training Job Recognition** ② **Parallelism Strategies Identification** ③ **Training Timeline Reconstruction** ④ **Performance Diagnosis**

- The first work using underlying network data to reconstruct detailed LLM training timelines
- Transforming the original black-box view into white-box, enabling precise diagnosis of performance issues within large-scale training platforms

Method: LLMPrism



① LLM Training Job Recognition ② Parallelism Strategies Identification ③ Training Timeline Reconstruction ④ Performance Diagnosis

Step-1
LLM Training Job Recognition

Spatial Communication Patterns

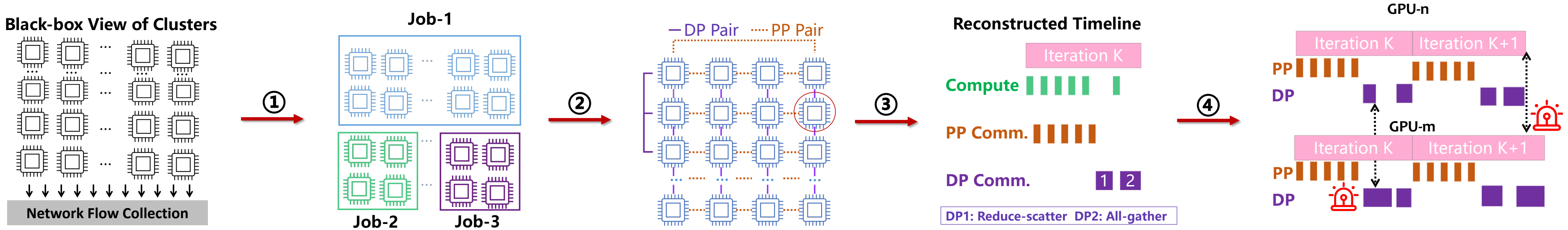
Algorithm 1: LLM Training Jobs Recognition

Input: RoCE Network Flows: $\mathcal{F} = \{f_1, f_2, \cdots\}$; List of GPUs: $\mathcal{X} = \{x_1, x_2, \cdots\}$; Physical Topology: T

Output: Recognized LLM task jobs: $\mathcal{C} = \{C_1, C_2, \cdots\}$;

$U \leftarrow$ Disjoint-set data structure
// (1) Search neighbors and build cross-machine clusters
for *each network flow* $f_i \in \mathcal{F}$ **do**
  srcAddr, dstAddr $\leftarrow f_i$
  if *U.findSet(srcAddr) != U.findSet(dstAddr)* **then**
    U.unionSet(srcAddr, dstAddr) // merge two GPUs
$\mathcal{CR} \leftarrow U$.getAllSets() // cross-machine clusters
// (2) Build task-level clusters based on physical topology T
for *each cross-machine clusters* $c_i \in \mathcal{CR}$ **do**
  get all server list s_i of c_i from physical topology
for *each pair* c_i *and* $c_j \in \mathcal{CR}$ **do**
  if *Jaccard Similarity* (s_i, s_j) = *1* **then**
    merge c_i and c_j
return $\mathcal{C} \leftarrow$ getMergedSet() // job-level clusters

Method: LLMPrism



① LLM Training Job Recognition ② Parallelism Strategies Identification ③ Training Timeline Reconstruction ④ Performance Diagnosis

Algorithm 2: Communication Type Identification

Input: RoCE Network Flows: $\mathcal{F} = \{f_1, f_2, \cdots\}$; Comm. pairs within a job: $\mathcal{P} = \{(u_1, v_1), \cdots (u_n, v_n)\}$
Output: Identified communication types: $\mathcal{T} = \{t_1, \cdots, t_n\}$;

```
G ← unordered graph of all GPUs within the job
for each pair (u, v) in P do
    // (1) Calculate communication interval
    extract related network flows: F_(u,v) = {f_1, ··· , f_t}
    compute comm. interval: Δt_i = f_{i+1}(time) − f_i(time)
    // (2) Step division
    employ Bayesian CPD: F_(u,v) = {F_step_1, F_step_2, ···}
    // (3) Type distinction
    for each step_k do
        count the distinct number of flow sizes N_k in F_step_k
    type t(u, v) ← PP if Mode(N_k) = 1 else DP
    add edge (u, v) to G if t(u, v) is DP
// (4) Noise refinement
DFS to find all connected component CC_i in G
for all pairs (u, v) in all CC_i do
    refine t(u, v) to DP if it is PP
return T = {t_1, ··· , t_n} // final communication types
```

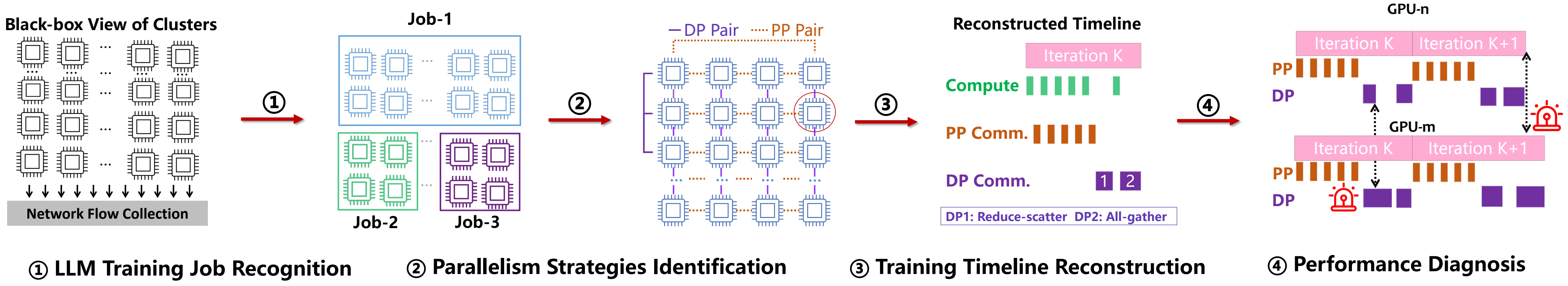
Step-2
Parallelism Strategies Identification

Distinctive Parallelism Communication

Bayesian Online Change-point Detection (BOCD) algorithm

$$r_t = \begin{cases} 0, & \text{if a change-point occurs at } t \\ r_{t-1} + 1, & \text{otherwise} \end{cases}.$$

Method: LLMPrism



Step-3
Training Timeline Reconstruction

Temporal Communication Patterns

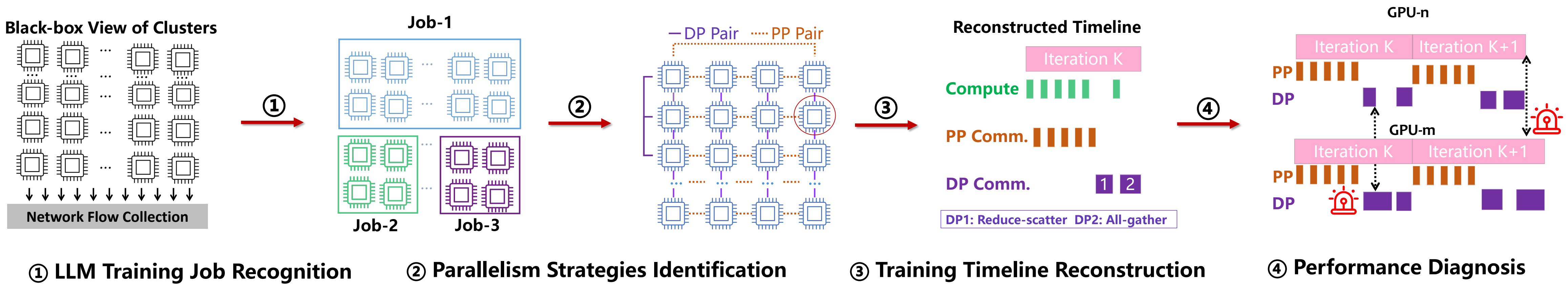
Key Findings:

- each training step concludes with a segment of DP collective communication traffic

Bayesian Online Change-point Detection (BOCD) algorithm

$$r_t = \begin{cases} 0, & \text{if a change-point occurs at } t \\ r_{t-1} + 1, & \text{otherwise} \end{cases}.$$

Method: LLMPrism



Step-4
Performance Diagnosis

Cross-step Diagnosis

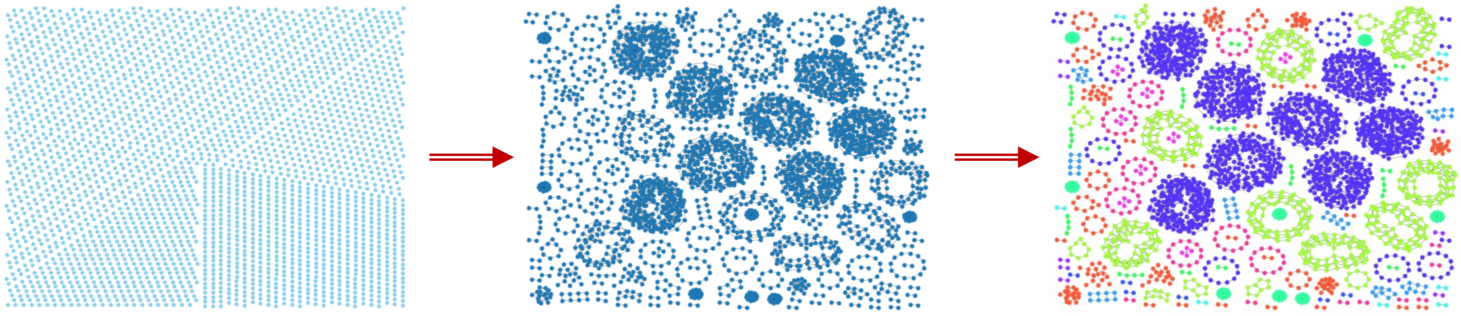
Cross-group Diagnosis

Switch-level Diagnosis

Evaluation

Part-1: Effectiveness of LLM Training Jobs Recognition

successfully identifying 19 training jobs



Black-box View | Cross-machine Clusters | Job-level Clusters

Internal Training Clusters with 2,880 XPUs

Evaluation

Part-2: Effectiveness of Parallelism Strategies Identification

TABLE I: Accuracy in identifying parallelism strategies via flows of varying-length periods for jobs with 1024 GPUs.

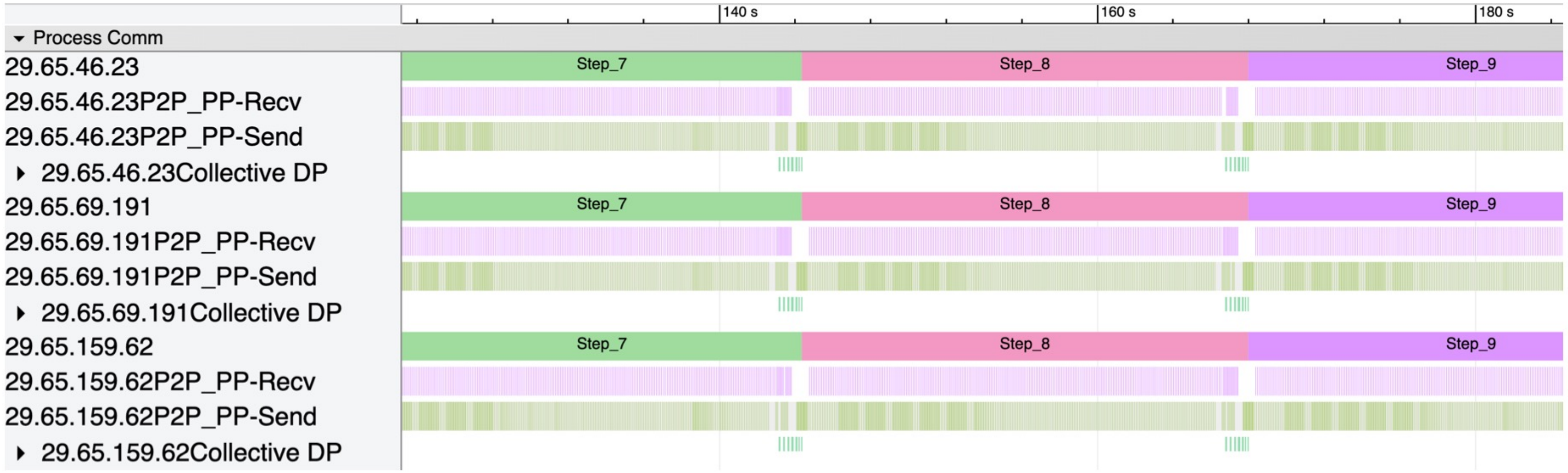
Methods	1 min Acc.	3 min Acc.	5 min Acc.	10min Acc.
LLMPrism w/o refinement	96.00%	97.93%	98.03%	99.61%
LLMPrism	100%	100%	100%	100%

5 LLM training jobs with 1024 XPUs with manual ground truths

- Without the noise refinement process, errors may occur in parallelism strategy classification, where some DP pairs are misclassified as PP pairs.
- This issue becomes more pronounced when network flow data is limited to shorter durations.
- Incorporating the noise refinement process allows LLMPrism to consistently achieve 100% accuracy across all training jobs

Evaluation

Part-3: Effectiveness of Training Timeline Reconstruction

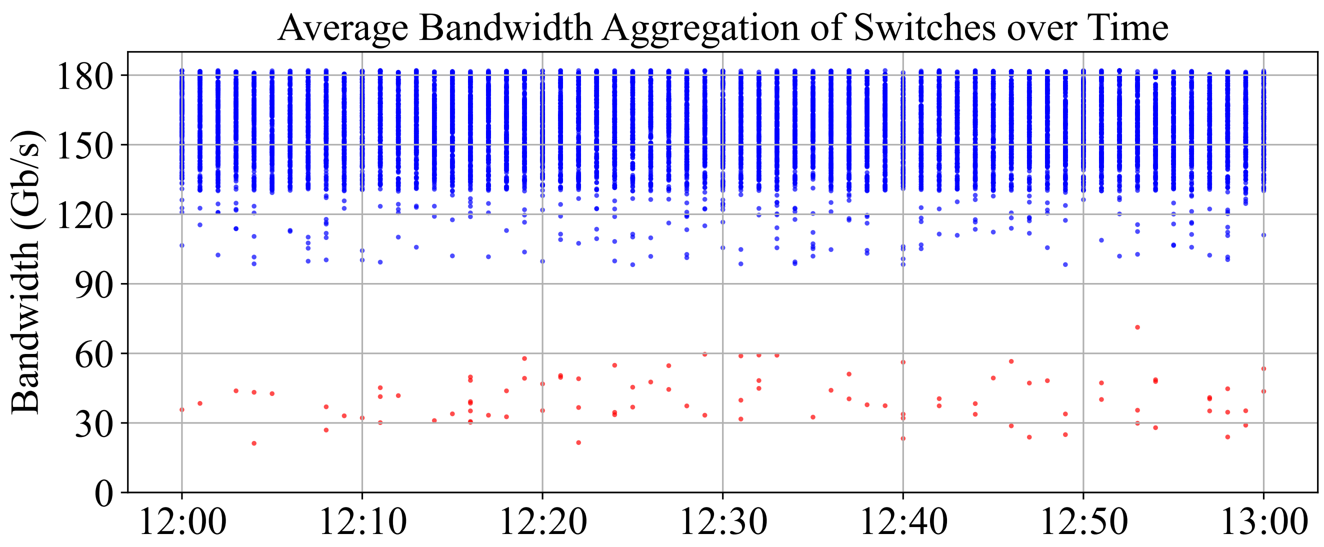


The reconstructed training timeline of a training job using 1,024 XPUs

reconstruction error of LLMPrism remained within 0.3%

Evaluation

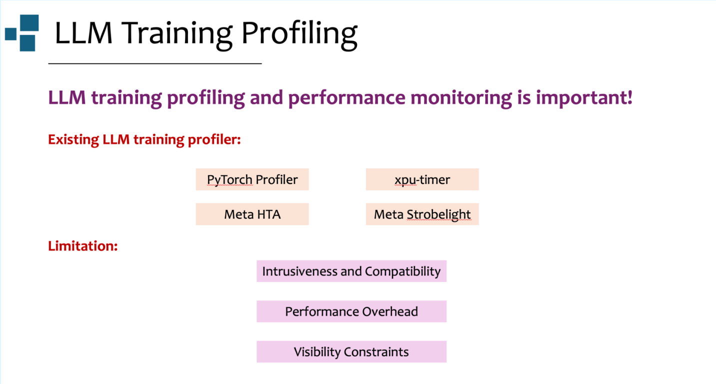
Part-4: Effectiveness of Performance Diagnosis



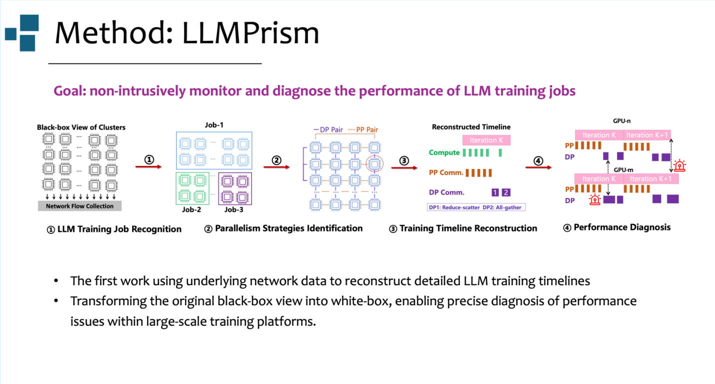
Switch-level diagnosis of LLMPrism using one-hour aggregated average bandwidths

successfully identified network issues and triggered alerts

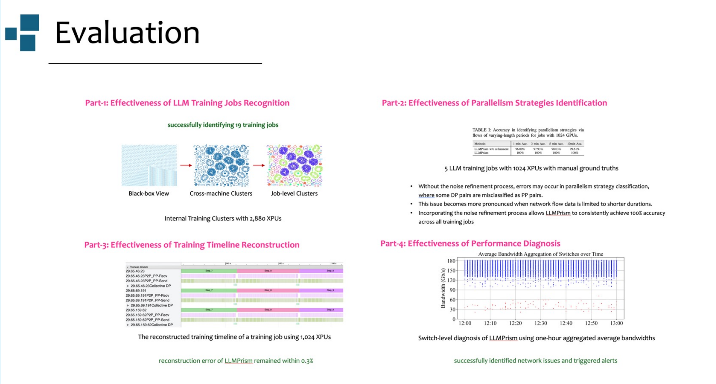
Conclusion



Motivation



Approach



Experiment

Thank you for listening

Q & A

Contact: zhjiang@link.cuhk.edu.hk