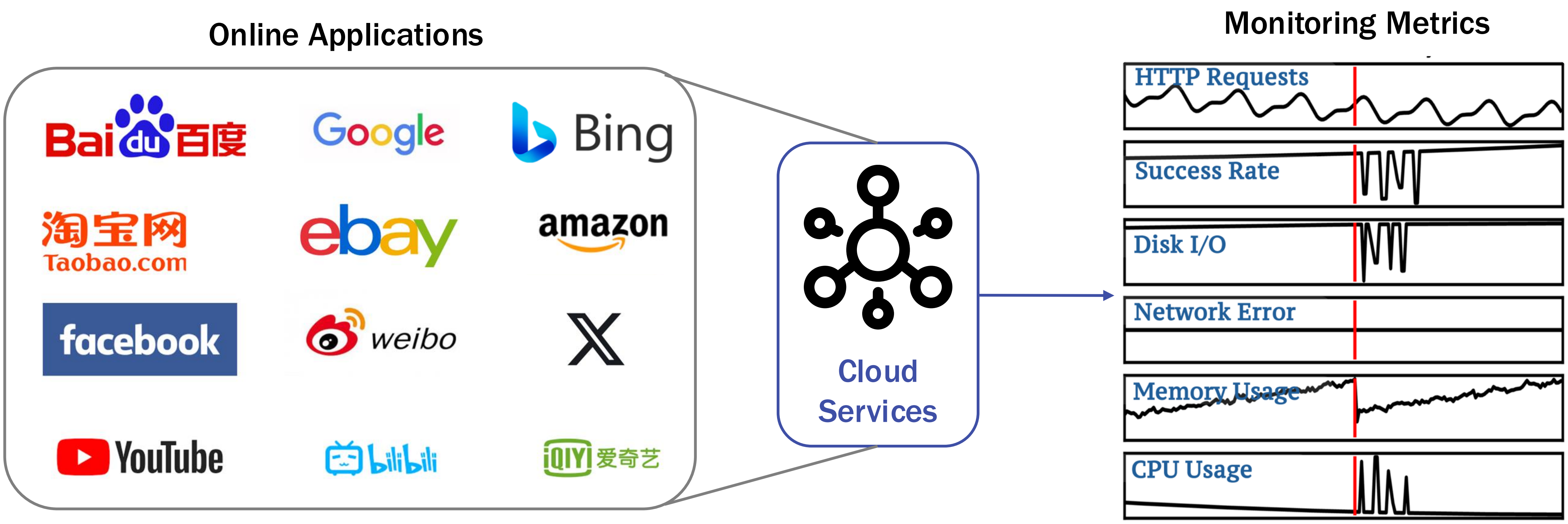
ICSE 2026

FoundRoot: Towards Foundation Model for Root Cause Analysis via Structured Deep Thinking

Zhe Xie[1], Zeyan Li, Xiao He, Shenglin Zhang, Longlong Xu, Yuzhuo Yang, Tieying Zhang, Jianjun Chen, Rui Shi, Dan Pei

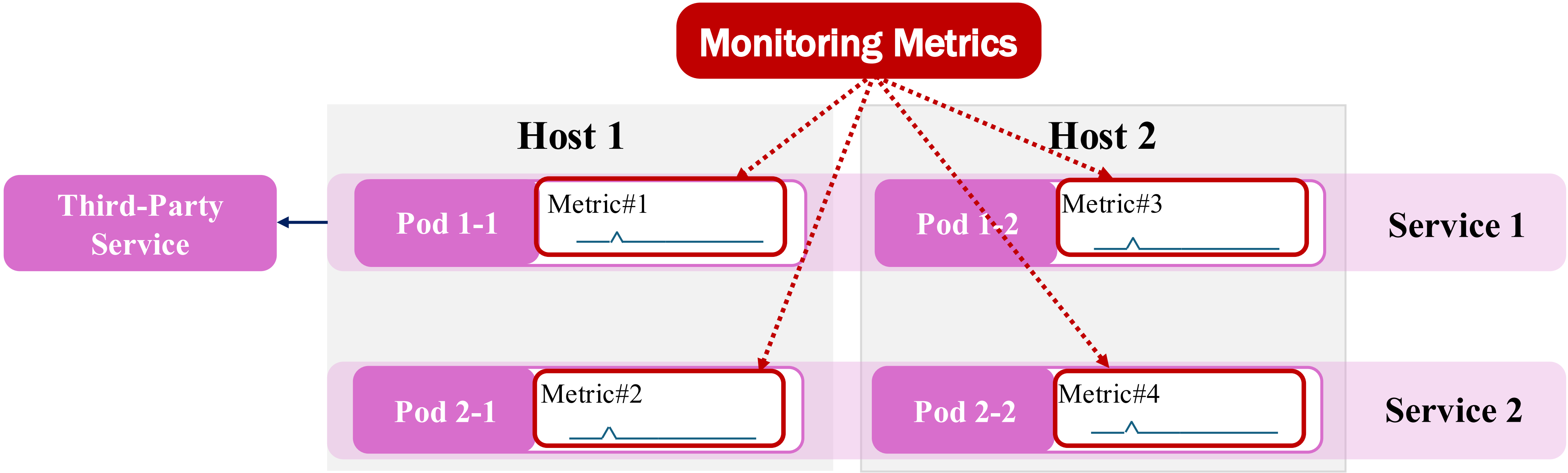
1. Presenter. Email: xiez22@mails.tsinghua.edu.cn

Background



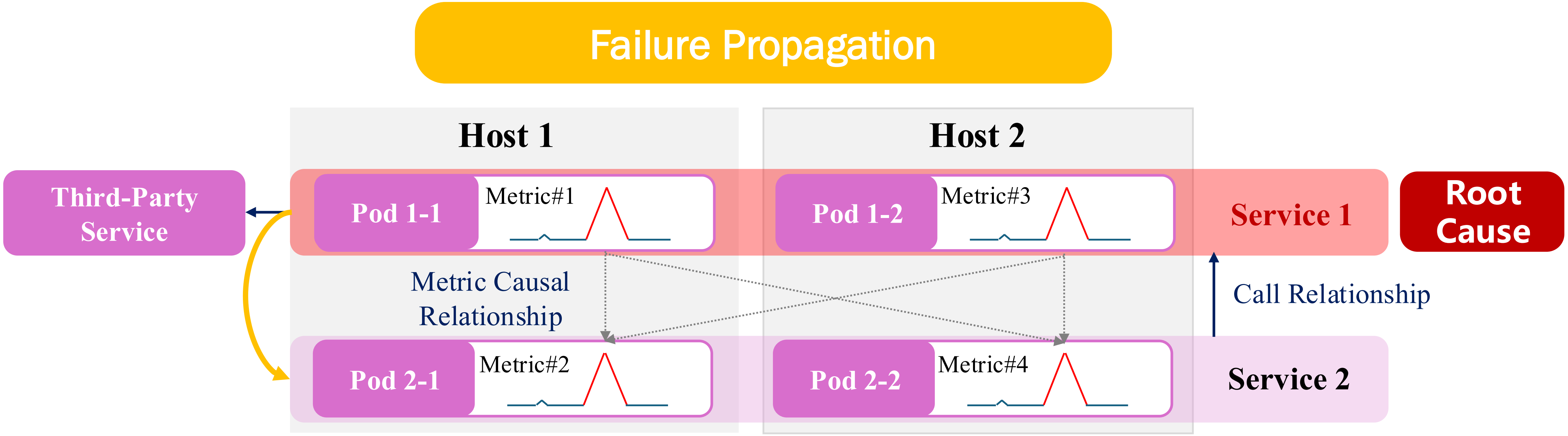
Online services produce lots of monitoring metrics

Root Cause Analysis



- However, system failures are inevitable due to frequent change and scaling
- In cloud services, observability tools are deployed to monitor the system status by collecting data (like metrics)

Root Cause Analysis



Root Cause Analysis (RCA)

- Rank the **Components** according to the metrics to identify **the root cause** of the failure

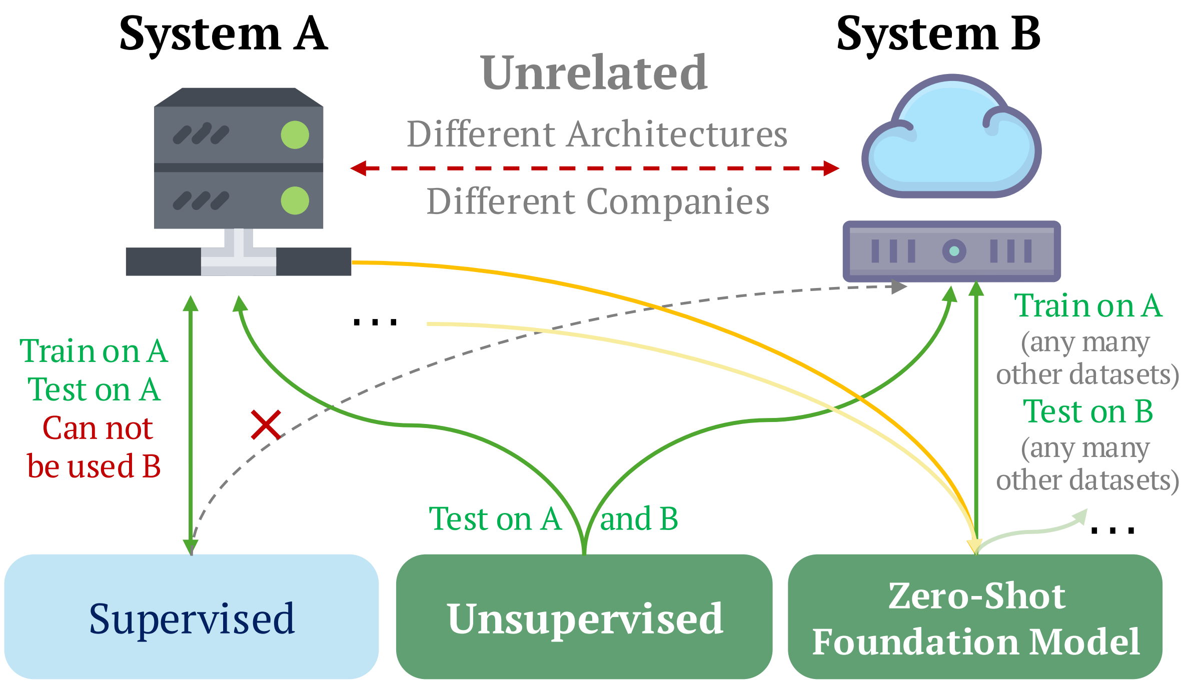
Zero-Shot RCA

Supervised Methods

✅ High accuracy within the same system

❌ Poor generalization (or even unusable) to unseen systems

❌ Requires costly labeled RCA data



Unsupervised / Zero-Shot Methods

✅ Generalizes to unseen systems

✅ No need for system-specific labels

Zero-shot foundation models enable scalable RCA across heterogeneous systems

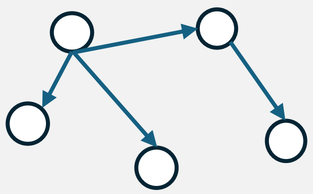
LLM-Based RCA

- Causal graphs are built with statistical methods (**time-consuming** causal discovery algorithms, *e.g.*, PC) and **manually set rules**
- Rank on causal graphs, **without considering physical meaning**

Classical Methods

1. Anomaly Detection

2. Build Causal Graphs



3. Rank

No. 1 Service-C
No. 2 Pod-B

LLM-based

1. Anomaly Detection

2. Analyzing Failure Propagation with LLM

First, I will start by …
Wait, maybe …

3. Rank and Explain

No. 1 Service-C,
because a spike of …

- Take **both physical meanings and the metric values** into consideration
- **Reflection** with a structured thinking process during reasoning
- Rank all the components in order, with **detailed analysis about the propagation**

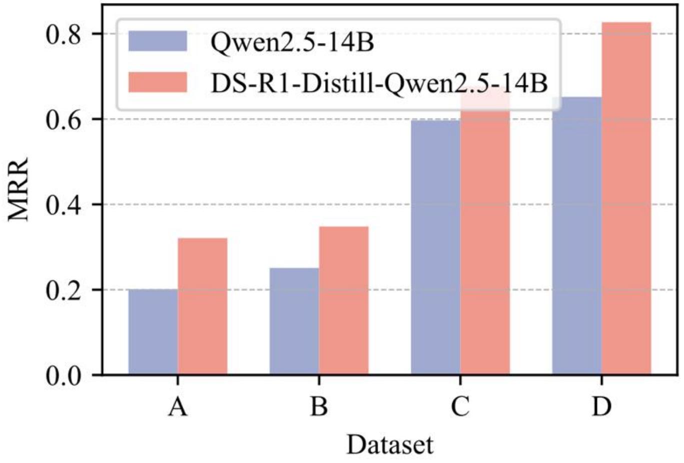
LLM-based methods **enable more accurate and interpretable** RCA by jointly leveraging **metric semantics** and values, while performing structured reasoning over failure propagation

Deep Thinking Helps — But Not Enough

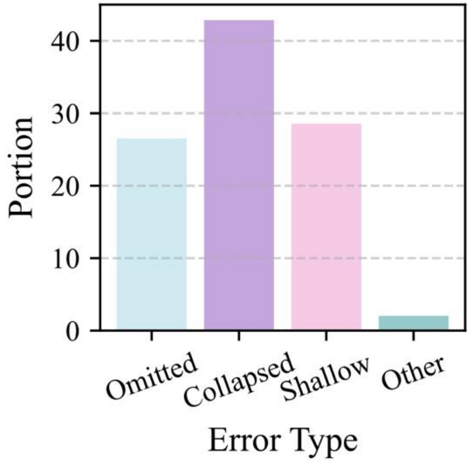
- Compared with the non-thinking base model, deep-thinking LLMs improve on all 4 evaluation datasets.
- So the bottleneck is not semantics alone: reasoning quality really matters.
- But a large fraction of cases still fail when the metric set becomes large.

w/o Deep Thinking

w/ Deep Thinking



(a) w/ or w/o deep thinking



(b) Error w/ deep thinking

Deep thinking improves RCA performance, but current LLMs still struggle with incomplete and inconsistent reasoning when handling large-scale metrics.

Typical Failure Modes

- LLMs often miss critical signals from large-scale metrics, either by *omitting key metrics* or *introducing contradictions* during reasoning
- Even when relevant information is considered, reasoning remains *shallow* and *prone to factual or causal errors*

Current LLM reasoning is neither comprehensive nor reliable for complex RCA tasks with ***large-scale metrics input***

Omitted: Root cause metric is completely omitted during reasoning

Example (Root Cause: os_018 Network)
<think>Alright, I'm trying to figure out...
First, looking at db_008, I've ... Looking further, docker_002 ...
...(metrics related to os_018 Network are completely omitted)...
So, the root cause seems to be **db_007 ...** </think>

Collapse: Metrics are analyzed initially but contradictions arose later

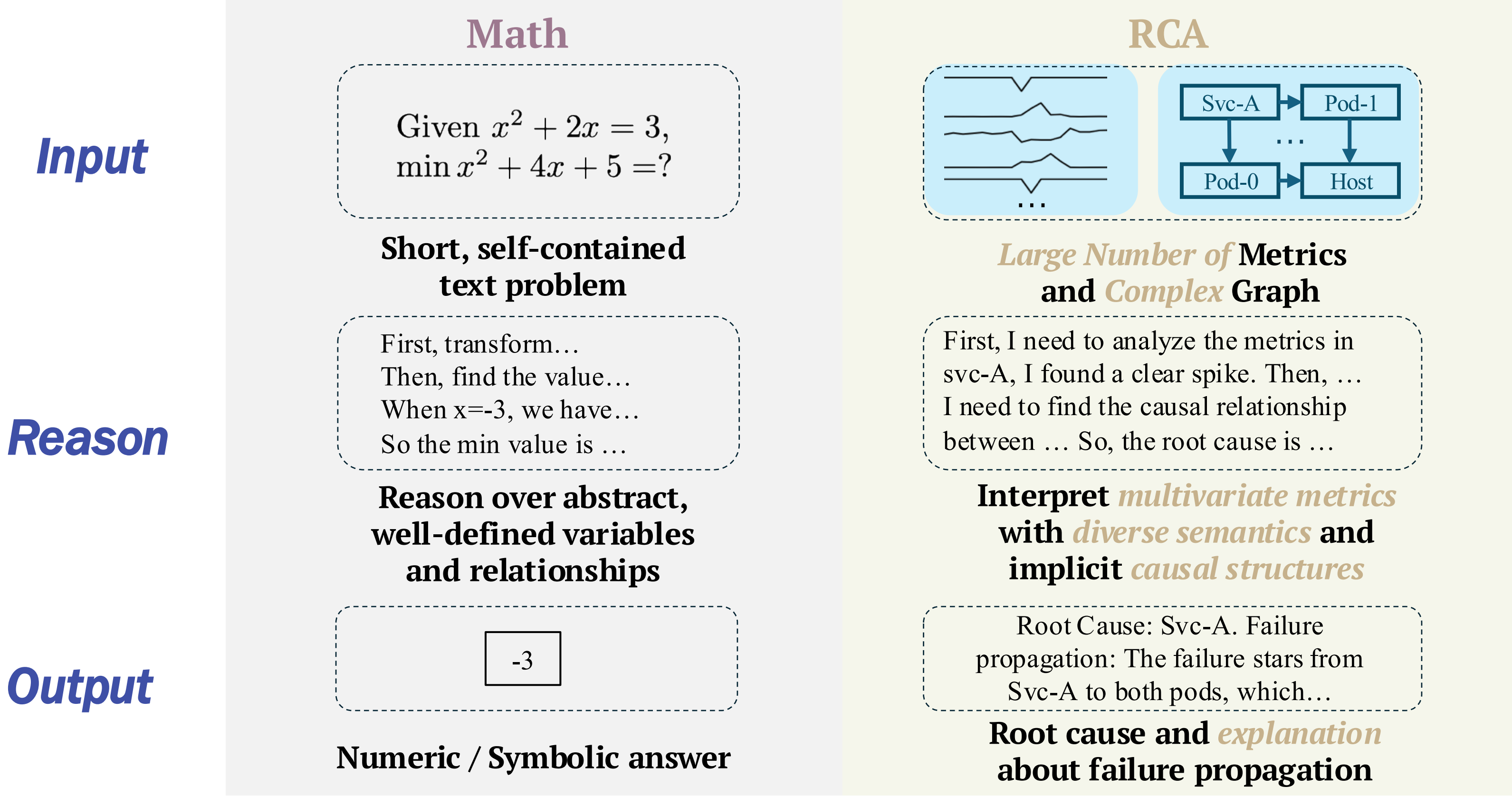
Example (Root Cause: docker_006)
<think>...docker_006##proc spikes **indicate increased processing**... db_003's drop **might be the downstream** of docker_006... Therefore, **db_007 is the root cause**, other components like **docker_006** is either **noise or normal...**

Shallow: Reasoning is shallow and contains some clear factual errors

Example (Root Cause: db_003)
<think>...looking at the component graph, I see that **docker_005 calls db_003**... Considering the component graph, if docker_005 fails, it would affect db_003...

Math Reasoning vs. RCA Reasoning

ICSE 2026



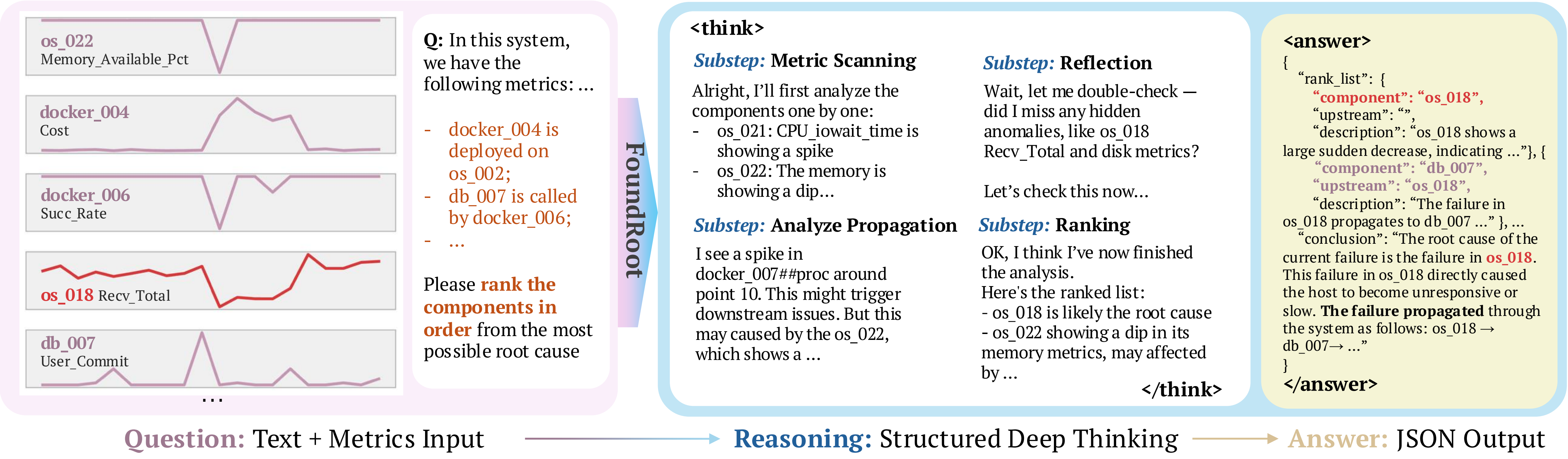
RCA reasoning fundamentally differs from *math/code reasoning* by requiring the integration of *large-scale multimodal inputs* and the understanding of *complex, implicit dependency structures*

What FoundRoot Adds

1. Structured deep thinking	Break down the RCA reasoning process into structured substeps.
2. Diverse RCA data curation	Curate 10 open-source datasets to ensure diversity and scalability.
3. SFT + RL Training	Enhance structured deep thinking for RCA through a two-stage training pipeline.

Key Message: Enable explicit, complete, and robust reasoning for RCA.

FoundRoot: Overview



- **Input:** Text + large-scale metric observations
- **Reasoning:** Structured deep thinking *(4 sub steps)* for causal analysis
- **Output:** Interpretable root cause ranking and propagation explanation

Structured Deep Thinking

<think>

Substep: **Metric Scanning**

Alright, I'll first analyze the components one by one:
- os_021: CPU_iowait_time is showing a spike
- os_022: The memory is showing a dip…

Substep: **Analyze Propagation**

I see a spike in docker_007##proc around point 10. This might trigger downstream issues. But this may caused by the os_022, which shows a …

Substep: **Reflection**

Wait, let me double-check — did I miss any hidden anomalies, like os_018 Recv_Total and disk metrics?

Let's check this now…

Substep: **Ranking**

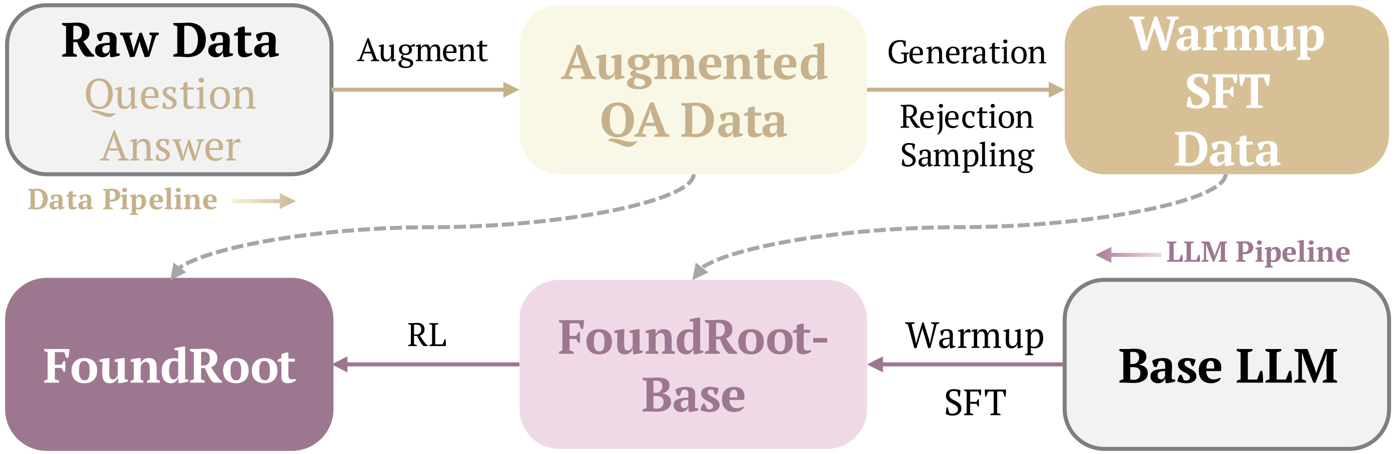
OK, I think I've now finished the analysis.
Here's the ranked list:
- os_018 is likely the root cause
- os_022 showing a dip in its memory metrics, may affected by …

</think>

By mimicking human operators, we design these four substeps as an ***internal deep thinking capability*** learned by the model.

1. Metric Scanning	Systematically lists all components and metrics to ensure no critical metrics are omitted.
2. Propagation Analysis	Infers causal relationships and identifies how failures propagate across system dependencies.
3. Reflection	Revisits previous reasoning to detect missing signals and resolve inconsistencies.
4. Ranking	Produces an ordered list of root causes with clear causal explanations.

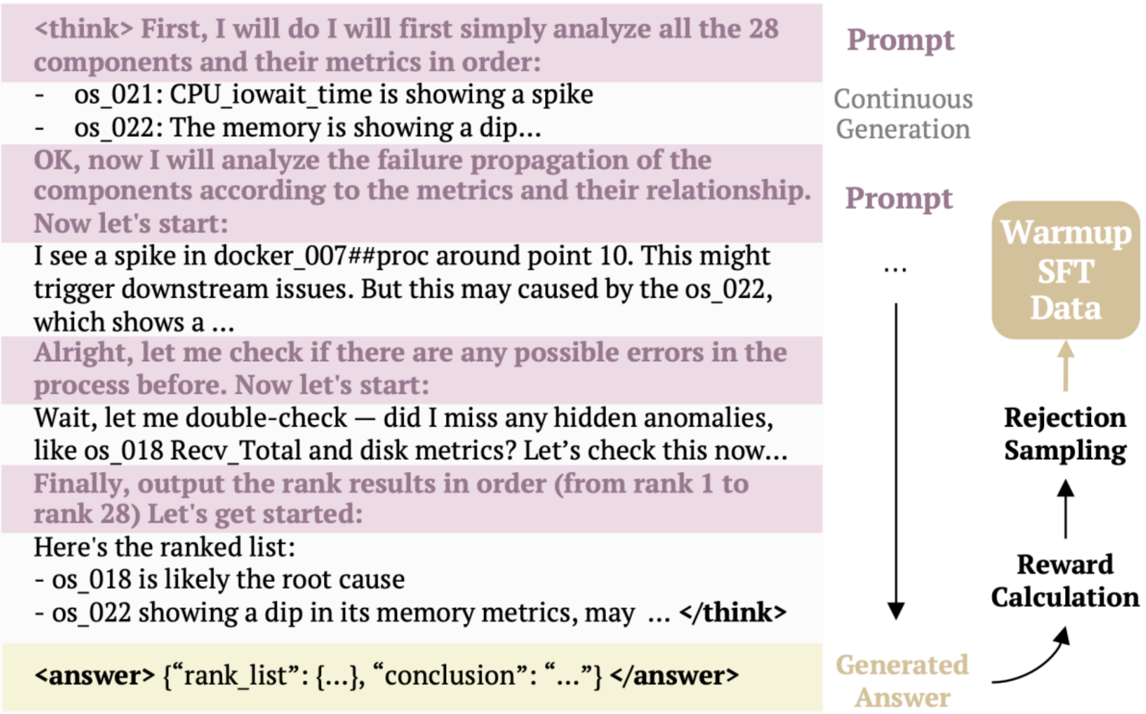
Overview of Training Process



- **Stage 1:** Warm-up SFT learns the structured deep thinking format.
- **Stage 2:** RL to further enhance the reasoning accuracy.

Construction of Warm-Up SFT Datasets

- Interleave prompts with **continuous generation** to guide each reasoning substep
- Insert template-based prompts (*e.g.*, ending with “I will …”) to ensure format-consistent continuation generation of LLMs
- Select high-quality reasoning traces via rejection sampling with **reward signals**



We generate structured reasoning data for warm-up SFT by *iterative continuous generation* with LLMs

Structured Output and Reward Functions


```
<answer>
{
  "rank_list": {
     "component": "os_018",
     "upstream": "",
     "description": "os_018 shows a large sudden decrease, indicating ..."},
{
     "component": "db_007",
     "upstream": "os_018",
     "description": "The failure in os_018 propagates to db_007 ..." }, ...
  "conclusion": "The root cause of the current failure is the failure in os_018. This failure in os_018 directly caused the host to become unresponsive or slow. The failure propagated through the system as follows: os_018 → db_007→ ..."
}
</answer>
```

Output Json

(Rank List + Text Conclusion)

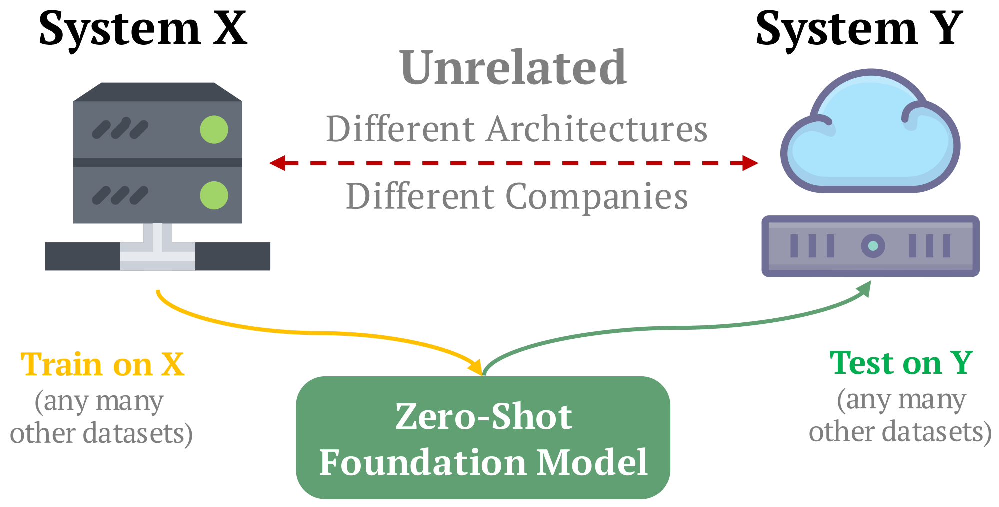
$$r(x, y) = \lambda_{\text{format}}\, r_{\text{format}} + \lambda_{\text{json}}\, r_{\text{json}} + \lambda_{\text{acc}}\, r_{\text{acc}} + \lambda_{\text{think}}\, r_{\text{think}}$$

- **Format reward:** Ensure outputs that contain both structured reasoning (<think>) and final answers (<answer>).
- **JSON reward:** Ensure the answer follows a valid JSON format with required fields.
- **Accuracy reward:** Promote correct root cause ranking using MRR (mean reciprocal rank):

$$r_{\text{mrr}} = \frac{1}{\text{rank}_{\text{gt}} + 1}$$

- **Structured thinking reward:** Enforce adherence to the four-step structured reasoning process.

Datasets



Dataset	# Cases (Aug.)	System	Split
A [22]	63	A service system of a major ISP	Test
B [22]	90	A service system of a major ISP	Test
C [15]	10	SocialNetwork	Test
D [5]	255	MicroSS (GAIA)	Test
E [22]	136 (1,158)	A service system of a bank	Train
F [21]	81 (769)	Oracle DB	Train
G [15]	43 (345)	TrainTicket	Train
H [50]	34 (259)	TrainTicket	Train
I [21]	94 (771)	TrainTicket	Train
J [50]	27 (200)	HipsterShop	Train

- 10 open-source RCA datasets in total.
- 6 datasets for training, 4 disjoint datasets for evaluation.
- Evaluation systems are unseen during training.

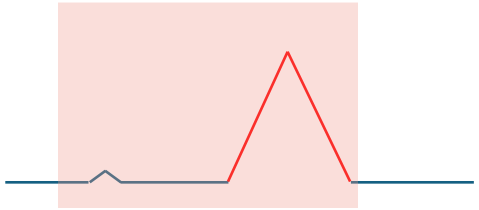
Training Data Augmentation

We use both *temporal sampling* and *metric sampling* for training data augmentation.

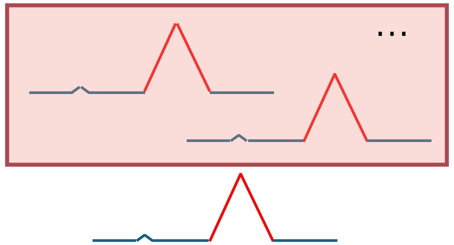
400+ → **3,500+**

Original Training Cases → Augmented Training Samples

Temporal Sampling	Sample variable-length windows (sliding window) from each failure case.
Metric Sampling	Sample both root-cause and non-root-cause metrics (to avoid hacking), but make sure at least one root-cause metric remains.



Temporal Sampling



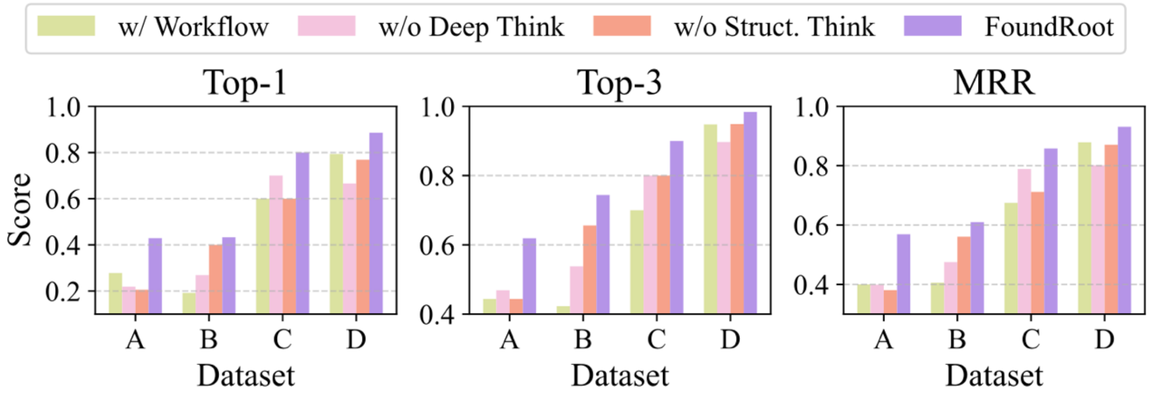
Metric Sampling

Evaluation Results

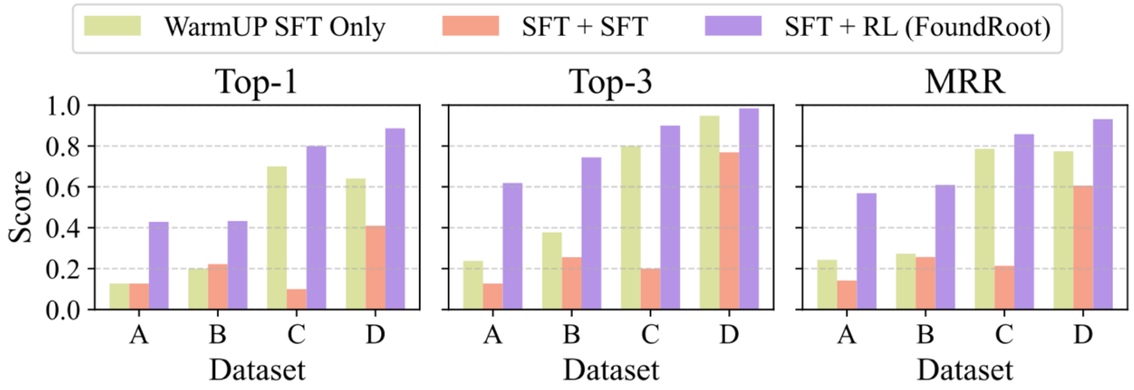
	Dataset	A			B			C			D		
Type	Method	Top-1	Top-3	MRR	Top-1	Top-3	MRR	Top-1	Top-3	MRR	Top-1	Top-3	MRR
Classical	MicroScope	0.143	0.159	0.221	0.089	0.233	0.211	0.300	0.400	0.462	0.436	0.795	0.645
	MonitorRank	0.063	0.095	0.154	0.078	0.167	0.228	0.200	0.200	0.296	0.461	0.897	0.687
	MicroCause	0.159	0.413	0.292	0.089	0.489	0.361	0.100	0.700	0.383	0.410	0.923	0.636
	CIRCA	0.079	0.302	0.272	0.144	0.400	0.329	0.400	0.700	0.571	0.282	0.871	0.553
	RCD	0.143	0.270	0.188	0.111	0.144	0.121	0.100	0.100	0.125	0.539	0.744	0.637
	BARO	0.048	0.222	0.212	0.133	0.333	0.290	0.400	**0.900**	0.583	0.615	0.949	0.776
	ART	0.095	0.127	0.177	0.133	0.289	0.242	0.600	0.700	0.708	0.256	0.564	0.486
LLM	Qwen2.5-14B	0.048	0.222	0.221	0.067	0.267	0.250	0.500	0.800	0.597	0.462	0.846	0.652
	GPT-4o	0.143	0.460	0.338	0.122	0.256	0.256	0.700	0.800	0.763	0.590	0.949	0.762
	o4-mini	0.079	0.175	0.168	0.056	0.133	0.162	0.700	0.800	0.754	0.820	0.949	0.891
	R1-14B	0.175	0.317	0.330	0.178	0.344	0.329	0.500	0.700	0.677	0.794	0.923	0.827
	R1-Full	0.254	0.413	0.383	0.200	0.467	0.426	0.700	0.800	0.811	0.785	0.929	0.859
	Doubao-Thinking	0.111	0.429	0.364	0.178	0.500	0.404	**0.800**	0.800	0.834	0.785	0.929	0.859
	RCA-Agent+R1-Full	0.032	-	-	0.178	-	-	0.700	-	-	0.462	-	-
	FoundRoot-14B	**0.429**	**0.619**	**0.569**	**0.433**	**0.744**	**0.610**	**0.800**	**0.900**	**0.858**	**0.886**	**0.984**	**0.931**
	Improvement	+68.9%	+34.6%	+48.6%	+116.5%	+48.8%	+43.2%	0%	0%	+2.9%	+8.0%	+3.7%	+4.5%

FoundRoot outperforms both classical and strong LLM baselines on *all 4 evaluation datasets.*

Ablation Studies



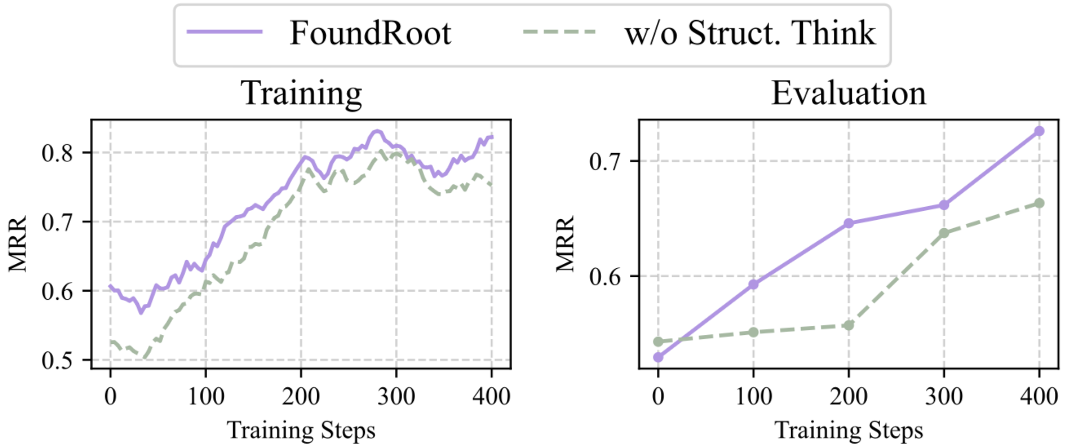
(a) Ablation Studies of Structured Deep Thinking



(b) Ablation Studies of RL

- *Structured deep thinking* consistently improves performance across all datasets and metrics
- Removing deep thinking leads to noticeable drops, especially in *complex datasets (A & B)*
- *RL* further boosts reasoning ability beyond SFT, achieving the best overall performance
- *SFT alone is insufficient* and may even limit reasoning quality compared to RL-enhanced models

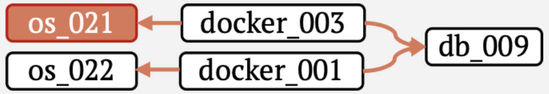
Why Structured Deep Thinking Works?



- Structured thinking yields similar training gains but significantly larger improvements on evaluation, indicating better generalization.

- Structured thinking performs more complete and self-corrected reasoning, mitigating missing signals and severe hallucination under with large-scale of input metrics.

Root Cause: os_021



FoundRoot-14B

<think> *(Step 1)* I will first analyze all the components...

(Step 2) I will analyze the propagation of the components...
... "os_021##proc" drops from 1217.85 to 1457.51, **but wait**, no, looking at the data, it's 1217.85, then 1457.51, but that's a spike, not a drop...because "docker_003" is host by "os_021", so if "os_021" crashes, "docker_003" drop...
(Note: FoundRoot analyzed all the metrics in Metric Scanning)
(Step 3) Now, I will check if there are any possible errors...
I should verify if "docker_003" could be the root cause. "os_021" is a deployment for "docker_003", so if "os_021" fails, "docker_003" can't run properly. Therefore, **"os_021" is the initiating failure point...**
(Note: In Reflection, os_021 is correctly identified as the root cause)
(Step 4) The final rank results... The root cause is "os_021"...**</think>**

R1-14B (Missing Key Metrics)

<think> The components to rank are quite a few, so I'll go through them one by one. Starting with "docker_001"... Next, "os_019 Memory"...
Other components like "os_019 Memory"...
In conclusion, the root cause is "docker_001"</think>
(Note: os_021 is not analyzed at all during thinking, leading to wrong RCA)

R1-14B + Workflow (Incomplete Reasoning)

<think> I will analyze ...
There is a drop in os_021##proc...
docker_003##count drops.... This would cause the services it supports to drop.
(Note: os_021 is hosting docker_003, it should be the upstream reason for the drop in docker_003##count, but it's ignored, leading to incorrect RCA)
In conclusion, the root cause is "docker_003"</think>

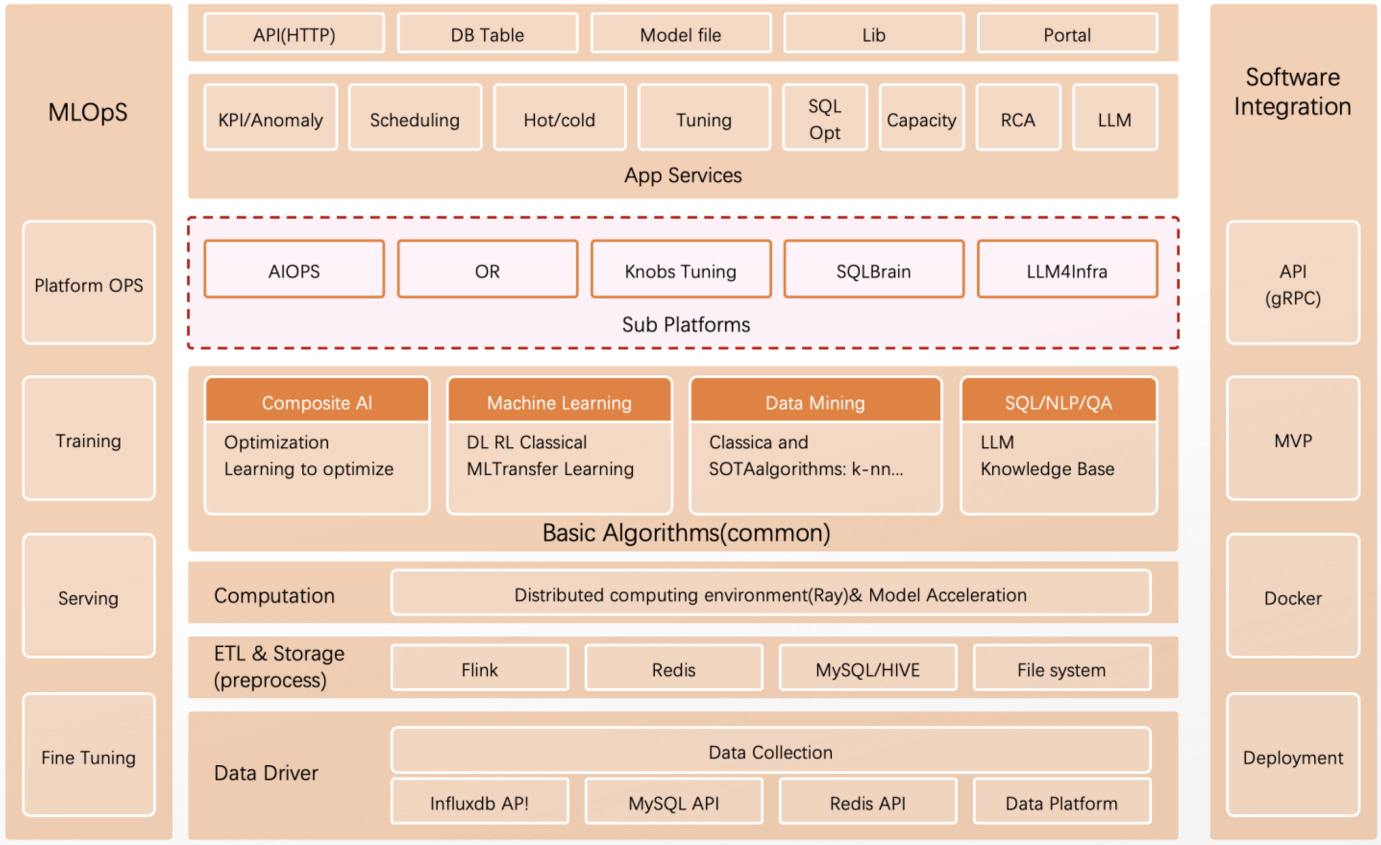
Conclusion

- **RCA Foundation Model:** We propose FoundRoot, a foundation model for *zero-shot RCA* that performs deep thinking over multivariate system metrics.
- **Structured reasoning framework:** We use a *structured deep thinking* process that explicitly models metric scanning, propagation analysis, reflection, and ranking.
- **Strong performance and generalization:** FoundRoot achieves *consistent improvements across multiple datasets*, demonstrating superior reasoning quality and cross-system generalization on RCA tasks.
- **Training paradigm:** We introduce a training pipeline combining structured SFT and RL to effectively learn and enforce deep reasoning behaviors.

AI for Infra Platform – ByteBrain

ByteDance 字节跳动

- AI/LLM services for infrastructure and systems
- Goal: Leverage AI to automatically optimize system performance and reduce labor and resource costs
- Area: natural language processing (NLP), AIOps, AI for DB/Storage/Networking, and Operations Research



ICSE 2026

Thank You!

FoundRoot: Towards Foundation Model for Root Cause Analysis via Structured Deep Thinking

(Source code and datasets will be released after the company review)