Microsoft

SuperBench: Improving Cloud AI Infrastructure Reliability with Proactive Validation

Yifan Xiong, Yuting Jiang, Ziyue Yang, Lei Qu, Guoshuai Zhao, Shuguang Liu, Dong Zhong, Boris Pinzur, Jie Zhang, Yang Wang, Jithin Jose, Hossein Pourreza, Jeff Baxter, Kushal Datta, Prabhat Ram, Luke Melton, Joe Chau, Peng Cheng, Yongqiang Xiong, and Lidong Zhou

Cloud AI infrastructure has massive incidents.

During the 3-month OPT-175B Training on 1,184 A100, incidents reported by Meta [1]

1.25

Failures/Regressions per Day

39.3%

VMs Involved

61K

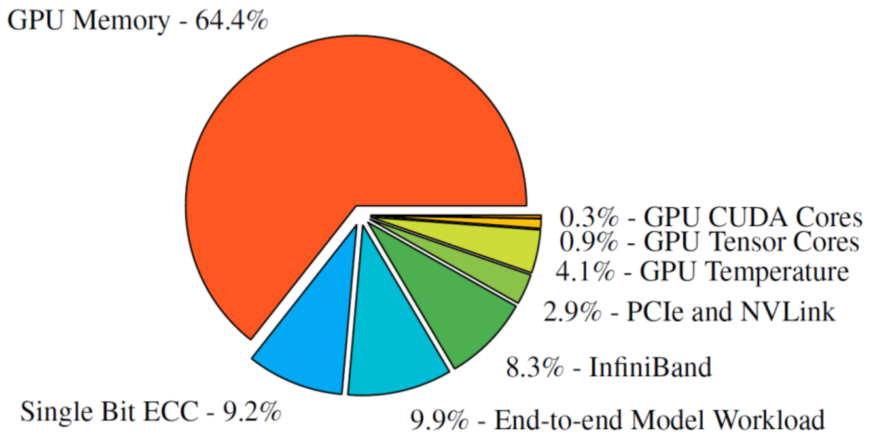
GPU Hours Affected

[1]. OPT-175 Logbook. https://github.com/facebookresearch/metaseq/blob/main/projects/OPT/chronicles/OPT175B_Logbook.pdf.

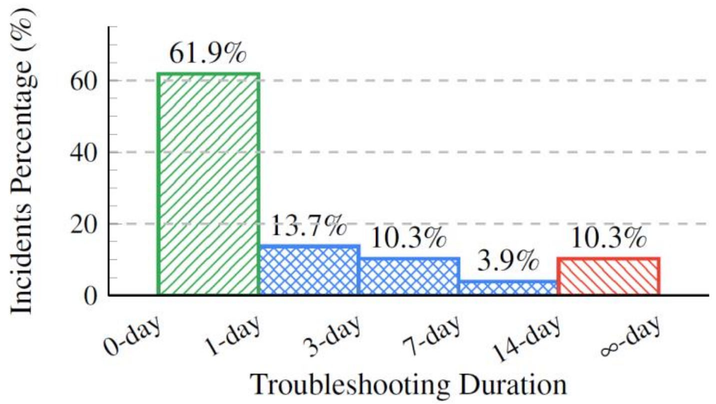
Incident Statistics in Azure Production Clusters

~2k incidents (regression or failure) in a 3-month period during 2022

- Many components involved: **>8** GPU related
- Long time to mitigate: **38.1%** >1-day, **10.3%** >1-week



Percentage of infrastructure incidents' sources



Incidents troubleshooting duration distribution

Why happens in cloud AI infrastructure?

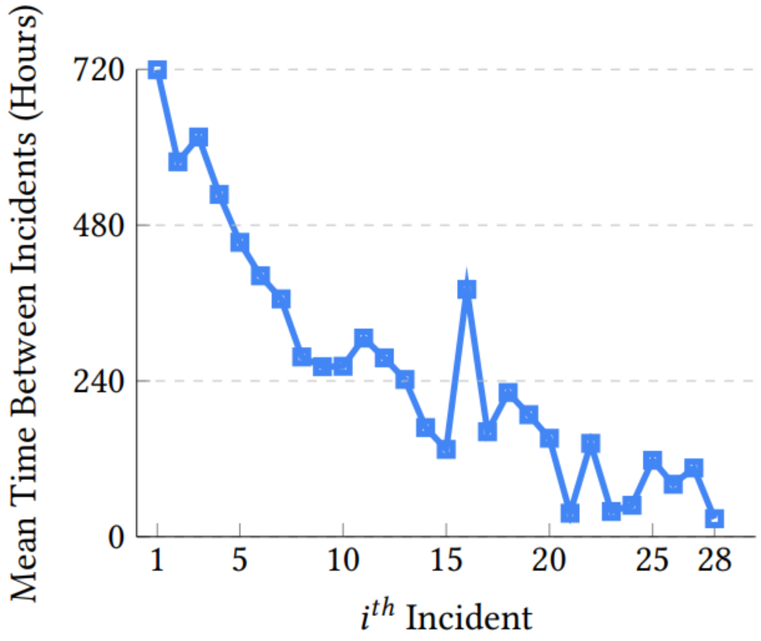
Emerging Issues in Cloud AI Infrastructure

- **Rapid hardware evolution**: e.g., hardware components are tested individually, fail to coverage regression in workloads
- **Cloud environment**: e.g., InfiniBand bit error rate can be 35x higher due to high temperature
- **Software immaturity**: e.g., single GPU issue can cause the entire distributed training to hang

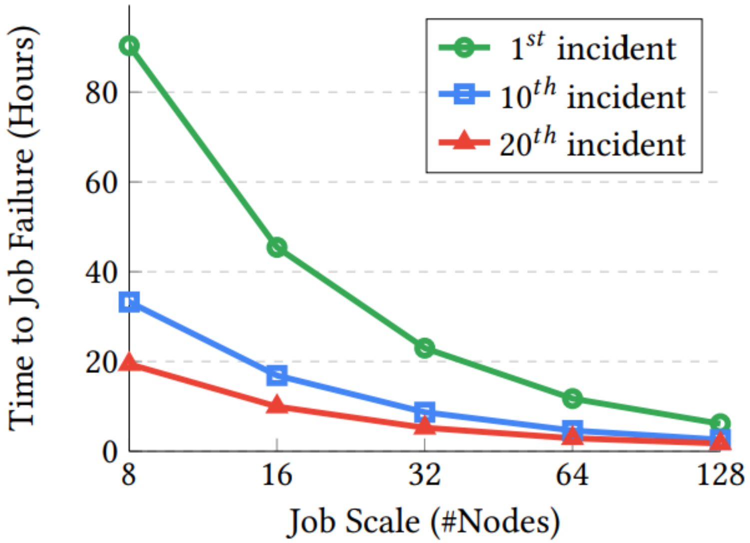
Therefore, redundancies are introduced to improve reliability.

Observations on Incidents

Even with redundancies, incidents still happen ***more frequently over time***.



Mean duration between i th and $i+1$ th incidents across all nodes that have $i+1$ incidents occurred

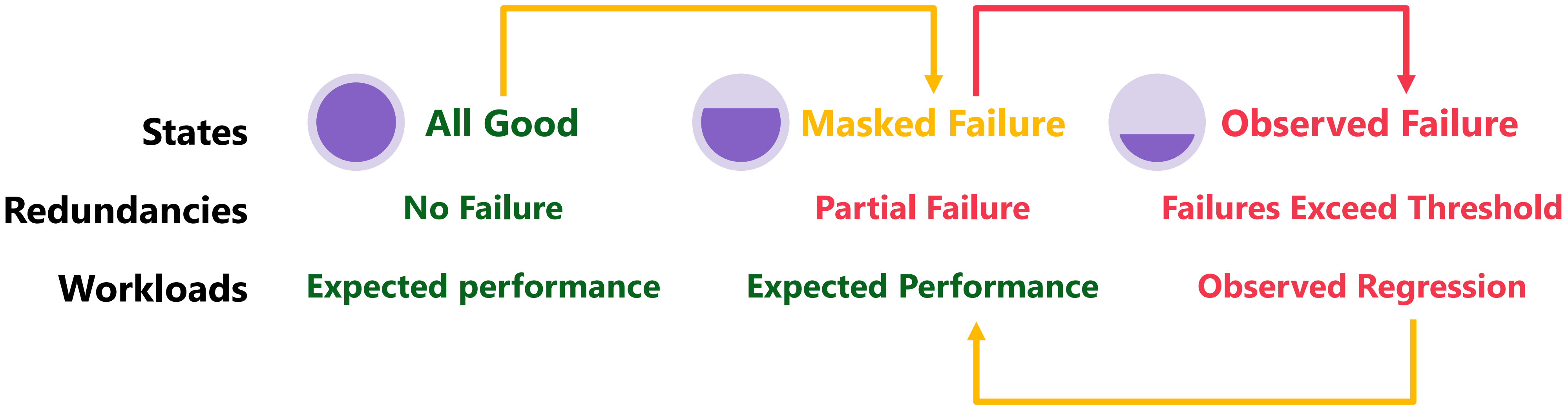


Time to failure for jobs if all nodes in the job have i th incidents occurred

Key Insight

Reactive troubleshooting can surprisingly compromise the reliability of cloud AI infra in unexpected ways, due to the existence of ***redundancies.***

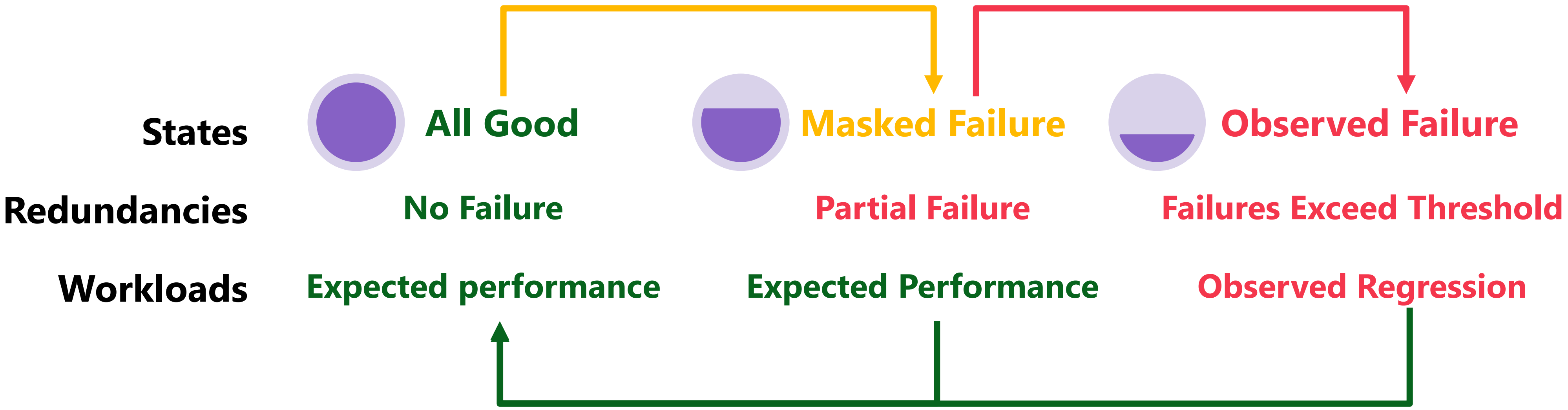
- partial redundancy failure can be masked in end-to-end workload performance
- reactive troubleshooting is performance oriented and only restores to **masked failure state**



Key Idea: Proactive Validation

Proactive validation improves reliability by **avoiding masked failure state**

- proactively run before incidents happen
- standalone tests to stress the hardware and pinpoint potential issues in redundancies



Key Questions

Three key questions on **how to do proactive validation**:

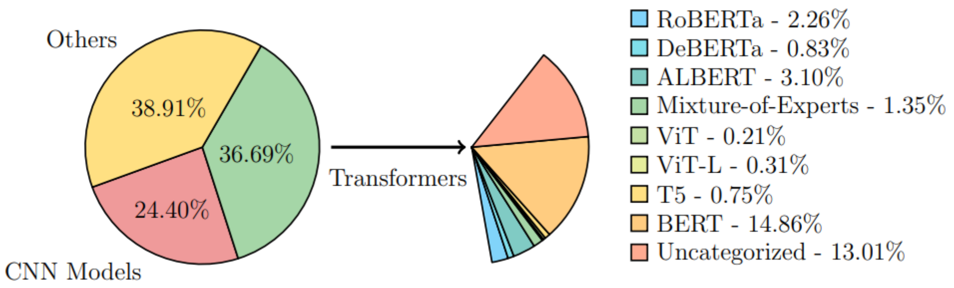
- **What to validate?**
- **What performance to expect?**
- **When to proactively validate?**

What to validate?

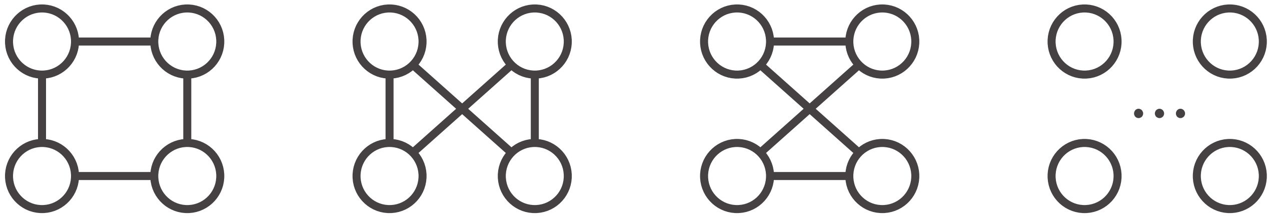
Hardware redundancies and customer workloads.

Challenge #1 – Huge Workload Space for Validation

- Diverse end-to-end customer workloads



- Exponential scale/node combinations



Solution #1 – A Small yet Representative Benchmark Set

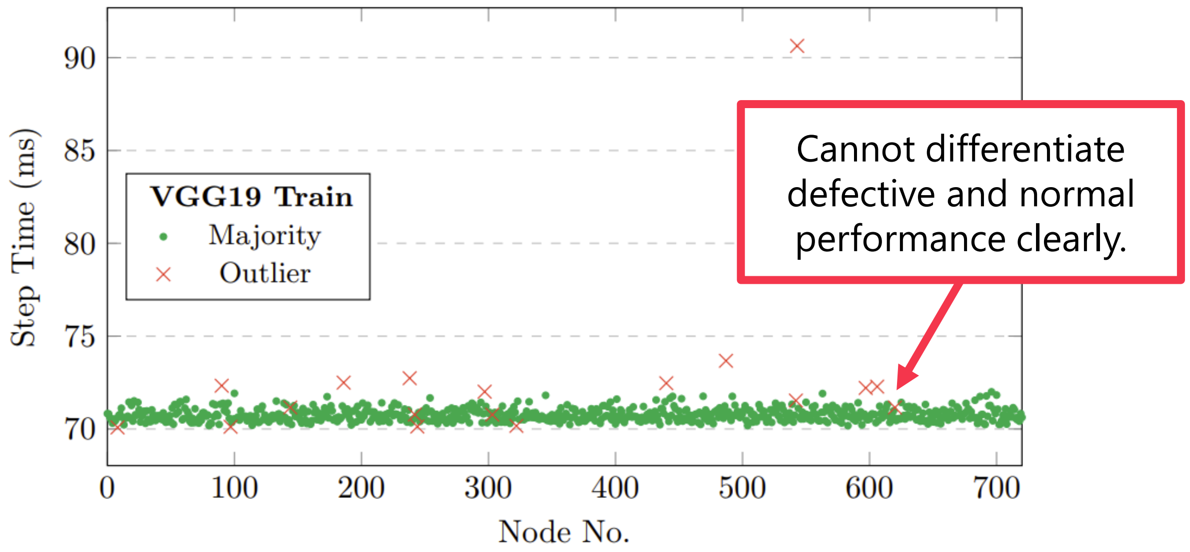
- **Representative end-to-end benchmarks**
  - Extract the most prevalent models and parameters from cluster job traces.
  - Continuously evolve with new models.
- **Comprehensive micro benchmarks**
  - Component-wise: stress individual hardware component one by one.
  - Pattern-wise: emulate workload patterns with multiple components used simultaneously.

What performance to expect?

Stable and high performance among healthy hardware replicas.

Challenge #2 – Unknown Ground Truth

- Gap between hardware spec and workload performance varies.
- Existing unsupervised outlier detection method doesn't work as expected.



Outlier detection on VGG19 training step time results

Solution #2 – Clear-cut Benchmark Criteria by Similarity Metric

- Define **similarity metric** between two benchmark samples S_1 and S_2, where $S_1 = \{S_{1,1}, \dots, S_{1,n}\}$ and $S_2 = \{S_{2,1}, \dots, S_{2,m}\}$.
  - Similarity = 1 – (integral area between S_1 and S_2 CDF curves) / (max integral area under S_1 and S_2 CDF curves)

- Offline train the **benchmark criteria** S_C for results from N nodes.

- Online **inference** defects **by similarity** between S_C and S_{New}.

When to proactively validate?

Frequently validate before incidents happen.

Challenge #3 – Trade-off between Duration and Coverage

- Predict node failures and partial regression in the future with dynamic failure rates.
- Select the most effective benchmarks according to current node status.

Cloud TPU

V4-2048

Location: Oklahoma

Number of chips: 1,024 (2,048 cores)

Total TPU Hours per month: 730

Commitment: Evaluation

USD 2,407,014
Implied price per chip-hour: USD 3.22

Total Estimated Cost: USD 2,407,014.40 per 1 month

1,024 chips price per month:

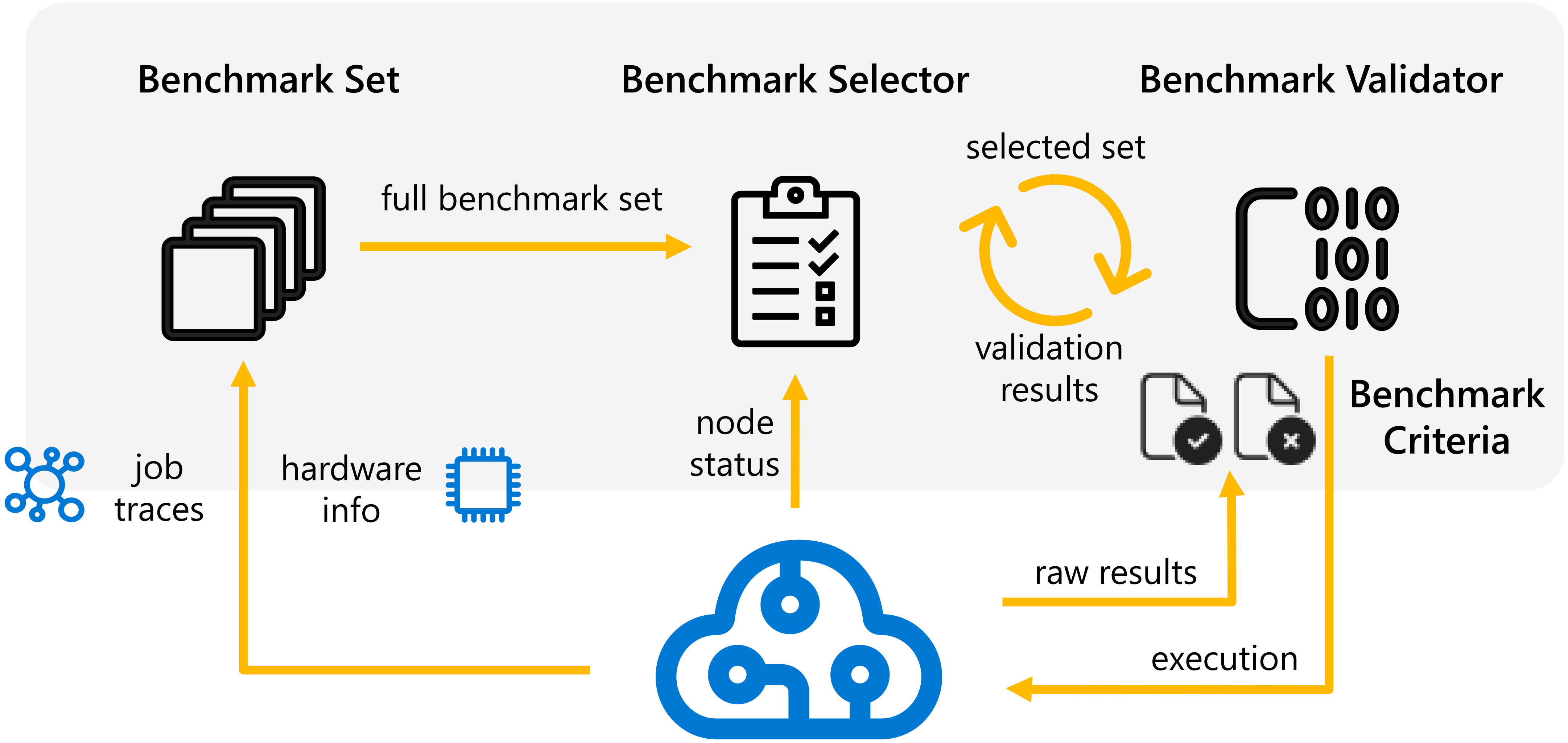
- Azure A100: $3.06M
- GCP TPUv4: $2.41M

Solution #3 – Efficiently Benchmark Selection

- Offline fit a **probability model** to predict time to next incident for each node
  - Input: total elapsed time, historical incident time, etc.
  - Output: distribution of time to next incident

- Online select an **efficient subset of benchmarks**
  - A subset of benchmarks with incident coverage C could decrease incident probability from p to $p \times (1 - C)$
  - Find a subset such that $p \times (1 - C) \leq p_0$ while minimize total benchmark time
  - Greedily select benchmarks with maximum $\frac{\Delta p}{time}$ in each iteration

The Anatomy of the SuperBench System



Evaluation

Evaluation on Benchmark Selection

Setup

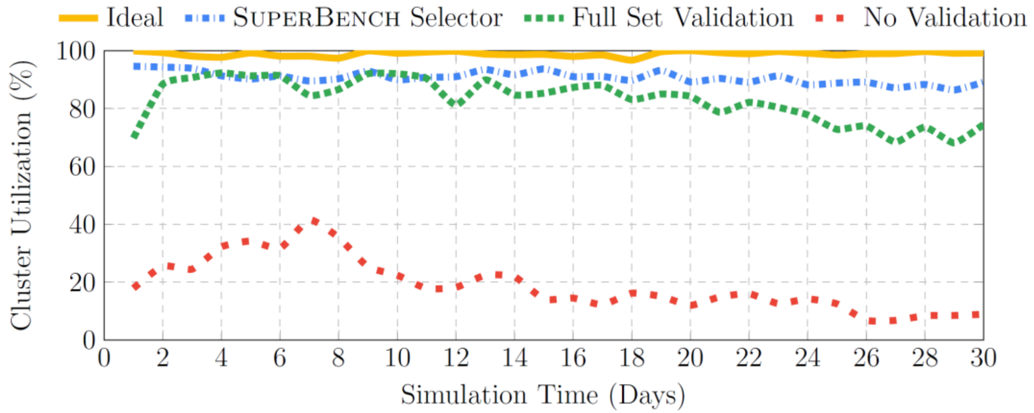
- Node Incident Trace
  - 4-month incident events
  - Internal clusters with 8k GPUs
  - Used to fit probability model

- Benchmark Results Dataset
  - 24 validation benchmarks in full set
  - 3k+ A100 VMs, 2,441 metrics per VM
  - Used to label defective nodes and calculate coverage for benchmark set

Results

Compared to full set validation,

- **9%** cluster utilization improvement, **381%** compared to no validation.
- **92.07%** validation time reduction.
- **11%** improvement on mean time between incidents.



Simulated avg. node util. with different selection policies

Evaluation on Benchmark Criteria

Setup

Run validation on internal GPU clusters:

- 1,152x AMD MI250X GPUs
- 512x NVIDIA H100 GPUs

Define *margin ratio* as metric:

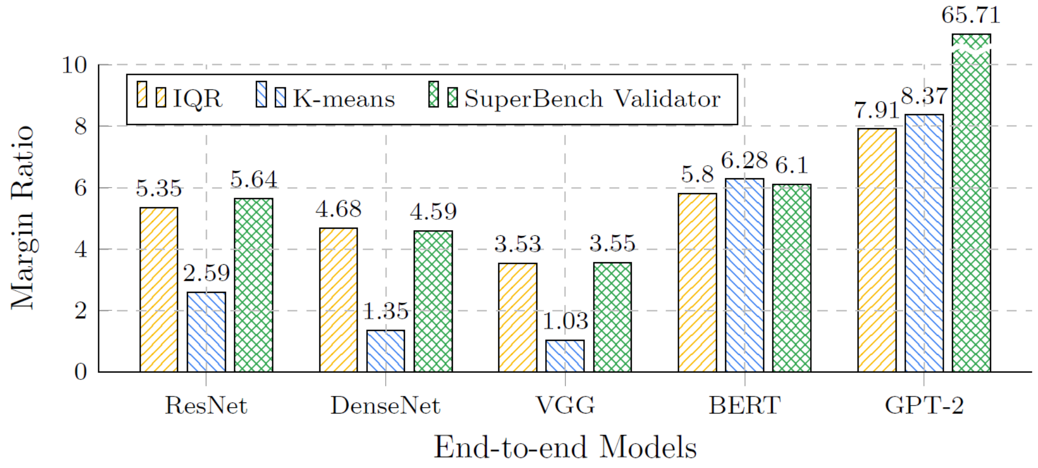
$$\frac{\min(1 - similarity(S_{defective}, S_C))}{\max(1 - similarity(S_{healthy}, S_C))}$$

Validation results are used to evaluate benchmark criteria on margin ratio metric.

Results

Compared to two baseline methods:

- Up to **7.31x** better margin ratios than IQR
- Up to **6.85x** better margin ratios than K-means



Margin ratios of different criteria methods

Evaluation on Cloud Deployment

Setup

- Validation in cluster build-out phase
- Over 24k+ A100 GPUs (3k+ VMs)
- Collect results in 90 days

Evaluate effectiveness in defective GPU node filtering.

Results

Filtered **10.36%** nodes as defects in total.

Validation Benchmarks	# Defects / # Total
IB HCA loopback	6.04%
H2D/D2H bandwidth	2.03%
BERT models	1.59%
CPU latency	1.33%
IB single-node all-reduce	1.10%
ResNet models	0.73%
GPT models	0.53%
LSTM models	0.46%
DenseNet models	0.40%
MatMul/all-reduce overlap	0.33%
NVLink all-reduce	0.30%
GPU GEMM	0.23%

Conclusion

- Reliability is crucial for cutting-edge AI infrastructure.
- However, **reactive troubleshooting** can surprisingly compromise the reliability due to **redundancies**.
- SuperBench is a **proactive validation system** for AI infrastructure to improve reliability.

Thank you!

Q & A