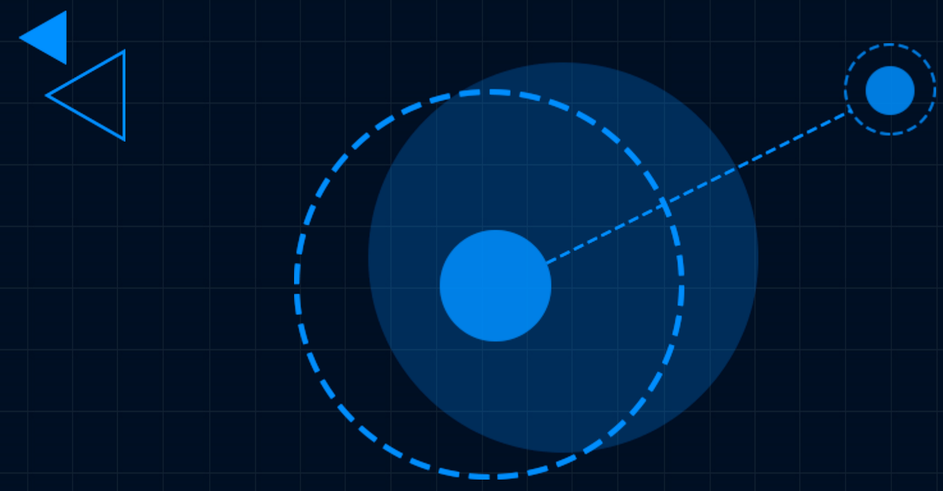
基于Kubernetes的大规模机器学习平台

Henry Zeng

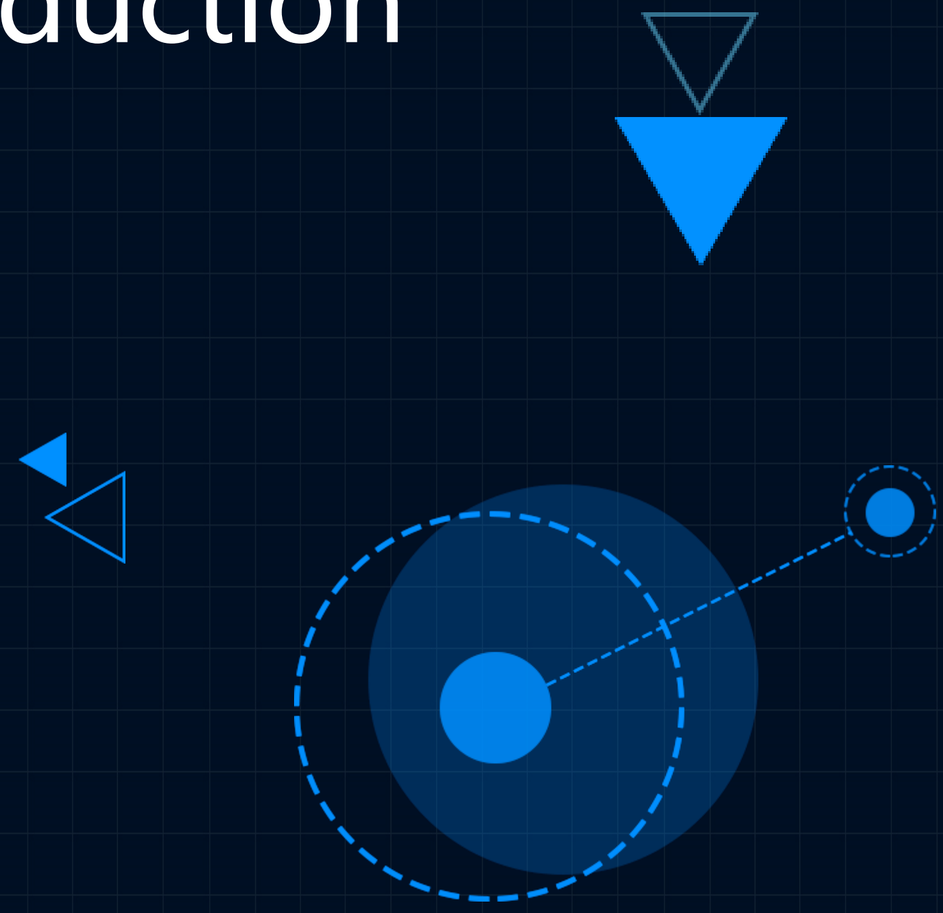
Principal Program Manager,
Azure Office of CTO



Agenda

- ❑Kubernetes ML Platform Introduction
- ❑Challenges and Solutions
- ❑Looking Forward
- ❑Azure Arc enabled Machine Learning

K8s ML Platform Introduction



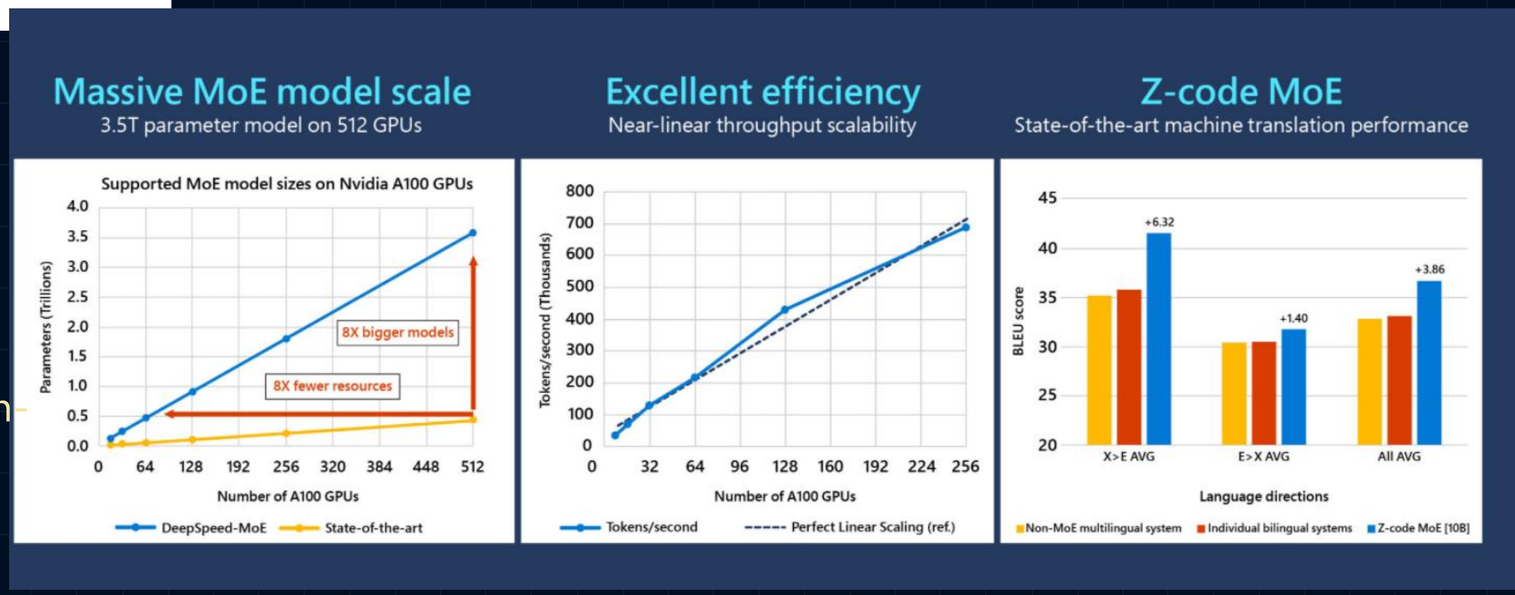
The Platform Train SOTA Model

Rank	Model	Participant	Affiliation	Attempt Date	Avg	Sentence-pair Classification	Structured Prediction	Question Answering	Sentence Retrieval
0		Human	-	-	93.3	95.1	97.0	87.8	-
1	Turing ULR v5	Alexander v-team	Microsoft	Sep 17, 2021	83.7	90.0	81.4	74.3	93.7
2	VECO	DAMO AliceMind Team	Alibaba	Sep 21, 2021	82.0	89.0	76.7	73.4	93.3
3	CoFe	HFL	iFLYTEK	Sep 21, 2021	81.9	88.4	75.7	73.9	93.5
4	Polyglot	MLNLC	ByteDance	Apr 29, 2021	81.7	88.3	80.6	71.9	90.8
5	Unicoder + ZCode	MSRA + Cognition	Microsoft	Apr 26, 2021	81.6	88.4	76.2	72.5	93.7
6	ERNIE-M	ERNIE Team	Baidu	Jan 1, 2021	80.9	87.9	75.6	72.3	91.9
7	HiCTL	DAMO MT Team	Alibaba	Mar 21, 2021	80.8	89.0	74.4	71.9	92.6
8	T-ULRv2 + StableTune	Turing	Microsoft	Oct 7, 2020	80.7	88.8	75.4	72.9	89.3
9	Anonymous3	Anonymous3	Anonymous3	Jan 3, 2021	79.9	88.2	74.6	71.7	89.0

Figure 1: XTREME leaderboard showing T-ULRv5 at the top.

T-ULRv5 shares a similar transformer architecture that's popular among the emerging foundation models and multilingual models like mBERT, mT5, XLM-R, and the previous version, T-ULRv2. Specifically, T-ULRv5 XL, the largest variant we pretrained, has 48 transformer layers, a hidden dimension size of 1,536, 24 attention heads, a 500,000-token multilingual vocabulary size, and a total parameter count of 2.2 billion.

DeepSpeed MoE overcomes these challenges through a symphony of multidimensional parallelism and heterogenous memory technologies, such as **Zero Redundancy Optimizer (ZeRO)** and **ZeRO-Offload**, harmoniously coming together to support massive MoE models—even on limited GPU resources—achieving efficiency, scalability, and ease-of-use. It enables 3.5 trillion parameter models on 512 GPUs, 8x larger than existing work



K8s Large Scale ML Platform

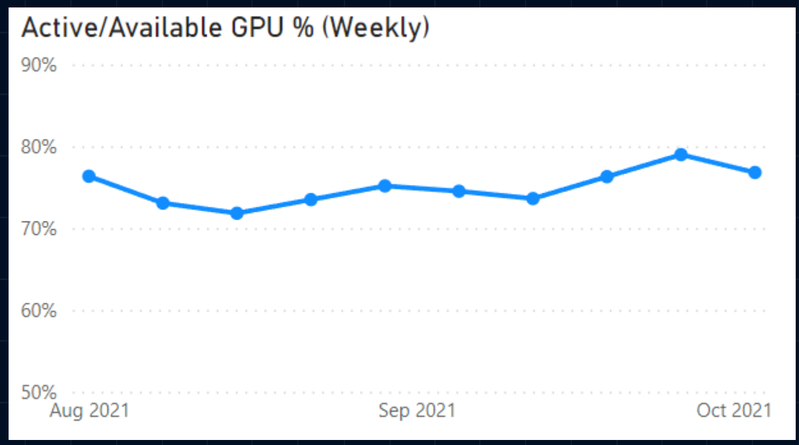
ML Workloads
Batch Training, Interactive Dev., Inference,
Data Process

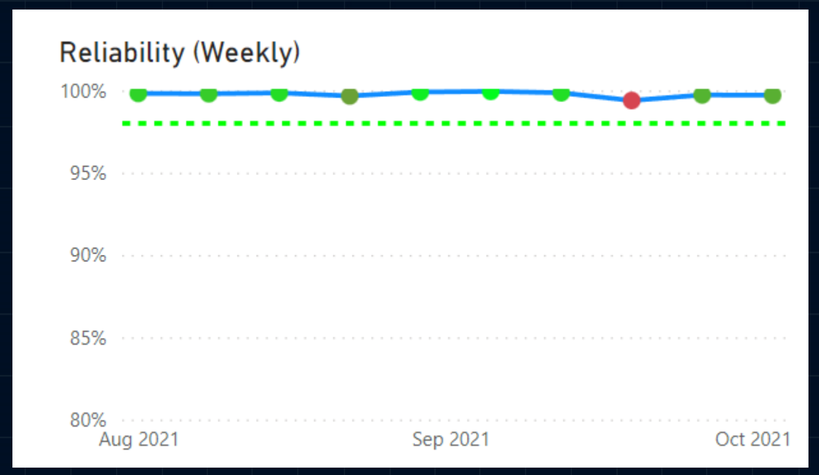
Experience
Portal, SDK, CLI, RESTAPI, Customized Lib

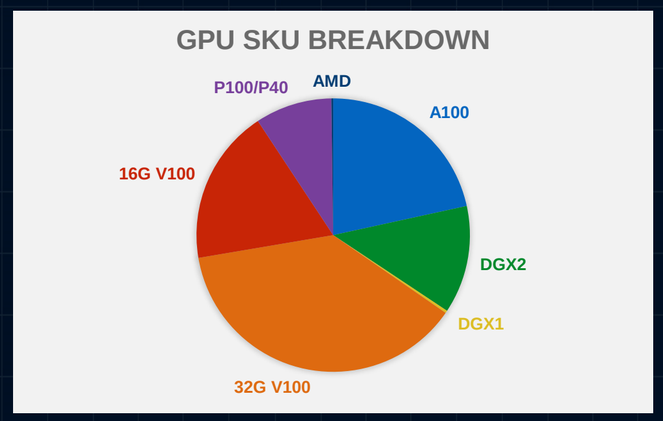
Spot-VM Support
Global Scheduler
Topology Awareness
High Perf Data Cache
Large Scale Training
Hybrid Training
Reporting
Policies
GPU/Job/WL Insights
Elastic Training

Kubernetes Clusters

Azure, Lab
(Huge mount of GPUs, many K8s clusters)
P40/P100/V100/DGX1/DGX2/A100/AMD…







Persona and Key Requirements

Data Scientist/ML Engineering

- Flexible ways of job submission, mgmt. and insights (Portal, SDK, RESTAPI, and Customized tools)
- Submit batch jobs for model training / interactive jobs for modeling building (SSH, iPython, Tensorboard, VSCode)
- Elastic training/inference based on available GPUs
- **Run large job with reliability**

Management/Admins

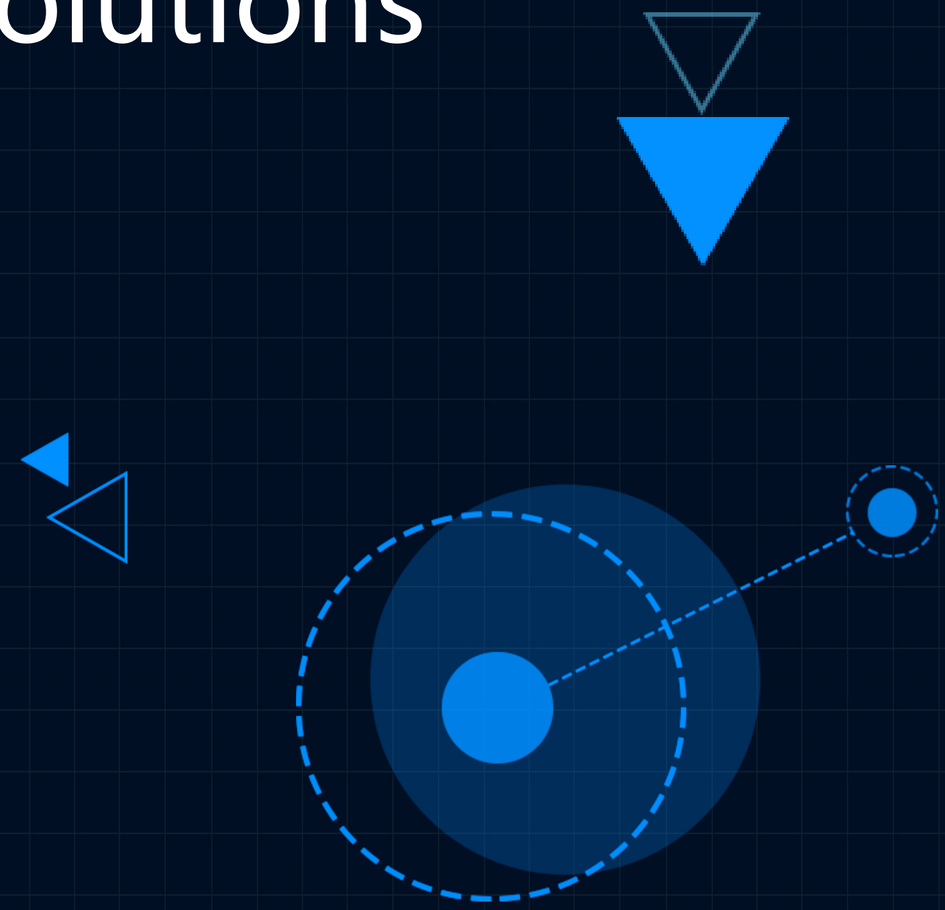
- **Drive GPU utilization of invested GPUs**
- Fairness of GPU resource among teams while important job can get priority
- View jobs and adjust priority, pause/resume/kill jobs
- Reports on GPU/user/jobs
- Self-service cluster policy and settings configuration

Ops

- Auto provision, setup and upgrade
- Observability and monitoring
- Self-healing of service and bad nodes
- HA, performance and scalability

The Challenges and Solutions

- GPU Node Stability
- Large Job Reliability
- Drive GPU Utilization



GPU Node Stability

The Challenges

- Expectation -- User can get 100% healthy allocated GPUs quota to run jobs with reliability
- Reality -- User job hang, failed or restarted due to bad nodes, IB issues, storage issue and other issues

Root Causes

GPU node is more fragile compared with CPU nodes in terms of GPU, IB, NvLink etc

64RowRemappingError	302
94ContainedECCError	108
NVLINKErr	80
NCCLTestIBChecker	33
79FallenOffBus	15

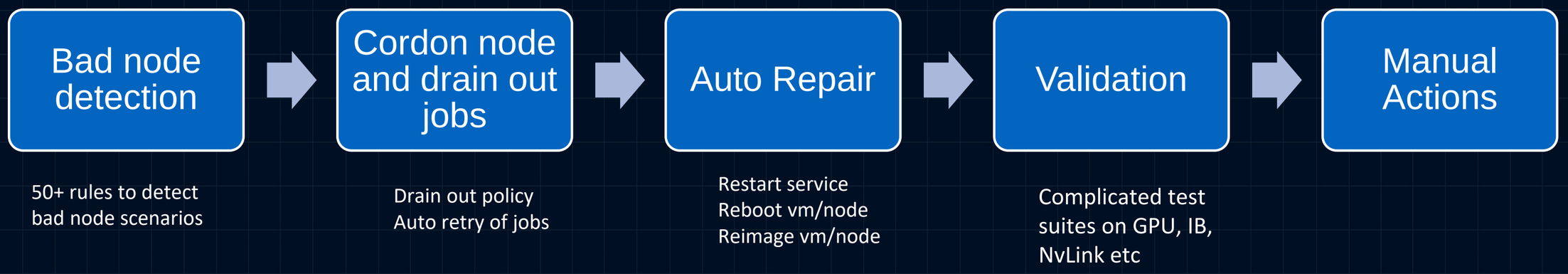
Solutions

- Automated process to detect and repair bad nodes
- User best practices to tolerate with bad nodes
- Bad node insights
- Elastic training

reason	Total
94ContainedECCError	122
NVLINKErr	105
95UncontainedECCError	73
79FallenOffBus	27
31GPUMemPageFault	13

Metrics

- Detection precision/recall and Time to Pool
- Job reliability

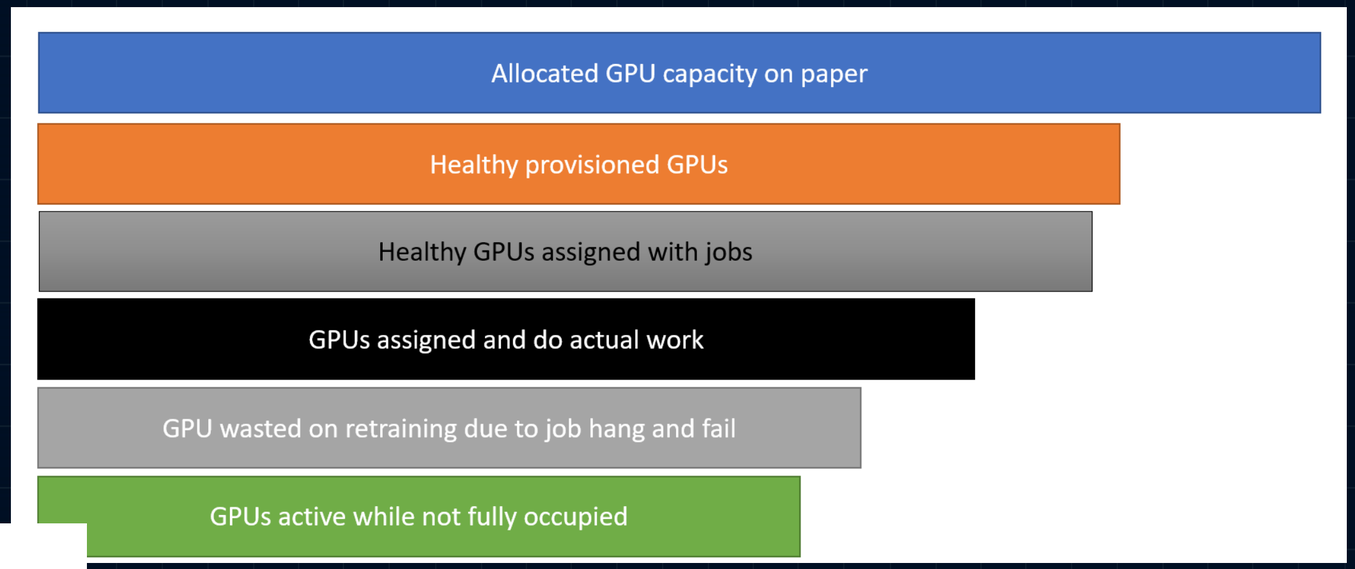


GPU Utilization

Why GPU utilization

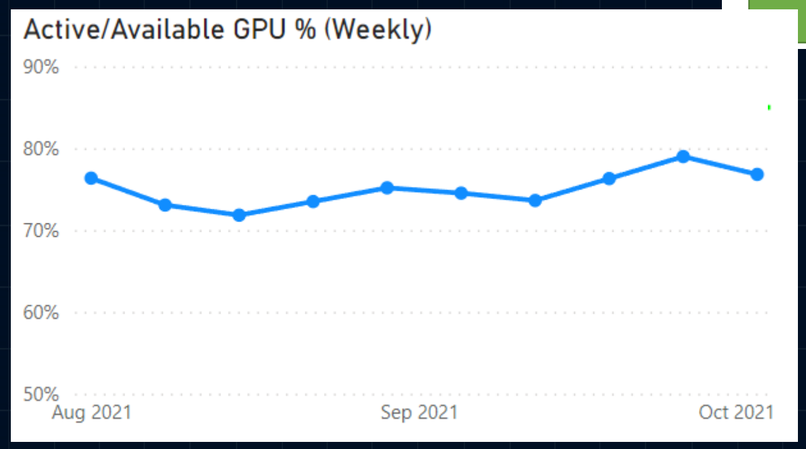
GPU hour is $$

Breakdown of GPU utilization



Metrics

- Healthy Ratio
- Assigned Ratio
- Idle Ratio



Solutions

How we drive unassigned ratio

- **Global Scheduler** to rebalance the workloads that allows any user to use underutilized GPU capacities in the pool.
- **Job Redispatch** to on-the-flight dispatch a queued job to another cluster with available GPUs.
- **Speed up** new cluster workload onboarding with blob and GJD
- **Elastic Training** to flexibly scale out/in the job according to cluster usage
- Improved cluster/node **stability** to build confidence for user to submit jobs

How we drive idle ratio

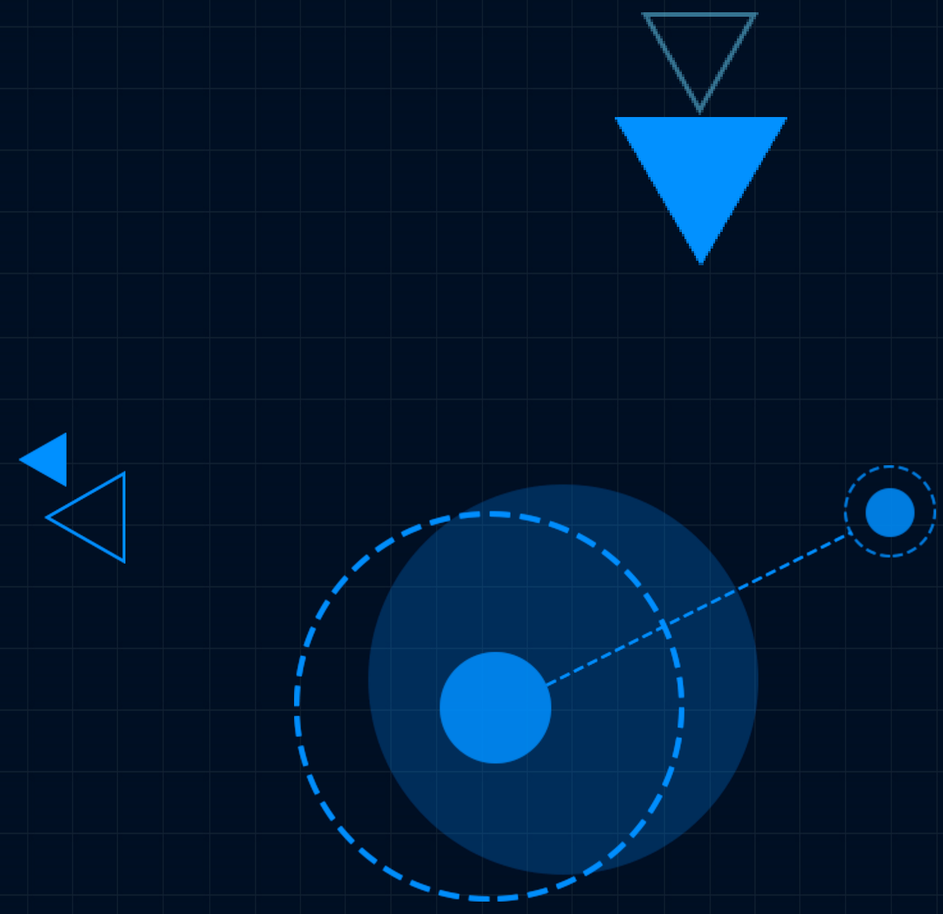
- **Idle Policy** to automatically kill the job being idle for long.
- **High Perf Data Cache** to reduce GPU waiting for IO
- **Job Reproducibility** to help recover the user job from system error, pre-orchestrate multiple tasks to reduce gaps
- **Workload Insights** to recommend idle GPU and handle imbalance

Large Job Reliability

- Expectation-- run large job for a long time (weeks) without failure/hang
- Reality -- can not run large job (init failure), job hang/fail after hours/days
  - The larger the job size, the higher probability of failure/hang, the higher demanding on node IB traffic, storage IO, ACR etc
  - Restart of a large job is time consuming – scheduling, setting up env, resume chkpts, load data, wasted GPU hours of since last ckpt, and wasted GPU hours for other healthy nodes
- Root Causes
  - Bad GPUs/IB/NvLinks etc
  - User workloads settings (not resilient with failures, not using latest image/lib or env variables)
  - External dependency such as storage, container registry etc
- Solutions

Looking Forward

What can be done better



Candidate Topics for Next Step

GPU utilization

- Node performance-based scheduling
- Topology awareness scheduling (IB, NvLink level)
- Cross region job re-dispatch
- Hybrid(CPU/GPU) scheduler based on usage
- Fairness of bad node distribution
- De-fragmentation of GPUs

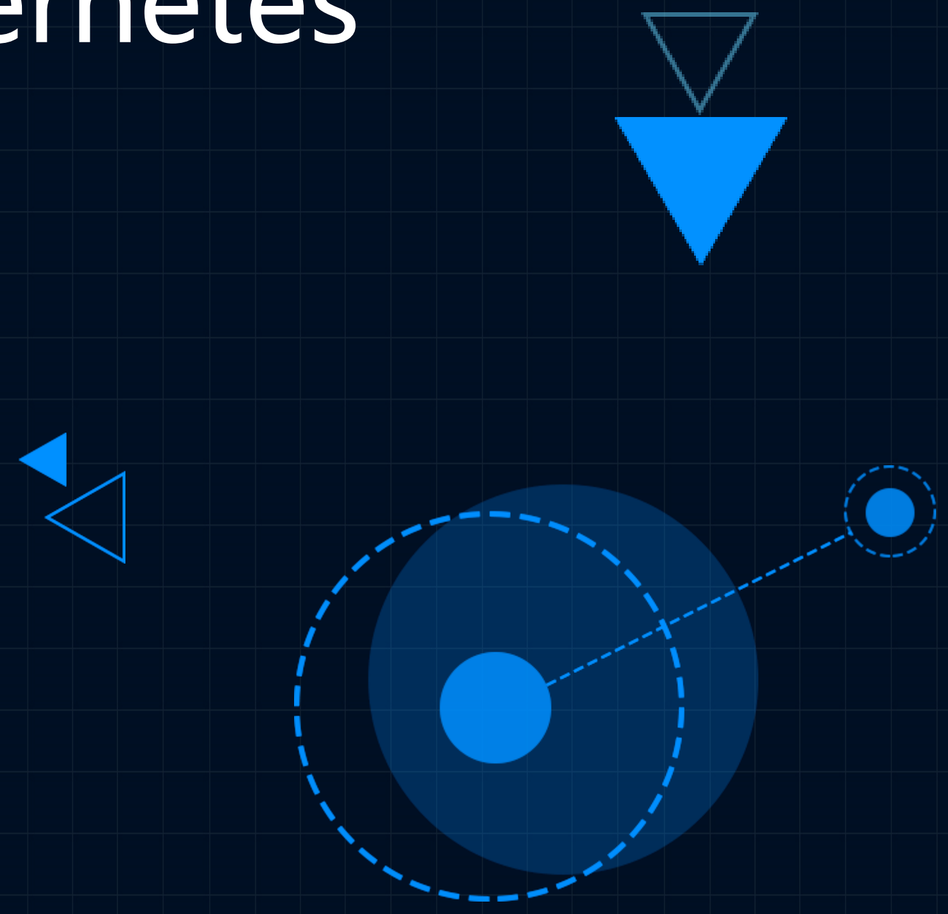
Bad nodes

- Bad node insights
- More accurate rules and streamline repair process
- Intelligent job drain out on bad nodes

Large job stability and performance

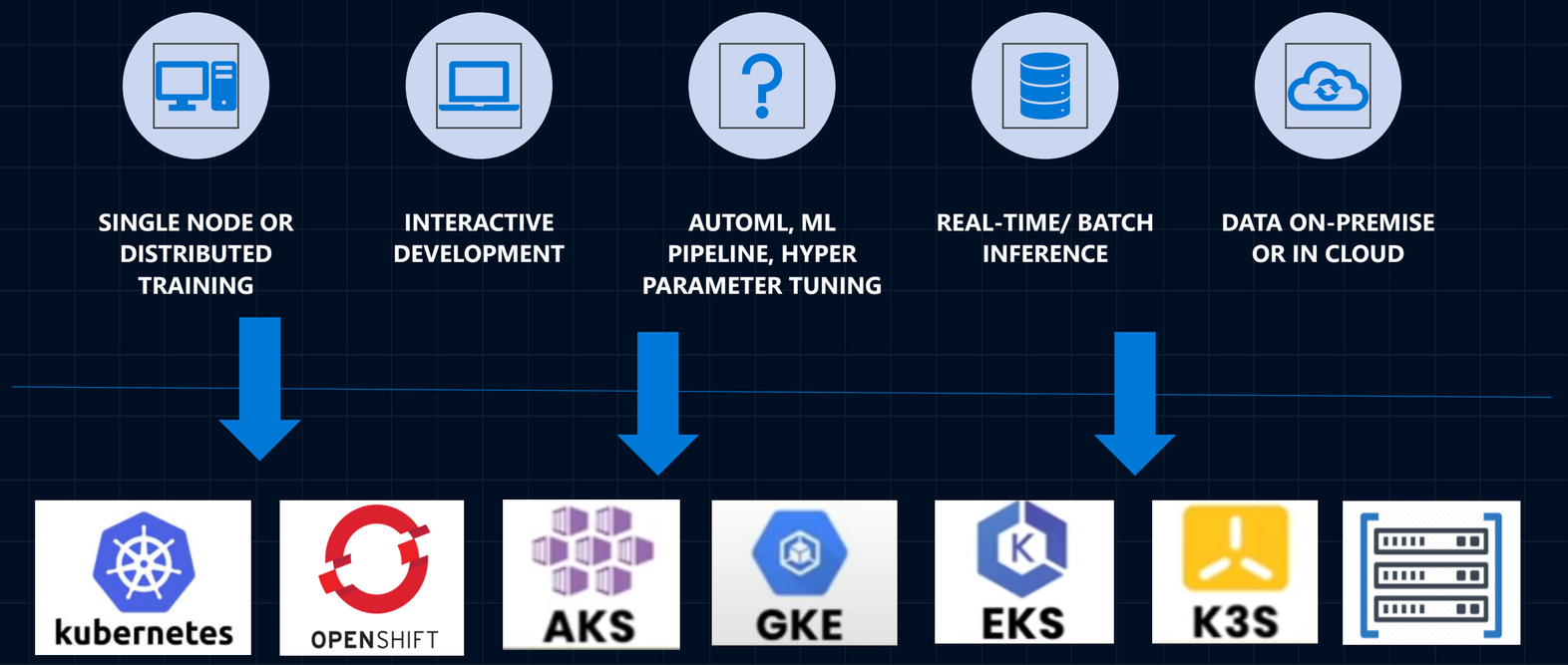
- High perf checkpointing
- Transparent elasticity
- Workload optimizer and recommender
- GPU Direct Storage (GDS) access
- Intelligent Failure Analyzer (IFA)
- Policy recommender

AzureML Arc for Kubernetes

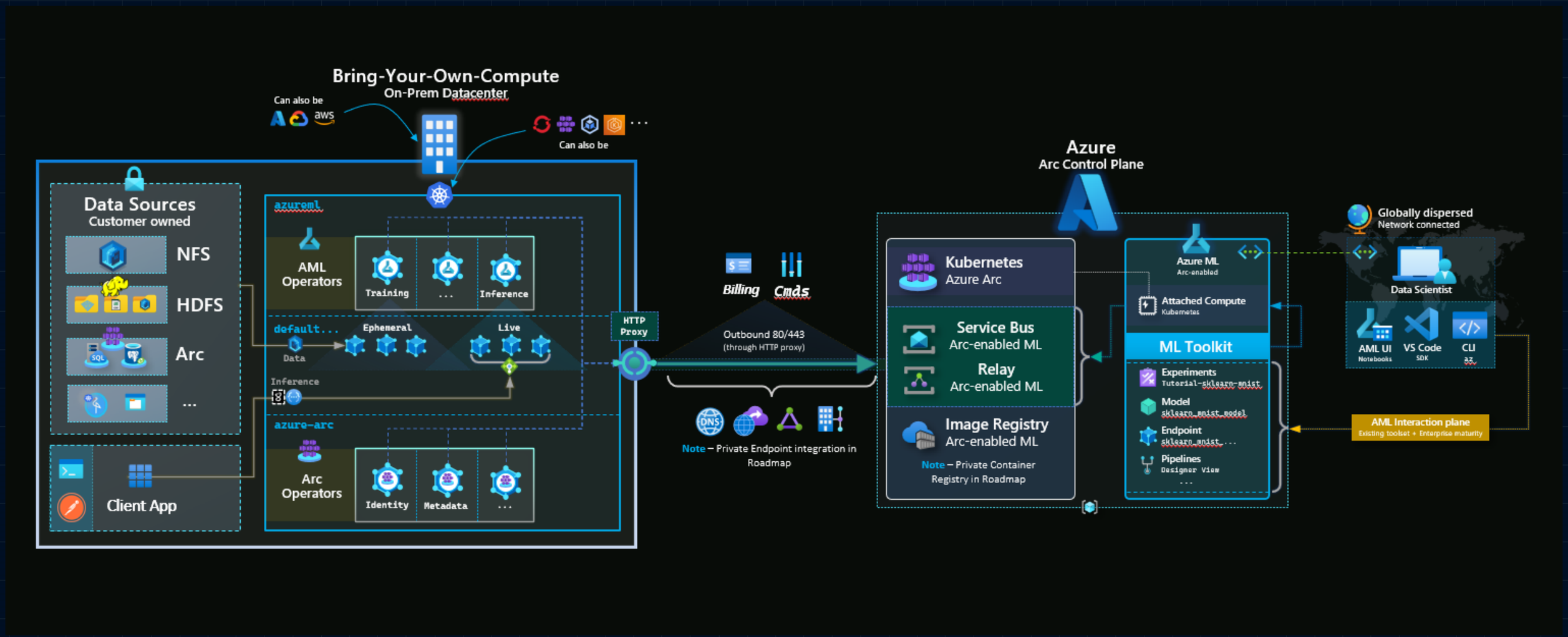


What is Azure Arc enabled ML (AMLArc)

- Bring your own Kubernetes compute anywhere in cloud or on premises to AzureML
- A unique compute target in AzureML for both training and inference
- Run any ML workload with seamless AzureML experience



AMLArc Deployment Reference Architecture



Try with Azure Arc Enabled ML

- Blog: https://aka.ms/amlarc/blog
- GitHub Repo: https://github.com/Azure/AML-Kubernetes
- Official documentation: https://aka.ms/amlarc/doc

Thanks and QA