Shared with dev@kubernetes.io for commenter
Shared with wg-serving@kubernetes.io for editing

Goals

This is a working doc that aims to:

- Track docs and proposals shared with the community for each workstream.
- Help identify problems we can potentially solve collectively and help organize our workstreams and exploration around these problems.

Docs and Proposals

Orchestration

- Blueprint API [Public] Blueprint Is All You Need proposes new Kubernetes workload APIs for deploying inference workloads, with a focus on LLMs and GenAI.
- [Public] KServe vs. Blueprint outlines the differences between Blueprint and KServe API, identifies gaps, provides the KServe roadmap for filling the gap by extending KServe.
- Serving Catalog
  - Proposal: [Public] K8s LLM Serving Catalog
  - Proposed structure: [Public] Kustomized Blueprints - Serving Catalog
- LLM Gateway
  - KServe's proposal Cloud Native LLM Gateway : higher level gateway for routing among LLM providers
  - [PUBLIC] Dense LLM Serving (+LoRA) for Inference Platform Teams : lower level gateway to route in clusters among LoRA adapters for LLMs; build a scheduler plugin into the envoy proxy for routing LLM requests. POC design: [PUBLIC] Kubernetes LLM Instance Gateway PoC Design
- [PUBLIC] Kubernetes: Disrupted pods should be eagerly removed from endpoints
- [External][CNCF WG-Serve] Ray Serve on Kubernetes

Multi-host/multi-node

- LeaderWorkerSet (LWS): repo, proposal
- KServe multi-node support proposal with focus on API
- [public] Network requirement for distributed and disaggregated inference

Autoscaling

- [External] Standardizing Large Model Server Metrics in Kubernetes
- KEP: OCI VolumeSource
- [public] KV-aware LLM Autoscaling and evaluation metrics
- Benchmarking Workloads for Performance Evaluation and Autoscaling in Kubernetes
- [PUBLIC] Kubernetes LLM Inference Autoscaling Examples
- KServe
    - KServe model cache proposal based on PVC [WIP] Model Cache Proposal
    - KServe OCI-based ModelCar feature
      Enhancing KServe Model Fetching with Modelcars
    - Integration of KEDA in KServe was proposed to provide more granular and metric-based auto scaling for inference services
- [PUBLIC] Model Deployment Patterns on k8s

DRA

- KEP-4680: Add Resource Health Status to the Pod Status for Device Plugin and DRA
- Revised DRA API and features for v1.31 K8s (PR)

Potential Problems to Address

What problems are in scope?

- Can be a project in the ecosystem and support multiple use cases well
- Standardization of common approaches - “commoditize a hard / annoying problem”
- Requires a change in core Kubernetes to accomplish

Orchestration

- Different LLM use cases cannot run concurrently on accelerators and swap easily, limiting density for multiple workloads
    - Share models fairly between different use cases via load balancing in the llm-instance-gateway -
      [PUBLIC] Dense LLM Serving (+LoRA) for Inference Platform Teams
        - Rolling out new versions of LLM models is painful / slow under capacity constraints due to not having enough nodes to reschedule existing service capacity
        - **Hard to define service requirements**

        - Service owners need to select hardware for their workload from what platform team offers (in terms of nodes), but the current mechanisms are either too low level or it's not clear what level we should be at
- It is hard to understand what parameterization and values do GenAI workloads vary on?
    - What are the key points of similarity and variation between people's production GenAI workloads that Kubernetes can more effectively support
        - Every Kubernetes deployment has a different way to define accelerator consumption beyond the basic accelerator request, but we anticipate GenAI workloads to be able to use multiple accelerator types
    - Create a catalog of recommended serving topologies and identify the variations (cloud, accelerator type, topology, etc)
- Hard to achieve density between serving and batch workloads because serving workloads don't have an obvious signal for how much capacity they can shed safely and Kueue needs more context from the serving workload
    - Can we directly model how much excess capacity workloads need (vs directly encoding it as a constant in HPA) so that the excess capacity can be disrupted first
- Sidecars that are intended to reconfigure the primary container without disruption, must disrupt the main workload in order to be updated (beyond changing the image)

Multi-host/multi-node

- Serving a large model requires multiple hosts which is poorly supported by orchestration tools
    - LeaderWorkerSet to support workloads needing replicas that are composed of multiple pods each, where each replica must be fully created
        - Can KServe leverage LWS or are there obstacles?
        - Are there other possible users of LWS in the stateful workload domain whose participation would accelerate adoption and guide feature design
- Deploying a disaggregated model server configuration is hard today and should be relatively common for large model serving in the future
- Intra-device communication https://github.com/efeslab/Nanoflow is a very promising work that SGLang community is heavily investing on
- Recent updates on disaggregate features. 09/04
    - TP & PP same parallelism between prefill & decode machines (kv transfer etc)

Autoscaling

- Autoscaling on device utilization / memory is not sufficient for production workloads, and it is challenging to identify and configure HPA to autoscale on model server metrics
    - [External] Standardizing Large Model Server Metrics in Kubernetes Attempt to standardize model server metrics
- Starting an accelerated workload is usually slow, discouraging autoscaling

    - ML Artifacts are even larger than most containers, but ML containers are also large
        - Reducing the amount of content that must be brought onto a node to start a pod
            - OCI image volumes
            - Model caching infrastructure
        - Starting image pull early
        - Placeholder pods with “placeholder replacement”
    - Can we directly model how much excess capacity workloads need (vs directly encoding it as a constant in HPA) so that startup is faster?
- Hard to benchmark how latency, throughput, and workload sharing interact with autoscaling so that deployers can achieve a target latency and contrast model servers and config settings for optimal performance
    - Can we streamline benchmarking of some/most GenAI workloads?

DRA

- **Hard to react to changing dynamics of traffic and capacity**
    - The dynamic nature of capacity in cloud, changing workload demands (traffic / load) over the day, and planning for adequate capacity to achieve workload goals is very complex - all of these are exacerbated by current accelerator nature
    - What do we need to define
    - A workload autoscaler can’t introspect the types and rough amounts of capacity that could become available and steer the type of pods it creates based on that (i.e. knowing there are A100’s available vs not may alter what label selectors you want to put on a pod)
- Topology aware placement to ensure high bandwidth network interconnect between the appropriate components of a replica (multi-host) or between services (disaggregated serving prefill and decode)
    - Currently this is handled on a cluster by cluster use cases

Instrumentation

- Inconsistency in operational metrics among standard model servers
    - When kube was introduced there was no broad industry “golden signals” consensus for microservices, which co-developed with and around kube and broader microservice ecosystem
    - [External] Standardizing Large Model Server Metrics in Kubernetes Attempt to standardize model server metrics
- **Hard to measure density** - Hardware utilization metrics are insufficient to help platform owners achieve cost density

- At cluster level, scheduler goodput is “are all provisioned resources assigned to a workload” (pod -> using it vs on a node and unused)
- No workload level view for utilization to manage cost density - i.e. for inference we might call this “inference goodput” where it’s a measure of how much room for improvement there in making the workload more dense