WG-Serving Use-Cases and Requirements

Note: members of dev@kubernetes.io have commenter privileges.
Author: Clayton Coleman
Status: Draft, converting seed ideas to formal list

Will be updated to contain formal requirements to target

Proposed List

A few use cases we are suggesting based on feedback so far. This list is open - please comment!

- There are a set of well known “standard model server configurations” that are fairly common across inference users, but advanced users might want to customize
    - The kserve ServingClass concept and kaito’s “preset” is reasonably aligned to that sort of standardization, and something multiple projects in the ecosystem might be able to leverage to simplify orchestrating and scaling model servers, and could also potentially support kserve’s RawDeployment mode in Kube? - discussed with Yuan Tang at KubeconEU
    - Make it easier for admins to upgrade model servers / models safely across many users (especially with A/B testing well integrated with the Gateway API)
- Some of the largest models may need more accelerators than a single host can present, and there is a need for ways to represent workloads where a single replica spreads across multiple hosts (e.g. LeaderWorkerSet) [sig-apps, sig-node, sig-scheduling]
    - A need for restart of all of the replicas within a single group if any fails, due to frameworks like MPI not being resilient to config membership changes - from Abdullah Gharaibeh
    - Because multi-host inference is often implemented using tools like MPI where specific indices for each worker must be known a priori, StatefulSets are more useful than Deployments, and some longstanding pain points with using indices in StatefulSets should be addressed
    - Multi-host inference requires scheduling that ensures all pods in a replica land in the same high speed network - from Ken Owens [sig-network]
    - Inference engines may introduce multiple roles that traditional rank based solutions (MPI) would not be able to orchestrate easily
- Autoscaling and load balancing accelerated workloads is very important for managing cost, but needs more than what HPA does out of the box today, and is slow [sig-network, sig-autoscaling, sig-instrumentation]
    - Better support for custom metrics in HPA and potentially first class accelerator utilization metrics

    - Inference has a different tradeoffs than web-applications between latency and throughput: sending more traffic to a busy inference server can increase throughput (and decrease cost) for a proportionally smaller latency hit. By understanding the latency objective and adding traffic shaping and smart autoscaling, we can potentially achieve optimal efficiency for a given workload automatically.
    - Due to the drawback of the DCGM metrics definition (e.g.GPU Utilization represents the fraction of time the GPU is not idle, but idle means at least one SM is active, it doesn't consider spatial dimension), so it's hard to find the linear relationship between resource metrics like SM_ACTIVE and replicas which makes the HPA estimation inaccurate, leading to more oscillations and SLO violations. - from Jiaxin Shan
- Running smaller pre-production serving workloads is hard relative to batch and typical web apps
    - Accelerated dev/test/prototype serving workloads aren't quite interruptible, but don't run forever and need to scale to zero when unused - which might benefit other mostly idle workloads [sig-scheduling, wg-batch]
    - It's hard to configure accelerators for sharing and there are tradeoffs for different types of sharing, which workloads might be able to represent in a standard way [sig-node, sig-scheduling]
- Large accelerated workloads are more vulnerable to disruption, slower to start, and need better primitives for mitigating disruption (with limited capacity)
    - Because keeping extra replicas around is expensive, inference might prefer to explicitly state how much safety margin the workload needs vs having to calculate it into the HPA scaling metric [sig-autoscaling]
    - Detecting and reacting to accelerator failures from both workloads and infrastructure
    - Make it easier for inference workloads to preempt training workloads on the same cluster for planned and unplanned disruption and later resumed - from Ken Owens
    - What can we do to reduce fragmentation in the scheduling domain for people who want to achieve high utilization across both batch and serving - is there a need to improve the base scheduler even more? - from Ken Owens, and others at KubeCon

Reading through the nVidia use cases for GPU that are helping to guide the DRA design, we also see additional opportunities to streamline or simplify those use cases for workloads so that users are mostly unaware of the complexity of the resource model. One example is:

- Accelerated inference workloads can easily indicate that their workload can run on 1 or 2 accelerators of two different types (A100 / H100), so that an administrator can provide different accelerators for cost or burst to new workload types