Efficient Deep Learning Inferencing in the Cloud using Kubernetes with Smart Provisioning of Arm Nodes

Cloud_Native Rejekts [NA'22]

Deep Learning (DL) models are being successfully applied in a variety of fields. Managing DL inferencing for diverse models presents cost and operational complexity challenges. The resource requirements for serving a DL model depend on its architecture, and its prediction load can vary over time, leading to the need for flexible resource allocation to avoid provisioning for the maximum amount of resources needed at peak load. Using the cloud to allocate resources flexibly adds operational complexity to obtain minimum-cost resources matching model needs from the large and ever-evolving sets of instance types. Selecting minimum-cost cloud resources is particularly important given the high cost of x86+GPU compute instances, which are often used to serve DL models.

This talk describes an approach to efficient DL inferencing on cloud Kubernetes (K8s) cluster resources. The approach combines two kinds of right-sizing. The first is right-sizing the inference resources, using Elotl Luna smart node provisioner to add right-sized compute to cloud K8s clusters when needed and remove it when not. The second is right-sizing the inference compute type, using cloud Ampere A1 Arm compute with the Ampere Optimized AI library, which can provide a price-performance advantage on DL inferencing relative to GPUs and to other CPUs.

The talk shows the benefits of the approach using inference workloads running on auto-scaled TorchServe deployments. For cloud K8s clusters from two vendors, the cost and operational complexity of right-sizing is compared against two non-right-sized approaches.

Anne Holler
Advisor
Elotl

Cloud_Native
Rejekts [NA'22]

Efficient Deep Learning Inferencing in the Cloud using Kubernetes with Smart Provisioning of Arm Nodes

Anne Holler, Advisor, Elotl
Twitter: @holler_anne

Deep Learning Inferencing at Scale: Resourcing Challenges

Deep Learning (DL) models successfully applied in many fields

- Including image and natural language processing

However, managing DL inferencing at scale, often run in the cloud for resource flexibility, presents challenges

- Cost Challenges
  - DL models differ greatly in serving resources needed, particularly wrt system memory to store weights
    - Economical to use cloud **instance shapes that are right-sized** for models
  - In production, a model's prediction load can vary significantly according to time-of-day and other factors
    - Desirable to automatically adjust the number of cloud serving **instances to match current load**
  - x86+GPU compute instances, often used to serve DL models, are costly compared to CPU-only instances
    - Worthwhile to evaluate if an inference use case can be handled well on **thriftier CPU-only instances**

- Operational Complexity Challenges
  - **Selecting currently-available minimum-cost cloud resources**
    - that match inferencing needs, which can change with model updates
    - from large and ever-evolving instance types, whose availability may vary over time

Deep Learning Inferencing at Scale: Talk Overview

An approach to managing the cost & operational complexity of DL inferencing at scale: Given cloud resources, organized in a Kubernetes (K8s) cluster for production container orchestration, the approach combines:

- **Right-sizing inference resources**
  - Use Elotl Luna smart node provisioner
  - Adds right-sized compute to cloud K8s cluster when needed and removes it when not

- **Right-sizing inference compute type**
  - Use Ampere A1 Arm compute along with Ampere Optimized AI library instance type
  - Can provide a price-performance advantage on DL inferencing relative to GPUs & other CPUs for low-latency low-batch-size use cases.

We show the benefits of the approach by comparing its cost & operational complexity against non-right-sized resource provisioning together with non-right-sized compute type (x86+GPU)

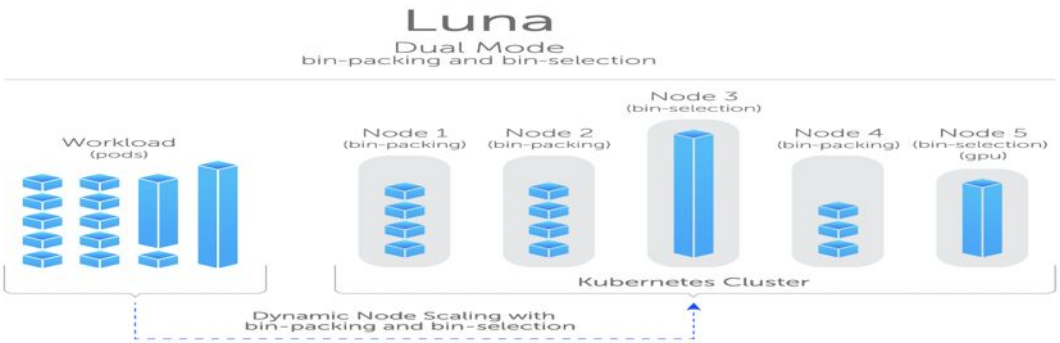
- Experimental runs were performed on an Oracle Cloud (OCI) OKE K8s cluster and on a Google Cloud (GCP) GKE K8s cluster

Right-Sizing Inference Resources

- Right-sizing inference resources leverages Elotl Luna
  - Luna adds nodes to cloud K8s for pending placements according to **bin-packing** vs **bin-selection** policy
  - Luna is suitable for handling **any bursty workloads** running on cloud K8s

- Previously reported on Luna's mgmt of x86+GPU compute for DL training workloads
  - Illustrated by validating Ludwig DL AutoML running on Ray Tune for tabular & text classification datasets

- Simple to adapt Luna to managing right-sized compute for DL inference workloads
  - Enabled Luna option to **consider Arm instances** for node provisioning in addition to x86 instances
  - Tuned **bin pack vs bin selection policy** to leverage granular shapes available for Ampere A1 compute
  - Added support for running **Luna on Arm**, to allow use of Arm-only K8s clusters, when more efficient



Right-Sizing Inference Compute Type

Right-sizing inference compute uses Ampere A1 Arm w/Ampere Optimized AI library

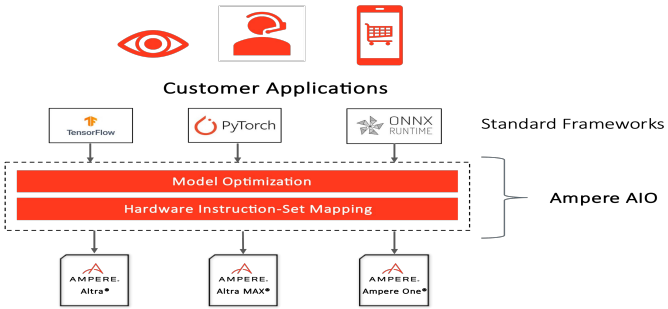
- CPU system tuned for DL inference performance without accuracy degradation.

Latest MLCommons datacenter inference benchmarks show GPUs delivering highest absolute performance for tested scenarios; however, YMMV

- Low-latency small-batch-size cloud server scenarios can run more cost-efficiently on CPU-only systems (see here)

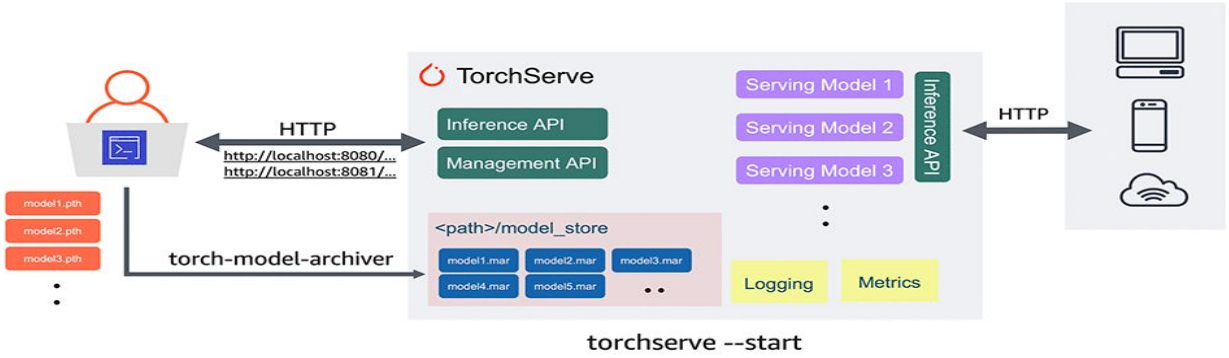
Pre-deployment tests established CPU-only deployment worked well for our use case

- **For right-size runs, no GPU requested; Luna chose Ampere A1 as less expensive than x86 CPU**
- Can be more performant; e.g. Article shows Ampere A1 outperforming x86 CPU shape on resnet_50_v15 model at fp32, same cost delivering >1.5x AMD's best E4 OCI shape & >2x Intel's Standard3 OCI shape



Demonstrating Benefits of Right-Sizing Inference

- Run on tunable DL inference system, deployable on cloud K8s & including autoscaling
  - Chose TorchServe, high-performance full-featured tool for serving PyTorch models
    - Supports tuning resource-related settings for each of the set of models being served
  - TorchServe can be deployed on Cloud-based K8s clusters
    - Description of use on AWS here, which is source of architecture graphic shown below
  - TorchServe includes load-triggered AutoScaling, via a K8s Horizontal Pod Autoscaler
    - HPA changes the number of DL serving replicas behind a TorchServe load-balancing endpoint



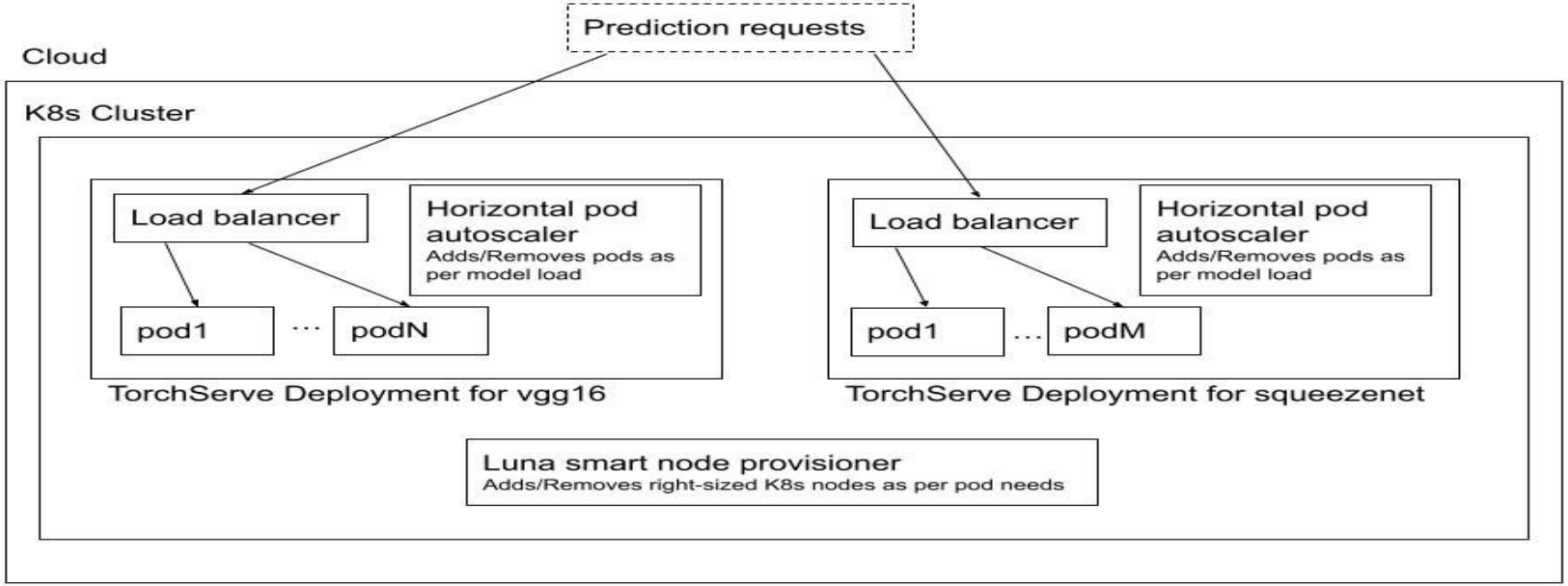
Right-Sizing Inference Expt: Workload Models

- Run workload of serving 2 standard DL models, shown in table below
  - Note large difference in number of weights between models

Model	Task	Weights
vgg16	Image Classification	138M
squeezenet1_1	Image Classification	1.2M

Right-Sizing Inference Expt: Workload Deployment

- Run with each model handled by a separate TorchServe deployment to allow:
  - Each TorchServe worker replica size to be **customized to model size**
  - Each TorchServe worker replica count to **scale for model load independently**



Right-Sizing Inference Expt: Configurations Compared

- We compare the following three configurations
  - **Max-sized configuration**
    - Static inference configuration with maximum-count maximum-sized x86+GPU compute nodes
    - **Operational model: Kubernetes Cluster set up to handle peak inference load for both models**
  - **Dynamic fixed-size configuration**
    - Dynamic inference configuration with variable-count fixed/maximum-sized x86+GPU compute nodes
    - **Operational model: K8s Cluster Autoscaler w/x86+GPU node pool defined**
  - **Right-sized configuration**
    - Dynamic inference configuration with variable-count right-sized Ampere A1 compute nodes
    - **Operational model: Luna K8s smart node provisioner adding/removing right-sized cost-efficient nodes**

Right-Sizing Inference Expt: Load and Scaling Range

- To generate each model's peak load
  - hey loadgen running 150 parallel threads requesting image classification on kitten_small.jpg
- TorchServe is configured for low latency, w/its worker and netty threads set to 1
- TorchServe Horizontal Pod Autoscaler
  - Set to scale up replica count based on CPU utilization >50%
    - Note: CPU utilization trigger works for both CPU-only & GPU-enabled pods
  - Set maxReplicas to maintain 99 percentile end-to-end latency <=~0.5 seconds at our peak load

```
anne@cloudshell:~/serve (elotl-dev)$ curl -X POST http://34.135.144.47:8080/predictions/vgg_16 -T docs/images/kitten_small.jpg
{
  "tabby": 0.530239462852478,
  "Egyptian_cat": 0.23884634673595428,
  "tiger_cat": 0.1343672275543213,
  "lynx": 0.06994840502738953,
  "Persian_cat": 0.009579462930560112
```

OKE Right-Sizing Inference Expt: Computing Resources

- K8s has 2 statically-allocated Ampere Arm CPU nodes for cluster mgmt, including Luna
  - Each is VM.Standard.A1.Flex shape w/2 OCPUs and 32GB [each: $0.0680/hr]
- TorchServe
  - Each model's TorchServe deployment set to create minimum of 1 worker pod
  - Note: Squeezenet1_1 worker pod memory size needed increase when GPU requested

Configuration	Vgg16 worker pod size	Vgg16 node instance type	Vgg16 max worker pod count	Squeezenet1_1 worker pod size	Squeezenet1_1 node instance type	Squeezenet1_1 max worker pod count
Max-sized	400m, 4GB, 1GPU	VM.GPU2.1 (P100 w/16GB)	3	400m, 2GB, 1GPU	VM.GPU2.1	3
Dynamic fixed-size	400m, 4GB, 1GPU	VM.GPU2.1 [$1.275/hr]	3	400m, 2GB, 1GPU	VM.GPU2.1	3
Right-sized	400m, 4GB	VM.Standard.A1. Flex w/1 OCPU, 4GB [$0.0160/hr]	4	400m; 1GB	VM.Standard.A1. Flex w/1 OCPU, 1GB [$0.0115/hr]	4

OKE Right-Sizing Inference Expt: Accuracy & Latency Validation

- Runs on right-sized and x86+GPU nodes produced same prediction results

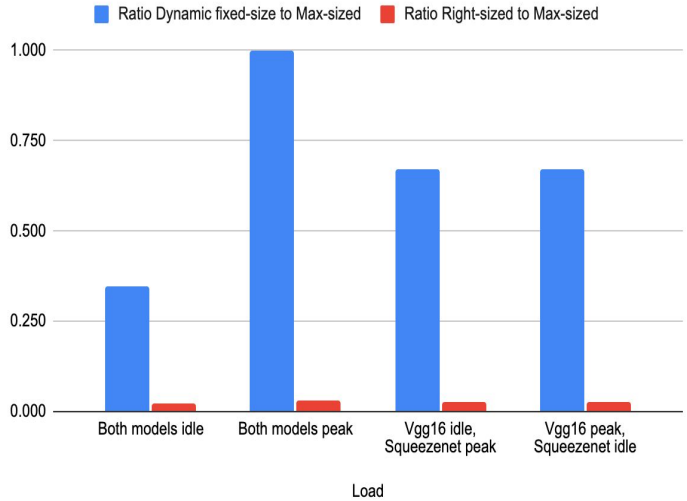
- Table shows 95 & 99 percentile E2E latency for models simultaneously handling peak load
  - Run on right-sized and on x86+GPU configurations
  - Both models meet desired E2E latency target (<=~0.5 secs) for presented workload and set up

Model	Right-sized 95% latency seconds	Right-sized 99% latency seconds	x86+GPU 95% latency seconds	x86+GPU 99% latency seconds
vgg16	0.2243	0.2703	0.2453	0.4457
squeezenet1_1	0.2176	0.2672	0.2975	0.4225

OKE Right-Sizing Inference Expt: Cost Compared

- Table presents costs per hour for three configurations at four operating points
  - Figure shows ratios of Dynamic fixed-size and Right-sized costs to Max-sized costs
    - When both models not at peak, Dynamic fixed-size configuration cost < Max-sized configuration cost
    - **In all cases, Right-sized configuration << Max-sized configuration cost, by nearly two orders magnitude**
    - Note: legacy VM.GPU2.1(P100)=$1.275/hr; current generation VM.GPU3.1(V100)=$2.950/hr

Load	Max-sized $/hr	Dynamic fixed-sized $/hr	Right-sized $/hr
Both models idle	7.7860	2.6860	0.1635
Both models peak	7.7860	7.7860	0.2460
Vgg16 idle, squeezenet peak	7.7860	5.2360	0.1980
Vgg16 peak, squeezenet idle	7.7860	5.2360	0.2115



GKE Right-Sizing Inference Expt: Computing Resources

- K8s has 2 statically-allocated Ampere Arm CPU nodes for cluster mgmt, including Luna
  - Each is t2a-standard-2 shape w/2 VCPUs and 8GB [each: $0.077/hr]
- TorchServe
  - Each model's TorchServe deployment set to create minimum of 1 worker pod
  - Note: Squeezenet1_1 worker pod memory size needed increase when GPU requested

Configuration	Vgg16 worker pod size	Vgg16 node instance type	Vgg16 max worker pod count	Squeezenet1_1 worker pod size	Squeezenet1_1 node instance type	Squeezenet1_1 max worker pod count
Max-sized	400m, 4GB, 1GPU	n1-standard-2 + T4 GPU (16GB)	7	400m, 3GB, 1GPU	n1-standard-2 + T4 GPU	8
Dynamic fixed-size	400m, 4GB, 1GPU	n1-standard-2 + T4 GPU [$0.445/hr]	7	400m, 3GB, 1GPU	n1-standard-2 + T4 GPU	8
Right-sized	400m, 4GB	T2a-standard-2 [$0.077/hr]	4	400m; 1GB	T2a-standard-1 [$0.0385/hr]	4

GKE Right-Sizing Inference Expt: Accuracy & Latency Validation

- Runs on right-sized and x86+GPU nodes produced same prediction results

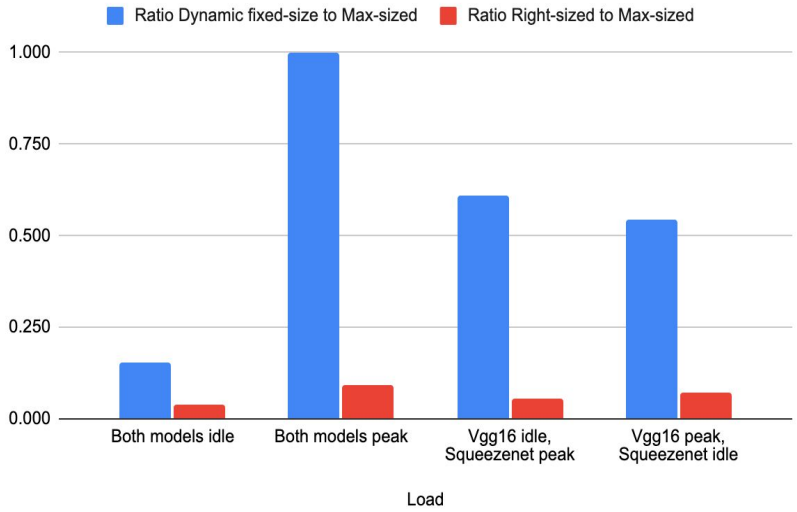
- Table shows 95 & 99 percentile E2E latency for models simultaneously handling peak load
  - Run on right-sized and on x86+GPU configurations
  - Both models close to desired E2E latency target (<=~0.5 secs) for presented workload and set up

Model	Right-sized 95% latency seconds	Right-sized 99% latency seconds	x86+GPU 95% latency seconds	x86+GPU 99% latency seconds
vgg16	0.3011	0.5153	0.3142	0.6095
squeezenet1_1	0.2996	0.4053	0.3335	0.5850

GKE Right-Sizing Inference Expt: Cost Compared

- Table presents costs per hour for three configurations at four operating points
  - Figure shows ratios of Dynamic fixed-size and Right-sized costs to Max-sized costs
    - When both models not at peak, Dynamic fixed-size configuration cost < Max-sized configuration cost
    - **In all cases, Right-sized configuration << Max-sized configuration cost, by nearly two orders magnitude**

Load	Max-sized $/hr	Dynamic fixed-sized $/hr	Right-sized $/hr
Both models idle	6.8290	1.0440	0.2695
Both models peak	6.8290	6.8290	0.6160
Vgg16 idle, squeezenet peak	6.8290	4.1590	0.3850
Vgg16 peak, squeezenet idle	6.8290	3.7140	0.5005



Right-Sizing Inference Expts: Operational Complexity Compared

- Operational complexity analysis below; entries marked “no” involve manual updates
  - Note: OCI discontinuing legacy VM.GPU2.1 is example of offerings update
  - **Right-sized configuration handled all three kinds of system changes automatically**

Configuration	**Automatically adapts to TorchServe max worker count needed changes**	**Automatically adapts to TorchServe worker size changes**	**Automatically handles cloud instance availability & offerings changes**
Max-sized	no	no	no
Dynamic fixed-size	yes	no	no
Right-sized	yes	yes	yes

Right-Sizing Inference: Conclusion

- Summary
  - Have shown that R**ight-sizing**
    - Right-sizing inference resources via using Elotl Luna
    - Right-sizing inference compute shapes by choosing Ampere A1 Arm w/Ampere Optimized AI library
  - **Can reduce cloud resource costs significantly, measured at 4 operating points for 2 cloud vendors**
    - **While avoiding operational complexity for changes in model serving resource needs & cloud offerings**

- But wait, just one more thing:
  - Note that K8s cluster that hosts DL serving workloads may be one of a K8s cluster set
    - E.g., there may be a separate K8s cluster comprising GPU nodes that hosts DL training workloads
  - Elotl Nova, a K8s service for cluster selection, can place K8s workloads into the appropriate cluster
    - Luna and Nova can be used together to simplify K8s resource management

Please Give Right-Sizing a Try on Your Workloads!

Here are resources to get you started:

- Elotl Luna
- Ampere Computing
- OCI Ampere A1 Compute, OCI OKE
- GCP Ampere A1 Compute, GCP GKE
- OCI Blog: Deep Learning inferencing at scale with Oracle Cloud A1 Compute with Elotl Luna

Thanks! Twitter: @holler_anne