Machine Learning Deployment Paradigms on Cloud: Related Papers

Srishankar Sundar

February 22, 2023

Github: https://github.com/somedamnauthor/mldep

1 Papers on Elasticity, Resource Utilization

- In [9], (USENIX 2019) Zhang et al. characterize Machine Learning (ML) workloads on Amazon Web Services (AWS) and Google Cloud Platform (GCP) across various paradigms - Infrastructure as a Service (IaaS - EC2), Container as a Service (CaaS - ECS) and Function as a Service (FaaS - Lambda). Further they present their own serving/deployment platform.

  Remarks: *The characterization is very interesting, and seems to cover what I had in mind for the broad topic for the thesis. Certain points of observation are given below:*

  - ***Workload diversity**: The characterization deals with two types of tasks - **Image Classification** using the two Inception models and **machine translation** using OpenNMT. We could push the diversity a lot more, with different use-cases. Further we could also look outside of just Deep Learning/Neural networks. Figure 1(b) in [8], where Richins et al. study end-to-end AI app performance, illustrates a use-case where video data is streamed in and face detection is performed, for example.*
  - ***Workload and Experiment Design**: The authors note that EC2 has long provisioning overhead, and thus does not do as effective a job at addressing demand surge. It is difficult to ascertain whether the average latency reported in the study takes into account provisioning overhead. Was the workload designed in such a way that provisioning and de-provisioning was tested/measured on EC2? This could be a possible area of study.*
  - ***IaaS-FaaS Hybrid**: The first key finding from the characterization is as follows: "IaaS achieves the best cost and latency performance for ML model serving, and combining it with FaaS can potentially reduce over-provisioning while remaining scalable to spiky workloads." The authors mention how this could be accomplished and it is a part of MArk, their system. Could be a possible area of study to implement a much lighter-weight system without the extra capabilities such as workload prediction, adaptive batching etc. in order to compare this against the pure IaaS and FaaS approaches.*
  - ***No advantage for CaaS**: The authors point out that ECS exhibits neither the quick scalability of Lambda nor the lower cost of EC2, and thus focus on EC2, Lambda and a combination thereof. This elicits the question as to which situation could call for ECS as the optimal solution. Investigating this could be a possible area of study.*
  - ***Do non-ML use-cases behave differently?** What makes these considerations specific to ML applications? Perhaps we could contrast the use-cases present with an additional use-case outside the broad umbrella of ML*

ML Model	EC2		ECS		Lambda	
	$	t (ms)	$	t (ms)	$	t (ms)
Inception-v3	5.0	210	9.17	217	19.0	380
Inception-ResNet	9.3	398	16.4	411	39.3	785
OpenNMT-ende	51.5	2180	96.3	2280	155	3100

Figure 1: Cost and Average Latency for 1M requests on 3 models deployed on AWS EC2, ECS and Lambda

Additional reading:
Gunasekaran et al. cite the above paper in their study on cloud resource heterogeneity [2] (Workshop on Serverless Computing, 2020).

Gunasekaran et al. also propose their own system titled 'Cocktail' with more considerations compared to MArk in [3] (USENIX 2022). ***Needs to be read in detail,*** *maybe there are great ideas here.*

- In [8] (IEEE HPCA 2020), Richins et al. take into account the entire pipeline involved in propping up an AI application and not just the aspect of serving the model. They study end-to-end application latency and provide breakdowns on each phase of a face detection use-case.

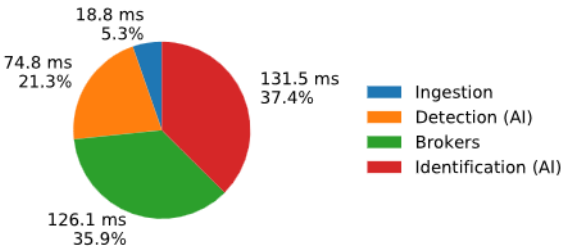

Figure 2: Breakdown of end-to-end frame latency. With inference steps in detection and identification stages, less than 60% of the latency arises from AI stages. Over a third of a frame's lifetime is spent in brokers.

Figure 2: End-to-end latency breakdown

Remarks: *"AI Applications Are More Than Just AI". This could be an interesting angle to take when approaching resource elasticity. Where this paper takes a use-case, one set architecture/-paradigm (the app is deployed as various containers via Kubernetes on a datacenter), we could try investigating the breakdown with the possibility of different paradigms - FaaS in place of containers, etc.*

- In [6] (MLSys 2020), Or et al. present an autoscaling engine for distributed deep learning workloads. Using its own scaling heuristics and decoupling the scaling from Tensorflow's in-built distribution mechanism, the autoscaler is able to achieve significant reductions in resource usage, tested on two different environments and workloads.

  Remarks: *Really liked this paper. To be noted that cloud platforms today support GPU scaling, but they seem to be based on a certain level of trial-and-error, as the user is expected to define the maximum GPU duty cycle before scaling up, etc. (2020).*
  Future work: *defined around more heuristics for scaling*
  Takeaways: *The way elasticity has been defined and analysed could be a possible takeaway - aspects studied include - Number of workers, completion time, GPU time (??), straggler removal*

2 Papers on MLOps

- In [7], Pölöskei presents a high-level view of the technology involved in a generic MLOps approach on a cloud environment. This includes the usage of Docker and Kubernetes in the context of deploying ML models, as well as a specific look at **KubeFlow** - which is a platform that faciliates creation of

workflows for ML model deployments using Kubernetes as a backbone. Further the author presents a case-study where adopting standard MLOps paradigms with KubeFlow resulted in reducing the gap between prototyping and deployment.

Remarks: *Very high-level paper but I thought it was a fair starting point for exploring ML deployments. For a more comprehensive survey refer to [5]*

- Section 5.2.2 in [4] (2022) reinforces the challenges in serving ML models - Achieving low latency and high throughput, startup/cold-start latency.

  Remarks: *Should be read further in detail in the context of the first set of papers on resource elasticity.*

- This leads us to [1], a study on the adoption of KubeFlow at CERN for the purpose of internally serving ML models. KubeFlow is integrated in this case with KALE, a JupyterLab extension in order to convert a notebook into a pipeline on KubeFlow, where separate cell(s) are parts of the pipeline. Each part of the pipeline is deployed on independent Kubernetes pods via KubeFlow.
  The paper studies the extent of scaling the KubeFlow deployment of ML models on GCP with respect to GPU resources, and also show that costs remain fairly stable across various resource allocations. GPU costs are also shown to be significantly higher than alternatives where pre-emptible TPUs (**???**) are used instead.

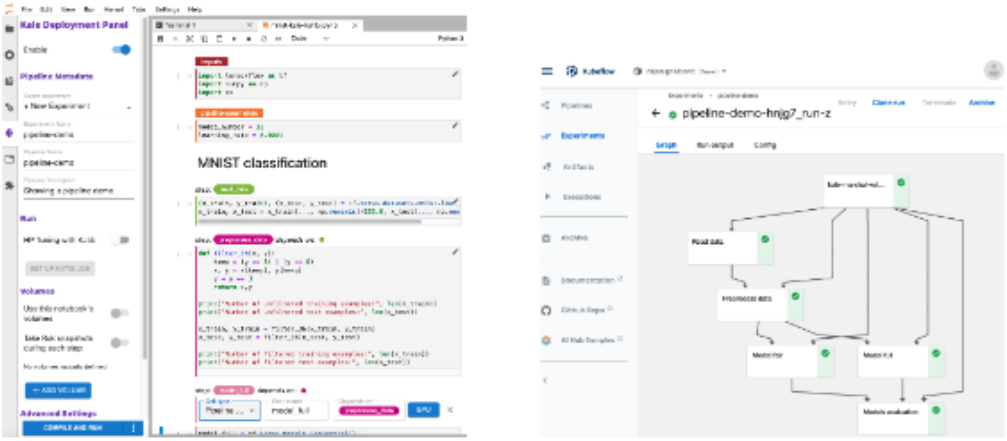

Figure 3: Notebook-to-pipeline

Remarks: *I really like the notebook-to-pipeline idea, should explore this later. The analysis of scaling and cost when using KubeFlow is nice, perhaps I could focus on analysis of* ***scaling on various ML deployment paradigms****?*
Future work on this paper talks about ***difficulties in integrating new data sources into pre-existing KubeFlow pipelines****, and* ***multi-cloud and multi-cluster integration for KubeFlow***

References

[1] Golubovic, Dejan and Rocha, Ricardo. Training and serving ml workloads with kubeflow at cern. EPJ Web Conf., 251:02067, 2021.

[2] Jashwant Raj Gunasekaran, Cyan Subhra Mishra, Prashanth Thinakaran, Mahmut Taylan Kandemir, and Chita R. Das. Implications of public cloud resource heterogeneity for inference serving. In Proceedings of the 2020 Sixth International Workshop on Serverless Computing, WoSC'20, page 7–12, New York, NY, USA, 2021. Association for Computing Machinery.

[3] Jashwant Raj Gunasekaran, Cyan Subhra Mishra, Prashanth Thinakaran, Bikash Sharma, Mahmut Taylan Kandemir, and Chita R Das. Cocktail: A multidimensional optimization for model serving in cloud. In USENIX NSDI, pages 1041–1057, 2022.

[4] Ask Berstad Kolltveit and Jingyue Li. Operationalizing machine learning models: A systematic literature review. In Proceedings of the 1st Workshop on Software Engineering for Responsible AI, SE4RAI '22, page 1–8, New York, NY, USA, 2023. Association for Computing Machinery.

[5] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. Machine learning operations (mlops): Overview, definition, and architecture, 2022.

[6] Andrew Or, Haoyu Zhang, and Michael Freedman. Resource elasticity in distributed deep learning. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, Proceedings of Machine Learning and Systems, volume 2, pages 400–411, 2020.

[7] István Pölöskei. Mlops approach in the cloud-native data pipeline design. Acta Technica Jaurinensis, 15(1):1–6, Apr. 2021.

[8] Daniel Richins, Dharmisha Doshi, Matthew Blackmore, Aswathy Thulaseedharan Nair, Neha Pathapati, Ankit Patel, Brainard Daguman, Daniel Dobrijalowski, Ramesh Illikkal, Kevin Long, David Zimmerman, and Vijay Janapa Reddi. Missing the forest for the trees: End-to-end ai application performance in edge data centers. In 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA), pages 515–528, 2020.

[9] Chengliang Zhang, Minchen Yu, Wei Wang, and Feng Yan. MArk: Exploiting cloud services for Cost-Effective, SLO-Aware machine learning inference serving. In 2019 USENIX Annual Technical Conference (USENIX ATC 19), pages 1049–1062, Renton, WA, July 2019. USENIX Association.