# Comparing Public Cloud Providers for Computation Intensive Workloads

Zhihan Guo, Zijun Ma
Department of Computer Sciences
University of Wisconsin-Madison

## ABSTRACT

As computation intensive workloads become more and more important, many cloud providers offer dedicated instances for hosting these workloads. When moving computation-intensive workloads to the cloud, it is always important for a customer to choose the optimal cloud provider suiting their needs. In this paper, we provide a comparative study on three main cloud providers with a focus on computation-intensive workload. The comparison covers typical CPU and GPU benchmarks, a machine learning model training task as a case study. We provide a comprehensive view of the performance, costs and user experiences across different providers. Our work could provide up-to-date valuable information for public cloud provider shopping to meet computation-intensive workloads.

## 1 INTRODUCTION

As computation-intensive workloads (e.g., machine learning and scientific modeling) becomes more and more important in the emerging big data era, organizations and professional developers keep searching for optimal ways to host these applications. To explore this niche market, popular public cloud providers (e.g., Amazon Web Service, Google Cloud Platform, Microsoft Azure) provides dedicated *Instance-as-a-Service* such as computing-optimized multi-core CPU and GPU instances. Customers can utilize this instances under a pay-as-you-go model and avoid setting up expensive local infrastructures.

This diversity of cloud providers leads to a practical questions: how to choose a cloud provider for computation intensive workload? Answering this question will benefit both cloud customers and providers. For a potential customer, the answer can help it choose a provider that best fits its performance and cost needs. For a cloud provider, such answers can point it in the right direction for improvements. For instance, a provider should pour more re- sources into optimizing table storage if the performance of its store lags behind competitors.

Although there have been many studies on comparing cloud providers, few of them have a main focus on computation intensive workloads and/or compute-optimized engines. [1] provides a comprehensive study on CPU, memory and disk I/O performance of general-purpose instances for four public providers. Because only one of the four providers has specialized instances, the study does not cover computing-optimized instances. Some recent blog posts [2] compare Machine Learning as a Service across different cloud providers, with focus on API and tool categories for automated, general-purpose machine learning services such as clustering, regression and object detection. Such comparison is helpful to services subscribers rather than customized machine learning model developers. [3] provides benchmarking results for TPU, GPU, and CPU platforms for Deep Learning on Google Cloud Platform only.

In this paper, we make the comparison among compute-optimized CPU instances and GPU instances across three main cloud providers (i.e., Amazon Web Service, Google Cloud Platform and Microsoft Azure). First, we justify the choice of cloud providers, virtual machines and metrics for a meaningful and relatively fair comparison. Second, we use sysbench [4] and mixbench [5] to benchmark these instances to show the performance and monetary cost for short-time computation intensive workload. Third, we conduct MNISI digit handwriting classification model training as a case study on these instances to show the performance and monetary cost for long-time computation intensive workload. Last but not least, we introduce the user experience to configure and utilize these instances. We highlight a few interesting findings below:

- Amazon has 2X initial CPU quota than Google and 4X initial CPU quota than Microsoft.

- Compared to Google and Amazon, Microsoft has much more complicated VM configuration process and longer machine provision time.
- Amazon has good performance across all tasks we tested in this report while it may not necessarily the cheapest as Google supports customizing resources (number of CPUs, GPUs, and size of RAM) in addition to fixed resource allocation.
- Azure shows no specific advantages in performance, cost and user experiences across all the tests we performed. However, our report does not cover more advanced usages such as auto scaling, etc.

# 2 MEASURE METHODOLOGY

## 2.1 Selecting Providers

Since Amazon Web Service, Google Cloud and Microsoft Azure takes up more than 50% of the world cloud market share and are still under rapid growth [6]. It is highly likely that a cloud customer would choose the service among the three. Therefore, we select these three cloud providers as our study targets.

## 2.2 Selecting Instance Types

We select multi-core CPU instances in the so-called compute-optimized family as well as GPU instances, since both CPU and GPU are suitable for a wide range of computation intensive task and are provided by all these three cloud providers. Due to the budget limitation, the highest-end machine to use in this project is 36 vCPU instance plus 1 GPU. We will not consider provider-specific instance types such as Google TPU, Microsoft FPGA and Amazon AI Chip.

## 2.3 Selecting Performance Metrics

We will compare the runtime cost and monetary cost for both short-time workload (as benchmark test) and long-time workload (as case study). Short-time workload metrics would help us understand the underlying infrastructure capacity of the virtual machine we created, meanwhile long-time workload metrics would help us know the isolation and resource multiplexing policy which are important to cloud customers in a multi-tenant production environment. What is more, we will also take user experience into consideration (e.g., whether the configuration process is clear and easy). The metrics we used are described below.

***Normalized Finishing Time.*** This metric measures how long the instance takes to complete one benchmark task in average. We show a normalized version to better show the differences. This metric is used for testing CPU performance in both basic benchmark and the case study and testing GPU performance in the case study.

***Normalized Peak Throughput.*** This metric is used to measure the maximum throughput of a GPU can get in average when performing certain computations in the basic GPU benchmark.

***Normalized Cost.*** This metric measures the monetary cost for both benchmark tests and case study. For GPU benchmark, we used normalized cost per performance. For other tests, we used the normalized cost per task. Combined with the two metrics described above, this provides a view of the performance-cost trade-offs across providers.

***Ease of Deployment.*** We performed qualitative evaluations over the ease of deployment based on factors such as launching procedures, provisioning time, flexibility in configuring an instance and other distinct features. This provide users information of which provider better suits their needs.

The metrics described above provide a comprehensive view that customers can take into consideration when choosing.

## 2.4 Repeated Tests

For each experiment, we launched three instances per configuration in the region where the data center provides lowest unit per-hour price and ran the same test for ten times on each instance.

# 3 BENCHMARK TEST

## 3.1 Instances

***CPU Instances.*** Table 1 shows CPU instances we measured in this experiment. Note that with default quota, Google Cloud can run virtual machines with up to 16 vCPUs and Azure can run machines with up to 8 vCPUs, while AWS can run machines with up to 36 vCPUs. For machines with same number of vCPUs, Google Cloud provides larger memory than Azure and AWS and charges more per hour as well. As the prime-calculation task only focuses computation and does not have much memory usage, the differences in memory size does not make much difference.

| Provider | Instance | vCPU (s) | CPU Type | Memory (GB) | Price ($/hr) |
|---|---|---|---|---|---|
| AWS | c5.xlarge | 4 | Intel(R) Xeon(R) Platinum 8124M CPU @ 3.00GHz | 8 | 0.17 |
| AWS | c5.2xlarge | 8 | Intel(R) Xeon(R) Platinum 8124M CPU @ 3.00GHz | 16 | 0.34 |
| AWS | c5.4xlarge | 16 | Intel(R) Xeon(R) Platinum 8124M CPU @ 3.00GHz | 42 | 0.68 |
| AWS | c5.9xlarge | 36 | Intel(R) Xeon(R) Platinum 8124M CPU @ 3.00GHz | 72 | 1.53 |
| Azure | F4s v2 | 4 | Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz | 8 | 0.17 |
| Azure | F8s v2 | 8 | Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz | 16 | 0.34 |
| Google Cloud | c2-standard-4 | 4 | Intel(R) Xeon(R) Platinum 8124M CPU @ 3.00GHz | 16 | 0.2088 |
| Google Cloud | c2-standard-8 | 8 | Intel(R) Xeon(R) CPU @ 3.00GHz | 32 | 0.4176 |
| Google Cloud | c2-standard-16 | 16 | Intel(R) Xeon(R) CPU @ 3.00GHz | 64 | 0.8352 |

**Table 1: CPU instances**

***GPU Instances.*** For this experiment, the GPU instances we used is shown in Table 2. All three providers requires a quota increase request to be able to use any GPUs. We submitted the request at the same time, but only Google Cloud and AWS approved the request by the time of evaluation.

## 3.2 Experiment Setup

***CPU Benchmark.*** We use *sysbench* [4] to perform prime selection task for integers within 1 million. For each instance we created, we conduct the task for 10 times under multi-threaded scenario, where the number of threads is equal to the number of vCPUs.

***GPU Benchmark.*** We use *Mixbench* [5] to evaluate performance bounds of GPUs on mixed operational intensity kernels. We perform three tasks – single precision flops (sp), double precision flops (dp) and integer multiply-addition (im) operations. A detailed description and demo of the benchmark is in their github repository [7]. We measured the peak throughput (GFLOPs) for each task.

## 3.3 CPU Quantitative Evaluations

*3.3.1 Normalized Finishing Time.* To normalize task finishing time, we use the amortized task finishing time of AWS 4-vCPU instance as 1. The amortized task finishing time on other instances are normalized based on it. As shown in figure 1, the amortized task finishing time decreases proportionally with the increasing number of vCPUs. However, two wired points need to be identified: for Google Cloud instance with 4 vCPUs and Azure instance with 8 vCPUs, the performance is quite worse. This might be attributed to the heavy-loaded CPU or defectable resource multiplexing policy for multi-tenants. Besides those two points, Google has shorter task finishing time than Amazon in terms of 8 and 16 vCPU instances.

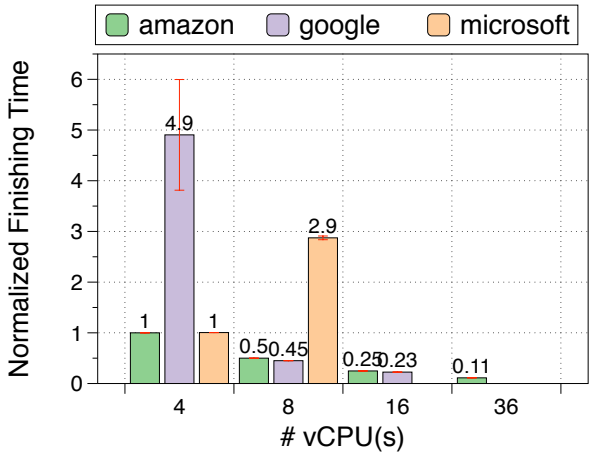


**Figure 1: CPU Normalized Finishing Time**

*3.3.2 Normalized Cost Per Task.* To normalize monetary cost per task, we use the monetary cost of AWS 4-vCPU instance as 1. The monetary cost per task on other instances are normalized based on it. As shown in figure 2, besides the two wired points, there is no significant difference in terms of monetary cost. Although Google Cloud has shorter task completion time on 8 vCPUs and 16 vCPUs, due to its higher unit hourly price, its monetary cost is slightly higher.

## 3.4 GPU Quantitative Evaluations

*3.4.1 Normalized Peak Throughput.* To normalize peak throughput, we use the amortized average peak throughput of AWS 4-vCPU 1GPU instance as 1. The amortized peak throughput on other instances are normalized based on it. As shown in Figure 3, AWS and

| Provider | Instance | GPU | vCPU(s) | Memory (GB) | GPU Memory (GB) | Price($/hr) |
|---|---|---|---|---|---|---|
| Amazon | p2.xlarge | 1 NVIDIA K80 | 4 | 61 | 12 | 0.9 |
| Google | n1-highmem-4 | 1 NVIDIA K80 | 4 | 26 | 12 | 0.481 |

**Table 2: GPU instances**

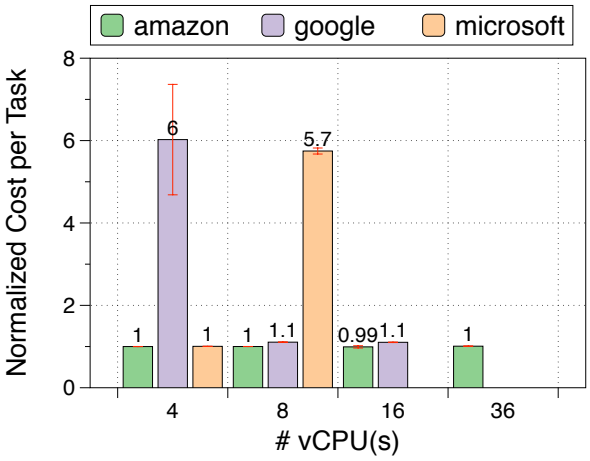


**Figure 2: CPU Normalized Cost Per Task**

Google Cloud has similar performance as expected, since they both use the same GPU device. However, the error bar indicates that Google Cloud shows more variances in the measure. We suspect that it may due to how the cloud providers manage their resources.


**Figure 3: GPU Normalized Peak Throughput**

*3.4.2 Normalized Cost Per Task.* To normalize cost per performance, we use the amortized average cost performance of AWS 4-vCPU 1GPU instance as 1 and the others are based on it. In Figure 4, it shows that Google takes almost half of the cost of AWS's since they have similar performance while Google Cloud's instance is nearly half cheaper than AWS's due to differences in the size of RAM. Note that Google Cloud supports flexible resource allocation unlike AWS which has a fixed unit, which can better suit the needs of users and save more money for them.

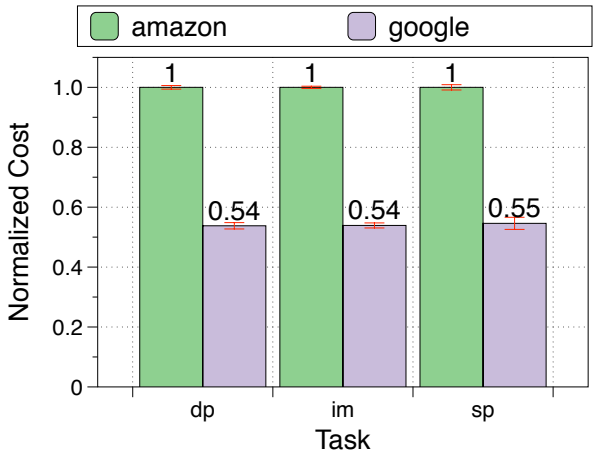


**Figure 4: GPU Normalized Cost Per Performance**

## 3.5 Qualitative Evaluations

*3.5.1 Ease of Deployment.* Azure requires eight steps to setup an instance while AWS requires two steps with 4 more optional steps and Google requires one step with folded up advanced settings. More importantly, every step in Azure has several require fields that users have to fill such as SSH public key, disk type etc, while AWS and Google has default options for all steps so that it allows users to just click and launch if there is no specific requests. Moreover, Google Cloud and AWS both support "launching from templates" so that users can instantiate nodes of same types from saved templates. *Takeaway: use AWS or Google Cloud if you want quick-launch without many specifications or re-launch with saved configurations*

*3.5.2 Flexibility in Configuration.* An unique feature Azure provides is *"Auto Shutdown"*. This feature is quite essential for Azure as it does not show running instance in the console dash board and as we will introduce later that if you create a VM instance in other services like Azure ML studio, it will not stop the VM even after you shutdown the service. Only Google Cloud allows users to *customize the CPU type or number of GPUs added for a selected instance.* AWS allows users to *specify the number of nodes for a same configuration* and provides an option of *"Auto Scaling"*.

*3.5.3 Provisioning Time.* Azure takes the longest time waiting for provisioning while the AWS takes the shortest among the three providers.

*3.5.4 Node Access.* Both AWS and Azure provide normal SSH access though Azure requires a SSH public for each configuration. Google Cloud requires users to install "gcloud" and use "gcloud ssh" from their local terminal to access. Otherwise, users could open a terminal in a new webpage with on click.

*3.5.5 Resource Availability.* One thing need to note here is that Google Cloud requires submitting a quota-increase request to use computation-optimized nodes and it has very high failure rate for instantiating such nodes due to limited resources in a region.

# 4 CASE STUDY: MACHINE LEARNING

## 4.1 Instances

***CPU Instances.*** We evaluated the machine learning task over AWS's c5.xlarge, Azure's F4s v2, and Google Cloud's c2-standard-4. As c2-standard-4 has larger memory, we turned off swapped options when running Azure's and AWS's instances. We also ran the same task on Azure's and AWS's instances with exactly the same size of memory as Google Cloud's to validate the fact that larger memory will not help in this experiment.
***GPU Instances.*** For this experiment, the GPU instances we used is shown in Table 2. To exclude the possible influence of memory, we also perform a simiar experiment as it is described above.

## 4.2 Experiment Setup

A CNN training for MNIST handwritten digits classification is used as the case study. The CNN, which is built in Pytorch, contains 2 convolutional layers and 2 fully connected layers. The Pytorch program is built with multi-processes and synchronous training pattern so that it could easily expand onto different CPU instances.

## 4.3 CPU Quantitative Evaluations

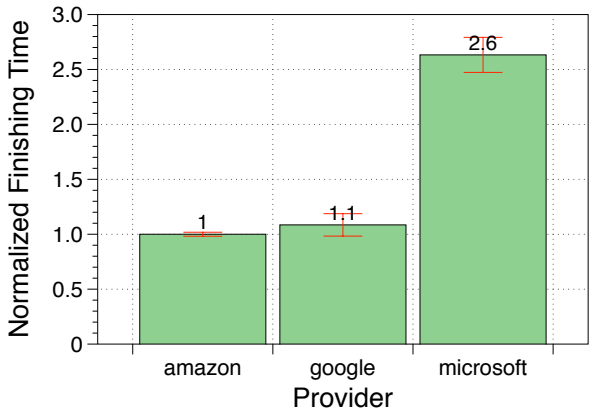


**Figure 5: CPU Normalized Finishing Time**

*4.3.1 Normalized Finishing Time.* To normalize task finishing time, we use the amortized task finishing time of AWS instance as 1. The amortized task finishing time on other providers are normalized based on it. As shown in figure 5, AWS has the shortest task finishing time meanwhile Azure has the longest task finishing time. It again indicates that AWS probably has the best resource multiplexing technique among the three. Also, it needs to be pointed out that, although Azure has very similar CPU configuration as AWS, it is actually 2x slower in our performance.

*4.3.2 Normalized Cost Per Task.* To normalize monetary cost per task, we use the amortized monetary cost of AWS instance as 1. The amortized monetary cost on other providers are normalized based on it. As shown in figure 6, AWS is the most cost-efficient meanwhile Azure is the least cost-efficient. This is mainly cost by the difference in their task finishing time.

## 4.4 GPU Quantitative Evaluations

*4.4.1 Normalized Finishing Time.* To normalize task finishing time, we use the amortized task finishing time of AWS instance as 1. The amortized task finishing time

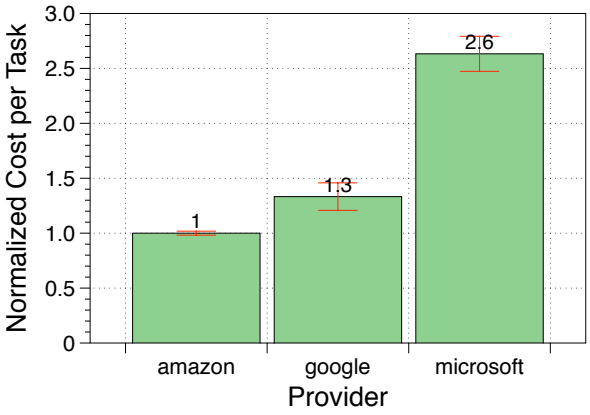


**Figure 6: CPU Normalized Cost Per Task**

on other providers are normalized based on it. As shown in figure 7, AWS takes nearly a third of time of Google Cloud's average time per task. This result contrasts with the basic GPU benchmark we described above. After excluding the influence of different memory size (described in section 4.1), we suspect that it may be still due to how differently they manage the nodes, as we can also see the greater variances showed by Google Cloud.

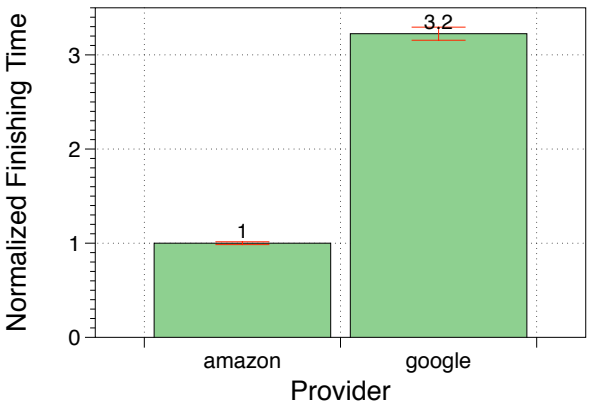


**Figure 7: GPU Normalized Finishing Time**

*4.4.2 Normalized Cost Per Task.* To normalize monetary cost per task, we use the amortized monetary cost of AWS instance as 1. The amortized monetary cost on other providers are normalized based on it. As shown in figure 8, AWS is more cost-efficient than Google Cloud even Google Cloud has less unit charges. This is due to AWS's the significant save in runtime for this specific task.

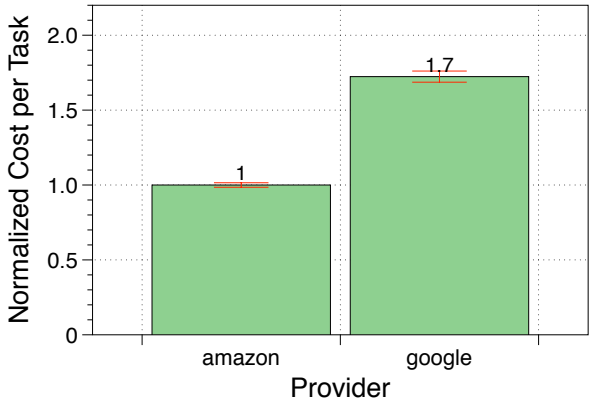


**Figure 8: GPU Normalized Cost Per Task**

## 4.5 Qualitative Evaluations

*4.5.1 Environment Setup.* All three providers support various disk images with predefined Machine Learning tools such as pytorch, tensorflow, CUDA etc, which can save a lot efforts for users. However, Google Cloud provides fewer choices for disk images for ML compared with the other two providers. A comparison of choices of disk images for ML is shown as follow:

*Azure (0 choices)* Azure provides no preset disk images for ML.

*Amazon (86 choices)* We will not list all the choices here. There have two types of images: one is "Base" with only cuda installed and the other is standard with multiple applications installed MXnet, PyTorch, Tensorflow, and Keras. Different disk images may have different extra applications like Caffe, Theno, CNTK, etc. Moreover, for each type of disk images, there have various choices of different versions.

*Google Cloud (4 choices)*

- Deep Learning Image: Base m37 (with CUDA 10.0): A Debian based image with CUDA 10.0.
- Deep Learning Image: PyTorch 1.2.0 and fastai m36 PyTorch 1.1.0 (and fastai) with CUDA 10.0 and Intel® MKL-DNN, Intel® MKL.

| Provider | AWS | Azure | Google Cloud |
|---|---|---|---|
| Service Name | Amazon SageMaker | Azure Machine Learning Studio | AI Platform Notebook |
| Git Support | yes | no | yes |
| One-Click Deployment | yes | no | yes |
| Provisioning Time | short | short | long |
| Features | one-click hyperparameter optimization; visual tracking & debugger | provide ML pipelines to build repeatable workflows; provide profile, validate, deploy, and analyze ML models in one workspace; provide visual analysis to help interpret ML models | Scale on demand (add CPUs, GPUs, RAM); Pull data from BigQuery, use Cloud Dataproc to transform it, and leverage AI Platform services or Kubeflow for distributed training and online prediction. |
| Limitations | | sandboxed execution; inability to customize python installation | limited choices of disk images (fewer applications and choices of versions) |

**Table 3: Comparison of Notebook Services**

- Deep Learning Image: PyTorch 1.3.0 and fastai m38: PyTorch 1.3.0 (and fastai) with CUDA 10.0 and Intel® MKL-DNN, Intel® MKL.
- Deep Learning Image: TensorFlow 1.15.0 m41 TensorFlow 1.15.0 with CUDA 10.0 and Intel® MKL-DNN, Intel® MKL.

*4.5.2 Use with Notebook.* *Jupyter Notebook* [8] is an open-source web application that allows people to create and share documents that contain live code, equations, visualizations and narrative text. It is a widely-used tool in python ecosystem and machine learning community. A comparison of notebook services of three providers is shown in Table 3.

# 5 FUTURE WORK

- Besides using machine learning training as a example computation intensive workload, different workloads such as large-scale data processing with Hadoop should also be investigated since these workloads typically incurs intra-network communication within a multi-node clusters, which is not covered in our case study.
- Besides compute-optimized instance, these cloud providers also offer memory-optimized instance (with hundreds of GB memory) as well as storage optimized instance (with NVMe SSD). Applications such as in-memory database or large-scale data storage could utilize these instances for better performance. We decide to explore these specialized instance in the future.

# 6 CONCLUSION

In this project, we did a comparative study among the compute-optimized instances in three popular cloud provider – AWS, Azure and Google Cloud via benchmark test and case study. For performance concern, AWS instances achieve the best across all tests. For monetary costs, Google Cloud shows great advantages by allowing customizing resource allocation. For the ease of deployment, Azure provides the most frustrating deployment experience among all three providers.

# REFERENCES


[1] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. Cloudcmp: comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2010.

[2] Comparing machine learning as a service: Amazon, microsoft azure, google cloud ai, ibm watson. Website. https://www.altexsoft.com/blog/datascience/comparing-machine-learning-as-a-service-amazon-microsoft-azure-google-cloud-ai-ibm-watson/.

[3] Gu-Yeon Wei, David Brooks, et al. Benchmarking tpu, gpu, and cpu platforms for deep learning. *arXiv preprint arXiv:1907.10701*, 2019.

[4] Scriptable database and system performance benchmark. Website. https://github.com/akopytov/sysbench.

[5] Elias Konstantinidis and Yiannis Cotronis. A quantitative roofline model for gpu kernel performance estimation using micro-benchmarks and hardware metric profiling. *Journal of Parallel and Distributed Computing*, 107:37–56, 2017.

[6] Aws vs azure vs google – detailed cloud comparison. Website. https://intellipaat.com/blog/aws-vs-azure-vs-google-cloud/.

[7] A gpu benchmark tool for evaluating gpus. Website. https://github.com/ekondis/mixbench.

[8] Project jupyter. Website. https://jupyter.org/.