ApproxDNN: Incentivizing DNN Approximation in Cloud

Seyed Morteza Nabavinejad
School of Computer Science
Institute for Research in Fundamental Sciences
Tehran, Iran
nabavinejad@ipm.ir

Lena Mashayekhy
Dept. Computer and Information Sciences
University of Delaware
Newark, DE
mlena@udel.edu

Sherief Reda
School of Engineering
Brown University
Providence, RI
sherief_reda@brown.edu

Abstract*—Service providers leverage discounted prices of reserved instances offered by cloud providers to amortize their operational costs. They reserve a certain number of instances to cover a significant portion of their computing resource requirements, and further employ on-demand instances to cover remaining requirements not satisfied by the reserved instances. Because of the higher price of on-demand instances, service providers seek lower usage of them to minimize their operational costs. In this work, we propose ApproxDNN approach for *Machine Learning as a Sevice* to reduce operational costs of service providers by incentivizing approximate results, based on the capabilities of cutting-edge GPUs and a discounted price model. When the deadline of jobs submitted by users are very tight, a service provider might not be able to execute all of them on reserved instances under the default precision. In such cases, ApproxDNN leverages the reduced-precision instructions to reduce the execution time of the jobs with slight reduction in their final accuracy, and consequently, minimize the employment of on-demand instances. To incentivize users to accept the approximate results of reduced-precision instructions, ApproxDNN offers them a discounted price for the service based on a newly designed pricing model. Our proposed pricing model of ApproxDNN guarantees lower or equal cost for service providers compared to the conventional method that solely depends on employment of on-demand instances in case of the reserved instance shortage. We employ real-world traces to conduct an extensive set of experiments and evaluate the performance of our proposed approach. The results show that ApproxDNN reduces the cost of service providers by 18%, while never exceeding the cost of the conventional method and slightly affecting the accuracy by 0.14%.

***Index Terms*—cloud computing, approximation, deep neural network, cost minimization**

I. Introduction

Cloud Infrastructure-as-a-Service (IaaS) providers aim to maximize the utilization of their resources, and consequently, increase their profit. Therefore, they offer a wide variety of instance types, in addition to conventional on-demand instances. For example, the spot instances [1] are low cost, relatively unreliable instances offered by cloud providers to utilize their transient spare capacity. The Reserved Instances (RIs) [2] are another type of discounted price instances offered in long-term contracts to guarantee that IaaS resources are rented for a long time, and hence, the IaaS provider does not need to be concerned about them being idle as users pay for the requested RIs regardless of actual use.

Service Providers (SPs) employ resources en mass from IaaS providers and deploy their services on them to serve final users. Since SPs aim to maximize their profit and/or minimize their service cost, they tend to rent discounted price instances such as RIs. Due to fluctuation in resource demand of SPs [3], it is impossible to cover all the requests with RIs without over-provisioning, which leads to idle resources and increased cost. Therefore, the employment of on-demand instances along with RIs is inevitable. Currently, the common practice is to rent a certain number of RIs to cover a large portion of requests over the course of time, and employ on-demand instances to compensate for the shortage of resources not covered by RIs [4]–[8]. Because of the wide gap between prices of RIs and on-demand instances (up to 75% [2]), SPs prefer to minimize the usage of on-demand ones in order to reduce their cost. A large body of research has focused on cost minimization considering this price gap by proposing various resource estimation, provisioning, and scheduling approaches [4], [6]–[8]. However, to the best of our knowledge, none of them has considered motivating approximation through pricing policy to address this challenge.

With the proliferation of Machine Learning (ML) applications, SPs have started offering them (e.g., AWS DeepRacer [9] and AWS DeepLens [10]). A large category of ML applications, including Deep Neural Networks (DNNs), can benefit from hardware accelerators such as GPUs and FPGAs. Hence, SPs tend to employ instances equipped by such accelerators. For example, Accelerated Computing instance families (P2, P3, G3, F1) offered by Amazon EC2 [11] have either GPUs or FPGAs. Cutting-edge GPUs such as Tesla P40 and Tesla V100 support reduced precision instructions, e.g., 32-bit floating points and 8-bit integer, in addition to the conventional 64-bit floating point. These reduced-precision instructions can accelerate the execution of ML applications such as DNN inference, however, they might affect accuracy of the results. Therefore, SPs can leverage this new capability of GPUs to reduce the execution time of the jobs, and consequently, minimize their cost. However, they should consider their negative impact on accuracy.

In this paper we propose a new approach, *ApproxDNN*, for Machine Learning-as-a-Service (MLaaS) providers. *ApproxDNN* helps SPs lessen the employment of on-demand instances, and consequently, reduce their operational costs. To reach this goal, it targets cost-sensitive users who cares about the final cost of the services they receive. *ApproxDNN* motivates those users to accept a negligible amount of accuracy reduction in exchange for a lower service price. *ApproxDNN* offers the users a discounted price for the service, in return for executing their requests by reduced-precision instructions, if needed. A discounted price can encourage cost-sensitive users to use the reduced-precision service. On the other hand, reduced-precision instructions shorten the runtime of requests and provide room on RIs to service more requests, and consequently, decrease the need for on-demand instances. By managing the discounts, *ApproxDNN* guarantees that SPs cost will never exceed the conventional approach of renting on-demand instances in case of resource shortage.

We employ a state-of-the-art GPU accelerator to evaluate the effect of reduced-precision arithmetic on performance of DNNs inference for the image classification application. Based on obtained results from our experiments and gathered information and real-world traces from IaaS providers, we conduct extensive experiments to study the effectiveness of *ApproxDNN* in minimizing the SPs costs, while maintaining the SLA (service level agreement) terms of each user. The benefits of our approach depends on difference between RIs and on-demand instances prices, as well as the willingness of end users to accept approximate results. The analysis shows that while the cost of SPs under our approach is significantly decreased compared with the conventional approach (employing on-demand instances in the presence of resource shortage), it has negligible impact on the accuracy. *ApproxDNN* can reduce the cost by up to 18%, while slightly reducing the accuracy by 0.14%. Our approach supports promotion of DNN Inference approximation in cloud through the following contributions:

- We employ several image classification DNNs and conduct experiments on a cutting-edge GPU accelerator to show how GPU architectures that support reduced-precision instructions can help accelerating the DNN inference applications. We also study their impact on the accuracy.
- We present a discounting model for MLaaS in cloud to incentivize DNN approximation such that the end users can pay less in return for accepting a slight reduction in accuracy of results, while SPs can gain more profit by reducing the usage of on-demand instances. Through formulation, we help SPs to offer the optimal discounts to the end users to motivate them accepting approximate results, while minimizing their own costs.
- Using simulation based on real-world data and traces, we evaluate the efficacy of our approach regarding an attractive discounted price for end users and reduced cost for SPs.

The rest of the paper is organized as follows. In the next Section, we present the motivation and background of this work. We then formulate our problem and describe our approach, *ApproxDNN*, in Section III. In Section IV, we evaluate the properties of our proposed approach by extensive experiments. We provide an overview of existing work in this domain in Section V. Finally, we summarize our results and conclude the paper in Section VI.

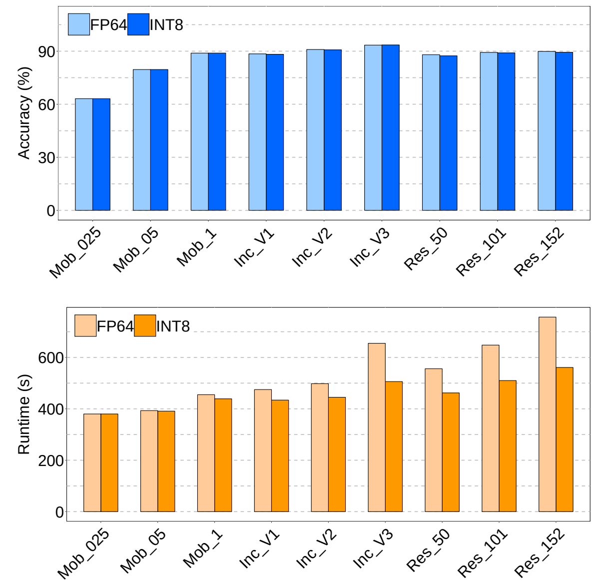

Fig. 1. The effect of reduced-precision on accuracy and runtime of DNN inference

II. Motivation and Background

A. DNN Inference Accuracy-Runtime Trade-off

We conduct a set of experiments to show the impact of reduced-precision arithmetic on the accuracy and runtime of DNN inference. We employ nine image classification DNNs with diverse characteristics such as number of layers, the type of layers, computational complexity, etc. These DNNs belong to MobileNet [12], Inception [13], and ResNet [14] families. For each DNN, we have its frozen graph model that is trained using conventional 64-bit floating point (FP64) precision. Using TensorRT [15], we generate their respective 8-bit Integer (INT8) models that can be used for inference. We use 50,000 images from ImageNet dataset [16] to evaluate the performance of networks under conventional and INT8 precision. The inference time and accuracy of results for both precision are presented in Fig. 1. For calculating the accuracy, the labels tagged to each image by networks is compared against the original labels provided by the dataset.

The results indicate that the effect of reduced precision on performance varies from one network to another. INT8 can reduce the runtime by up to 25% (Res_152) compared against the conventional precision. However, it reduces the accuracy by only around 0.6% for the same network. Considering these results, one can leverage the INT8 to meet the deadline of more jobs on RIs, and hence, decrease the employment of on-demand instances. However, the possible accuracy reduction should be considered and the users should be incentivized to use it.

B. TensorRT

TensorRT is a tool designed to optimize DNN inference. It optimizes DNN models trained in most frameworks such as

TensorFlow by calibrating the weights to lower precision with a slight effect on accuracy. In our work, we use TensorRT to quantize the weights from FP64 to INT8. It employs Symmetric linear quantization to scale FP64 to INT8. TensorRT needs a saturation threshold for calibrating the weights. Values above (below) that threshold are mapped to +127 (-127) (max range of INT8), and the rest of them are mapped to a value between -127 and +127. TensorRT runs FP64 inference on a calibration dataset (a few images) for several times to find the best saturation threshold. In each iteration, a quantized distribution based on a different saturation threshold is generated. The saturation threshold that leads to least amount of information loss is selected with the help of Kullback-Leibler divergence. Next, FP64 weights are quantized to INT8 based on the best obtained saturation threshold to generate the calibration table and INT8 model of the DNN [17].

C. Reserved Instances

In addition to conventional on-demand instances, cloud providers offer a miscellany of other instance types to satisfy the various requirements of users, while increasing their own profit. They provide their temporary idle resources in the form of spot instances which are low cost, but with low availability level. To satisfy the needs of users that want to rent instances for a long period of time such as SPs, cloud providers offer RIs. These instances are offered in the form of long-term contracts (1-year, 3-year) with significant discounts (up to 75%) compared to on-demand instances. SPs can employ RIs to cover a large portion of their computing resource demand, and hence, minimize the use of on-demand instances. Using data from Amazon EC2 [2], [18], we show the difference between RI and on-demand prices of several GPU-enabled instances in Fig. 2.

III. DNN Approximation in Cloud

A. Problem Statement and Formulation

A SP aims to deploy a set of DNN jobs on its computing resources. A certain number of RIs are rented and a scheduler is employed to schedule the jobs on them. For each job i, its estimated runtime for both conventional and INT8 precision, ${ERT_i}^{\text{FP64}}$ and ${ERT_i}^{\text{INT8}}$, and its deadline D_i are available.

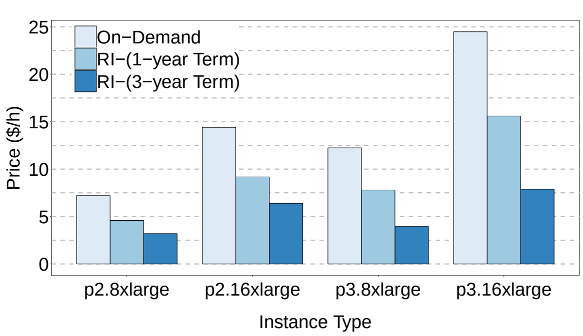

Fig. 2. The gap between prices of RIs and on-demand instances for several instance types

TABLE I
Notations Used in the Paper

Parameter	Definition
D_i	Deadline of job i
ST_i	Start time of the job (submit time + wait time in queue)
IP_{reserved}	Price of reserved instance ($/hour)
$IP_{\text{on-demand}}$	Price of on-demand instance ($/hour)
${ERT_i}^{\text{FP64}}$	Estimated runtime of job i using FP64 precision
${ERT_i}^{\text{INT8}}$	Estimated runtime of job i using INT8 precision
${C_i}^{\text{reserved}}$	Monetary cost of deploying job i on reserved instance
${C_i}^{\text{on-demand}}$	Monetary cost of deploying job i on on-demand instance
C_{SLAV}	Monetary cost of SLA violation
SLA_{P}	SLA violation penalty fee coefficient

For DNN inference, estimating the runtime can be obtained by sampling a few inference tasks (e.g., inferencing a few image in image classification DNNs). Our experiments on a set of image classification DNNs show that such an approach is valid and can yield fairly accurate estimations. Generating INT8 version of the DNN models is also very fast and imposes negligible overhead on the system. After scheduling all the jobs on RIs, the scheduler might decide to deploy some jobs on on-demand instances to satisfy their deadlines.

When scheduling the jobs on RIs, the scheduler first considers the conventional precision and its estimated runtime for the job. The reason is to avoid a penalty fee associated with INT8 precision due to possible violation of accuracy mentioned in SLA. Note that this is a common practice for SPs to pay their users in the case of SLA violation. For example, Amazon Compute pays its users back if the monthly uptime percentage of the instances is less than a certain value [19]. The scheduler then considers the INT8 precision to decrease the number of employed on-demand instances, and consequently, their monetary cost. However, it should consider the SLA violation penalty fee. Before using INT8 for processing a job, the SP should ask the owner of that job. If the owner disagree, the SP must execute the job with the conventional precision. As can be seen in Fig. 1, the effect of INT8 precision on the accuracy of DNNs is different. It renders the accuracy low in some networks, while the other ones are not affected. Because of this uncertainty regrading effects of INT8 on accuracy, the SP is required to pay a penalty fee to the users whose jobs are executed with INT8 precision. To incentivize the users to accept the INT8 precision, a SP can maintain the amount of SLA violation penalty fee. Finally, if a job is deployed on on-demand instances, it will be executed by FP64 precision to avoid an extra cost due to the SLA violation penalty fee.

In the following, we provide the optimization model for solving the described problem. All the parameters used in the optimization model are listed in Table I. The objective function of the optimization model is to minimize the monetary cost of an individual job (job i) by either deploying that job on reserved or on-demand instances. We define the following decision variables. The decision variable x_i shows the precision selected for the job:

$$x_i = \begin{cases} 1, & \text{conventional precision is selected for job } i \\ 0, & \text{INT8 precision is selected for job } i \end{cases} \quad (1)$$

We use another decision variable, y_i, to show either job i is deployed on reserved or on-demand instances.

$$y_i = \begin{cases} 1, & \text{job } i \text{ is scheduled on reserved instance} \\ 0, & \text{job } i \text{ is scheduled on on-demand instance} \end{cases} \quad (2)$$

If the job is scheduled on an on-demand instance ($y_i = 0$), then it would be definitely executed by conventional precision according to the problem statement (i.e., $x_i = 1$). Hence, the cost of deploying the job on on-demand instances, $C_i^{\text{on-demand}}$, can be calculated having the estimated runtime of FP64 (ERT_i^{FP64}) and the price of on-demand instance ($IP_{\text{on-demand}}$) as follows:

$$C_i^{\text{on-demand}} = (1 - y_i) \times x_i \times ERT_i^{\text{FP64}} \times IP_{\text{on-demand}} \quad (3)$$

However, if the job is executed on a reserved instance ($y_i = 1$), it is either executed by FP64 or INT8 precision. If it is executed by INT8 ($x_i = 0$), the service provider should pay the SLA violation penalty fee, in addition to the reserved instance cost. Otherwise ($x_i = 1$), only the monetary cost of reserved instance should be considered. Therefore, the cost of deploying the job on reserved instances, C_i^{reserved}, is calculated using the following equation:

$$\begin{aligned} C_i^{\text{reserved}} &= y_i \times (x_i \times ERT_i^{\text{FP64}} \times IP_{\text{reserved}}) \quad + \\ & y_i \times (1 - x_i) \times (ERT_i^{\text{INT8}} \times IP_{\text{reserved}} + C_{\text{SLAV}}) \end{aligned} \quad (4)$$

Note that one of C_i^{reserved} or $C_i^{\text{on-demand}}$ would be zero, and the other one determines the total cost.

We now formulate optimization model as follows:

$$\text{Minimize} \quad C_i^{\text{reserved}} + C_i^{\text{on-demand}} \quad (5)$$

Subject to:

$$ST_i + x_i \times ERT_i^{\text{FP64}} + (1 - x_i) \times ERT_i^{\text{INT8}} \leq D_i \quad (6)$$

The objective function is to minimize the monetary cost of job i and it is constrained by the job's deadline.

B. *ApproxDNN*

In this section, we present our proposed approach, *ApproxDNN*, to reduce the monetary cost that service providers should pay for resources they use, and consequently, increase their profit. *ApproxDNN* leverages the capability of GPUs that support reduced-precision instructions to accelerate the execution of DNN jobs, and hence, reduce the cost of service providers. With the help of an illustrative example shown in Fig. 3, we describe how *ApproxDNN* works and discuss its properties compared to conventional methods. As can be seen, we have three DNN inference jobs that we want to (preferably) schedule on a reserved instance. Scheduling all of them with FP64 precision on the reserved instance leads to job 3 missing its deadline (Fig. 3.a). In this case, the conventional methods that do not leverage reduced-precision instructions of GPUs, would schedule job 3 on an on-demand instance (as shown in Fig. 3.b), which means more cost due to price differences between an on-demand and reserved instance (see Fig. 2). Unlike conventional methods, *ApproxDNN* takes advantage of INT8 precision provided by GPU to accelerate the execution of job 3. Hence, it can successfully deploy all three jobs on reserved instance and avoid extra cost of renting an on-demand instance (Fig. 3.c). Note that *ApproxDNN* leverages INT8 provided that the two following conditions hold true: 1) the INT8 would be able to sufficiently reduce the execution time of the job, such that it can meet its deadline on a reserved instance; and 2) the user that has submitted the job would agree with a possible accuracy reduction of the job due to employment of INT8-precision instructions by the SP. In the following, we describe the overall flow of *ApproxDNN*.

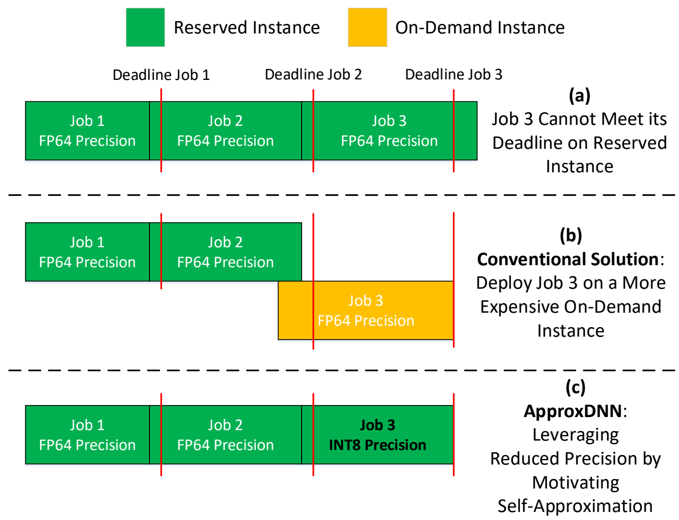

Fig. 3. Illustrative example that shows difference between ApproxDNN and conventional approaches.

ApproxDNN offers the SLA violation penalty fee to users as a multiple of cost of running the job on a reserved instance using INT8, by employing SLA_{P} coefficient. Hence, the monetary cost of SLA violation for job i would be calculated as follows:

$$C_{\text{SLAV}} = SLA_{\text{P}} \times ERT_i^{\text{INT8}} \times IP_{\text{reserved}} \quad (7)$$

Therefore, the total cost of running job i on a reserved instance using INT8 precision is the sum of the cost of deploying the job on a reserved instance and the cost of SLA violation:

$$(ERT_i^{\text{INT8}} \times IP_{\text{reserved}}) + (SLA_{\text{P}} \times ERT_i^{\text{INT8}} \times IP_{\text{reserved}}) \quad (8)$$

ApproxDNN aims to guarantee that the cost of using the reduced precision for the SP will never exceed the cost of the conventional approach (i.e., employing on-demand instance). Hence, it finds the maximum value of SLA_{P} (called $SLA_{\text{P-Max}}$) such that the cost of deploying the job on a

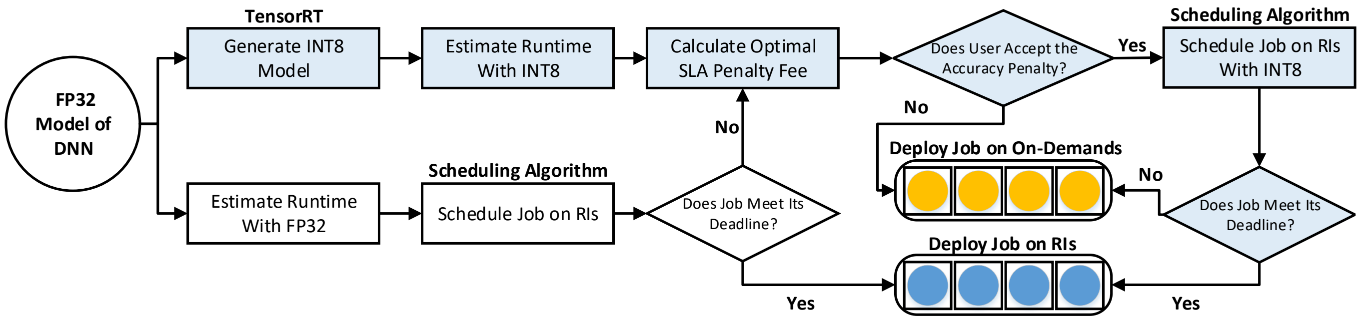

Fig. 4. The overall flow of *ApproxDNN*. The colored parts are the contributions of *ApproxDNN*.

reserved instance using INT8 would be less than or equal to the cost of deploying the job with the conventional precision on an on-demand instance:

$$(ERT_i^{\text{INT8}} \times IP_{\text{reserved}}) + (SLA_{\text{P}} \times ERT_i^{\text{INT8}} \times IP_{\text{reserved}}) \leq ERT_i^{\text{FP64}} \times IP_{\text{on-demand}} \quad (9)$$

To calculate $SLA_{\text{P-Max}}$, *ApproxDNN* solves the first derivative of Eq. (9) for SLA_{P}. After obtaining $SLA_{\text{P-Max}}$, *ApproxDNN* formulates the expected cost of deploying the job on a reserved or on-demand instance. At this step, *ApporxDNN* needs to have knowledge of the probability of acceptance of reduced-precision results by the user. We assume that this probability is a function of SLA_{P} as follows:

$$P_{\text{acceptance}} = \frac{SLA_{\text{P}}}{SLA_{\text{P-Max}}} \quad (10)$$

Considering $P_{\text{acceptance}}$, the expected cost is formulated as follows:

$$P_{\text{acceptance}} \times (1 + SLA_{\text{P}}) \times (ERT_i^{\text{INT8}} \times IP_{\text{reserved}}) + (1 - P_{\text{acceptance}}) \times ERT_i^{\text{FP64}} \times IP_{\text{on-demand}} \quad (11)$$

If the user accepts the offer of executing his/her job with INT8 precision (with probability $P_{\text{acceptance}}$), the cost will be the first term of Eq. (11). Otherwise, if the user does not accept the offer (with probability $1 - P_{\text{acceptance}}$), then the job should be executed by the conventional precision on an on-demand insance, and the cost is the second term of Eq. (11). The only variable in Eq. (11) is SLA_{P}, and hence, *ApproxDNN* solves the first derivative of Eq. (11) to find it. The result is the optimal value of SLA_{P}, which we call $SLA_{\text{P-Optimal}}$, that minimizes the monetary cost that SP should pay for serving the job. *ApproxDNN* makes sure that $SLA_{\text{P-Optimal}}$ is not greater than $SLA_{\text{P-Max}}$. Otherwise, it replaces $SLA_{\text{P-Optimal}}$ with $SLA_{\text{P-Max}}$. Having $SLA_{\text{P-Optimal}}$, *ApproxDNN* offers $SLA_{\text{P-Optimal}} \times ERT_i^{\text{INT8}} \times IP_{\text{reserved}}$ to the user as the SLA violation penalty cost. The overall flow of *ApproxDNN* is shown in Fig. 4. The colored parts are the contributions of *ApproxDNN* to the conventional approach. The pseudo-code of *ApproxDNN* is also presented in Algorithm 1.

Algorithm 1 ApproxDNN

Input: $IP_{\text{reserved}}, IP_{\text{on-demand}}, ERT_i^{\text{FP64}}, ERT_i^{\text{INT8}}$
Output: Assignment of job i to reserved or on-demand instance plus its precision (FP64 or INT8)
//We assume that it is not possible to schedule the job with conventional precision on reserved instance such that it can meet its deadline

```
SLA_P-Max = Solve(first derivative of Eq. (9) for SLA_P)
P_acceptance = SLA_P / SLA_P-Max
SLA_P-Optimal = Solve(first derivative of Eq. (11) for SLA_P)
Compare SLA_P-Optimal with SLA_P-Max
Offer SLA_P-Optimal × ERT_i^INT8 × IP_reserved penalty fee to user
if User accepts the offer then
    Deploy job i with INT8 precision on reserved instance
    Service provider cost = (1 + SLA_P-Optimal) × ERT_i^INT8 × IP_reserved
else // User does not accept the offer
    Deploy job i with FP64 precision on on-demand instance
    Service provider cost = ERT_i^FP64 × IP_on-demand
```

IV. Evaluation

We conduct an extensive set of experiments using real-world workload to evaluate the efficacy of *ApproxDNN* and compare its performance against other approaches.

A. Experimental Setup

Workload. We employ traces provided by *Microsoft Azure* [20] to create our workload. Each trace includes specification of VMs launched in one of Azure's datacenters. We consider each VM as a job in our experiments. For each VM, we extract its deployment time and finish time from the trace and calculate its runtime accordingly. We have considered the runtime of the VMs as the FP64 execution time, and then generated INT8 execution as a coefficient of the FP64 execution (between 0.75 to 1 of the FP64 execution time, according to ratios we have from image classification DNNs for conventional and INT8 precision). There is no information for deadline of the jobs, so we consider it from 1.5 to 3 times of the FP64 execution time. That means, from the start time of the job, it has that much time (deadline) to complete. It is a common practice to consider the deadline of jobs as a coefficient of their runtime [21], [22].

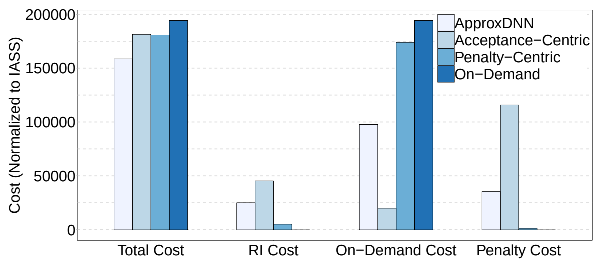

Fig. 5. The overall monetary cost results obtained for each approach.

At the first step, we use a simple scheduler which represents the common schedulers that schedule the jobs in a round-robin fashion on RIs, considering the FP64 execution time. When scheduling the jobs on RIs, if a job cannot meet its deadline, it is scheduled on on-demand ones. The only extra step is that this scheduler checks to see if the job that will be scheduled on on-demand instances is able to be scheduled on RIs and meet its deadline if INT8 is employed. If so, it saves the information of that job as a candidate job for approximation. In our experiments we use the list of the saved jobs as the input to evaluate *ApproxDNN* and other approaches. Please note that any other scheduler can be used instead of the simple scheduler we have used. The total number of the saved jobs in the workload is 18,187.

Systems compared. To evaluate the efficacy of *ApproxDNN*, we compare it against the following approaches:

- *On-Demand*: This approach does not consider approximation and solely schedules the jobs that cannot meet their deadline using RIs, on on-demand instances. This approach is similar to previous works [4], [5] that schedule jobs on a mixture of RIs and on-demand instances. It is also a baseline approach to show the advantage of using approximation.
- *Acceptance-Centric*: This approach aims to increase the acceptance rate of approximation offered to users, and consequently, reduce the number of on-demand instances rented for the jobs that cannot meet their deadlines. Therefore, it offers high SLA violation penalty fees (high value of SLA_{P}). In the experiments, we set the SLA_{P} of this approach to 0.9 of the maximum SLA_{P} (i.e., $SLA_P = 0.9 \times SLA_{P-Max}$). Similar to *ApproxDNN*, this approach leverages INT8 precision to reduce the execution time of the jobs.
- *Penalty-Centric*: Unlike the previous approach, this one aims to minimize the cost imposed by the SLA violation penalty fee. Hence, it offers a low SLA violation penalty fee coefficient to users. In the experiments, we have $SLA_P = 0.1 \times SLA_{P-Max}$ for this approach. This approach also employs reduced-precision instructions similar to *ApproxDNN*.

VM Instance. We use the specifications of Amazon EC2 *p3.2xlarge* instance [23] in the experiments. This instance is equipped with a Tesla V100 GPU that supports reduced-precision instructions. The on-demand price of this instance is \$3.06 per hour and for the reserved price of it, we have used standard 3-year term contract (all upfront) which is \$0.985 per hour. All the prices are for US East (Ohio) region and the Linux operating system.

B. Experimental Results

Conducting the experiments, we gather various results regarding monetary cost of the different approaches. The total monetary cost of each approach for all the jobs, as well as the monetary cost spent for RIs, on-demand instances, and penalty cost are shown in Fig. 5. In addition, the cumulative distribution of the results per job is illustrated in Fig. 6. The results show that *ApproxDNN* can improve the total

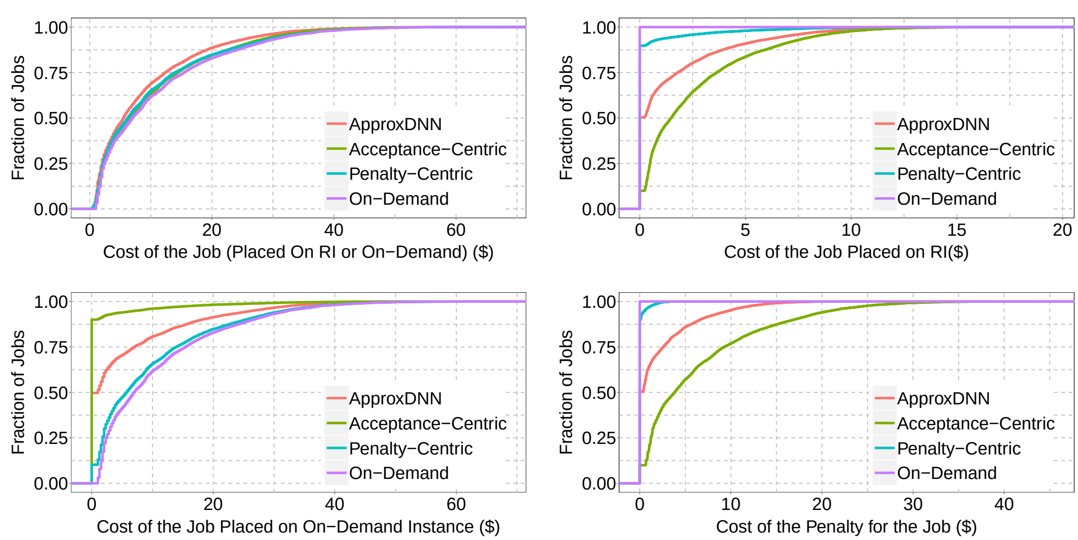

Fig. 6. Cumulative distribution of cost of each job under different approaches. The horizontal axis indicates the RI, on-demand, or penalty cost of a job (depending on the plot) and the vertical axis shows the fraction of jobs with equal or less cost than the certain value indicated by horizontal axis.

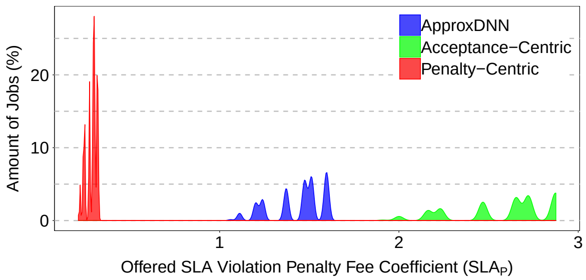

Fig. 7. Distribution of SLA_P offered for each job under different approaches.

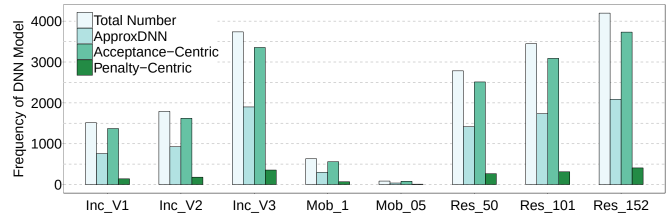

Fig. 8. Frequency of each DNN type in the workload, and the number of DNNs from each DNN type that has been executed with reduced-precision models by different approaches.

monetary cost of all the jobs by around 18.3%, 12.5%, and 12.2% compared with On-Demand, Acceptance-Centric, and Penalty-Centric approaches, respectively. Since Acceptance-Centric offers high SLA_P, which leads to high approximation acceptance rate, its RI cost is high and on-demand cost is low. However, its high SLA_P causes highest penalty cost. On the other hand, Penalty-Centric has low RI and penalty cost since it does not offer enough SLA_P to incentivize users to accept approximation on RIs. However, it deploys a large portion of jobs on on-demand instances that impose significant cost. Since *ApproxDNN* aims to find a balance between acceptance rate of approximation by users and penalty cost, its results stand between Penalty-Centric and Acceptance-Centric for RI, on-demand, and penalty cost in both Fig. 5 and Fig. 6. Note that since On-Demand approach neither employs RIs nor offers penalty, there is no bar for it in RI Cost and Penalty Cost in Fig 5.

To go deeper into the details, we study the value of SLA_P offered to each user by different approaches. The continuous histogram of the SLA_P offered by each approach is shown in Fig. 7. The horizontal axis shows the value of SLA_P, and the vertical axis indicates the percentage of the jobs with a certain value of SLA_P. As expected, Penalty-Centric has the lowest SLA_P at 0.285 on average and Acceptance-Centric has the highest at 2.33. Again, *ApproxDNN* stands between them at 1.294.

In the following, we shed light on the impact of using reduced-precision instructions on the accuracy of the workload. First, in Fig. 8 we show the number of each DNN type in the workload and number of each DNN that has been executed using reduced-precision instructions under each approach. For example, 1513 out of 18,187 jobs are using Inception-V1 (INC_V1) image classification DNN. *ApproxDNN*, Acceptance-Centric, and Penalty-Centric have executed 758, 1371, and 140 of these 1513 jobs with the reduced-precision model of the DNN, respectively. Conducting experiments on a Tesla P40 GPU, that supports INT8-precision instructions, by the setup introduced in Section II, we have the Top-5 label accuracy of image classification application for each DNN under each precision (FP64 and INT8) as presented in Table II.

Considering the results shown in Fig. 8 and Table II, we have the following values for the overall accuracy of the workload under different approaches (note that since On-Demand does not employ reduced-precision instructions, its accuracy is the highest possible accuracy): 90.12% for On-Demand, 89.99% for *ApproxDNN*, 89.88% for Acceptance-Centric, and 90.10% for Penalty-Centric. While *ApproxDNN* significantly reduces the total cost of the workload compared with On-Demand, it has negligible effect on the accuracy and slightly decrease it by around 0.14%. *ApproxDNN* even achieves higher accuracy compared with Acceptance-Centric, which has higher monetary cost than *ApproxDNN*. Penalty-Centric yields higher accuracy than *ApproxDNN* since it rarely uses reduced-precision mode of DNNs. Note that this reduced

TABLE II
ACCURACY OF DNN MODELS UNDER DIFFERENT PRECISION

DNN Model	Original Accuracy	INT8 Accuracy
Inception-V1 (INC_V1)	88.49	88.24
Inception-V2 (INC_V2)	90.95	90.79
Inception-V3 (INC_V3)	93.43	93.53
MobileNet-V1-1 (Mob_1)	88.91	88.93
MobileNet-V1-05 (Mob_05)	79.62	79.62
MobileNet-V1-025 (Mob_025)	63.14	63.14
ResNet-V2-50 (Res_50)	87.99	87.40
ResNet-V2-101 (Res_101)	89.30	89.05
ResNet-V2-152 (Res_152)	89.89	89.40

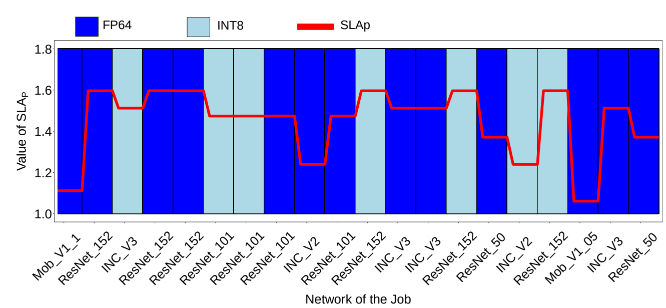

Fig. 9. The SLA_P offered to each job and the precision that the DNN is executed with for the first 20 jobs under *ApproxDNN* approach. The background color indicates the precision of the network (dark blue: FP64, light blue: INT8) and the red line shows the value of SLA_P.

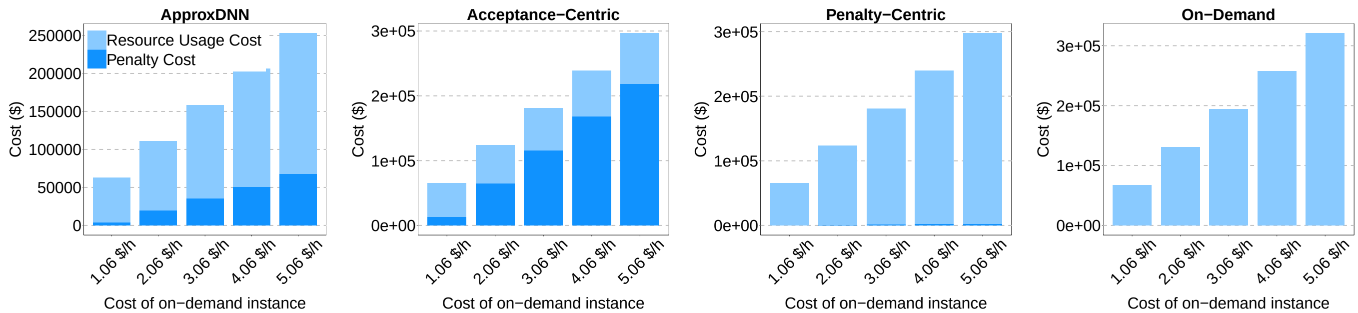

Fig. 10. The impact of price gap between RIs and on-demand instances on the resource usage and penalty cost of each approach.

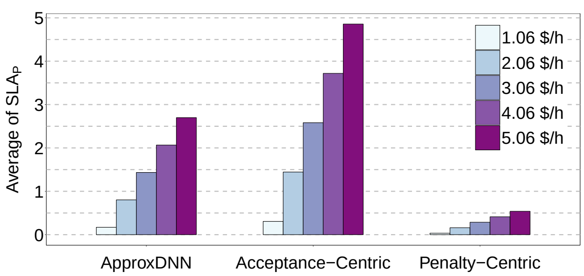

Fig. 11. Impact of price gap on the average SLA_P under different approaches.

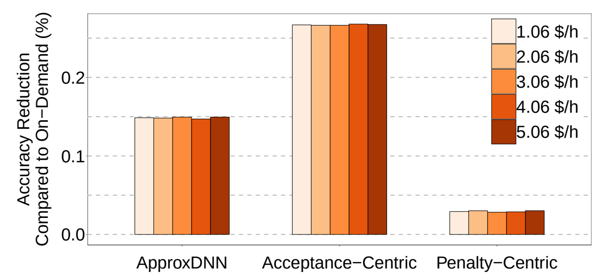

Fig. 12. Impact of price gap on the accuracy of the workload for all the approaches.

accuracy of *ApproxDNN* is acceptable by the users since it is compensated by the SLA violation penalty fee.

Finally, in Fig. 9, we show the dynamic behavior of *ApproxDNN* by presenting its results for the first 20 jobs (due to the space limit, we only depict 20 jobs). This figure shows the value of SLA_P offered to each job and the precision that the DNN is executed with (the background color). The results show that SLA_P offered to each job has a direct relationship with the runtime difference of FP64 and INT8 versions of its DNN. For example, the runtime of ResNet networks can be significantly reduced with INT8 precision (see Fig. 1). Therefore, *ApproxDNN* offers a high value of SLA_P to them. On the other hand, the Mobile networks such as Mob_V1_1 are offered a lower SLA_P since the runtime difference between their FP64 and INT8 models is not as large as ResNet.

C. Sensitivity Analysis

1) Price Difference Between RI and On-Demand Instances: In this section, we study the impact of difference between price of RI and on-demand instances on the performance of *ApproxDNN* and other approaches. To analyze how the price gap of RIs and on-demand instances affects each approach, we keep the RI price intact and change the price of on-demand instance from 1.06 $/h to 5.06 $/h by step of one. The results in Fig. 10 shows how each approach reacts to the price gap variation. The resource usage cost in Fig. 10 shows the sum of the cost of RIs and on-demands used by each approach, and the penalty cost shows the amount of SLA violation penalty fee paid to users. As expected, a higher on-demand instance price increases the total cost of all the approaches. However, the portion of resource usage and penalty cost is differently affected depending on the approach. In *ApproxDNN*, the portion of these two costs remains almost the same regardless of the price of the on-demand instance. However, the amount of resource usage cost increases slower than the penalty cost in Acceptance-Centric, as it tends to reduce the resource usage cost by offering high penalty fees. On the contrary, the resource usage cost dominates the penalty cost in the Penalty-Centric as the price gap increases. This is expected since Penalty-Centric tries to avoid paying the penalty fee, and thus, it has to spend more on on-demand instances. As On-Demand does not offer any penalty fee, all of its cost belongs to the resource usage, regardless of the price gap.

The average value of SLA_P offered to users by each approach for various on-demand instance prices (depicted in Fig. 11) justifies the cost-related results we discussed. All the approaches offer higher SLA_P as the price gap increases. However, the values offered by Acceptance-Centric are always much higher than Penalty-Centric. *ApproxDNN*, as always, stands between Acceptance-Centric and Penalty-Centric. While the price gap variation significantly affects the total cost and SLA_P of the approaches, their accuracies experience slight fluctuation. Fig. 12 shows the amount of accuracy reduction of each approach compared to On-Demand (that dose not employ approximation) when changing the on-demand instance price. As can be seen, the amount of variation in all the approaches is negligible.

TABLE III

IMPACT OF INT8 RUNTIME REDUCTION ON THE PERFORMANCE OF THE APPROACHES. ADNN: APPROXDNN, AC: ACCEPTANCE-CENTRIC, PC: PENALTY-CENTRIC

INT8 Runtime Reduction	SLA_{P}			RI Cost			On-Demand Instance Cost			Usage Cost			Penalty Cost			Total Cost			Accuracy		
	ADNN	AC	PC	ADNN	AC	PC	ADNN	AC	PC	ADNN	AC	PC	ADNN	AC	PC	ADNN	AC	PC	ADNN	AC	PC
Original	1.434	2.581	0.287	25134	45381	5282	97699	20101	173888	122833	65482	179170	35648	115783	1496	158481	181265	180666	0.149	0.266	0.028
10%	1.649	2.968	0.330	22620	41005	4339	97699	19577	175641	120319	60582	179980	36905	120193	1415	157224	180775	181395	0.149	0.266	0.028
20%	1.918	3.452	0.384	20223	36544	4198	97281	19104	173957	117503	55648	178155	38313	124633	1597	155816	180281	179752	0.150	0.268	0.031
30%	2.263	4.074	0.453	17551	31886	3697	97953	19454	173888	115504	51339	177585	39313	128511	1654	154817	179850	179240	0.149	0.267	0.031
40%	2.724	4.902	0.545	15080	27448	2899	97699	18760	175527	112779	46208	178426	40675	133129	1570	153454	179337	179997	0.149	0.268	0.030

2) Runtime Difference Between FP64 and INT8: Runtime difference between FP64 and INT8 versions of the DNNs is the other parameter that we study its impact on the performance of the approaches. We decrease the INT8 runtime of the DNNs while keeping the FP64 runtime constant. The INT8 runtime is decreased by 10%, 20%, 30%, and 40% compared with the initial experiments. We present the results in Table III. While all the approaches can leverage the decreasing runtime of INT8 to improve the total cost, the rate of cost reduction varies from one approach to another one. *ApproxDNN* has the highest cost reduction, Penalty-Centric has the least reduction, and Acceptance-Centric stands between them. All the three approaches offer higher SLA_{P} as the runtime gap increases. However, the offered SLA_{P} by Penalty-Centric are still very low, and hence, it cannot leverage the gap significantly. Therefore, while its RI cost is slightly reduced, its on-demand cost and penalty cost, which are increasing, neutralize the RI cost reduction. For Acceptance-Centric, both RI and on-demand cost are decreasing, but the penalty cost is increasing. Finally, the decreasing RI cost in *ApproxDNN* dominates the increasing penalty cost. Since the on-demand cost is almost constant and RI cost surpasses the penalty cost, the total cost of the *ApproxDNN* is decreasing.

The bottom line is that when the INT8 runtime reduces, *ApproxDNN* and Acceptance-Centric pay less for the reduced-precision DNNs that are deployed on the RIs. On the other hand, the higher SLA_{P} offered by them causes more penalty cost (INT8 runtime and SLA_{P} both affect penalty cost according to Eq. (11). Since the SLA_{P} increase is more significant than INT8 runtime reduction, the overall penalty cost increases). The reduction of RI cost is higher than the penalty cost increase, and hence, the total cost is decreasing. The jobs deployed on an on-demand instance are executed by the conventional accuracy, and hence, the reduction in INT8 runtime does not affect their monetary cost. Since Penalty-Centric tends to deploy most of the jobs on an on-demand instance, it cannot leverage much from the reduced INT8 runtime. Therefore, its total cost reduction is negligible compared with *ApproxDNN* and Acceptance-Centric.

V. RELATED WORK

Combining RIs and on-demand instances to reduce the monetary cost has studied in a large body of research [4]–[8]. RISA [4] employs stochastic optimization to find the best number of RIs that increases the total monetary cost. RISA considers the fluctuation in resource demand of big data jobs and deploys the jobs that are not covered by RIs on on-demand instances. It models the problem as a variant of News Vendor Problem, a well-know problem among stochastic problems. CoH-R [8] considers the resource demands of jobs in an hourly granularity to model the cost of RIs and on-demand instances. Having these models, it deploys the jobs on different instances. Reserved Instance Provisioning strategy based on Autoregressive Model (RIPAM) [5] also considers the cost difference of RIs and on-demand instances when scheduling the jobs. These studies have focused on the resource allocation and scheduling techniques to reduce the monetary cost by combining RIs and on-demand instances. However, none of them leverages approximation techniques. *ApproxDNN* can be used as complementary to these approaches.

Using approximate techniques to improve the performance of DNNs has been on the increase. CANNA [24] is interested in both training and inference phases of neural networks (NN). To accelerate the training phase, CANNA proposes Gradual Training Approximation (GTA). GTA starts from deep approximation to achieve as much acceleration as possible. However, it gradually reduces the approximation level to achieve a sufficient amount of accuracy based on internal error of NN. In inference phase, CANNA relaxes the computation in each layer of NN to gain speed up, while maintaining the accuracy. CANNA employs a floating point unit (FPU), which is a hardware configurable unit, to control the level of approximation in each layer at runtime. Similar to CANNA, Koteshwara et al. [25] propose an incremental precision based approach for reducing the energy consumption of classification applications. The first component of the proposed approach is a threshold calculation unit which decides on the level of approximation needed to classify the samples properly. The second component, incremental-precision fast Fourier transform, controls the level of approximation for feature computation. ApproxANN [26] leverages an error-tolerant feature of neural networks to apply approximation on both memory access and computation. ApproxANN identifies the less critical neurons that have slight effect on accuracy of the network. Then, it applies approximation on their memory access and computation to improve energy efficiency while considering accuracy requirements. Power-Inference accuracy Trading (PIT) is another approach that leverages reduced-precision instructions to improve response time and energy consumption of DNN jobs submitted to a queue. It dynamically changes DVFS of GPU and precision of DNNs by taking into account runtime and slack time of jobs waiting in the queue. The main concern of the aforementioned approaches is response time, power, or energy, and none of them considers the monetary cost of the resources.

Proteus [27] aims to improve cost and execution time of ML training by leveraging spot instances. TensorFlow and similar frameworks use a parameter server architecture in which parallel workers work independently from each other and communicate through a key-value store. It employs a combination of on-demand and spot instances to reduce the cost of training while improving the execution time. It has two modules: 1) AgileML parameter server that combines several reliability tiers and deploys essential functions on reliable on-demand instances and less critical ones on spot instances. 2) BidBrain which is responsible for resource allocation. It acquires the resources from market by monitoring the prices and bidding on new instances when they would improve work per dollar of the system. Proteus focuses on training phase and does not leverages approximation. However, *ApproxDNN* is mainly concerned about inference phase and aims to improve the monetary cost by employing approximation.

VI. Conclusion

In this paper we introduced *ApproxDNN* approach that leverages the reduced-precision instructions of the cutting-edge GPUs to improve the monetary cost of SPs. *ApproxDNN* encourages cost-sensitive users to use a reduced-precision model of DNNs, which results in negligible accuracy reduction, in exchange for a discounted service price. Employing reduced-precision models, which leads to reduced execution time, enables the SPs to lessen their on-demand instance usage, and hence, pay less for such resources. Instead, they can deploy more requests on less expensive RIs. The experiments using real-world traces emphasizes the efficacy of our proposed approach in reducing monetary cost compared to three rival approaches. The results show that *ApproxDNN* can successfully reduce monetary cost, and it has negligible effect on the accuracy of the requests.

Acknowledgment

This research was supported in part by NSF grant CNS-1755913.

References

[1] "Spot instances," https://aws.amazon.com/ec2/spot/, accessed June. 22th 2019.

[2] "Reserved instances," https://aws.amazon.com/ec2/pricing/reserved-instances/, accessed June. 22th 2019.

[3] S. Shen, V. van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," in *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 2015, pp. 465–474.

[4] S. M. Nabavinejad and M. Goudarzi, "Profit maximization of big data jobs in cloud using stochastic optimization," *IEEE Transactions on Cloud Computing*, 2019.

[5] Y. Ran, J. Yang, S. Zhang, and H. Xi, "Dynamic iaas computing resource provisioning strategy with qos constraint," *IEEE Transactions on Services Computing*, vol. 10, no. 2, pp. 190–202, 2015.

[6] S. Niu, J. Zhai, X. Ma, X. Tang, W. Chen, and W. Zheng, "Building semi-elastic virtual clusters for cost-effective hpc cloud resource provisioning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 7, pp. 1915–1928, 2015.

[7] D. Candeia, R. A. Santos, and R. Lopes, "Business-driven long-term capacity planning for saas applications," *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 290–303, 2015.

[8] S. Shen, K. Deng, A. Iosup, and D. Epema, "Scheduling jobs in the cloud using on-demand and reserved instances," in *European conference on parallel processing*. Springer, 2013, pp. 242–254.

[9] "Aws deepracer," https://aws.amazon.com/deepracer/?n=sn&p=sm, accessed November 30, 2019.

[10] "Aws deeplens," https://aws.amazon.com/deeplens/?n=sn&p=sm, accessed November 30, 2019.

[11] "Amazon ec2 instance types," https://aws.amazon.com/ec2/instance-types/, accessed June. 22th 2019.

[12] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[15] "Nvidia tensorrt," https://developer.nvidia.com/tensorrt, accessed June. 22th 2019.

[16] O. Russakovsky, J. Deng, H. Su, and J. K. et al., "ImageNet Large Scale Visual Recognition Challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[17] S. Migacz, "8-bit inference with tensorrt," in *GPU technology conference*, vol. 2, 2017, p. 7.

[18] "On-demand instances," https://aws.amazon.com/ec2/pricing/on-demand/, accessed June. 22th 2019.

[19] "Amazon compute service level agreement," https://aws.amazon.com/compute/sla/, accessed December 4, 2019.

[20] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 153–167.

[21] A. Tumanov, T. Zhu, J. W. Park, M. A. Kozuch, M. Harchol-Balter, and G. R. Ganger, "Tetrisched: global rescheduling with adaptive plan-ahead in dynamic heterogeneous clusters," in *EuroSys'16*. ACM, 2016, p. 35.

[22] J. W. Park, A. Tumanov, A. Jiang, M. A. Kozuch, and G. R. Ganger, "3sigma: distribution-based cluster scheduling for runtime uncertainty," in *EuroSys'18*. ACM, 2018, p. 2.

[23] "Amazon ec2 p3 instance product details," https://aws.amazon.com/ec2/instance-types/p3/, accessed November 30, 2019.

[24] M. Imani, M. Masich, D. Peroni, P. Wang, and T. Rosing, "Canna: Neural network acceleration using configurable approximation on gpgpu," in *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*. IEEE Press, 2018, pp. 682–689.

[25] S. Koteshwara and K. K. Parhi, "Incremental-precision based feature computation and multi-level classification for low-energy internet-of-things," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 4, pp. 822–835, 2018.

[26] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu, "Approxann: An approximate computing framework for artificial neural network," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium, 2015, pp. 701–706.

[27] A. Harlap, A. Tumanov, A. Chung, G. R. Ganger, and P. B. Gibbons, "Proteus: agile ml elasticity through tiered reliability in dynamic resource markets," in *Proceedings of the Twelfth European Conference on Computer Systems*. ACM, 2017, pp. 589–604.