GENIE: QoS-guided Dynamic Scheduling for CNN-based Tasks on SME Clusters

Zhaoyun Chen*†, Lei Luo*, Haoduo Yang*†, Jie Yu*, Mei Wen*† and Chunyuan Zhang *†
*College of Computer, National University of Defense Technology, Changsha, China
†National Key Laboratory for Parallel and Distributed Processing, Changsha, China
Email: l.luo@nudt.edu.cn

***Abstract*—Convolutional Neural Network (CNN) has achieved dramatic developments in emerging Machine Learning (ML) services. Compared to online ML services, offline ML services that are full of diverse CNN workloads are common in small and medium-sized enterprises (SMEs), research institutes and universities. Efficient scheduling and processing of multiple CNN-based tasks on SME clusters is both significant and challenging. Existing schedulers cannot predict the resource requirements of CNN-based tasks. In this paper, we propose GENIE, a QoS-guided dynamic scheduling framework for SME clusters that achieves users' QoS guarantee and high system utilization. Based on a prediction model derived from lightweight profiling, a QoS-guided scheduling strategy is proposed to identify the best placements for CNN-based tasks. We implement GENIE as a plugin of Tensorflow and experiment with real SME clusters and large-scale simulations. The results of the experiments demonstrate that the QoS-guided strategy outperforms other baseline schedulers by up to 67.4% and 28.2% in terms of QoS-guarantee percentage and makespan.**

***Index Terms*—QoS-guided, SME Cluster, Scheduling, CNN, Multi-task**

I. Introduction

The rapid advances in convolutional neural networks (CNNs) have transformed the field of machine learning. The wide availability of large training datasets and the ease of access to commodity accelerators (like GPUs) have increased CNNs' adoption in critical processes in business and research. Compared to online ML services that offer image or speech recognition leveraged by trained CNN models, such as Intelligent Personal Assistants (IPAs) [1], offline ML services that encompass CNN training and inference are common in research and development processes for most small and medium-sized enterprises (SMEs), research institutes, and universities [2]. Moreover, together with the computationally intensive nature and growth in computational requirements of CNN processing, a heterogeneous distributed cluster known as the SME Cluster is adopted to further increase the efficiency of processing in industry and academia [3]. Compared with datacenters or HPC systems, a SME cluster has a limited scale of computing nodes and has simple network topology between nodes.

Although a SME cluster provides many advantages for computing capability, it brings new challenges in task scheduling for the purpose of obtaining optimal performance. As shown in Fig. 1, in these offline ML scenarios, a heterogeneous distributed cluster is shared by different users with diverse requirements and configurations. The overall performance of the system depends on each task's Quality-of-Service (QoS) guarantee and system utilization. Most previous scheduling algorithms allocate resources based on historical and heuristics information [4]. However, these methods are not accurate in predicting the resource requirements of each task. Researchers [5] have proposed scheduling designs for batch CNN inference tasks, which are more appropriate for online ML scenarios than offline ML scenarios. Considering the specific offline ML scenarios that encompass diverse CNN-based tasks, in this paper, we propose GENIE, a QoS-guided dynamic scheduling framework to support multi-task optimizing and release users from complex task managements so that they can focus more on application development.

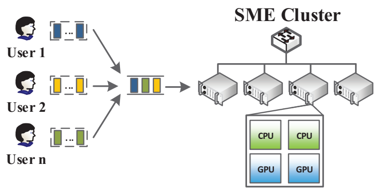


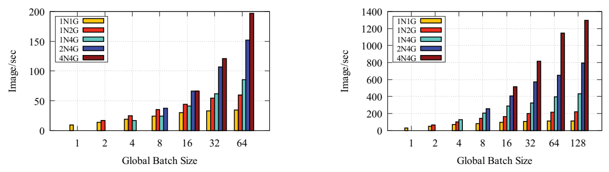
Fig. 1: Different users submit diverse CNN-based tasks to a shared SME heterogeneous cluster in offline ML scenarios.

The major contributions of this paper to the field are as follows:

- A characterization under diverse placements on a distributed system is presented to show its impacts on the computational efficiency for CNN-based tasks. Based on a lightweight profiling method, a prediction model is proposed and extended to all types of CNN-based tasks and any scale of distributed systems.
- According to the prediction model, a QoS-guided dynamic scheduling framework GENIE is proposed to identify the effective placements for tasks and to schedule them on the clusters.
- An evaluation of a real distributed cluster is presented, demonstrating that the QoS-guided scheduling strategy outperforms other baselines in terms of QoS-guarantee percentage and makespan. Moreover, a trace-driven simulation on larger distributed clusters indicates the good scalability of a QoS-guided strategy.

TABLE I: Platform Specifications

	Specifications
Node	Powerleader 4U PR4712GW Chassis * 4
System	CentOS 7.0, Tensorflow 1.7.0
CPU	Intel Xeon E5-2660 v3, 10C, 2.60GHz * 2 / node
Memory	64GB DDR3 Memory / node
GPU	NVIDIA Tesla K80 * 4 / node
Interconnection	56Gbps Infiniband, PCIe 3.0 in node, no NVlink



(a) Training (b) Inference

Fig. 2: Performances of *Inception-v3* with varying batch sizes under diverse placements. For diverse placements, *N* and *G* represent respectively the number of nodes and the number of GPUs in one node.

II. Evaluating and Modeling the Impacts of Placement Strategies on a SME cluster

Researchers [2] have attempted to provide a beneficial profiling method for CNN-based tasks on a single multi-GPU server. In order to evaluate the impacts thereof under diverse placements, we further make a comprehensive characterization of various CNN models on a distributed cluster. The specifications of the platform are shown in Table.I.

A. Evaluating the Impacts of Placement Strategies

To simplify the statement, we denote **the number of nodes** and **the number of GPUs in each node** to indicate the scale of computing resources provision in SME clusters. Based on **data parallelism**, for a CNN-based task with a fixed global batch size, the more nodes and GPUs are divided and placed on, the smaller local batch size is on each GPU. We then evaluate the impacts of CNN-based tasks under diverse placements.

We present the results of *Inception-v3* in Fig.2 to show the impacts thereof under diverse placements. The other typical CNN models have similar trends and properties. We can draw some observations from the evaluation: (1) For a single GPU in a single node, the performance (the yellow bars in Fig.2) increases at first and then plateaus when the batch size reaches a certain point, which is because the GPU's computing resources have become saturated. (2) For multi-GPUs in one node or multiple nodes, the overall performance is determined by the performance of each GPU and the communication overheads between GPUs. (3) Only when the global batch size is increasing in size, the training performance increases sharply along with the increasing number of GPUs (batch size $>$ 16). This is because the single GPU's utilization is becoming saturated and the communication frequency and overhead percentages between GPUs decrease. (4) For inference tasks, due to an absence of communication between GPUs, performance achieves almost linear scaling along with the increasing number of GPUs, as shown in Fig. 2b.

B. Modeling for the Performance of a Single CNN-based Task

Therefore, we propose an analytical model to predict the performance of a single CNN-based task under varying placements. For a given task type and a given platform, we denote $\mathbf{t}_i = \langle b_i, r_i \rangle$ of global batch size b_i and iterations r_i as a task. We denote $\mathbf{s}_{ij} = \langle n_{ij}, g_{ij} \rangle$ as the j-th placement of task $\mathbf{t}_i$, including the number of nodes n_{ij} and the number of GPUs in one node g_{ij}. As shown in the profiling results above, the overall performance of the task is determined by **the performance of a single GPU** and **the communication overheads between GPUs**. Therefore, the performance of diverse CNN-based tasks under varying placements, which is denoted by p, is described as:

$$p_{ij} = f_{\text{perf}}(\mathbf{t}_i, \mathbf{s}_{ij}) = (n_{ij} \cdot g_{ij} - o_{ij}) \cdot \bar{p}_{ij}, \tag{1}$$

where $\bar{p}_{ij}$ and o_{ij} are the performance of a single GPU and the communication overheads respectively. $\bar{p}$ is only affected by **local batch size** $\bar{b}$ assigned to the local GPU. According to the profiling results, we observe a quadratic curve relationship between them. Furthermore, the response latency l of task $\mathbf{t}_i$ under j-th placement is calculated as:

$$l_{ij} = f_{\text{lat}}(\mathbf{t}_i, \mathbf{s}_{ij}) = \frac{b_i \cdot r_i}{p_{ij}} + \nu, \tag{2}$$

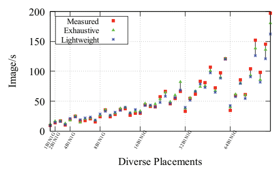
where ν represents the constant startup overhead of GPUs and $b_i \cdot r_i$ represents the total computations of the task.

When the communication overheads between nodes and GPUs are ignored, the ideal performance is the product of the total number of GPUs and the performance of the single GPU. However, the communication overheads introduce an extra performance breakdown, which is determined by the topology between nodes and GPUs. We propose **average path length between GPUs** as a penalty term to quantify the performance breakdown. The communication overheads are described as:

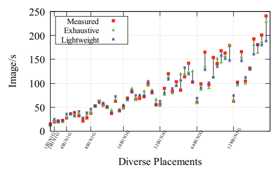
$$\begin{aligned} o_{ij} &= f_{\text{O/H}}(\mathbf{s}_{ij}) \\ &= \begin{cases} 0 & n_{ij} = g_{ij} = 1, \\ \frac{\lambda_1^{(i)}(n_{ij}-1)\cdot g_{ij} + \lambda_2^{(i)}(g_{ij}-1)}{n_{ij}\cdot g_{ij}-1} & n_{ij}, g_{ij} > 1, \end{cases} \end{aligned} \tag{3}$$

where $\lambda_1^{(i)}$ and $\lambda_2^{(i)}$ respectively represent the weights between nodes and between GPUs for calculating the penalty term. For inference tasks, $\lambda_1^{(i)}$ and $\lambda_2^{(i)}$ are equal to 0 due to the absence of communication.

Clearly, the performance prediction models are derived by fitting according to profiling results. In order to decrease the high cost of exhaustive profiling, we conduct a lightweight profiling method that can obtain sufficient results. For a given task type, assuming that there are B_i, N_i and G_i different configurations for the global batch size, the number of nodes, and the number of GPUs in one node on a SME cluster, our proposed lightweight profiler can reduce the number of experiments from $B_i \times N_i \times G_i$ to $B_i + N_i + G_i$ and achieve comparable accuracy as shown in Fig.3. Each type of CNN task only needs to be profiled once and the lightweight profiler can apply to any clusters of diverse topologies and scales, including datacenters and HPC systems.



(a) Inception-v3



(b) ResNet50

Fig. 3: *Inception-v3* and *ResNet50* training performances under diverse placements on a 4-node and 16-GPU SME cluster. 'Measured', 'Exhaustive' and 'Lightweight' respectively represent the real value and the theoretical values calculated by fitting models of exhaustive and lightweight profiling.

III. GENIE: A QoS-guided Scheduling Framework

In order to identify the proper placements for CNN tasks, we propose GENIE, a QoS-guided scheduling framework that makes decisions based on the modeling explained above.

A. User's QoS Definition

In this paper. we adopt the Expected Completion Time (ECT) e as the key indicator of the QoS. If the finish time exceeds the ECT, users' QoS satisfaction would decrease. More specifically, we denote the double baseline response latency $\bar{l}$ of the task execution on a single GPU as the normal users' ECT [6]. In emergencies of varying degrees, users can apply for higher priorities to shorten their ECTs. For task $\mathbf{t}_i$, we define three classes of user priorities and their associated ECTs e_i as:

$$e_i = \begin{cases} 0 & Urgent, \\ \bar{l}_{i1} & Prior, \\ 2\bar{l}_{i1} & Normal, \end{cases} \tag{4}$$

where $\bar{l}_{i1} = f_{\text{lat}}(\mathbf{t}_i, \langle 1, 1 \rangle)$.

B. A Cost-Effective Model with Diverse Placements

Due to the resource-intensive nature of CNN tasks, resource allocations should be efficient for multi-task scenarios. Therefore, we introduce cost model c, which is determined by the total number of GPUs and the topology between them, to judge resource occupancy. For the j-th placement $\mathbf{s}_{ij}$ of task $\mathbf{t}_i$, the cost model can be described as:

$$c_{ij} = f_{\text{cost}}(\mathbf{s}_{ij}) = \frac{n_{ij} \cdot g_{ij}}{\hat{N} \cdot \hat{G}} + \theta \frac{n_{ij}}{\hat{N}}, \tag{5}$$

where $\hat{N}$ and $\hat{G}$ represent the upper limits of n_{ij} and g_{ij} provided by the cluster. θ is the weight coefficient for adjusting the cost impacts of nodes' topology. Based on Eq.1 and 5, the cost-effective ratio (CER) r can be calculated by $r_{ij} = \frac{p_{ij}}{c_{ij}}$.

C. A QoS-guided Scheduling Strategy

Due to the limited computing resources and QoS constraints of this project, we propose a QoS-guided scheduling strategy by determining the scheduling orders and placements of tasks, **guaranteeing users' QoS and improving system utilization as best-effort delivery**.

Assuming that the current time is t_c, the tasks that arrived at time t_c are pushed into a waiting queue $\mathbf{Q}$. For a task $\mathbf{t}_i$ with ECT e_i in $\mathbf{Q}$, there are q_i placements in a placement set $\mathbf{M}_i$, whose response latencies and cost-effective ratios are respectively $l_{i1}, l_{i2}, ...l_{iq_i}$ and $r_{i1}, r_{i2}, ...r_{iq_i}$. In order to guarantee the user's QoS, we select the placements $\mathbf{s}_{ij} \in \mathbf{M}_i$ which can satisfy the QoS by $l_{ij} + t_c <= e_i$. We then push them into a subset $\mathbf{M}'_i$. The placement $\mathbf{s}_{ij^*}$ of the highest r among $\mathbf{M}'_i$ is adopted as the solution for task $\mathbf{t}_i$. A key indicator for current task priority, denoted as **waiting allowance** w, is calculated based on the placement $\mathbf{s}_{ij^*}$ by $w_i = e_i - (l_{ij^*} + t_c)$. All tasks in the waiting queue are ordered according to w and the task with the smaller w has a higher priority. Moreover, when there are no placements that can meet the user's ECT, the placement with the highest cost-effective ratio among $\mathbf{M}_i$ is adopted as the solution. Then, the scheduler monitors the cluster load and launches the tasks according to the reordered waiting queue. GPUs are adopted as exclusive accelerators and tasks have private access to GPUs. As tasks finish and new tasks arrive, the order and placement solution of each task in the waiting queue may keep changing. The overview of the scheduling is shown in Alg.1. We implemented a prototype of the scheduling framework GENIE as a plugin for Tensorflow, including our proposed scheduling strategy.

Algorithm 1: QoS-guided Scheduling

```
Input: WaitingQueue $\mathbf{Q}$, CurrentTime $t_c$
Output: ReorderdQueue $\mathbf{Q}'$
for each $\mathbf{t}_i \in \mathbf{Q}$ do
    $e_i$ = CalculateECT($\mathbf{t}_i$);
    $\mathbf{M}_i$ = GetAllPlacement($\mathbf{t}_i$);
    for each $\mathbf{s}_{ij} \in \mathbf{M}_i$ do
        $(l_{ij}, r_{ij})$ = CalculaterLatencyAndCER($\mathbf{t}_i$, $\mathbf{s}_{ij}$);
        if $l_{ij} + t_c \leq e_i$ then
            push($\mathbf{s}_{ij}$) into $\mathbf{M}'_i$;
    if $\mathbf{M}'_i \neq \varnothing$ then
        $\mathbf{s}_{ij^*}$ = SelectHighCER($\mathbf{M}'_i$);
    else
        $\mathbf{s}_{ij^*}$ = SelectHighCER($\mathbf{M}_i$);
    $w_i = e_i - (l_{ij^*} + t_c)$;
$\mathbf{Q}'$ = Reorder($\mathbf{Q}, w$);
return $\mathbf{Q}'$;
```

IV. Evaluation

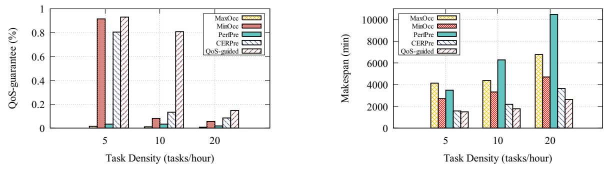
A. Experimental Setup

Platform: In this section, we adopt the same 16-GPU distributed platform, as shown in Table.I.

Workloads: Considering that no public ML traces are available, we refer to a public production trace [7] and generate task queues containing diverse CNN-based tasks with Poisson request arrivals. The specifications of task generation are shown in Table.II. For each CNN-based task, model type, calculation type, and application configurations are generated with uniform distribution.

TABLE II: Specifications of Task Queue Generation

	Specifications
Arrival Time	Poisson Distribution
Model Type	Alexnet, GoogleNet, ResNet50, Inception-v3
Calculation Type	Training, Inference
QoS-Priority Proportion	Urgent : Prior : Normal = 5% : 35% : 60%
Task Queue Duration	24 hours
Task Density	5, 10, 20 tasks/hour



(a) QoS-guarantee Percentage (higher is better) (b) Makespan (lower is better)

Fig. 4: Performance Comparisons for Diverse Scheduling Strategies

Baselines: We compare our proposed QoS-guided scheduling strategy against the following baselines. (1) MaxOcc: the scheduler allocates the maximum available resources for each task according to system load. (2) MinOcc: the scheduler allocates the minimum available resources for each task. (3) PerfPre: Considering the prediction model, the scheduler adopts the solution with the highest performance for each task. (4) CERPre: Considering the prediction and cost models, the scheduler adopts the solution with the highest r for each task.

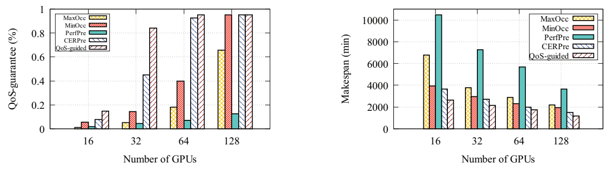
Metrics: The motivation of this paper is meeting users' QoS and improving system utilization. Therefore, the metrics we adopt here include makespan and QoS-guarantee Percentage, the percentage of the tasks that do not exceed ECTs.

B. Performance Analysis on a Real Cluster

We evaluate our proposed QoS-guided scheduling strategy with four baselines. As shown in Fig. 4, the results demonstrate that with an increasing task density, the QoS-guarantee percentage decreases and the makespan increases. MinOcc and CERPre outperform MaxOcc and PerfPre. This may be because MinOcc and CERPre focus more on resource sharing and not on single task performance. The QoS-guided strategy takes users' ECTs and cost-effective model into consideration concerning scheduling and has a stronger tolerance and robustness for diverse task densities. The QoS-guided strategy achieves improvements of up to 67.4% in QoS-guarantee percentage and 28.2% reduction in makespan than the best of the baselines. However, when the task density is too high (> 20), users' QoS is hard to guarantee due to limited computing resources. Moreover, our evaluation shows that the $95th$ percentile of accuracy error for prediction model is $< 13\%$ and the scheduling overhead is less than 1% compared with makespan.

C. Large-Scale Simulation

In order to evaluate the scalability of GENIE, we propose a trace-driven simulation on larger distributed clusters. Considering the increasing scale of clusters, the workload trace we



(a) QoS-guarantee Percentage (higher is better) (b) Makespan (lower is better)

Fig. 5: Performances of Large-Scale Simulation

adopted here was configured with 20 tasks/hour. The results are shown in Fig. 5. As the scales of the platform increase from 16 GPUs to 128 GPUs, most scheduling strategies have enough processing capabilities to meet users' QoS and decrease the makespan. However, the QoS-guided strategy can identify the most efficient placements to improve the QoS-guarantee percentage and maximize system utilization. Compared with other baselines, the results shows that the QoS-guided strategy has better scalability on diverse larger distributed clusters.

V. Conclusion

This paper presented GENIE, a QoS-guided scheduling framework for offline ML scenarios, including offline lightweight profiling and online dynamic scheduling. Based on the prediction models, GENIE identifies the best placements for CNN-based tasks and schedules them on the cluster. Numerous experiments on real clusters and simulations demonstrate that GENIE achieves a higher QoS guarantee and system utilization than other baselines. In the future, we plan to merge GENIE into cluster manager frameworks like Kubernetes, Yarn, or Mesos.

VI. Acknowledgments

The authors gratefully acknowledge supports from National Key Research and Development program under No.2016YFB1000400; National Key Program JZX2017-1585/Y479; National Nature Science Foundation of China under NSFC No. 61872377, 61802417, 61802420 and 61502509.

References

[1] J. Hauswald, et al., "Sirius: An Open End-to-End Voice and Vision Personal Assistant and Its Implications for Future Warehouse Scale Computers," in *ASPLOS*, 2015.
[2] Z. Y. Chen, et al., "Multiple CNN-based Tasks Scheduling across Shared GPU Platform in Research and Development Scenarios," in *HPCC*, 2018.
[3] S. Foghani, et al., "Promoting clusters and networks for small and medium enterprises to economic development in the globalization era," in *SAGE Open*, vol. 7, no. 1, 2017.
[4] S. A. Jyothi, et al., "Morpheus: Towards Automated SLOs for Enterprise Clusters," in *OSDI*, pp. 117-134, 2016.
[5] F. Yan, et al., "SERF: Efficient Scheduling for Fast Deep Neural Network Serving via Judicious Parallelism," in *SC*, pp. 300-311, 2016.
[6] Q. Chen, et al., "Prophet: Precise QoS Prediction on Non-Preemptive Accelerators to Improve Utilization in Warehouse-Scale Computers," in *ASPLOS*, pp. 17-32, 2017.
[7] A. Thusoo, et al., "Hive: A Warehousing Solution Over a Map-Reduce Framework," in *VLDB*, pp. 1626-1629, 2009.