SLO-aware DL Job Scheduling for Efficient FPGA-GPU Edge Cloud Computing

Taewoo Kim[1][0000−0003−4290−6460], Minsu Jeon[1][0000−0002−2739−8149], Changha Lee[1][0000−0003−3687−2989], Fawaz AL-Hazemi[2][0000−0003−3212−8941], and Chan-Hyun Youn[1][0000−0002−3970−7308]

[1] Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea
{taewoo_kim, msjeon, changha.lee, chyoun}@kaist.ac.kr
[2] University of Jeddah, Jeddah 21959, Saudi Arabia
fmalhazemi@uj.edu.sa

Abstract. Deep learning applications have become increasingly popular in recent years, leading to the development of specialized hardware accelerators such as FPGAs and GPUs. These accelerators provide significant performance gains over traditional CPUs, but their efficient utilization requires careful scheduling configuration for given DL requests. In this paper, we propose a SLO-aware DL job scheduling model for efficient FPGA-GPU edge cloud computing. The proposed model takes into account variant service-level objectives of the DL job and periodically updates the accelerator configuration of DL processing while minimizing computation costs accordingly. We first analyze the impact of various DL-related parameters on the performance of FPGA-GPU computing. We then propose a novel scheduling algorithm that considers the time-variant latency SLO constraints and periodically updates the scheduling configuration. We evaluated our scheduler using several DL workloads on a FPGA-GPU cluster. Our results demonstrated that our scheduler achieves improvements in terms of both energy consumption and SLO compliance compared to the traditional DL scheduling approach.

Keywords: DL Scheduling · FPGA-GPU Computing. · Edge Computing for DL Serving

1 Introduction

In edge cloud computing, where resources are limited and power consumption is a key issue for operators, efficient resource usage is critical for delivering high-performance deep learning (DL) services. With the increasing diversity of DL applications, such as image and natural language processing, the latency service level objective (SLO) can vary depending on the application requirements. Heterogeneous accelerators, such as field-programmable gate arrays (FPGAs) and graphics processing units (GPUs), can be used in edge cloud computing to accelerate DL tasks and reduce latency. One of the main challenges in DL job scheduling for FPGA or GPU is to ensure efficient utilization of resources

while satisfying the SLO requirements of different applications. To address this challenge, several approaches have been proposed in the literature, including batch size adjustment and spatial-temporal scheduling schemes to maximize the throughput of resources, mainly targeting a homogeneous GPU cluster. However, the effective scheduling of DL jobs on heterogeneous FPGA-GPU clusters is still a challenging problem due to its different characteristics of processing performance in terms of throughput and energy consumption.

In this paper, we present a novel scheduling approach that focuses on optimizing the use of heterogeneous accelerators while ensuring latency SLO of DL applications in FPGA-GPU edge cloud computing. Our proposed method considers various scheduling primitives depending on the type of accelerator and adjusts the scheduling configuration according to the time-varying latency SLO and DL requests. We conducted experiments on a heterogeneous FPGA-GPU cluster to evaluate the performance of our proposed scheduler.

2 Related Work and Problem Description

In this section, we discuss various approaches to handling GPU scheduling for DL jobs and some approaches addressing heterogeneous FPGA-GPU scheduling. The conventional GPU schedulers like Nexus [7] and Clippers [3] exploit a one-at-a-time scheduling mechanism, where the entire GPU resources including parallel cores and memory are occupied by a single DL job. In this situation, they fix a batch size as the maximum value which can be accepted in the GPU memory while satisfying the latency SLO to get a high throughput (requests per second). One limitation of this mechanism is that it can result in significant idle time for GPUs when serving DL jobs with low request rates, which can decrease overall resource utilization in edge cloud computing. To address the limitation, NVIDIA developed multi-process service (MPS) [1] to support concurrent execution of DL jobs over multiple workers. Dynamic spatial-temporal scheduling scheme [5] has also been shown to improve performance of DL processing. Other GPU scheduling approaches, such as GSLICE [4] and Salus [8], use fine-grained partitioning of parallel cores and spatial-temporal scheduling to maximize capacity and reduce memory occupancy for deploying given DL models. On the other hand, [6] uses layer-level management scheme on FPGA-GPU clusters to derive optimal data distribution and batch size while minimizing energy costs for heterogeneous FPGA-GPU clusters. Nevertheless, there is still potential for enhancing the effectiveness of resource utilization in the context of FPGA-GPU scheduling by implementing a concurrent execution mechanism, which can conserve computational resources in an edge cloud setting.

3 A Proposed SLO-aware DL Job Scheduling Model

In this section, we describe our proposed approach for SLO-aware DL job scheduling in FPGA-GPU computing, as illustrated in Fig. 1. The system receives DL inference jobs, each with a request set (i.e. a bunch of input data) and latency

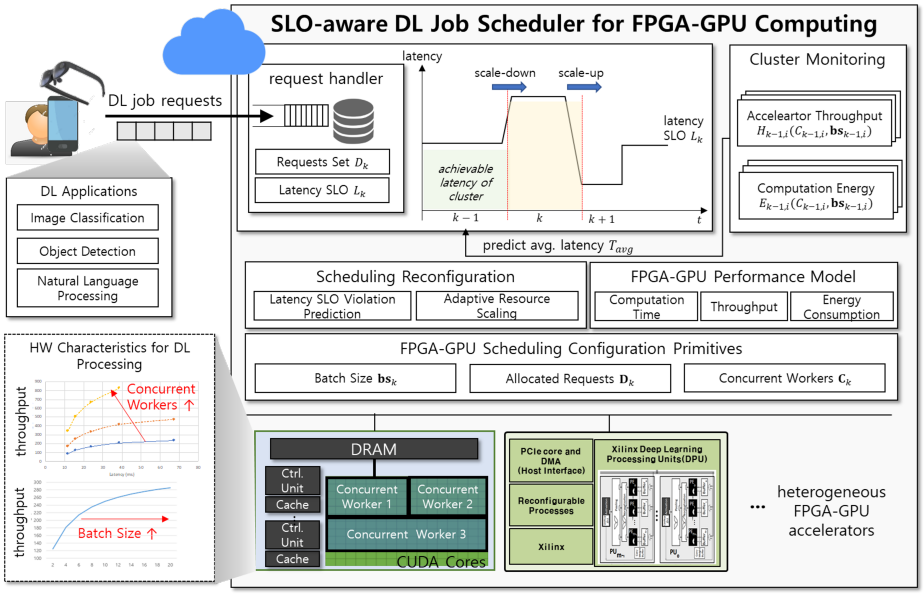

Fig. 1: The proposed architecture of SLO-aware DL job scheduling for FPGA-GPU edge cloud computing.

SLO. The latency SLO represents the required time for processing a given request set. The scheduler continuously monitors the feasibility of satisfying the SLO with the current scheduling configuration in the cluster. If it is determined that the current configuration cannot meet the required latency SLO, the scheduler updates the configuration based on FPGA-GPU performance predictions. Otherwise, it can adaptively reduce resources for computation costs in edge cloud computing when the current scheduling configuration is sufficient for serving requests.

In terms of edge cloud computing resources, we describe a computing environment in k-th time slot. We refer $\mathbf{S}_k^{GPU}$ and $\mathbf{S}_k^{FPGA}$ as a set of N_k^{GPU} GPUs and N_k^{FPGA} FPGAs available in an edge cloud environment, respectively. The total number of FPGA-GPU accelerators is $N_k = N_k^{GPU} + N_k^{FPGA}$. We assume that the available accelerators are limited for each particular time slot. The latency consumed for processing DL inference can vary depending on the state of the scheduling configuration. The latency SLO L_k and service requests $\mathcal{D}_k$ can also be time-variant but we assume that it is fixed in the same time slot. In terms of components for processing DL inference, especially in accelerators, they consist of the pre-processing time conducted in host platforms such as CPU, memory, and storage, feed-forward computation time executed in an accelerator, and return time notifying inference result to a service user. Our scheduling algorithm only considers the computation time of an accelerator since the other components are consistent regardless of accelerator type.

3.1 Performance Model for FPGA-GPU Computing

In edge cloud computing with heterogeneous FPGA-GPU accelerators, we conduct the performance modeling when processing DL inference. To reduce processing costs and maximize the availability of user requests, especially in GPU, we utilize a concurrent execution mechanism [5, 4, 8, 2] that executes multiple workers on an accelerator. In particular, it is possible to increase the overall utilization of GPU by sharing simultaneous streaming multiprocessor and global memory.

As the controllable factors for computation time in an accelerator, we consider two factors; batch size and concurrency (i.e. the number of concurrent workers). We derive the computation time $T^j_{k,i}$ for processing a single batch in j-th worker of i-th accelerator as follows:

$$T^j_{k,i}(C_{k,i}, bs^j_{k,i}) = (\alpha_i \cdot bs^j_{k,i} + \beta_i) \cdot Q_{k,i}(C_{k,i}), \tag{1}$$

where $Q_{k,i}(C_{k,i}) \in \mathbb{R}$ is slowdown ratio due to concurrent workers (i.e. $Q_{k,i}(1) = 1$), and α_i, β_i are coefficients determined by an accelerator and a DL model. In FPGA, we fix a single worker (i.e. $C_{k,i} = 1$, $\forall i \in \mathbf{S}^{FPGA}_k$). As described in $\alpha_i \cdot bs^j_{k,i} + \beta_i$, the computation time is a linear model by a batch size, and the time increases due to contention of shared resources by multiple workers. From the computation time modeling, we can derive the throughput and energy cost of an accelerator. From Eq.1, the throughput (requests per second) of i-th accelerator $H_{k,i}$ can be derive as follows:

$$H_{k,i}(C_{k,i}, \mathbf{bs}_{k,i}) = \sum_{j=1}^{C_{k,i}} \frac{bs^j_{k,i}}{T^j_{k,i}(C_{k,i}, bs^j_{k,i})} = \sum_{j=1}^{C_{k,i}} \frac{bs^j_{k,i}}{(\alpha_i \cdot bs^j_{k,i} + \beta_i) \cdot Q_{k,i}(C_{k,i})}, \tag{2}$$

where we denote $\mathbf{bs}_{k,i} \in \mathbb{R}^{C_{k,i}}$ as a set of batch size in each worker. It can be presented as the sum of the throughput of all workers simultaneously executed in i-th accelerator. From Eq. 2, the energy cost consumed for processing a unit request in i-th accelerator can be given to:

$$E_{k,i}(C_{k,i}, \mathbf{bs}_{k,i}) = \frac{P_{k,i}(C_{k,i})}{H_{k,i}(C_{k,i}, \mathbf{bs}_{k,i})} = \sum_{j=1}^{C_{k,i}} \frac{(\alpha_i \cdot bs^j_{k,i} + \beta_i) \cdot Q_{k,i}(C_{k,i})}{bs^j_{k,i}} \cdot P_{k,i}(C_{k,i}), \tag{3}$$

where the active power of i-th accelerator with $C_{k,i}$ concurrent workers is denoted as $P^{k,i}(C_{k,i})$. It implies that the energy consumption of an accelerator is dependent on j-th concurrent worker having the slowest computation time.

3.2 Proposed Scheduling Scheme

Based on the performance model of FPGA-GPU accelerators, we describe the SLO-aware DL job scheduling in k-th time slot under the aforementioned environment. The variable for the number of concurrent workers allocated to each

accelerator is denoted as $\mathbf{C}_k = \{C_{k,i} \,|\, \forall i \in \mathbf{S}_k^{FPGA} \cup \mathbf{S}_k^{GPU}\}$ and its batch size is represented as $\mathbf{bs}_k = \{bs_{k,i}^j \,|\, \forall i \in \mathbf{S}_k^{FPGA} \cup \mathbf{S}_k^{GPU}, \forall j \in [C_{k,i}]\}$. The number of requests (i.e. DL inputs) assigned to each accelerator is denoted as $\mathbf{D}_k = \{D_{k,i}^j \,|\, \forall i \in \mathbf{S}_k^{FPGA} \cup \mathbf{S}_k^{GPU}, \forall j \in [C_{k,i}]\}$. Let $|D_{k,i}| = \sum_j D_{k,i}^j$ be the total number of requests, and If $D_{k,i}^j = 0, \forall j \in [C_{k,i}]$, i-th accelerator is deactivated (i.e. there are no DL jobs allocated).

We now present the scheduling problem of edge cloud computing with heterogeneous FPGA-GPU accelerators. Given latency SLO L_k in k-th time slot, the objective is to maximize the overall throughput of FPGA-GPU accelerators while reducing the energy consumption in an edge cloud environment. We now formulate an optimization problem as follows:

$$\underset{\mathbf{C}_k, \mathbf{bs}_k, \mathbf{D}_k}{\text{maximize}} \sum_{i=1}^{N_k} \frac{|D_{k,i}|}{|\mathcal{D}_k|} \cdot \frac{H_{k,i}(C_{k,i}, \mathbf{bs}_{k,i})}{E_{k,i}(C_{k,i}, \mathbf{bs}_{k,i})}, \tag{4}$$

$$\text{subject to } \; T_{k,i}^j(C_{k,i}, bs_{k,i}^j) \cdot D_{k,i}^j \leq L_k, \; \forall i \in [N_k], \forall j \in [C_{k,i}], \tag{5}$$

$$\sum_{i=1}^{N_k} \sum_{j=1}^{C_{k,i}} D_{k,i}^j = |\mathcal{D}_k|, \tag{6}$$

where $|\mathcal{D}_k|$ is the total number of received requests. The objective function in Eq. 4 is to maximize the energy efficiency for DL processing, which is represented as achievable throughput per energy while multiplying the ratio of allocated requests as a weighting factor. This can achieve efficient resource usage in edge cloud computing with low power consumption. The constraints is follows; Eq.5 describes the satisfaction of latency SLO, which we only consider computation time constraints for simplicity, and Eq. 6 implies that the received requests should be allocated to an accelerator.

Since the optimization problem is non-convex (multiplication of control variables), we propose a heuristic configuration method with updating scheduling parameters periodically. For this purpose, we introduce a simple way to verify computing loads in the starting of k-th time slot. Given the volume of requests $\mathcal{D}_k$ and its latency SLO L_k, the scheduler evaluates the feasibility of the average expected computation time with previous scheduling parameters. If the number of requests is $|\mathcal{D}_k|$ in k-th period, the average expected computing time is as follows:

$$T_{avg} = \frac{|\mathcal{D}_k|}{\sum_i H_{k-1,i}(C_{k-1,i}, \mathbf{bs}_{k-1,i})}. \tag{7}$$

Algorithm. 1 shows the heuristics for the proposed scheduler. It utilizes Eq. 7 to determine the optimal batch size and concurrency for each accelerator that maximizes overall throughput. This configuration is set as a baseline in the initialization step, which serves as a benchmark for efficient resource utilization over time slots. Additionally, all accelerators are activated by the scheduler.

At the beginning of $k \geq 2$ time slot, the proposed approach evaluates whether the latency SLO is met by the scheduling configuration at $k-1$ by checking if

Algorithm 1 SLO-aware DL Job Scheduling for FPGA-GPU Computing.

Input: received requests $\mathcal{D}_k$, latency SLO L_k, $\mathbf{S}_k^{GPU}$ GPUs and $\mathbf{S}_k^{GPU}$ FPGAs, total number of accelerators N_k.

$k \leftarrow 1$
$\mathbf{I} \leftarrow Arr\,[0 : N_k]$.
for all $i \in \mathbf{S}_k^{GPU} \cup \mathbf{S}_k^{FPGA}$ **do** ▷ initialize
    Initiate $\mathbf{bs}_{k,i}^*, C_{k,i}^*$ maximizing Eq. 2
    $\mathbf{I}[i] \leftarrow 1$
end for
while true **do**
    $k \leftarrow k + 1$
    if $T_{avg} \geq L_k$ **then** ▷ tight workloads
        while $|\mathcal{D}_k| / \sum_i H_{k,i}(C_{k,i}, \mathbf{bs}_{k,i}) \leq L_k$ **do**
            $\mathbf{Z} \leftarrow \{i | \mathbf{I}[i] = 0\}$
            $z^* \leftarrow \arg_{z \in Z} \max H_{1,z}(C_{1,z}^*, \mathbf{bs}_{1,z}^*) / E_{1,z}(C_{1,z}^*, \mathbf{bs}_{1,z}^*)$
            $\mathbf{I}[z^*] \leftarrow 1$
            Update $\sum_i H_{k,i}(C_{k,i}, \mathbf{bs}_{k,i})$, $\forall \{i | \mathbf{I}[i] = 1\}$
        end while
    else ▷ loose workloads
        while $|\mathcal{D}_k| / \sum_i H_{k,i}(C_{k,i}, \mathbf{bs}_{k,i}) \leq L_k$ **do**
            $\mathbf{Z} \leftarrow \{i | \mathbf{I}[i] = 1\}$
            $z^* \leftarrow \arg_{z \in Z} \min H_{1,z}(C_{1,z}^*, \mathbf{bs}_{1,z}^*) / E_{1,z}(C_{1,z}^*, \mathbf{bs}_{1,z}^*)$
            $\mathbf{I}[z^*] \leftarrow 0$
            Update $\sum_i H_{k,i}(C_{k,i}, \mathbf{bs}_{k,i})$, $\forall \{i | \mathbf{I}[i] = 1\}$
        end while
    end if
    Assign $\mathcal{D}_k$ proportion to throughput of each accelerators
end while

the time required to complete the requests is within the latency SLO L_k. If the task can be completed within the SLO, the scheduler reduces the allocated accelerators one by one, starting with the least energy-efficient one. This is done to minimize power consumption by deactivating unnecessary accelerators. However, if the task cannot be completed within the required latency, the scheduler increases the allocated accelerators one by one, starting with the most energy-efficient one, until the latency SLO is met. Finally, it divides the requests $\mathcal{D}_k$ among the available accelerators in proportion to their respective throughput.

4 Performance Evaluation and Discussion

In order to evaluate the performances of the proposed scheduling algorithm in an edge cloud environment, we constructed the heterogeneous FPGA-GPU computing environments. We prepared the two server nodes that consist of 4 NVIDIA RTX 3080 GPUs, 2 NVIDIA 2080 Super GPUs, and 2 Xilinx Alveo U200 FPGAs. To evaluate the performance of the proposed scheduling method,

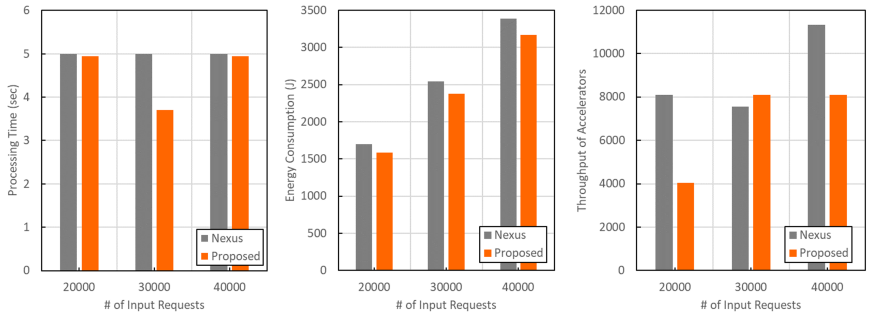

Fig. 2: Comparison of processing time (left), energy consumption (center), and throughput of accelerators (right) in FPGA-GPU cluster under relaxed latency SLO $L_k = 5$ sec.

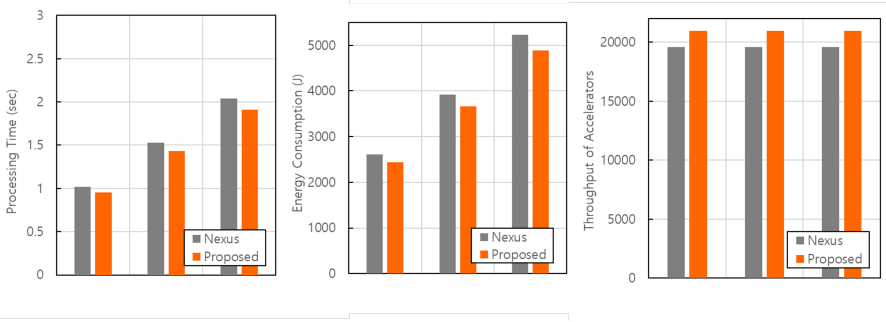

Fig. 3: Comparison of processing time (left), energy consumption (center), and throughput of accelerators (right) in FPGA-GPU cluster under tight latency SLO less $L_k = [1, 1.5, 2]$ sec.

we compared it to the scheduler in Nexus [7] as a baseline that allocates only the maximum batch size with a single worker to achieve the maximum throughput regardless of the power consumption. For performance metrics, we measured the processing time delayed from all received requests, the total energy consumption to complete the inference, and the total throughput of the allocated accelerators.

Initially, we evaluated the performance of the proposed model by varying the input loads $\mathcal{D}_k$ under a relaxed latency SLO of $L_k = 5$ sec. We measured the performance metrics for input loads $|\mathcal{D}_k| = [20000, 30000, 40000]$. Fig. 2 depicts the comparison results, where the proposed model demonstrates lower energy consumption 6.6% than the baseline by finding a better configuration through the allocation of multiple workers in each accelerator. Moreover, the proposed model allocates fewer accelerators for $|\mathcal{D}_k|$=20000 and 40000 but still satisfies the $L_k = 5$ sec latency SLO.

Furthermore, we evaluated the model's performance under a narrower latency SLO of $L_k = 2$ sec. As shown in Fig. 3, the proposed model searches for the

optimal configuration to minimize the total processing time while meeting the latency SLO. Compared to the baseline, which slightly violates the given latency SLO, the proposed scheduler reduces the processing time by 7% and shows lower energy consumption 6.5%.

5 Conclusion

In this paper, we proposed an SLO-aware DL job scheduling scheme for efficient FPGA-GPU edge cloud computing. By considering the different characteristics of heterogeneous accelerators, we aim to optimize resource usage while ensuring that the application's latency SLO is met. Through experiments on a heterogeneous FPGA-GPU cluster, we demonstrated that our proposed scheduler achieved better performance in terms of resource utilization and meeting SLO requirements compared to existing scheduling schemes. Future work could explore how to further improve the performance of our proposed scheduler by considering additional factors such as power consumption, network bandwidth, and system heterogeneity.

Acknowledgements This work is supported by Samsung Electronics Co., Ltd.

References

1. NVIDIA multi-process service. https://docs.nvidia.com/deploy/mps/index.html.
2. Choi, S., Lee, S., Kim, Y., Park, J., Kwon, Y., and Huh, J. Multi-model machine learning inference serving with gpu spatial partitioning. *arXiv preprint arXiv:2109.01611* (2021).
3. Crankshaw, D., Wang, X., Zhou, G., Franklin, M. J., Gonzalez, J. E., and Stoica, I. Clipper: A {Low-Latency} online prediction serving system. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)* (2017), pp. 613–627.
4. Dhakal, A., Kulkarni, S. G., and Ramakrishnan, K. Gslice: controlled spatial sharing of gpus for a scalable inference platform. In *Proceedings of the 11th ACM Symposium on Cloud Computing* (2020), pp. 492–506.
5. Jain, P., Mo, X., Jain, A., Subbaraj, H., Durrani, R. S., Tumanov, A., Gonzalez, J., and Stoica, I. Dynamic space-time scheduling for gpu inference. *arXiv preprint arXiv:1901.00041* (2018).
6. Kim, W.-J., and Youn, C.-H. Cooperative scheduling schemes for explainable dnn acceleration in satellite image analysis and retraining. *IEEE Transactions on Parallel and Distributed Systems 33*, 7 (2021), 1605–1618.
7. Shen, H., Chen, L., Jin, Y., Zhao, L., Kong, B., Philipose, M., Krishnamurthy, A., and Sundaram, R. Nexus: A gpu cluster engine for accelerating dnn-based video analysis. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles* (2019), pp. 322–337.
8. Yu, P., and Chowdhury, M. Salus: Fine-grained gpu sharing primitives for deep learning applications. *arXiv preprint arXiv:1902.04610* (2019).