# Learning to Dynamically Select Cost Optimal Schedulers in Cloud Computing Environments

Shreshth Tuli[1], Giuliano Casale[1], Nicholas R. Jennings[1,2]
[1]Imperial College London, UK , [2]Loughborough University, UK

## ABSTRACT

The operational cost of a cloud computing platform is one of the most significant Quality of Service (QoS) criteria for schedulers, crucial to keep up with the growing computational demands. Several data-driven deep neural network (DNN)-based schedulers have been proposed in recent years that outperform alternative approaches by providing scalable and effective resource management for dynamic workloads. However, state-of-the-art schedulers rely on advanced DNNs with high computational requirements, implying high scheduling costs. In non-stationary contexts, the most sophisticated schedulers may not always be required, and it may be sufficient to rely on low-cost schedulers to temporarily save operational costs. In this work, we propose MetaNet, a surrogate model that predicts the operational costs and scheduling overheads of a large number of DNN-based schedulers and chooses one on-the-fly to jointly optimize job scheduling and execution costs. This facilitates improvements in execution costs, energy usage and service level agreement violations of up to 11%, 43% and 13% compared to the state-of-the-art methods.

## 1 INTRODUCTION

The onset of the Artificial Intelligence (AI) and Deep Learning (DL) era has led to a recent shift in computation from hand-encoded algorithms to data-driven solutions for resource management in cloud systems [2]. This has given rise to efficiently harnessing the data processing capacities of multiple devices and provides services at scale with high Quality of Service (QoS). However, the increasing computational demands of modern Internet of Things (IoT) applications makes it crucial to curtail the operational costs of cloud machines. This calls for efficient resource management schemes, such as task scheduling policies, to execute workloads on cloud systems within tight cost budgets. AI and DL offer promising solutions that rely on accurate surrogate models that are inexpensive to evaluate at run-time.

**Background and Motivation.** In recent years, state-of-the-art resource management solutions, which particularly focus on optimal placement of tasks on cloud virtual machines (VMs), have increasingly explored data-driven DL methods [7, 9, 10, 11, 12]. Such methods typically rely on a trace of resource utilization characteristics and QoS metrics of tasks and cloud hosts and are referred to as *trace-driven* schedulers. They utilize such traces to train a deep neural network (DNN) to estimate a set of Quality of Service (QoS) parameters and run optimization strategies to find the best scheduling decision for each incoming task. However, most prior work run inference using such trace-driven DNN models on a broker node to generate scheduling decisions, while the incoming tasks are executed on worker nodes [11]. In such cases, having a dedicated broker is often cost inefficient due to the idle times between decisions in discrete-time control settings, wherein the task placement decisions are taken at fixed scheduling intervals [11]. To tackle this, we resort to paradigms such as Function as a Service (FaaS) that allow execution of DL schedulers as serverless functions, only incurring costs for the inference run time of each DNN model.

**Contributions.** In this work, to leverage the recent advantages brought by DL in task scheduling and build a policy agnostic solution, we choose a set of state-of-the-art schedulers. This work aims to solve the meta-problem of on-the-fly selection of scheduling policies for a cloud computing environment wherein the incoming tasks are executed on worker nodes and scheduling decisions are run as serverless functions. To solve this meta-problem, we develop the proposed solution that we call MetaNet. It uses a DNN as a surrogate model to predict the task execution costs and scheduling time for each policy. We then select the most cost-efficient policy at each scheduling interval using the online estimates generated by this surrogate. As we dynamically update the policy to tradeoff between simple and sophisticated DL schedulers, we reduce overall operational costs, energy consumption and Service Level Agreement (SLA) deadline violation rates by up to 11%, 43% and 13% respectively compared to state-of-the-art schedulers.

## 2 BACKGROUND AND RELATED WORK

Recent work in scheduling for cloud computing environments has demonstrated that AI-based solutions are not only faster, but can also scale efficiently compared to traditional heuristic and optimization techniques [9, 10, 11, 12].

**Evolutionary Optimization.** This class of methods forecasts QoS using non-DL solutions such as ARIMA [8] or DL based models such as Long-Short-Term-Memory (LSTM) neural networks [6]. It then applies evolutionary search strategies such as Ant Colony Optimization (ACO) [1], to converge to a locally optimal scheduling decision.

**Surrogate Optimization.** This class of methods uses differentiable function approximators, particularly neural networks, to act as surrogates of the QoS for a future state of the system. For instance, GRAF [7] uses a graph neural net-


Figure 1: MetaNet System Model.

work (GNN) as a surrogate model to predict service latencies and operational costs for a given scheduling decision and uses gradient-based optimization to minimize the service latencies or execution costs. To do this, it uses the concept of neural network inversion [13], wherein the method evaluates gradients of the objective function with respect to inputs and runs optimization in the input space. Other methods, such as Decision-NN [12], GOBI [11] and its second-order generalization GOSH [10], combine the prediction and optimization steps by modifying the loss function to train and optimize in tandem.

## 3 PROPOSED METHOD

**System Model.** In this work, we target a standard heterogeneous cloud computing environment where all nodes are in the same Wide Area Network (see Figure 1 for an overview). Tasks are container instances that need to be processed, generated from an IoT layer and transferred to the compute nodes via gateway devices. We do not have any cloud broker; instead, we run the scheduling policies as FaaS functions on a serverless platform such as AWS Lambda or Azure Serverless. As is common in prior work [1, 9, 12], we consider a discrete-time control problem, *i.e.*, we divide the timeline into fixed-size execution intervals (of $\Delta$ seconds). We denote the overall cost of the system amortized by the number of completed tasks in the $t$-th interval as $\phi_t$. This includes the operational costs of the worker nodes as well as that of running the schedulers as serverless functions.

**MetaNet Methodology.** We create a surrogate model $f_\theta$ that is a DNN with parameters $\theta$. We consider a set of scheduling policies $\mathcal{P}$ of size $q$. Given a system state at the start of the $t$-th interval, each scheduler produces a scheduling decision for this interval. The state is denoted by $C_{t-1}$ and includes the resource (CPU, RAM and disk) utilization characteristics of the active workloads in the system $W_{t-1}$, resource utilization of cloud hosts $H_{t-1}$ and scheduling decision of the previous interval $S_{t-1}$. Thus, $C_{t-1} = [W_{t-1}, H_{t-1}, S_{t-1}]$. Now, $f_\theta$ estimates the average execution cost (sum of the cost of task execution and serverless run) for each scheduler $p^k \in \mathcal{P}$ in the $t$-the interval as $\hat{\phi}_t^k$. The collection of $\hat{\phi}_t^k$ for all $k$ in $I_t$ is denoted by $\hat{\phi}_t$. In such a case, our problem can be formulated as

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \sum_{t=1}^{T} \phi_t^\pi \\ \text{subject to} \quad & S_t = p^k(t), \forall\ t \\ & \pi = \arg\min_k \hat{\phi}_t^k, \forall\ t \\ & \hat{\phi}_t = f_\theta(W_{t-1}, H_{t-1}, S_{t-1}), \forall\ t, \end{aligned} \tag{1}$$

where $f_\theta$ is a surrogate of the average execution cost.

**Surrogate Model and Training.** We use a feed-forward neural model that takes as input the state of the cloud system $C_{t-1} = [W_{t-1}, H_{t-1}, S_{t-1}]$ and outputs a single scalar. We use a fully connected neural network with 3 layers, each with 64 nodes and ReLU activation, except the last layer

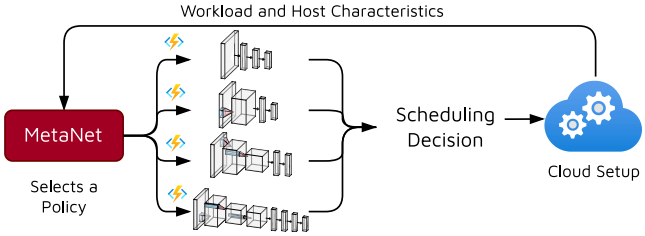


Figure 2: MetaNet Pipeline.

where we use the Sigmoid activation to bring the output in the range $(0, 1)$. Thus, for any input $(W_{t-1}, H_{t-1}, S_{t-1})$, the complete neural network can be described as

$$\hat{\phi} = f_\theta(W_{t-1}, H_{t-1}, S_{t-1}). \tag{2}$$

To train the above described MetaNet neural network $f_\theta$, we collect traces from a cloud computing environment. To collect data for training, we execute the scheduling policies $\mathcal{P}$, each for $\Gamma$ scheduling intervals, and collect a dataset as

$$\Lambda = \{(k, W_{t-1}, H_{t-1}, S_{t-1}, \phi_t^k, \omega_t^k)\}_{t=1}^{\Gamma \cdot q}, \tag{3}$$

where $q$ is the number of scheduling policies in $\mathcal{P}$. We initialize $W_0$ as $H_0$ zero-matrices and $S_0$ as an empty graph. We find the maximum cost from the dataset as $\phi_{max}^k$ for each scheduler $p^k$. This allows us to denormalize the neural network output and bring it to the same range as the one in the dataset. We then train the model using the loss function

$$L = \|\phi^k - \phi_{max}^k \cdot \hat{\phi}^k\|_2. \tag{4}$$

**Dynamic Policy Selection Using MetaNet.** We first store the saved surrogate model $f_\theta$ into a central network-attached-storage (NAS) that is accessible to all worker nodes. At start the scheduling interval $I_t$, we get the workload and host characteristics $W_{t-1}, H_{t-1}$ with the scheduling decision $S_{t-1}$. As we do not have any broker node in our setup, we run the MetaNet model in the worker node with the least CPU utilization. On this worker node we use the surrogate model to predict cost and decide the scheduling policy as

$$p^\pi, \text{ s.t. } \pi = \arg\min_k \hat{\phi}_t^k. \tag{5}$$

Overall, MetaNet selects a scheduling policy on-the-fly as per the system states. To do this, at the start of each scheduling interval, it predicts the task execution cost and scheduling time of each policy. It then selects the one with the minimum cost estimate. This policy optimizes the scheduling decision, initialized as $S_{t-1}$, to generate $S_t$. The complete pipeline is summarized in Figure 2. We also tune $f_\theta$ with each new datapoint to adapt to non-stationary settings.

## 4 PERFORMANCE EVALUATION

**Setup.** We consider the complete set of hosts $\mathcal{H}$ to be static with time as is common in prior work for a fixed cloud platform [9]. We use different VM types from the Microsoft Azure Platform, *i.e.*, 60 `Azure B2s` with a dual-core CPU and 4GB RAM (in UK-South), 20 `Azure B4ms` with a quad-core CPU and 16GB RAM and 20 `Azure B8ms` with an octa-core CPU and 32 GB RAM (in East-US). To save on costs, we define a host to be active when the CPU utilization is $> 0\%$. We utilize the `Azure Automation` (`https://azure.microsoft.com/services/automation/#overview`) service to hibernate and resume VMs based on their CPU utilization.

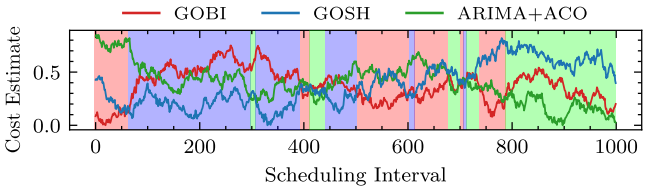


Figure 3: Predicted costs (line plots) and dynamic scheduler selection (background color) in MetaNet.

**Workloads.** To generate the tasks in our system, we use the *AIoTBench* applications [5]. AIoTBench is an AI based cloud computing benchmark suite that consists of various real-world computer vision application instances. The seven specific application types correspond to the neural networks they utilize. These include three typical heavy-weight networks: ResNet18, ResNet34, ResNext32x4d, as well as four light-weight networks: SqueezeNet, GoogleNet, MobileNetV2, MnasNet. At the start of each scheduling interval, we create new tasks, sampled uniformly from the seven applications, where the number of new tasks comes from a Poisson distribution with rate $\lambda = 1.2$.

**Baselines.** We compare MetaNet against seven baselines, which also form our policy set $\mathcal{P}$. We integrate the ACO algorithm with two demand forecasting methods: AutoARIMA and LSTM, and call these ARIMA+ACO and LSTM+ACO. We also include Decision-NN, Semi-Direct and GRAF, GOBI and GOSH. This makes the size of the policy set $q = 7$. We also use a Multi-Armed Bandit (MAB) model using Upper-Confidence-Bound exploration [3] and a Deep-Q-Network (DQN) [4] that choose a policy based on pre-trained models using data $\Lambda$.

**Implementation and Training.** We use implement MetaNet on the COSCO framework [11]. We collect the dataset $\Lambda$ by executing all baselines for $\Gamma = 100$ intervals. Similarly, we execute all approaches for $T = 1000$ scheduling intervals to generate QoS scores, with each interval being $\Delta = 10$ seconds long, giving a total experiment time of nearly 2 hours 46 minutes for each method.

**Visualization of MetaNet.** Figure 3 visualizes the MetaNet approach running on the setup and workloads described above. The x-axis denotes the scheduling interval and the y-axis denotes the cost estimate $\hat{\phi}_t^k$ for the three most frequently selected policies in $\mathcal{P}$ for readability: ARIMA+ACO, GOBI and GOSH. The highlighted bands indicate the selected scheduling policy. The selected policy corresponds to the one with the least estimated cost. The cost estimates are non-stationary, further corroborating the need for dynamic selection of the scheduling policies in volatile workload and host setups.

**Comparison with Baselines.** Table 1 shows the results of our experiments (cost in USD, energy in KW-hr, resp. time in seconds). Across all metrics, MetaNet outperforms the baselines. MetaNet improves the energy consumption by allocating tasks, *i.e.*, to the same hosts to minimize execution costs and consequently the active hosts in the system. This is shown by the highest CPU utilization of MetaNet, *i.e.*, 87.4%. MetaNet also gives the lowest response time and consequently the lowest SLA violation rates. Dynamic optimization baselines perform poorly due to the stateless assumption in MAB that ignores environment dynamism, and DQN being slow to adapt in volatile settings [11].

Table 1: Comparison with the baseline.

| Model | Cost | Energy | Resp. T. | SLA V. | CPU % |
|---|---|---|---|---|---|
| ARIMA+ACO | 0.794 | 5.432 | 20.391 | 0.241 | 62.4 |
| LSTM+ACO | 0.862 | 5.217 | 16.921 | 0.212 | 76.5 |
| Decision-NN | 1.072 | 4.224 | 12.255 | 0.211 | 80.2 |
| Semi-Direct | 1.021 | 4.095 | 17.092 | 0.131 | 72.2 |
| GRAF | 0.993 | 4.228 | 14.722 | 0.127 | 76.1 |
| GOBI | 0.752 | 3.827 | 14.293 | 0.122 | 80.2 |
| GOSH | 0.911 | 3.267 | 13.292 | 0.118 | 84.3 |
| MAB | 0.874 | 2.921 | 13.921 | 0.121 | 79.9 |
| DQN | 0.907 | 3.121 | 13.877 | 0.119 | 80.4 |
| MetaNet | **0.667** | **2.029** | **11.273** | **0.102** | **87.4** |

## 5 Conclusions

This work presents MetaNet, which dynamically selects scheduling policies for cost-efficient task processing in cloud systems. Future work would explore other DNNs as surrogates and extend MetaNet to other types of resource management decisions such as dynamic VM provisioning and autoscaling.

## References

[1] Muhammad Aliyu et al. "Efficient metaheuristic population-based and deterministic algorithm for resource provisioning using ant colony optimization and spanning tree". In: *International Journal of Cloud Applications and Computing (IJCAC)* 10.2 (2020), pp. 1–21.

[2] Hongming Cai et al. "IoT-based big data storage systems in cloud computing: perspectives and challenges". In: *IEEE Internet of Things Journal* 4.1 (2016).

[3] Volodymyr Kuleshov and Doina Precup. "Algorithms for multi-armed bandit problems". In: *arXiv:1402.6028* (2014).

[4] Yuxi Li. "Deep reinforcement learning: An overview". In: *arXiv:1701.07274* (2017).

[5] Chunjie Luo et al. "AIoT bench: towards comprehensive benchmarking mobile and embedded device intelligence". In: *International Symposium on Benchmarking, Measuring and Optimization*. Springer. 2018, pp. 31–35.

[6] Soukaina Ouhame, Youssef Hadi, and Arif Ullah. "An efficient forecasting approach for resource utilization in cloud data center using CNN-LSTM model". In: *Neural Computing and Applications* (2021), pp. 1–13.

[7] Jinwoo Park et al. "GRAF: a graph neural network based proactive resource allocation framework for SLO-oriented microservices". In: *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*. 2021, pp. 154–167.

[8] Parminder Singh, Pooja Gupta, and Kiran Jyoti. "TASM: Technocrat ARIMA and SVR model for workload prediction of web applications in cloud". In: *Cluster Computing* 22.2 (2019), pp. 619–633.

[9] Peter J Stuckey et al. "Dynamic programming for predict+optimise". In: *AAAI*. Vol. 34. 02. 2020.

[10] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. "GOSH: Task Scheduling using Deep Surrogate Models in Fog Computing Environments". In: *IEEE Transactions on Parallel and Distributed Systems* (2022).

[11] Shreshth Tuli et al. "COSCO: Container Orchestration Using Co-Simulation and Gradient Based Optimization for Fog Computing Environments". In: *IEEE Transactions on Parallel and Distributed Systems* 33.1 (2021), pp. 101–116.

[12] Bryan Wilder, Bistra Dilkina, and Milind Tambe. "Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization". In: *AAAI*. Vol. 33. 01. 2019.

[13] Eric Wong and J Zico Kolter. "Neural network inversion beyond gradient descent". In: *Advances in Neural Information Processing Systems, Workshop on Optimization for Machine Learning* (2017).