Indian Journal Of Artificial Intelligence Research (INDJAIR)
Volume 2, Issue 2, July-December 2022, pp. 1-9. Article ID: INDJAIR_02_02_001
Available online at https://iaeme.com/Home/issue/INDJAIR?Volume=2&Issue=2
Journal ID: 4739-1529

DEVELOPING AN EFFICIENT SCHEDULING MODEL FOR CLOUD-BASED AI TRAINING USING PREDICTIVE WORKLOAD PATTERNS

Andrew Jason
USA.

ABSTRACT

Cloud-based AI training introduces unique challenges in managing computing resources efficiently under dynamic workloads. Predictive workload models can anticipate resource needs and optimize scheduling, reducing latency and energy consumption. This paper proposes a predictive scheduling model integrating LSTM-based workload forecasting and dynamic resource allocation tailored for AI training in cloud environments. We evaluate its performance against conventional models using real trace data, showing significant improvements in resource utilization and training throughput.

Keywords: AI training, cloud computing, predictive scheduling, workload modeling, LSTM, resource optimization

Cite this Article: Andrew Jason. Developing an Efficient Scheduling Model for Cloud-Based AI Training Using Predictive Workload Patterns. *Indian Journal of Artificial Intelligence Research (INDJAIR)*, 2(2), 2022, pp. 1-9.

https://iaeme.com/MasterAdmin/Journal_uploads/INDJAIR/VOLUME_2_ISSUE_2/INDJAIR_02_02_001.pdf

1. Introduction

The exponential rise in demand for cloud-based AI model training has led to complex challenges in resource scheduling due to unpredictable and non-linear workload patterns. AI training, particularly deep learning, involves compute-intensive tasks that can spike based on dataset sizes, training iterations, and concurrent processes. Static scheduling approaches often lead to resource underutilization or bottlenecks.

To address this, cloud providers seek intelligent scheduling models that incorporate **predictive workload analysis**. By forecasting workload patterns using time-series models (e.g., LSTM, GRU), the system can allocate resources just-in-time, improving efficiency and reducing costs.

This paper develops and evaluates a scheduling model using LSTM-based prediction integrated with a dynamic resource manager. The proposed solution aligns with the broader movement toward **autonomous cloud infrastructure** for AI services.

2. Literature Review

Effective resource scheduling in cloud environments, particularly for AI workloads, has drawn considerable attention due to the rise of data-intensive machine learning tasks. The literature before 2021 showcases a transition from static scheduling to intelligent, predictive models leveraging deep learning and reinforcement learning. This review categorizes foundational works into **three main trends**: (1) Predictive Modeling for Workload Forecasting, (2) Intelligent Scheduling Mechanisms, and (3) Cloud-AI System Optimization Frameworks.

2.1 Predictive Modeling for Workload Forecasting

Early research emphasized forecasting techniques that could predict cloud workloads by learning temporal patterns. Liu et al. (2017) developed an adaptive prediction model capable of distinguishing workload classes, a key step toward automating cloud orchestration. Similarly, Bhagavathiperumal et al. (2019) applied time-series models, showing how workload variation patterns could be used for early provisioning.

Lu et al. (2019) advanced this with a GRU-based framework, enabling fine-grained predictions that considered daily and hourly cyclic trends. Kumar and Umamaheswari (2020) further improved prediction accuracy using LSTM architectures trained on nonlinear patterns observed in enterprise cloud data.

These models emphasized that **accurate temporal forecasting is critical** to enable just-in-time resource allocation.

2.2 Intelligent Scheduling Mechanisms

As predictive models matured, focus shifted toward **integration with scheduling frameworks**. Wei et al. (2018) proposed DRL-scheduling, a deep reinforcement learning-based scheduler that optimized Quality-of-Service (QoS) by learning policy decisions from experience. Al-Asaly et al. (2019) suggested a cognitive provisioning approach where intelligent agents adjusted resource allocation based on inferred demand.

Tuli et al. (2020) contributed a hybrid architecture using A3C reinforcement learning combined with residual neural networks. This allowed dynamic adaptation under stochastic environments, ideal for AI training tasks that are inherently unpredictable.

These works indicate a clear **paradigm shift from reactive to proactive scheduling**, enabling cloud systems to adapt in real-time.

2.3 Cloud-AI System Optimization Frameworks

Another stream focused on **framework-level design** to handle end-to-end AI training pipelines in cloud environments. Masdari and Khoshnevis (2020) provided a survey categorizing workload forecasting techniques and emphasized the lack of AI-specific schedulers.

He et al. (2018) proposed "HaaS," a real-time analytics platform with a heterogeneity-aware scheduler—addressing hardware diversity in cloud datacenters. Likewise, Hummer et al. (2019) proposed "ModelOps," a lifecycle management approach that ensured reliable AI execution under varying resource and performance constraints.

Chen et al. (2020) leveraged reinforcement learning for feedback-based scheduling and introduced QoS-aware metrics into resource orchestration.

3. Model Architecture

3.1 System Design

The proposed system comprises:

- **Predictive Layer:** Uses LSTM trained on historical traces (e.g., Google Cluster Trace).
- **Scheduling Engine:** Translates predicted loads into resource demands.

- **Deployment Layer:** Assigns cloud nodes dynamically via Kubernetes.

4. Experimental Setup

To validate the effectiveness of the proposed predictive scheduling model for AI training workloads in cloud environments, a carefully structured experimental framework was implemented. The experiment was designed to evaluate both the forecasting accuracy of the LSTM model and the practical efficiency of the dynamic scheduler integrated with it. The setup includes real-world workload traces, a custom simulation environment, and comparative baseline models for evaluation.

The experimental pipeline is composed of three main stages: (1) data acquisition and preprocessing, (2) training and testing of forecasting models, and (3) deployment of scheduling algorithms in a simulated cloud cluster. Each stage is optimized to mimic real-world cloud conditions as closely as possible, with a special focus on resource-intensive AI workloads.

4.1 Dataset

The Google Cluster Trace dataset was chosen for this study due to its rich detail, high volume, and real-world representativeness. This dataset contains logs of over 40 million tasks executed across 12,000 physical machines in a Google datacenter over a 29-day period. These tasks vary in complexity and are well-suited for exploring both regular and bursty workload patterns typical of AI model training.

The dataset includes valuable fields such as:

- Job and task identifiers
- Resource requests and usage metrics (CPU, memory, disk)
- Task start and end times
- Job scheduling class and priority
- Evictions and failures

This wealth of information allows us to simulate job submission behaviors, predict workload dynamics, and test how well the proposed scheduling model adapts under different stress conditions.

4.2 Data Preprocessing and Feature Engineering

To transform the raw traces into a form suitable for machine learning and scheduling analysis, several preprocessing steps were applied:

- **Task Filtering:** Only tasks with high CPU/memory requests (indicative of AI training jobs) were selected.
- **Time-Series Construction:** Task metrics were resampled into 5-minute intervals to form continuous time-series suitable for LSTM training.
- **Temporal Encoding:** Features such as "hour of day" and "day of week" were encoded to capture periodic workload fluctuations.
- **Normalization:** All continuous features (CPU, memory) were scaled using Min-Max normalization to accelerate LSTM convergence.
- **Train-Test Split:** Data was split into 80% training and 20% testing, preserving time order to prevent data leakage.

This preprocessed dataset serves as the input for both the forecasting models and the workload-driven simulation of the cloud scheduler.

4.3 Forecasting Models Setup

The primary forecasting model used is a Long Short-Term Memory (LSTM) neural network, trained on sequences of resource usage data. Key configurations include:

- **Input sequence length:** 30 time steps (2.5 hours)
- **Prediction window:** 10 future steps (50 minutes ahead)
- **Loss function:** Mean Squared Error (MSE)
- **Optimizer:** Adam
- **Epochs:** 50
- **Batch size:** 64

Baseline models include:

- **GRU (Gated Recurrent Unit):** Similar setup as LSTM for fair comparison.
- **ARIMA:** Traditional statistical model trained on CPU usage only.

4.4 Scheduler Simulation Environment

To evaluate the real-world impact of predictions, we built a Python-based cloud simulation environment that mimics Kubernetes-like behavior:

- **Scheduler logic:** Allocates containers to nodes based on predicted vs. actual workload.
- **Node pool:** 500 virtual machines with variable CPU/memory configurations.
- **Job queue:** Populated using the Google Cluster job submission patterns.

Three scheduling strategies were tested:

- **Static Scheduling:** Based on average historical demand
- **Greedy Scheduling:** Allocates max resources to each incoming job
- **LSTM-Based Predictive Scheduling:** Uses real-time workload forecasts to dynamically adjust allocations

Performance was monitored across key metrics such as average latency, resource utilization, and deadline violations.

5. Results and Evaluation

In this section, we present a comprehensive evaluation of our predictive scheduling model using the LSTM architecture. The model is assessed based on two primary aspects: **workload forecasting accuracy** and **scheduling performance under varying AI workload patterns**. Our goal is to demonstrate not only that the model can effectively forecast future resource demands but also that these forecasts can be operationalized to enhance overall cloud scheduling efficiency.

We compare the LSTM model with baseline techniques such as GRU (Gated Recurrent Unit) and ARIMA (AutoRegressive Integrated Moving Average). Each model was trained and tested on the preprocessed Google Cluster Trace dataset. Evaluation metrics for forecasting include **Mean Absolute Error (MAE)** and **Root Mean Square Error (RMSE)**. For scheduling effectiveness, we analyze **resource utilization**, **average job latency**, and **deadline violation rate**.

5.1 Workload Prediction Accuracy

The LSTM model achieved strong results in forecasting CPU and memory usage, which are crucial for resource scheduling in AI training tasks. With its ability to capture both short-term fluctuations and long-term temporal dependencies, LSTM consistently outperformed both ARIMA and GRU across all evaluation metrics. Specifically, LSTM achieved a **MAE of 1.43** and an **RMSE of 1.87**, indicating low average deviation from actual values.

Comparatively, the GRU model also performed well but was slightly less accurate, recording a MAE of 1.52. The ARIMA model, while effective for linear trends, struggled with the nonlinear and bursty patterns typical of AI workloads, showing a significantly higher MAE

of 2.65. These results validate the suitability of LSTM for cloud environments where **workload patterns are irregular and dynamic**.

Model	MAE	RMSE
LSTM	1.43	1.87
GRU	1.52	1.96
ARIMA	2.65	3.44

The graph below illustrates the prediction trends over a 24-hour period, comparing actual vs. forecasted CPU usage. LSTM predictions closely follow the true workload curve, with minor deviations during burst intervals. This forecasting fidelity is critical to ensuring accurate scheduling decisions downstream.

5.2 Scheduler Performance Evaluation

To assess how the predictions improve scheduling, we implemented a custom dynamic scheduler that reacts to the 10-step-ahead LSTM forecasts. The results were benchmarked against a **static scheduling algorithm**, where resources are allocated based on average past usage, and a **greedy algorithm**, which assigns maximum resources on job arrival without prediction.

The LSTM-enhanced scheduler yielded notable improvements in key performance indicators. It achieved an **average resource utilization of 91%**, significantly higher than the static model's 72%. This demonstrates that predictive scheduling minimizes idle resources while also avoiding over-provisioning. Additionally, the **average job latency dropped by 41.5%**, from 530 ms (static) to 310 ms (LSTM), indicating faster execution of training jobs.

Metric	Static Scheduler	LSTM-Based Scheduler
Avg. Latency (ms)	530	310
Resource Utilization	72%	91%

Deadline Violations	17%	4.5%

Perhaps most significantly, the deadline violation rate—a critical metric in training pipelines where jobs are scheduled sequentially—fell to just 4.5% in the LSTM model. This contrasts sharply with the 17% violation rate observed in static scheduling. The lower rate of missed deadlines implies smoother training workflows and less need for reallocation or re-training, which can be costly in production environments.

The results strongly suggest that integrating time-series predictions into scheduling logic can **dramatically improve operational efficiency and reliability** in AI training systems. This approach is especially beneficial in multi-tenant cloud platforms, where demand fluctuation and contention are common.

6. Conclusion

Predictive scheduling is key to managing resource-hungry AI training workflows in cloud environments. The proposed LSTM-enhanced scheduling model demonstrated superior performance in handling fluctuating workloads, reducing latency, and optimizing cloud resource use. Future work includes exploring federated learning to decentralize model updates and enhance data privacy.

References

[1] Wei, Y., Pan, L., Liu, S., Wu, L., Meng, X. (2018). DRL-scheduling: An intelligent QoS-aware job scheduling framework for applications in clouds. *IEEE Access*, 6, 55112–55125.

[2] Subramanyam, S.V. (2019). The role of artificial intelligence in revolutionizing healthcare business process automation. International Journal of Computer Engineering and Technology (IJCET), 10(4), 88–103.

[3] Al-Asaly, M.S., Hassan, M.M., Alsanad, A. (2019). A cognitive/intelligent resource provisioning for cloud computing services. *Soft Computing*, 23, 10151–10163.

[4] Tuli, S., Ilager, S., Ramamohanarao, K., Buyya, R. (2020). Dynamic scheduling for stochastic edge-cloud computing environments. *IEEE Transactions on Mobile Computing*, 19(10), 2358–2372.

[5] Liu, C., Shang, Y., Chen, S., Cheng, B. (2017). Adaptive prediction approach based on workload pattern discrimination. *Journal of Network and Computer Applications*, 89, 1–12.

[6] Subramanyam, S.V. (2022). AI-powered process automation: Unlocking cost efficiency and operational excellence in healthcare systems. International Journal of Advanced Research in Engineering and Technology (IJARET), 13(1), 86–102.

[7] Lu, Y., Liu, L., Panneerselvam, J., Yuan, B. (2019). A GRU-based prediction framework for intelligent resource management. *IEEE Transactions on Services Computing*, 13(6), 1040–1052.

[8] Kumar, K.D., Umamaheswari, E. (2020). HPCWMF: Hybrid predictive cloud workload management using improved LSTM. *Cybernetics and Information Technologies*, 20(4), 37–47.

[9] Chen, Z., Min, G. (2020). Resource allocation using prediction-enabled feedback control. *IEEE Transactions on Parallel and Distributed Systems*, 31(10), 2363–2376.

[10] Subramanyam, S.V. (2021). Cloud computing and business process re-engineering in financial systems: The future of digital transformation. International Journal of Information Technology and Management Information Systems (IJITMIS), 12(1), 126–143.

[11] Masdari, M., Khoshnevis, A. (2020). A survey of workload forecasting methods in cloud computing. *Cluster Computing*, 23(3), 2393–2410.

[12] Garg, S.K., Toosi, A.N., Gopalaiyengar, S., Buyya, R. (2014). SLA-based virtual machine management. *Journal of Network and Computer Applications*, 45, 108–120.

[13] He, J., Chen, Y., Fu, T.Z.J., Long, X. (2018). Haas: Real-time analytics with heterogeneity-aware scheduling. *IEEE ICPADS*, 196–203.

[14] Yin, J., Lu, X., Chen, H., Zhao, X., Xiong, N.N. (2014). System resource prediction under bursty workloads. *Information Sciences*, 278, 708–725.

[15] Bhagavathiperumal, S., Goyal, M. (2019). Workload analysis using ML prediction. *IEEE APSIPA*, 896–903.

[16] Christidis, A., Moschoyiannis, S., Hsu, C.H. (2020). Enabling serverless AI workloads. *IEEE Access*, 8, 91123–91136.

[17] Herbst, N., Amin, A., Andrzejak, A., Grunske, L. (2017). Online workload forecasting. *Lecture Notes in Computer Science*, 10070, 219–233.

[18] Hummer, W., Muthusamy, V., Rausch, T. (2019). ModelOps: Cloud lifecycle for AI. *IEEE IC2E*, 11–20.