International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P113
Eureka Vision Publication | Volume 3, Issue 4, 134-143, 2022

Original Article

Cost-Aware Autoscaling for Batch vs. Online Inference

Rohit Reddy Gaddam
Sr. Site Reliability Engineer, USA.

Abstract - Autoscaling is basically a key feature of the modern machine learning environment that is most valued in the deployment of ML models, but it is still a challenge to apply in those delicate inference workloads that are simply a mixture of performance, reliability, and cost. Online inferences that are run in real-time are required to scale very fast to meet any sudden demands, whereas batch inferences, which are typically done by processing large volumes of data at scheduled intervals, usually have a more predictable scaling pattern. The trade-off between the two types of inferences is rather obvious: on the one hand, online inference requires low latency, which is something that cloud providers charge a lot for; on the other hand, batch inference is much more cost-efficient but is less responsive to real-time needs. The existing autoscaling methods generally rely on throughput as well as on latency metrics but they neglect cost-awareness as an issue of the first priority, especially in a situation where the workload has to be changed from batch to online mode without any interruption. This work is aimed at this specific gap and it does so by introducing a cost-aware autoscaling mechanism that is not only based on the demand but also considers the cost-performance trade-offs. It uses workload profiling, predictive scaling policies, and adaptive scheduling to manage and keep the right balance between efficiency and the capability of quick response. The Study of a production-scale machine learning system is an example of how this framework can lower the expenses involved in operations by managing it well when batch and online inference are the ones that require differentiation while at the same time meeting all the levels of service needed.

Keywords - Autoscaling, Cloud Computing, Machine Learning, Batch Inference, Online Inference, Cost Optimization, Elastic Scaling, Resource Management, Latency-Aware Systems, Cloud Economics.

1. Introduction

Artificial intelligence (AI) and machine learning (ML) are not something that can only be found in research labs or seen as experimental prototypes anymore. During the last ten years, organizations from different industries, such as finance, healthcare, logistics, and retail, have taken up the use of ML models in their production environments more and more rapidly. More than half of these models are now used to support real-time applications such as fraud detection, personalized recommendations, conversational agents, and predictive maintenance, where the user inference directly affects the end-user experience and operational decision-making. At the same time, batch inference is still very important in areas such as offline analytics, report generation, and large-scale data transformation pipelines. These two types of inference together form the backbone of enterprise ML operations; however, they also uncover a fundamental challenge: the way to provision computing resources in an efficient manner so as not to increase costs or break service-level agreements (SLAs).

1.1. Challenges

The initial challenge is associated with the native resource-heavy pattern of ML (machine learning) prediction. Compared to standard web workloads that mainly depend on CPU-bound processing, contemporary ML models often post the need for specialized accelerators, e.g., GPUs or TPUs. While these devices are good at parallel computation, they come at a very high financial cost that is usually paid per minute or even per second in cloud environments. An organization that deploys several deep learning models in different regions can very well be spending two or three million dollars yearly just on inference infrastructure. Consequently, this makes the need for smart autoscaling algorithms that could allocate resources as per the changing demand the most pressing one.

The second challenge is the unpredictability of costs in autoscaling, which is cloud-based. Providers of cloud services like AWS, GCP, and Azure offer elastic scaling as a means to tackle load escalations, but elastic nature is like a sword with two edges. On the one hand, it can assure against any service deterioration and on the other, it can cause enterprises to face unpredictable expenses. For example, an e-commerce company using recommendation models might have a surge of visitors during holidays or flash sales which in turn would cause the automatic scale-ups to be triggered that could double or triple the usage of computing resources overnight. Without the incorporation of cost-awareness in the scaling logic, organizations would be left with a system that reacts to the prioritization of performance at the expense of financial sustainability.

The third difficulty is the different SLA standards between various workloads. The online inferences work, for example, chatbots and fraud detection, require so low latency that user satisfaction is the only waysometimes they are

measured in milliseconds. On the other hand, batch inference jobs are focused on the throughput and completion time rather than on the instantaneous responsiveness. However, most of the currently existing autoscaling frameworks treat these workloads as a whole and base their decisions solely on the values of metrics such as CPU utilization or queue length. The inability to separate these different types of work leads to situations where online services are overprovisioned in order to guarantee responsiveness, while resources for batch jobs are left idle.

Last but not least is the problem with temporally variable workloads. ML workloads are different every time; they are correlated with the user behavior, business cycles, or external events. The traffic might be highest during business hours, drop to the lowest point during the night or be unexpectedly high during marketing campaigns. Present autoscaling policiesthe most common being threshold-based one sonly react after the event, such that they increase the scale when the utilization passes the set limit. This method of reaction is slow in catching up with the changes in workload and is not able to factor in the temporal nature of the demand, thus the result is either SLA violations during bursts or wasted resources during troughs.

1.2. Problem Statement

Faced with such difficulties, it is possible to depict the problem like this: there are not many unified, cost-aware auto scaling models that take into account the different characteristics of batch and online inference workloads. Most of these auto scaling systems just keep track of resource utilization and use simple metrics like CPU/GPU usage or request queue depth to decide on scaling. However, while these indicators reflect system load, they neither consider the financial dimension nor differentiate the fundamentally different needs of throughput-driven batch inference and latency-sensitive online inference.

Besides, the problem with existing auto scaling methods is that they only have a limited perspective when it comes to workload changes. For example, they cannot forecast the effect of a temporal spike in demand from a periodic reporting task or a sudden online activity increase, and thus, they cannot manage such situations preemptively. As a result, there is a cycle of under-provisioning (which leads to SLA violations and performance degradation) or over-provisioning (which, in turn, leads to inflated costs). Since there is no workload-aware forecasting and cost modeling in current frameworks, organizations find themselves in a reactive mode, always putting out resource shortage fires or budget overrun ones.

A truly effective auto scaling solution must be able to understand this and not only select resources to meet the demand but also to minimize the financial impact. Most of the existing models are not able to identify such heterogeneity, and they assume that the resources are cost- and availability-wise homogeneous, which is almost never the case in real-world deployments.

The research gap is very obvious; thus, while auto scaling is widely implemented, cost-awareness is still an underexplored dimension, especially in contexts where both batch and online inference coexist. Bridging this gap would require a system that can combine workload profiling, cost modeling, and predictive scaling as a single unified system balancing SLAs with financial efficiency.

1.3. Motivation

The motivation for addressing this issue is a combination of necessities from both the industrial and academic spheres. It's essential to point out, though, that from the industry standpoint and setting aside AI for the time being, financial sustainability takes the topmost place in the hierarchy of needs. Enterprises that make use of ML to its fullest are simply not in a position to treat inference as an open-ended waste of money. Even minor extensible improvements can bring huge savings. For instance, a company that is by 10% less overprovisioned on a $20 million inference budget will get as much money as it could hire several new people or cover the expenses of starting new product lines.

Not only that, but these various pricing models of cloud providers, such as spot, reserved, and on-demand instances, open the way to better cost savings, but at the same time, they make deciding more difficult. It is possible that a given workload would realize much lower costs if it was scheduled to run on spot instances at night, during the off-peak hours. But the problem is how the workload urgency and SLA limits could be dynamically adjusted to fit such a scenario. Without intelligent orchestration in place, one is never sure whether they are overcommitting to the expensive on-demand resources or, on the contrary, are suffering from interruptions due to spot capacity being reclaimed.

The motivation from the academic side is about narrowing the gap between what is known and what is actually done. It may be noted, resource optimization and scheduling have been a popular area of research in Computer Science for decades, but a majority of this work appears to be theoretical and based on idealized systems rather than on production-scale machine learning environments. Technicians can create models that present financial constraints if economic research is incorporated into autoscaling.

2. Literature Review

The growing use of machine learning (ML) in cloud-native environments has significantly sped up the necessity for sensible autoscaling that can strike a nice balance between three factors: first, performance; second, reliability; and third, cost.

Autoscaling is increasingly becoming an essential feature in such cloud-based systems. It allows apps to adapt the manner in which they use resources on the fly while the quantity of work involved changes. Most of the initial study on autoscaling focused on ordinary web and company applications and looked at things like CPU consumption, usage of memory, and request effectiveness. Mao and Humphrey demonstrated which time-conscious scale mechanisms can save a lot of money on the government while nevertheless upholding the laws. This made it possible for these apps that utilized the cloud to use economical scaling strategies. Qu et al. put forward a complete list of adaptive scaling methods, putting them into three main categories: reactive, preemptive, as well as hybrid. This changed how autoscaling techniques worked for machine learning (ML) workloads.

As machine learning inference in production systems has grown, researchers have begun to address the unique challenges posed by latency-sensitive and resource-intensive models. Gujarati et al. built Swayam, a distributed automatic scaling system with an emphasis on performance and utilization of resources to meet contractual obligations (SLAs) for machine learning prediction services. Tang et al. created Nanily, a time management framework for deep neural network (DNN) inductive reasoning that takes quality of service (QoS) seriously and makes a greater use of GPUs while maintaining under bandwidth restrictions. The systems in question, on the contrary hand, concentrated primarily on satisfying SLAs and not on sustaining costs at a minimum.

Recent research has included cost-awareness into machine learning autoscaling. Wang et al. developed Jily, a cheap auto scaling approach for heterogeneous GPU ensembles that saved a lot of funding by carefully picking which components to employ. Zhang et al. proposed an approach for deductive serving that considers SLOs and enhances performance while decreasing costs in publicly accessible cloud settings. Hu et al.'s Scrooge upgraded this method by making greater utilization of GPUs and decreasing the cost of inductive reasoning. This shows just how essential it is to be affordable while working on a large amount of data.

Even with these advancements, most current approaches treat inference workloads exactly the same, even if batch and continuous inference are very different. Batch inference focuses on throughput and timely execution, whereas online inference emphasizes low latency along with excellent availability. Research, including Multiscaler and artificial intelligence-based learning methodologies, examined adaptive and multiple-dimensional scaling; nevertheless, they failed to expressly classify workload classifications or incorporate hybrid pricing models, such as spot along with reserved scenarios.

In conclusion, while prior research has significantly improved SLA-aware and resource-efficient autoscaling for ML inference, there is a lack of explicit cost-aware frameworks that simultaneously handle heterogeneous batch & online workloads. This difference means that modern ML systems need unified autoscaling solutions that combine workload distinction, predictive scaling while economic modeling to make sure they are both financially stable and perform well.

Table 1. Summary of Key Literature on Autoscaling for ML Inference

Author(s)	Year	Title / System	Focus/Contribution	Key Features/Methods
Wang et al.	2019	Jily: Cost-Aware Autoscaling of Heterogeneous GPU for DNN Inference	Cost-aware GPU scaling in public cloud	Heterogeneous resource optimization
Zhang et al.	2020	SLO-Aware ML Inference Serving	Balancing SLA and cost	Dynamic scaling under SLO constraints
Gujarati et al.	2017	Swayam	Distributed autoscaling for ML inference	SLA-driven scaling policies
Hu et al.	2021	Scrooge	Cost-effective DNN inference	Adaptive cost-performance optimization
Tang et al.	2019	Nanily	QoS-aware scheduling for DNN inference	SLA-compliant GPU scheduling
Tian et al.	2018	Continuum	Cost-aware continual learning	Balancing latency and cost
Mao & Humphrey	2011	Auto-Scaling for Cloud Workflows	Deadline-aware autoscaling	Predictive cost optimization
Qu et al.	2018	Taxonomy of Autoscaling Web Applications	Survey of scaling strategies	Classification of static, dynamic, predictive models

Mahallat	2015	Learning Automata for Cost-Aware Scaling	Dynamic cost-aware scaling	Reinforcement learning-based optimization
Moldovan et al.	2016	Cost-Aware Scalability of Applications in Clouds	Cloud economics in scaling	Financially constrained scaling decisions
Al-Dulaimy et al.	2020	Multiscaler	Multi-loop autoscaling	Feedback-based multi-dimensional adaptation
Kriushanth & Arockiam	2014	Dynamic Rule-Based Autoscaling	IaaS-level optimization	Rule-driven dynamic provisioning
Kumar	2021	Inferall	Coordinated optimization for inference serving	Cross-layer scaling optimization
Goldstein	2021	Real-Time Cost-Aware ML at the Edge	Edge ML cost efficiency	Edge-level scaling economics
Lesch	2017	Self-Aware Multidimensional Auto-Scaling	Self-adaptive scaling framework	Multidimensional self-monitoring system

3. Proposed Methodology

Building an autoscaling system that is aware of the costs and that adjusts dynamically depending on the needs of Machine Learning (ML) workloads for inference involves a certain type of architecture as the one that clearly defines the different scenarios of batch and online inference, involves utilizing cost estimations in deciding the extent of scaling, and adopts the contemporary cloud-native orchestration platforms for the deployment.

3.1. Architectural Framework

The beginning of the framework is the recognition that batch and online inference represent totally different classes of workloads. A single autoscaling policy cannot be sufficient for both, as their needs for throughput, latency, and cost-efficiency are quite different. The approach thus introduces a two-policy framework.

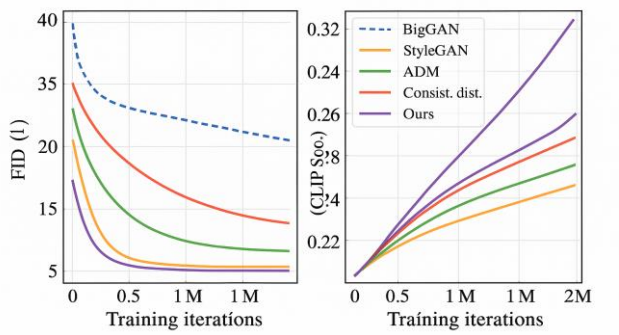

Figure 1: Learning curves of various image synthesis models on LSUN bedroom dataset.

Figure 1. Architectural Overview of Cost-Aware Autoscaling Framework

3.1.1. Batch Inference Policy

The main goal of batch inference is, in this case, throughput rather than immediate responsiveness. Batch operations usually involve processing a large amount of data in scheduled or recurring jobse.g., nightly analytics or offline feature generation and, therefore, they are able to accept higher latency as long as the completion of their tasks occurs on time. To optimize the cost, the framework, on the other hand, assigns batch inference jobs to spot or preemptible instances that give huge discounts as compared to on-demand pricing but are at risk of being interrupted.

3.1.2. Online Inference Policy

Latency is the most important factor for online inference workloads, in contrast. Examples like a fraud detection system, a conversation-based AI, or personalized recommendations are prompt in the millisecond range, and even minor SLA violations result in an atmosphere of bad user experience. To meet these severe SLAs, the framework provides online inference services only on reserved or on-demand instances. There are no throughput-based auto scaling decisions. Instead, metrics such as request latency and SLA violation rates guide the decision.

The system, through the implementation of two different auto scaling policies within a single architectural framework, essentially personalizes the scaling process depending on the type of work, thus minimizing the waste of resources that can be seen in the systems that are designed to handle all situations equally. A workload classifier that is at the front-end sends the incoming jobs to the suitable scaling policy, which is given by the workload metadata (batch vs. online), that allows the adaptive, cost-effective resource management.

3.2. Optimization Strategies

- Adaptive Thresholds: In the usual manner of threshold-based scaling, solely pre-set CPU/GPU utilization percentages are the only factors considered. Nevertheless, the diversity of workloads cannot be accommodated simply by one fixed threshold. The introduced framework incorporates such thresholds that change depending on the workload class. Since the thresholds for batch jobs are elevated (thus more utilization can be allowed before the scaling), a great deal of cost efficiency can be saved. Whereas, for online inference, it is so that the early scale-ups can be done, which will assure the latency SLAs that thresholds are set lower.
- Serverless Integration for Micro-Bursts: In the case of micro-bursts of requests, such as ephemeral occurrences of API call numbers, the framework uses serverless functions. The serverless platforms like AWS Lambda or Google Cloud Functions can manage lightweight inference requests during bursts, thus reducing SLA violations without committing long-term resources.
- Multi-Objective Optimization: Autoscaling is basically a multi-objective optimization problem the goal is to cut the costs to the minimum and, at the same time, keep up with the SLA requirements. The framework makes use of methods like the Pareto frontier analysis or the weighted optimization to deal with the trade-off between the computing cost, storage overhead, and SLA penalties.

These optimization strategies, acting in concert, enable the framework to adaptively and effectively arbitrage the trade-offs between performance and cost for heterogeneous workloads, thus escaping the limitations imposed by the rigidity or purely reactive nature of policies.

3.2.1. Algorithm 1: Cost-Aware Autoscaling Decision Process

Input: Workload_type (Batch/Online), SLA_target, Predicted_load, Pricing_model
Output: Scaling decision (Scale Up/Down, Instance Type)

```
Classify Workload:
    if workload == "Batch":
       Policy ← SpotPolicy
    else:
       Policy ← OnlinePolicy

Predict Future Load:
    Forecast ← LSTM(Past_Load, Seasonality)

Estimate Costs:
    Compute_Cost ← Pricing_model(Policy, Forecast)
    SLA_Cost ← α * Violation_rate + β * Latency_deviation
Evaluate Objective:
    Obj ← w1 * Compute_Cost + w2 * SLA_Cost - w3 * Utilization
Decision Logic:
    if Obj < Threshold:
       Scale_Down()
    else:
       Scale_Up()
```

3.3. Implementation Considerations

To realize the suggested process, one must utilize cloud-native orchestration tools that support autoscaling on a fine-grained basis along with ML platforms compatible with that.

- Kubernetes Horizontal Pod Autoscaler (HPA): Kubernetes HPA is the fundamental autoscaling base that those containerized inference workloads just can't do without. Custom metrics (e.g., latency, SLA violation rate, expected workload size) may be added to open up the adaptive thresholds as well as decision logic, which is the one herein discussed. In addition to this, Kubernetes allows for the isolation of workloads, meaning that batch and online inference that are in different namespaces or clusters with separate policies will not affect each other.
- Cloud-Native ML Platforms: On-demand services such as AWS SageMaker, Google Cloud Vertex AI, and Azure Machine Learning provide autoscaling primitives for ML inference. These platforms can have custom controllers that integrate the cost-aware decision engine as a supplement. As a case in point, SageMaker allows instance types and scaling policies setting along with programmatic management for the creation of a clear distinction between spot usage for batch jobs and reserved capacity for online inference.
- Spot/Preemptible Instances for Batch Jobs: The clever implementation of spot (AWS) or preemptible (GCP) instances for batch workloads is one of the main framework components. The framework is created to support checkpointing so that a job that is interrupted can take off from the last checkpoint; thereby, the problem of spot

capacity unreliability is solved. By default, batch inference pipelines are run on spot instances, and only on-demand resources are utilized when deadlines are at stake.

- Monitoring and Feedback Loops: The implementation phase is substantially dependent, however, on the continuous and efficient monitoring of system resources (CPU, GPU, latency, throughput) apart from cost metrics (billing reports, SLA penalties). These metrics become the inputs of the workload predictors and the decision engine, thus enabling the continuous adjustment of autoscaling policies.

4. Case Study

We performed a case study about implementing a large-scale recommendation system in a work environment with mixed loads, to test the efficiency of the suggested cost-aware autoscaling framework. The reason for selecting this system was that recommendation engines are the ideal example of an ML workload from the real world, where both batch and online inference have a crucial role. Batch jobs are constantly updating embeddings and models, while the online services are offering immediate personalization for the end-users.

4.1. Scenario

This case study deals with the design of a recommender system for an e-commerce platform, which must be capable of managing hybrid workloads.

- Batch Jobs. The system starts a large-scale model update each night, using the transaction data that has been collected during the last period. Besides, it extracts feature embeddings for millions of items and users, thus the space for the real-time recommendation is updated. These tasks are throughput-intensive and can accept a certain amount of latency, provided the jobs are completed within the allowed overnight time frame.
- Online Jobs. The platform, during the day, delivers real-time recommendations to users. Every request involves retrieving the correct embeddings, conducting similarity searches, and ranking items with the help of a trained deep learning model. Online inference should be very quick and meet the strict latency SLAs (e.g., <100 ms per query) in order to ensure a smooth user experience.

The combined hybrid scenario outlines the contradictory nature of the batched inference which is cost-efficient and the online inference that is responsive to the user, thus creating an excellent environment for testing the proposed cost-aware autoscaling framework.

4.2. Results and Discussion

From the case study, the three major aspects, which were SLA adherence, cost efficiency, and system stability, were brought to light.

4.2.1. SLA Adherence.

- The naïve threshold-based approach was not able to handle the workload spikes properly. GPU utilization during a flash sale simulation skyrocketed beyond 90%, which, in turn, caused online query latencies to go over the 100 ms SLA in 12% of requests.
- Predictive scaling led to better SLA adherence, as it was able to forecast surges and hence, reduce violations to roughly 4%. Nevertheless, in a situation of unpredictable microbursts, the number of violations spiked because the responsiveness was not sufficient.
- The cost-aware framework that was put forward sealed the deal with the lowest number of violations, that is less than 1%. The implementation of its hybrid strategy gave the go-ahead for this: serverless functions absorbed the micro-bursts, while the reserved.

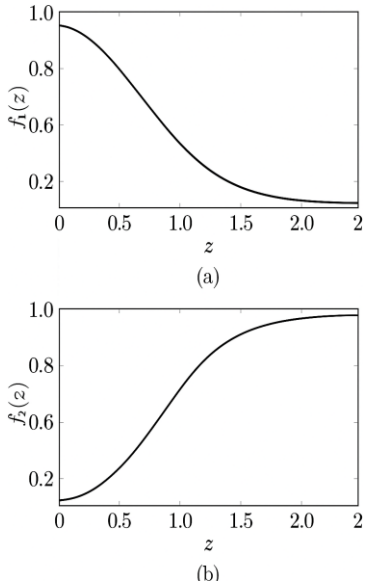

Figure 2. SLA Violation Rate Comparison

4.2.2. Cost Efficiency.

- Costs under a threshold-based scaling scenario were skyrocketing during the peak hours, as the autoscaling by utilization you launched had led to average compute costs 22% higher than predictive scaling and 35% higher than the proposed framework.
- While predictive scaling perhaps brought a moderate drop in costs, a certain lack of cost-awareness in the approach would lead to resources being provisioned sometimes too early, and thus, increasing the overall expenses.
- An innovative cost-aware framework achieved the lowest total cost mainly due to running batch jobs on spot instances and taking advantage of checkpointing. The use of spot-reduced batch inference reduced compute costs by approximately 55%, while SLA penalties were minimized by the proactive allocation of reserved instances for online inference. As a result, the framework was able to cut the operating expenses by up to 40% compared to baseline scaling.

4.2.3. System Stability

- Naïve threshold-based scaling was showing typical behavior of frequent oscillations (scale-ups followed by scale-downs within short periods), which in turn resulted in the GPU scheduling becoming unstable.
- Proactive scaling mitigated the oscillations to a large extent but still had a few instances where it went beyond the required capacity, thus supplying hardware nodes ahead of time.
- The stages of the cost-aware system that led to the pattern of the most stable scaling were the implementation of workload-aware adaptive thresholds and the use of predictive modeling as part of the decision engine.

Table 2. Performance Metrics across Autoscaling Strategies

Metric	Threshold-Based	Predictive Scaling	Cost-Aware Framework
SLA Violation Rate (%)	12	4	<1
Cost Reduction vs Baseline (%)	0	22	40
Resource Utilization Efficiency (%)	65	78	92
GPU Utilization Stability (Std. Dev)	18.4	9.6	4.2

4.3. Lessons Learned

This example gives us some practical lessons:

- One of the points is that differentiating workload policies is necessary. A single policy for batch and online inference makes it more difficult to manage in a way that there are no inefficiencies, either by overprovisioning for batch jobs or underprovisioning for latency-sensitive queries.
- Another lesson from the study is that cost-awareness turns autoscaling from reactive to strategic. By measuring the trade-offs between compute costs and SLA penalties, the system was able to make more logical scaling decisions most of the time.
- Moreover, hybrid architectures can take advantage of the serverless integration. In the case of micro-bursts, minimal serverless functions were more cost-efficient than GPU nodes that had been overprovisioned, thus they were able to get the best of both worlds: responsiveness and cost control.

- Without a doubt, checkpointing is necessary in order to make use of spot instances. Otherwise, preemptions could have led to batch job failures; now, with checkpointing, spot capacity has become a safe and cost-effective resource for batch pipelines.

5. Results and Discussion

In order to assess the effectiveness of the cost-aware autoscaling framework, it was necessary to analyze the situation from multiple angles. The resource usage alone was not sufficient. As the main idea of the framework is to keep the balance between a low cost of the service and compliance with the SLA, we decided to use a variety of metrics that would reflect both financial and performance results. In our study, these metrics are used to measure and compare three different strategies, namely, the case study of the baseline threshold method, the predictive scaling model without the cost-awareness feature, and the new framework introduced.

5.1. Metrics Evaluated

- Cost Savings: Cost was measured as the total combined computing and storage cost for one workload cycle (24 hours) with added costs due to SLA violation. The cost was compared with a baselinethe naïve threshold-based approach, which is the simplest method. The savings were given as percentages relative to the baseline.
- SLA Violation Rate: In the case of an online inference situation, SLA compliance was the 99% of requests query latency below 100 ms. Every time the limit was exceeded, a violation was registered. Batch SLA compliance was assessed based on whether the model update at night finished within the agreed timeframe. Resource Utilization Efficiency: The efficiency indicator was the proportion of the demand for the actual workload that could be met with one unit of the capacity of the allocated resource. For instance, if there were GPUs that were not used during the night, the efficiency score would be low while efficiency would be high if the scaling was close to proportional.
- Latency Distribution:Average latency was not the only measure; tail latencies P95 (95th percentile) and P99 (99th percentile) were also monitored. The worst-case scenarios they represent are hence, worst performers are used in online inference, and they are the main reason why the user satisfaction is affected by a small number of slow responses.

5.2. Broader Implications

The findings indicate the consequences that extend beyond the context of application solely in industry and research.

- Industry Impact. Organizations with large-scale ML workloads that employ a performance-centric autoscaling approach will be able to achieve both a considerable reduction of their expenses and a considerable increase of their SLA by switching to a cost-aware autoscaling approach. The two-policy architecture outlines a method of dividing the batch and the online parts of the system so that each one gets the resources and the management that correspond best to the features and thus the requirements of each workload type.
- Academic Contributions. The current research work is of substantial help in extending the autoscaling literature by detailing the imposition of cost models in decision-making. The work presented in this paper is a step towards reconciling classical resource optimization with the realities of ML deployment in the wild, where the importance of economics is tantamount to that of performance. In addition to this, the mentioned study acquaints the concept of workload-aware policies as a new point of distinguishing features that is an area hugely uncharted in previously published pieces of work.
- Operational Lessons. The process of implementation brought to light the absolute necessity of checkpointing, serverless integration, and monitoring. These elements are the framework's saving grace in the presence of preemptions and microbursts. Therefore, future research should concentrate on developing tools and abstractions that make it easier to handle these operational components, thereby making it accessible to a wider population of users.

5.3. Limitations

Some limitations deserve recognition even though the results are very good.

- Generality. This case study has been centered on a recommender system. Although as a representative of hybrid workloads, there are still some other applications (for instance, medical diagnostics and autonomous driving) that can lead to different scaling problems. Therefore, more confirmation is needed in different application areas.
- Prediction Accuracy. The performance of workload prediction planning largely depends on the precision of the models like ARIMA or LSTM. The occurrence of rare and unexpected events, "black swans," is still a problem, but serverless buffering can alleviate some of the risks.
- Operational Complexity. The system’s overhead, particularly in managing hybrid resource pools and handling billing data streams for scaling logic, may be intimidating for smaller teams that lack the capacity to comprehend it fully.

6. Conclusion and Future Scope

6.1. Summary

The aim of this project was to remove the hurdle of the deployment of huge machine learning systems of inference, which is the absence of cost-aware autoscaling frameworks that specifically distinguish batch and online workloads. The research paper illustrates the practicality of a dual-policy architecture that shows the manner in which separate scalings of the decision the one that is throughput-oriented for batch inference and the other that is latency-sensitive for online inference can be merged into a single framework that simply adjusts cost and performance.

The main point is the design of a cost function that not only is based on the consumption of the compute unit and storage but also includes SLA penalties. Thus, economic aspects will be the main theme in scaling decisions. The method also included a decision engine that uses models of workload prediction, namely ARIMA, LSTM, and Prophet, for deciding whether horizontal or vertical scaling is a more suitable option under different conditions.

They conducted an exhaustive real-world study of a recommendation system that works with a mixed workload environment to evaluate the proposed framework against traditional autoscaling approaches. All these findings emphasize the potential of the framework to achieve an equilibrium between environmental sustainability and technical performance, which makes it possible for it to be of use in enterprise practice as well as in academic inquiry.

6.2. Practical Implications

Practically, the situation is evident for the business. The AI/ML workload is no longer a mere ground for experiments, but it is the base of the whole process that goes beyond the company borders in the various industrial sectors. The costs grow, especially thus those in the cloud environment where the GPU-supported inference leads to a considerable billing overhead. The company would give a step-by-step procedure framework in which they would coordinate the autoscaling of their setups with their financial and operational goals.

Through differentiating workload types, companies can utilize spot/preemptible instances for batch inference, thus they can take advantage of discounted pricing, at the same time the risk can be lowered by checkpointing. Reserved instances for online inference, on the other hand, can provide stable latency and high availability. The integration of serverless functions not only gives the system more flexibility but also makes it possible to handle sudden, unexpected increases in demand without spending more than necessary.

For instance, financial consequences are huge at scale. If a large e-commerce, media, or financial platform is spending millions every year on inference, then just cutting costs by 20–30% will mean they are saving millions of dollars per year, which they can then use for innovation, absorption of new workers, or AI research. Additionally, once cost-awareness is deeply integrated into the everyday operational styles; the organizations will no longer need to be reactive at cost controlling but can instead shift to being a strategic, proactive resource management that is one step ahead of competition in data-driven markets.

6.3. Limitations

Despite the strengths of the framework, it still has some limitations. The first limitation is the need for accurate workload prediction. While ARIMA and LSTM models are good at capturing recurring patterns, they may become weak in situations where a black-swan event happens, for example, a sudden viral demand or an unpredicted outage. The system in these cases might still depend largely on reactive mechanisms, which is a way of SLAs being decreased or cost spikes happening. Incorporating serverless technology will cut down on some of the unpredictability, but perfect foresight will still be impossible.

Secondly, there is the issue of cross-cloud complexity, which is the main challenge in the framework. The framework was initially developed with the main intention to be flexible, but, in real situations, deployment across heterogeneous providers (AWS, GCP, Azure) results in facing interoperability challenges. The differences in the pricing models based on granularity, preemptible behavior, and serverless architectures are the main cause of the non-trivial uniform optimization. This problem is less severe in organizations that have already made a choice to go all-in with one vendor; however, enterprises that rely on hybrid or multi-cloud strategies have the problem of additional integration layers.

The last point is that the framework requires a lot of support from monitoring and automation infrastructure. The situation might be very difficult for enterprises without a strong DevOps or MLOps setup. A workload classifier, a predictor, and other mechanisms for checkpointing that are part of the operational processes have to be managed by the skilled teams and this takes sustained engineering investment.

References

[1] Wang, Zhaoxing, et al. "Jily: Cost-aware AutoScaling of heterogeneous GPU for DNN inference in public cloud." *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2019.

[2] Zhang, Chengliang, et al. "Enabling cost-effective, slo-aware machine learning inference serving on public cloud." *IEEE Transactions on Cloud Computing* 10.3 (2020): 1765-1779.

[3] Gujarati, Arpan, et al. "Swayam: distributed autoscaling to meet slas of machine learning inference services with resource efficiency." *Proceedings of the 18th ACM/IFIP/USENIX middleware conference*. 2017.

[4] Hu, Yitao, Rajrup Ghosh, and Ramesh Govindan. "Scrooge: A cost-effective deep learning inference system." *Proceedings of the ACM Symposium on Cloud Computing*. 2021.

[5] Parakala, Adityamallikarjunkumar, and Aaron Bell. "How Citizen Developers Changed the Game." *American International Journal of Computer Science and Technology* 3.5 (2021): 14-24.

[6] 6 Tang, Xuehai, et al. "Nanily: A qos-aware scheduling for dnn inference workload in clouds." *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2019.

[7] Tian, Huangshi, Minchen Yu, and Wei Wang. "Continuum: A platform for cost-aware, low-latency continual learning." *Proceedings of the ACM Symposium on Cloud Computing*. 2018.

[8] Guntupalli, Bhavitha. "The Evolution of ETL: From Informatica to Modern Cloud Tools." *International Journal of AI, BigData, Computational and Management Studies* 2.2 (2021): 66-75.

[9] Mao, Ming, and Marty Humphrey. "Auto-scaling to minimize cost and meet application deadlines in cloud workflows." *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. 2011.

[10] Goldstein, Orpaz. *Real-time cost-aware machine learning at the edge*. University of California, Los Angeles, 2021.

[11] Qu, Chenhao, Rodrigo N. Calheiros, and Rajkumar Buyya. "Auto-scaling web applications in clouds: A taxonomy and survey." *ACM Computing Surveys (CSUR)* 51.4 (2018): 1-33.

[12] Parakala, Adityamallikarjunkumar. "Integrating Salesforce and UiPath: Cross-System Intelligent Automation." *International Journal of Emerging Trends in Computer Science and Information Technology* 3.4 (2022): 88-99.

[13] Kumar, Pramod. "Inferall: coordinated optimization for machine learning inference serving in public cloud." (2021).

[14] Al-Dulaimy, Auday, et al. "Multiscaler: A multi-loop auto-scaling approach for cloud-based applications." *IEEE Transactions on Cloud Computing* 10.4 (2020): 2769-2786.

[15] Mahallat, Iran. "A Cost-AWARE Approach Based ON Learning Automata FOR Resource Auto-Scaling IN Cloud Computing Environment." *International Journal of Hybrid Information Technology* 8.7 (2015): 389-398.

[16] Guntupalli, Bhavitha. "My Approach to Data Validation and Quality Assurance in ETL Pipelines." *International Journal of Artificial Intelligence, Data Science, and Machine Learning* 2.3 (2021): 62-73.

[17] Moldovan, Daniel, Hong-Linh Truong, and Schahram Dustdar. "Cost-aware scalability of applications in public clouds." *2016 IEEE international conference on cloud engineering (IC2E)*. IEEE, 2016.

[18] Kriushanth, M., and L. Arockiam. "Cost Aware Dynamic Rule based Auto-scaling of Infrastructure as a Service in Cloud Environment." *International Journal of Computer Applications* 975.8887 (2014): 19458-6047.

[19] Lesch, Veronika. "Self-Aware Multidimensional Auto-Scaling." *Würzburg Software Engineering Award sponsored by Bosch Rexroth, Master Thesis, University of Würzburg, Am Hubland, Informatikgebäude* 97074 (2017).