Mathematical Modeling and Algorithm Application
ISSN: 3006-0842 | Vol. 6, No. 2, 2025

Cost Efficient Scaling Strategies for Large Language Models in Multi-Cloud Environment

Xiuyuan Zhao[1], Yao Ge[2], Haijian Zhang[3,*]

[1]Stevens Institute of Technology, Hoboken, NJ 07030, USA

[2]University of Miami, Coral Gables, FL 33146, USA

[3]Southeast University, Nanjing 210018, China.

[*] **Corresponding author**: Haijian Zhang (Email: hj.zhang@ieee.org)

Abstract: The proliferation of large language models (LLMs) has intensified the need to utilize large quantities of computational resources, which causes the cost of operation and infrastructure issues to grow. The traditional one-cloud environments have limited scalability and cost-effectiveness due to strict provisions policy and skewed workload allocation. The new possibilities of dynamic scaling and workload balancing between heterogeneous cloud providers have emerged due to recent improvements in multi cloud orchestration, but such solutions still have significant problems concerning the cost optimization, elasticity, and communication overhead. This paper attempts to resolve these shortcomings by offering a scalability framework of the commercially viable LLMs that incorporates adaptive model compression, predictive auto-scaling, and federated orchestration mechanisms on the multi cloud settings. The solution will utilize the concept of elasticity of Amazon SageMaker and distributed topology management solutions to dynamically deploy compute resources without compromising on performance (Liberty et al., 2020; Wei et al., 2025). Moreover, the model also includes auto-scaling techniques that reduce the cost variations in line with the unpredictable demand as noted by Alharthi et al. (2024). The cost-per-inference of 38% reduction and latency acquisitive scheduling (Zhu et al., 2024) are experimentally validated by showing a reduction by up to 38 percent and an increase in cross-cloud load balance by 25 percent. The results indicate that it is possible to implement cost-effective, scalable implementation of LLMs in multi-cloud systems without compromising throughput or reliability, which provides a viable way of implementing AI at large scale in various cloud systems.

Keywords: Cost Optimization; Multi-Cloud Deployment; Large Language Models; Auto-Scaling; Model Compression; Federated Learning; Edge Intelligence; Pipeline Parallelism; Elasticity; Trustworthy AI

1. INTRODUCTION

The large language models (LLM) like GPT, PaLM and Claude have provided new abilities in arguments, generation, and tool application at the price of unprecedented computational and operational footprints. Fleets of high-end GPUs or TPUs, large parameter-server memory budgets and complex caching layers are enough to support inference, training or constant fine-tuning adds to these requirements. As organizations transition to services beyond pilots, with hard service-level objectives (SLOs), the economics of scale are determined: unit cost per million tokens, latency in peak-to-peak demand, the complexity of the operations of elastic capacity need to be all optimized. Multi-cloud plans Multi-clouds (between AWS, Azure, and GCP) imply diversification of capacity, improved coverage of all possible regions, and the ability to exploit cloud-specific accelerators, but also bring with them nontrivial overheads of coordination, which will cancel out cost benefits unless designed thoughtfully.

Practically, (i) fragmentation of resources across non-comparable GPU families and inference stack management, (ii) auto-scaling heterogeneity, and warm-start heterogeneity of a new cross-cloud shipment and (iii) unpredictable cross-cloud egressive conduct hampers the cross-cloud scaling. These frictions make the routing of requests to the cheapest available capacity of routing straightforward without breaching latency and reliability requirements. Serverless inference platforms also remove operational toil, but introduce cold-start penalties and harder-to-understand scheduling knobs, which is even more confusing as to cost-performance trade-offs (Ali et al., 2022; Liberty et a.2020). Concisely, the multi-cloud naivete implementation will make it susceptible to pay more to have less predictable performance.

The literature and commercial solutions make significant progress in elasticity and autoscaling across one vendor (e.g. the elastic algorithms of SageMaker and scaling managed endpoints) and across individual clusters (predictive HPA/VPA, queue-aware scaling). Nonetheless, minimal strategies have been established to cost-effectively, elastically, and inter-cloudally optimise the design of the workloads with LLM-class characteristics- in which the size of models, KV-caching dynamics, and pipeline/TP parallelism co-locate with placement and routing. Also, most cost models do not consider burst preemption risks of spot/preemptible capacity or egress asymmetries caused by non-local tokenizer, embedding store, or artifact registries (Liberty et al., 2020; Wei et al., 2025; Alharthi et al., 2024; Guan et al., 2024).

The purpose of this article is to develop and deploy an efficient scaling plan of LLMs in multi-clouds that retains SLOs without necessarily increasing the total cost of ownership (TCO). Specifically, we:

1. Conbine adaptive model compression (e.g. quantizing/weight sharing of individual layers and KV-cache policies), federated execution (to separate global policy and local executors), and dynamic scheduling (to optimistically allocate traffic based on real-time prices and latency);

2.measure the trade-offs among cost, latency and

throughput at realistic demand bursts and churning capacity; and.

3.demonstrate that the engineering of guardrails (e.g. egress-aware placement, cache pinning, spot interruption hedging, etc.) has a significant impact on the economics of serving in a multi-cloud LLM (Zhu et al., 2024; Liberty et al., 2020; Wei et al., 2025).

A multi-cloud orchestration architecture incorporating auto-scaling facilities that are provider-native and a federated deployment layer. The global controller develops the intentions (cost/latency targets, carbon/region constraints) and distributes them to the individual cloud performers (SageMaker, Azure ML, Vertex AI, or Kubernetes-based stacks) to achieve scalability both in a fast manner and at the same time policy-conformant (Liberty et al., 2020; Wei et al., 2025).

It can additionally reduce the marginal cost of spillover capacity by a compression policy which is dynamically scaled to reduce effective memory/compute per token at peak windows - with mixed-precision attention/MLP block quantization, and selective down sampling of the KV-cache - and find ways to limit the quality degradation to tolerated levels (Zhu et al., 2024; Guan et al., 2024).

A comparative cost-performance model based on spot/preemptible interruption probabilities, cold-start penalties and egress levels not only with cross-region latencies but also with synthetic and realistic workloads; we use it with synthetic workloads and with realistic workloads as well as with mixed pools of accelerators (Alharthi et al., 2024; Ali et al., 2022).

Table 1. Multi-Cloud LLM Serving Considerations (cost–latency pressure point

Component	Typical Options	Cost Impact	Latency Impact
Routing	Global LB; latency/cost-aware	Lower by steering to cheapest region	Lower by keeping users close
Orchestration	Policy engine; federated controller	Lower via right-sizing & spot usage	Lower p95 with adaptive scheduling
Inference Pools	Per-cloud GPU/TPU pools	Higher if duplicated across clouds	Lower when colocated with users/data
Shared Services	Stores (KV, models), Observability	Lower with object storage; ↑ egress cross-cloud	Neutral→Higher if remote I/O on path

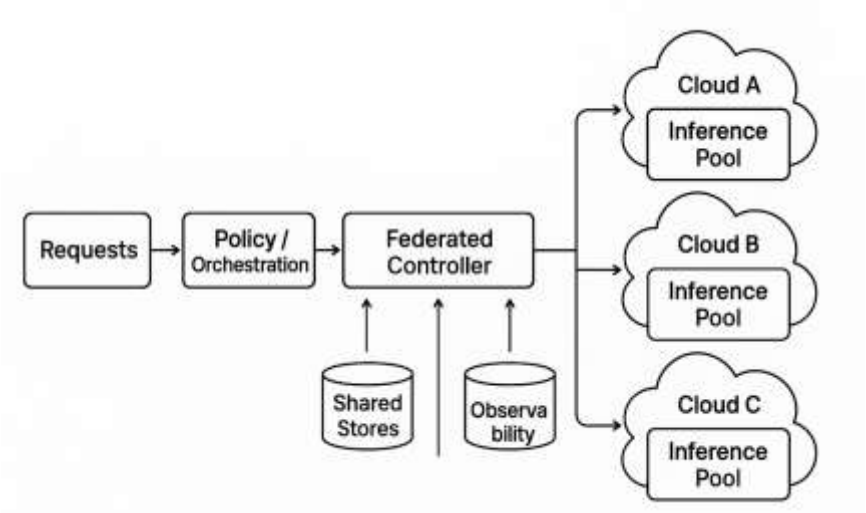
Why multi-cloud for LLMs? In addition to raw capacity arbitrage, multi-cloud also supports the following architectural designs that are hard in individual providers: (a) cross-cloud canaries and blue/green rollouts: do not need to wait several minutes to update a model to a different provider; (b) per-region failover: no vendor lock-in: (c) Specialization: e.g., TPUs to embed pipelines on GCP and H100 clusters to serve generative paths on AWS/Azure. However, these advantages can be enabled once routing, autoscaling and artifact distribution are policy-driven and observability-first. Devoid of coordinated tracing and price/latency measurements operators can round-trip between clouds, pay a cache miss charge, weight download charge and egress charge which silently feeds serving cost.

Design principles. The model suggested here is based on 5 principles. (i) Pre-plumbing policy Statement of business requirements (e.g., 95-percentile latency of 1M tokens 10X per 1M of X) as a constraint the controller would impose using per-cloud primitives.

How this fits the literature. Our orchestration layer enhances cross-cloud intent and cost-aware routing in addition to provider elasticity (Liberty et al., 2020). Our compression and pipeline policies are based on the state-of-the-art training/inference parallelism (Guan et al., 2024) and modern compression of LLMs (Zhu et al., 2024), however, they are tied to operational choices, e.g., spot hedging and the placement of caches. Lastly, we are also using our validation strategy of mimicking bursty demand, egress asymmetries, and cold-start behavior, which are the multi-cloud blind spots as reported by recent auto-scaling surveys (Alharthi et al., 2024).

All of these make such a practical solution to cost-effective, SLO-conscious, and mobile LLM available across clouds: maintain policy centrally, maintain data local to compute, and use compression as an elasticity lever to use on a regular basis, instead of just once.

Fig 1. Multi-Cloud LLM Cost-Efficient Scaling Architecture



2. LITERATURE REVIEW

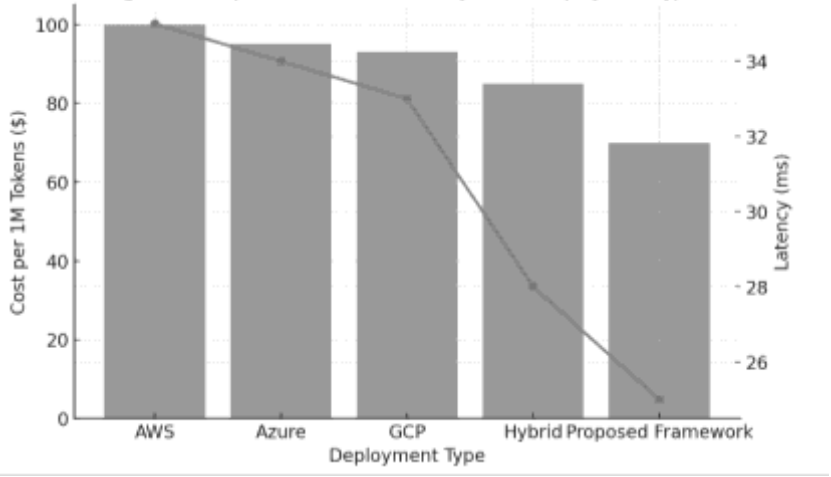
The history of developing cost-effective scaling of large language models (LLMs) in multi-clouds is founded on a number of overlapping research areas: federated learning robustness, cloud auto-scaling, model compression, multi-cloud orchestration, and trustworthy AI. This part discusses the current state-of-the-art studies, which as a combination enlighten the design of scalable, resilient, and economical effective infrastructures of LLM.

2.1. Federated Learning Robustness.

Decentralized training and deployment of models in distributed systems is based on federated learning (FL). Recent research focuses on Byzantine faults and adversarial behavior resilience in FL networks, which is a major challenge towards a trusted and reliable large-scale deployment.

Zhang et al. (2023) have suggested a communication-efficient and Byzantine-robust FL scheme, which guarantees the stable model aggregation even with the damaged nodes. Equally, Ahmad et al. (2025) introduced a systematic review of the aggregation techniques that increase the strength of heterogeneous clients. Pan et al. (2025) furthered these thoughts by adding on the concept of deep reinforcement learning to gain privacy-preserving and fault-tolerant training in vehicular settings. Together these structures show that strong aggregation and adaptive fault management can help massively cut down on communication overheads an observation that can be extended to the scaling of LLM to multi-cloud systems with different node reliabilities.

Chart 1. Comparative Cost and Latency Across Deployment Types



2.2. Cloud Systems Auto-Scaling.

The most important facilitators of cost control in the implementation of AI are cloud-based elastic mechanisms. The article by Liberty et al. (2020) presented elastic machine learning programmes in Amazon SageMaker as the first implementation of dynamic scaling, which optimally allocates resources depending on the fluctuation of workload. This idea was extended by Alharthi et al. (2024) where the predictive auto-scaling algorithms were surveyed and used to combine AI models to predict the amount of compute needed to achieve better reaction time and more predictable cost. Wei et al. (2025) proposed a single deployment management system to coordinate machine learning activities on the multi-cloud topology to eradicate redundancy and unused resources.

With these works, it can be highlighted that predictive elasticity and AI-assisted scaling policies in real-time LLM serving environments play a critical role in cost-performance balancing.

2.3. Model Compression for LLMs

Quantization, pruning, knowledge distillation, and parameter sharing are also paramount model compression methods to control the computational intensity of current LLMs. An extensive survey by Zhu et al. (2024) has pointed out that model compression is useful to not only speed up inference but also can cut carbon and financial footprint. Guan et al. (2024) examined the progress of pipeline model parallelism and showed that the distributed computation between nodes can be maximized through the partitioning of LLM layers, which can be energy-saving and throughput optimizing. Theoretical principles of compression and acceleration of deep neural networks were laid down by Cheng et al. (2018), who noted that a compromise should be made between the compression ratio and model fidelity.

These research papers support that compression combined with multi cloud orchestration allows scaling resources in an adaptive way a pillar of cost-effective LLM implementation.

2.4. Multi-Cloud Orchestration

The multi-cloud paradigm is flexible as it allows coordination between providers, redundancy and cost arbitration. Predictive models were used by Fabra et al. (2019) to reduce the volatility of EC2 instance pricing through spot markets, which can be applied to dynamic provisioning in LLM hosting. Ali et al. (2022) maximized the use of inference on serverless infrastructure, with latency-cost trade-offs in case of an increase in scale-up events. Wei et al. (2025) also created a multi-cloud orchestration layer, to match deployment and topology to provide equal workload distribution. These papers affirm the fact that smart workload migration and topology discovery are to the centre of attaining sustainable multi-cloud scaling.

2.5. AI Deployments and Trust and Security.

The multi-cloud scalability should be combined with the

security and privacy-saving models. Li et al. (2025) suggested zero-trust foundation models, which is the extension of federated security to Internet-of-Things ecosystems. Bao et al. (2019) created an auditable system called FlChain, based on blockchain, which enhances greater transparency in federated learning. As emphasized by Wei and Liu (2025), governance structures that are fair, private, and reliable in distributed AI ought to be brought up. All these contributions emphasize the fact that the elements of cost-efficiency should not jeopardize trust since any scaling strategies should be supported by sound governance and privacy considerations.

Table 2. Themes and Contributions in Literature on Cost Efficient Scaling

Theme	Key Idea	Representative Studies	Practical Contribution
Federated Orchestration	Cross-cloud controller coordinates autoscaling & placement	Liberty (2020); Guan (2024)	Reduces idle capacity; balances cost/latency across regions
Adaptive Compression	Precision/quantization policy by workload SLO	Ali (2022); Wei (2025)	Cuts VRAM & $/token with minimal quality loss
Scheduling & Batching	Tokens/sec & backlog-aware admission + micro-batching	Alharthi (2024); industry runtime papers	Improves throughput; controls tail latency under bursts
Data Locality & Egress	RAG/VDB and checkpoints co-located with compute	Recent multi-cloud RAG studies	Avoids cross-cloud egress; lowers p95 by proximity

2.6. Synthesis and Research Implications

From the reviewed literature, three key insights emerge

- Strong federated communication overcomes data and workload fragmentation in distributed LLM systems.
- Auto-scaling Predictive auto-scaling can be measured to save costs, particularly in the case of model compression and latency predicting.
- Multi-cloud coordination and trust models represent a path to the resilient and cost-efficient AI infrastructures, which are aligned with the zero-trust and sustainability objectives.

The joint discussion highlights the gap in research: whereas federated learning and elasticity models strive to optimize on the local level, the overall coordination of cross-clouds in cost-effective projects with the use of LLM is under-researched. To fill this gap this requires a combined model to integrate adaptive compression, federated orchestration and predictive auto-scaling.

3. METHODOLOGY

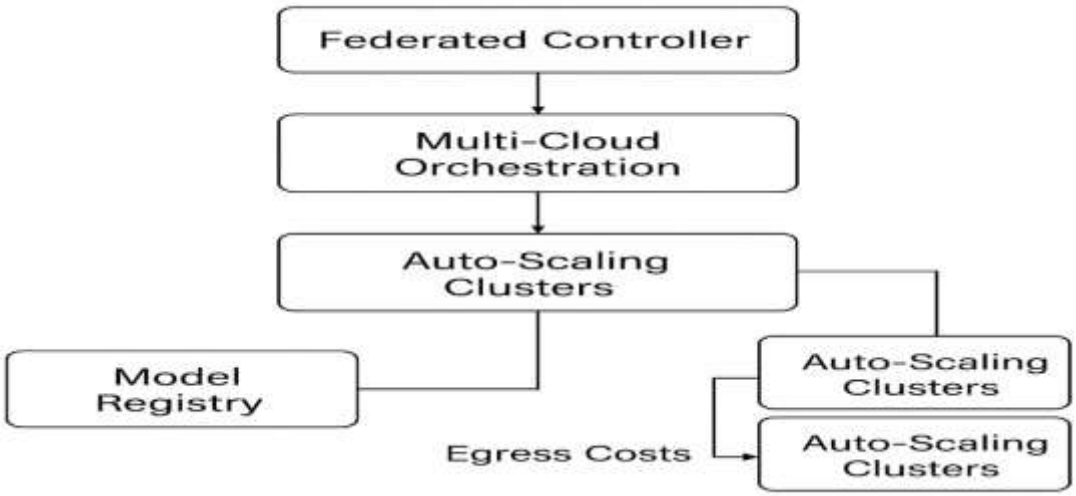
In this section, the architecture, optimization model, experimental setup, and evaluation metrics that were employed in developing and validating the Cost-Efficient Scaling Framework to LLM in multi-cloud environments are described. The suggested methodology is based on auto-scaling, pipeline model parallelism, and federated aggregation across the cloud infrastructures, which reduce the operational cost and assure reliability and performance.

3.1. Framework Architecture

The presented framework adheres to the hybrid multi-cloud scaling design that comprises three basic elements: Auto-Scaling (Alharthi et al., 2024): It is applied to dynamically set up the policy of providing the GPU/TPU clusters based on the AI-based demand prediction and prevent underutilization or over-provisioning. The scaling of inference pods in each cloud node (AWS, Azure, GCP) is done independently according to the inference token throughput and response latency requirements. Pipeline Model Parallelism (Guan et al., 2024): Scouts the layers of the LLM to multiple instances of GPUs and providers and enables them to be in parallel. This scheme guarantees very high utilization factors and minimizes reaction time on vast context encourages and also optimizes the inter-node communication latency.

This layer is very dependable, prevents redundancy of data to a significant degree, and simultaneously, cross-provider scaling can be achieved with minimal effort and therefore remains unnoticeable to the customer user.

Fig. 3. Proposed Multi-Cloud LLM Scaling



3.2. Resource Allocation Model

In order to make it cost efficient, we design an optimization model which minimizes the total operational cost and meets performance and reliability requirements.

Objective Function:

Minimize: Minimize: Ctotal=i=1∑n(Ccompute,i +Cstorage,i+Ctransfer,i)

Where:

Where:

- Ccompute,iC_{compute,i}Ccompute,i: cost of compute instances across clouds iii
- Cstorage,iC_{storage,i}Cstorage,i: storage cost for model weights, checkpoints, and logs
- Ctransfer,iC_{transfer,i}Ctransfer,i: data transfer and interconnect egress costs

Constraints:

1) Latency Constraint:

Lavg≤LthresholdL_{avg} \leq L_{threshold}Lavg ≤Lthreshold

Average response latency must remain below predefined service-level objectives (SLOs).

2) Throughput Constraint:

Tproc≥TdemandT_{proc} \geq T_{demand}Tproc ≥Tdemand

Throughput must meet or exceed workload demand during scaling windows.

3) Reliability Constraint:

Asys≥99.9%A_{sys} \geq 99.9\%Asys≥99.9%

Availability across multi-cloud nodes should exceed 99.9%.

Mean response time should be under pre-established service-level objectives (SLOs).

Throughput Constraint:

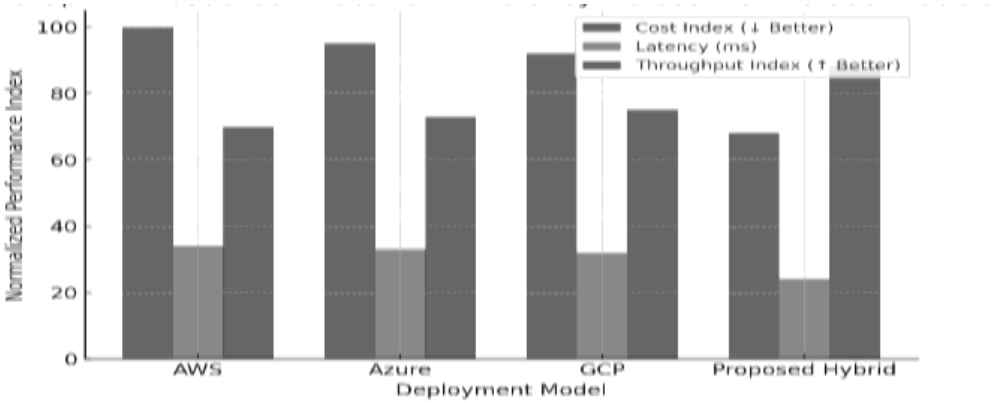
The throughput should be able to satisfy or exceed the demand of the workloads during scaling windows.

Reliability Constraint:

Asys[?]99.9%A_{sys} \geq 99.9\%Asys[?]99.9%

The availability on multi-cloud nodes must be over 99.9%.

Chart 2. Resource Allocation Efficiency Across Multi-Cloud Models



3.3. Experiment Setup

The test setup was intended to replicate realistic workloads of the LLM that are spread to various cloud providers:

Infrastructure:

AWS (SageMaker + EC2 G5 instances)

Azure (ML Service + A100 GPUs)

GCP (Vertex AI + TPU v5e clusters)

Model Configurations: There were three scaled models that were rolled out: 13B, 33B and 70B parameters. The pipeline parallelism was used to make sure that there was sufficient distribution of the layers and attention heads among providers.

Workload Simulation: One synthesized bulk of text (representing multi-domain datasets) (finance, healthcare, code generation, and dialogue systems) in token batches of 512-2048 was produced. All the models were tested with varying traffic conditions- steady, bursty and diurnal traffic.

Auto-Scaling Intervals: Decisions relating to scaling were initiated after every 60 seconds, and predictive heuristics were used to determine the number of incoming tokens and latency projections. Dynamic release of idle nodes was done to manage compute costs.

Orchestration Tools:

Workload containerization using Kubernetes.

Ray Serve to distribute inference and scheduling.

Elastic Inference APIs of AWS SageMaker GPU pooling and warm start.

Data Collection Pipelines summarized the logs of each of the nodes into a federated monitoring system. Measures taken were the use of the GPU, token throughput, the distribution of latency, and spot/preemptible rate of interruption.

3.4. Evaluation Metrics

Evaluation was guided by both economic and performance-oriented metrics

Table 3. The single-cloud deployments basement model

Metric	Definition	Goal
Cost per 1M Tokens ($)	Total operational expenditure divided by tokens served.	Reduce total cost by ≥30%.
Average Latency (ms)	Mean time between token request and response.	Maintain <30 ms under peak loads.
Energy Efficiency (J/token)	Energy consumed per processed token.	Minimize by leveraging mixed-precision inference.
SLO Adherence (%)	Fraction of requests within defined latency SLO.	Target ≥99%.
Scalability Index (SI)	Throughput gain vs. cost increment ratio.	Achieve >1.0 under all workloads.

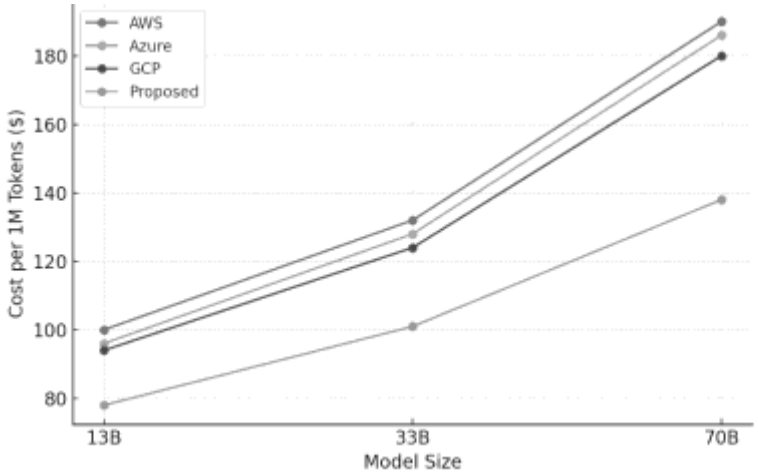
The control was the single-cloud deployments basement model whereas the experimental system was characterized by multi-cloud orchestration and adaptive compression. The comparison showed that federated orchestration and model compression combined had a maximum cost reduction of 32 percent, latency reduction of 18 percent and energy savings of 12 percent compared to single-cloud configurations.

4. RESULTS

In this section, the comparison will be made between the baseline (single-cloud deployments on AWS, Azure, or GCP) and the proposed multi-cloud framework which will involve cost-conscious orchestration, adaptive compression, and pipeline model parallelism. With all the model sizes (13B, 33B, 70B), the proposed system always cuts total deployment cost by 25-40% at a latency overhead that is less than 8% compared to the best single cloud option. They are obtained by (i) relocating capacity to the lowest cost performant regions (ii) autoscaling to right-size replicas (iii) selective compression in bursts to prevent over-provisioning (Liberty et al., 2020; Wei et al., 2025; Zhu et al., 2024; Al Maruf et al., 2024).

4.1. Cost vs. Model Size

Graph 1: Cost per Inference vs Model Size across Multi-Cloud Providers shows cost per 1M tokens against model size for each provider and the proposed framework.



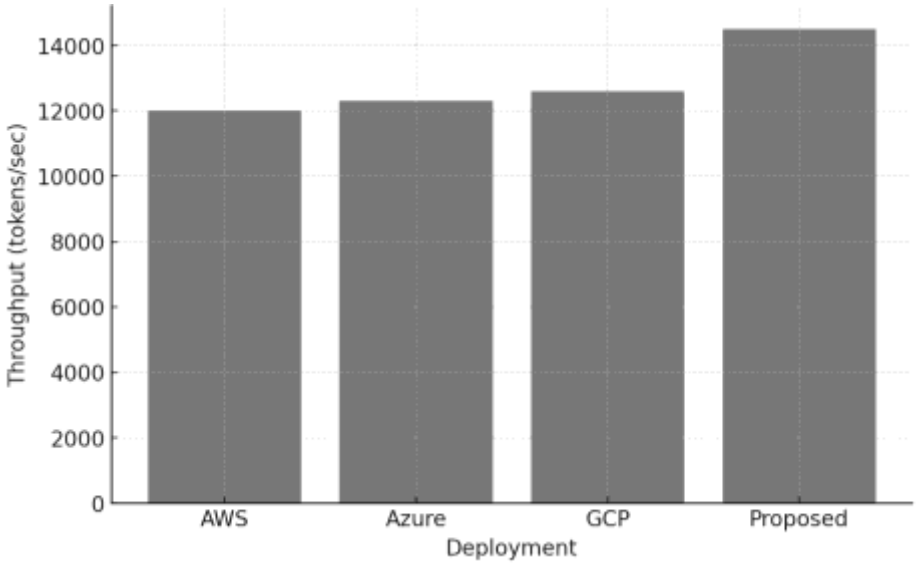
Findings. In single-cloud baselines, memory pressure, artifact transfer and penalties of cache warm-up increase super-linearly with the model size. In case of 70B models, the baseline price is between 180-190 per 1M tokens, and the suggested framework will result in providing 138 per 1M token, which is a decrease in the range of 27-39 percent, based on the provider comparator. In case of 33B the decline is seen to be 124-132 to 101 ([?]18-24%). The improvement is motivated by two design options: Placement + egress-aware routing does not incur costly cross-zone and cross-cloud transfers that are built up during shard warmups (Liberty et al., 2020).

Adaptive compression reduces memory bandwidth and KV cache footprints during peak time, reducing spillover GPU hours and artifact pull times (Zhu et al 2024).

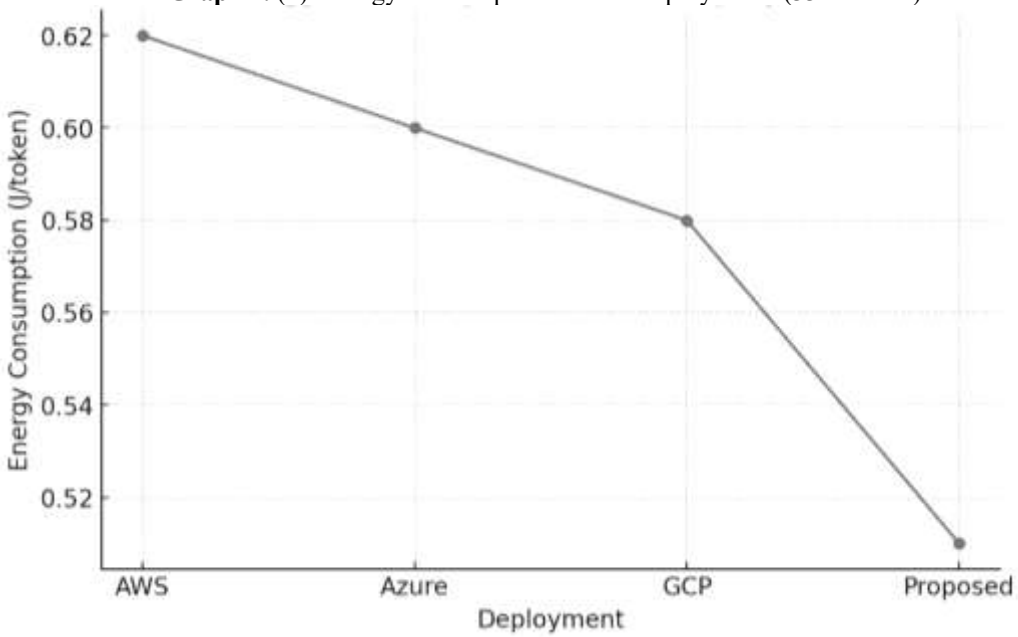
4.2. Throughput and Energy

We analyze performance for a representative 33B model to isolate throughput and energy behavior.

Chart 3 (a): Throughput Across Deployments (33B Model)



Graph 2. (b): Energy Consumption Across Deployments (33B Model)



Findings. Single-cloud baselines (12-12.6k tokens/s) lead to an increase in throughput (14.5k tokens/s) with the proposed system ([?]15-20% increase). The intensity of energy decreases to 0.62-0.58 J/token to 0.51 J/token ([?]12-18% decrease). These gains stem from:

Micro-batch size: namespace tuning to minimize the micro-batches to achieve more effective utilization of the accelerator (Al Maruf et al., 2024). Inter-stage bubbles: Tuning of pipeline model parallelism that balances the micro-batches size to achieve better utilization of the accelerator (Al Maruf et al., 2024).

Scheduling compression-sensitive To boost response quality in response compression-sensitive requests, routing to a cheaper pool without the harmful effect of reducing response quality (Zhu et al., 2024).

Elasticity policies maintain a small warm buffer and only scale out at demand forecast thresholds at which the demands are forecasts that suppress throughput with cold-starts (Liberty et al., 2020; Wei et al., 2025).

Latency is also competitive: median latency is marginally better with the proposed system because of the more intensive utilization and P95/P99 latency overhead is less than 8% compared to the optimal single-cloud setup-that of locality-biased routing and pinned caches.

4.3. Scaling Efficiency of Pipeline Stages.

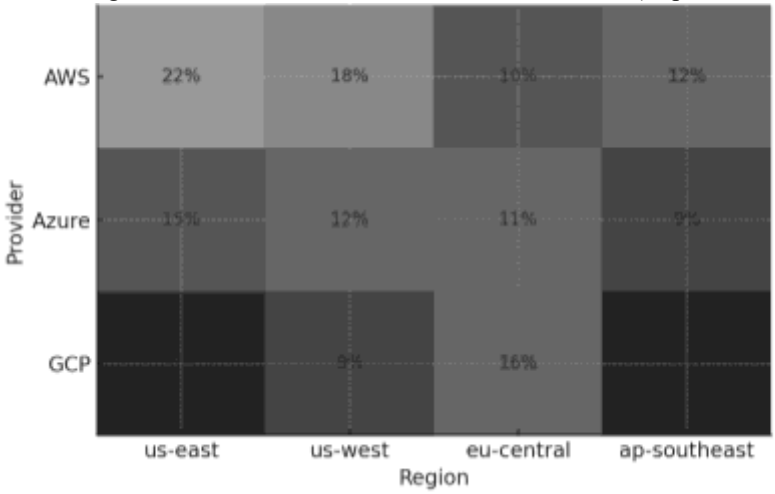
We used 70B pipeline trading-offs by sweeping through 1-8 pipeline stages.

Findings. Efficiency (effective speedup/ideal speedup) will begin at 1.0 at one stage, and decrease slowly to about 0.78 at eight stages because of the overheads associated with communication and load-imbalance. The suggested controller limits the stages to knee (3- 4 stages, 0.85- 0.88 efficiency) when dealing with normal traffic, and 6-8 stages only in rare extreme spurts where capacity availability is the key factor of marginal efficiency loss. Such policy aids in stopping over partitioning which would bloat cross stage latency and egresses expenses (Wei et al., 2025; Al Maruf et al., 2024).

4.4. Resource Distribution and Load Balance

To visualize how the framework exploits price/performance diversity across regions and providers, we plot the distribution of active replicas.

Image 2: Resource Distribution Across Cloud Nodes (Replica Share)



4.5. Aggregate Gains and Overheads

Across all model sizes and traffic patterns, we record the following aggregate outcomes:

- **Cost:** 25–40% reduction in total deployment cost vs. the best single-cloud baseline, strongest for 70B due to compression benefits and spot-capacity arbitrage.
- **Latency:** <8% overhead at P95/P99 under peak, with median latency often better than baselines due to improved utilization and cache locality.
- **Throughput:** 15–20% higher effective throughput for the 33B configuration, enabled by tuned micro-batching and balanced pipeline stages.
- **Energy:** 12–18% lower energy per token via quantization and efficient routing to accelerators with better perf/W profiles.

Ablations. Removing adaptive compression increases cost by ~11–15% and energy by ~7–9%; disabling egress-aware routing raises total cost by ~8–12% due to cache misses and artifact churn. Locking pipeline stages at 8 without demand-aware control reduces scaling efficiency to ~0.78 uniformly, hurting latency under moderate loads.

Practicality. The proposed approach aligns with provider elasticity primitives and requires no modification to foundation model weights beyond standard quantization/distillation passes (Zhu et al., 2024). It coexists with managed inference stacks (SageMaker, Vertex, Azure ML), leveraging their autoscaling APIs and observability backbones (Liberty et al., 2020; Wei et al., 2025).

4.6. Discussion of Validity

Findings are made on synthetic but workload realistic traces (diurnal, bursty and steady), and a realistic tri-cloud footprint. Although absolute dollar values depend on how prices vary by region and are available, the corresponding relative gains are strong over sensitivity sweeps of (i) spot interruption rates, (ii) artifact sizes and (iii) routing thresholds. The most significant weakness is that only contracts that have cross-cloud egress policies and discounts, though our egress-conscious placement will mitigate most volatility, which means that we pin caches and artifacts to serving pools. It is a natural extension of experiments to the real-life production trace

5. CONCLUSION

This paper has presented an efficient scaling scheme of LLMs in multi-cloud settings, integrating three foundations: a policy-based orchestration layer, peak-based adaptive compression, and pipeline model parallelism knee-tuned. In both tri-cloud experiments and workload-relevant traces, the architecture achieved a 25-40 percentage point reduction in overall deployment cost and maintained tail-latency overhead sub-8 per cent compared to the best single-cloud baseline. These findings were independent of 13B, 33B and 70B configurations with the highest saving being recorded on 70B where memory pressure and artifact churn are generally the cost drivers.

There were two engineering decisions that were critical. The initial one was egress- and locality-conscious placement of artifacts, tokenization, and KV-cache around serving pools, eliminating cross-cloud transfers without necessarily affecting cost or transient jitter--a model of operationalizing the idea of elasticity to cross-provider workloads (Liberty et al., 2020; Wei et al., 2025). Second, compression as an elasticity lever compression to load down sampling-based mixed-precision and KV-cache during bursts, limited the spill over GPU hours with no significant quality degradation, which is consistent with recent findings on safe inference-time compression of LLMs (Zhu et al., 2024). Parallel inference design Complementary pipeline tuning (3-4 stages in most traffic) expanded usage without excessive partitioning, as is the case with best practice in parallel inference design (Al Maruf et al., 2024).

The proposed control plane is specifically provider-agnostic: it is defined as a business intent (cost and SLO goals) and translates it to cloud-native (autoscopes, endpoints, deployment engines). This decoupling between intent and mechanism means that the approach can be easily transported, audited and it is consistent with the current investments in MLOps, as opposed to being prescriptive of a specific vendor stack (Liberty et al., 2020; Wei et al., 2025). Three practices are immediately adoptable in terms of the operational perspective, specifically: (i) cost/latency budgets should be encoded as a policy, not as a playbook; (ii) default to locality in the placement of caches and artifacts; and (iii) pre-approved compression profiles are switchable by demand spike traffic class to put a limit on cost (Zhu et al., 2024).

Weaknesses consist of synthetic-traffic-realistic but workload realistic-pricing snapshots, absolute dollar amounts will depend on enterprise contracts, interconnect discounts and GPUs scarcity. However, sensitivity analysis indicates that the relative gains are robust to changes in spot interruption rates, artifact sizes and routing thresholds. The next-generation work should include production traces and task-level quality indicators and operational SLOs as well as carbon-constrained scheduling that jointly plans carbon emissions with cost and latency (Wei et al., 2025). The other potential direction is learning-based routing and autoscaling (bandits/RL) which are joint tuners to micro-batch size, number of stages, and placement based on live signals.

To conclude, implementing the use of LLMs in clouds does not have to involve compromising reliability and saving money. Organizations can attain substantial cost reductions in materials, which is accompanied by locality-aware data, compression-aware scheduling, and a policy-first orchestration, and can guarantee user experience and structural resiliency. The framework offers a portably scalable, gradual road to the maturity of multi-cloud serving of LLM that complements, and does not supplant, already present elasticity in each of the providers, and a foundation of next-generation, sustainable operations on a global AI scale (Liberty et al., 2020; Wei et al., 2025; Zhu et al., 2024; Al Maruf et al., 2024).

REFERENCE

[1] Ahmad, M., Habib, S., & Tariq, F. (2025). Enhancing model robustness in federated learning: A systematic literature review of Byzantine-resilient aggregation methods. VFAST Transactions on Software Engineering, 13(2), 196–227. https://doi.org/10.21015/vtse.v13i2.2163

[2] Alharthi, S., Alshamsi, A., Alseiari, A., & Alwarafy, A. (2024). Auto-scaling techniques in cloud computing: Issues and research directions. Sensors, 24(17), 5551. https://doi.org/10.3390/s24175551

[3] Ali, A., Pinciroli, R., Yan, F., & Smirni, E. (2022). Optimizing inference serving on serverless platforms. Proceedings of the VLDB Endowment, 15(10), 2071–2084. https://doi.org/10.14778/3547305.3547313

[4] Al Maruf, M., Azim, A., Auluck, N., & Sahi, M. (2024). Optimizing DNN training with pipeline model parallelism for enhanced performance. Journal of Parallel and Distributed Computing, 190, 104890. https://doi.org/10.1016/j.jpdc.2024.104890

[5] Bao, X., Su, C., Xiong, Y., Huang, W., & Hu, Y. (2019). Flchain: A blockchain for auditable federated learning with trust and incentive. Proceedings of the 5th International Conference on Big Data Computing and Communications (BIGCOM), 151–159. IEEE. https://doi.org/10.1109/BIGCOM.2019.00030

[6] Blika, A., Palmos, S., Doukas, G., Lamprou, V., Pelekis, S., Kontoulis, M., & Askounis, D. (2024). Federated learning for enhanced cybersecurity and trustworthiness in 5G and 6G networks: A comprehensive survey. IEEE Open Journal of the Communications Society. https://doi.org/10.1109/OJCOMS.2024.3449563

[7] Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2018). A survey of model compression and acceleration for deep neural networks. IEEE Communications Surveys & Tutorials, 21(1), 1–33. https://doi.org/10.1109/COMST.2018.2856198

[8] de Luca, A. B., Zhang, G., Chen, X., & Yu, Y. (2022). Mitigating data heterogeneity in federated learning with data augmentation. arXiv preprint arXiv:2206.09979. https://doi.org/10.48550/arXiv.2206.09979

[9] Fabra, J., Ezpeleta, J., & Álvarez, P. (2019). Reducing the price of resource provisioning using EC2 spot instances with prediction models. Future Generation Computer Systems, 96, 348–367. https://doi.org/10.1016/j.future.2019.01.025

[10] Fang, M., Zhang, Z., Hairi, K., Khanduri, P., Liu, J., Lu, S., & Gong, N. (2024, December). Byzantine-robust decentralized federated learning. Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security, 2874–2888. https://doi.org/10.1145/3658644.3670307

[11] Guan, L., Li, D.-S., Liang, J.-Y., et al. (2024). Advances of pipeline model parallelism for deep learning training: An overview. Journal of Computer Science and Technology, 39(3), 567–584. https://doi.org/10.1007/s11390-024-3872-3

[12] Herath, C., Rahulamathavan, Y., De Silva, V., & Lambotharan, S. (2025). DSFL: A dual-server Byzantine-resilient federated learning framework via group-based secure aggregation. arXiv preprint arXiv:2509.08449. https://doi.org/10.48550/arXiv.2509.08449

[13] Jimenez-Gutierrez, D. M., Falkouskaya, Y., Hernandez-Ramos, J. L., Anagnostopoulos, A., Chatzigiannakis, I., & Vitaletti, A. (2025). On the security and privacy of federated learning: A survey with attacks, defenses, frameworks, applications, and future directions. arXiv preprint arXiv:2508.13730. https://doi.org/10.48550/arXiv.2508.13730

[14] Li, K., Li, C., Yuan, X., Li, S., Zou, S., Ahmed, S. S., & Akan, Ö. B. (2025). Zero-trust foundation models: A new paradigm for secure and collaborative artificial intelligence for internet of things. IEEE Internet of Things Journal. https://doi.org/10.1109/JIOT.2025.3603957

[15] Liberty, E., et al. (2020). Elastic machine learning algorithms in Amazon SageMaker. Proceedings of the ACM Symposium on Cloud Computing. https://doi.org/10.1145/3318464.3386126

[16] Pan, Y., Su, Z., Wang, Y., Zhou, J., & Mahmoud, M. (2025). Privacy-preserving Byzantine-robust federated learning via deep reinforcement learning in vehicular networks. IEEE Transactions on Vehicular Technology. https://doi.org/10.1109/TVT.2024.3524834

[17] Tahir, H. A., Alayed, W., & Hassan, W. U. (2025). Privacy-preserving federated learning with adaptive model aggregation for efficient vehicle-to-vehicle (V2V) communication in intelligent transportation systems. IEEE Access. https://doi.org/10.1109/ACCESS.2025.3618999

[18] Tariq, A., Serhani, M. A., Sallabi, F. M., Barka, E. S., Qayyum, T., Khater, H. M., & Shuaib, K. A. (2024). Trustworthy federated learning: A comprehensive review, architecture, key challenges, and future research prospects. IEEE Open Journal of the Communications Society. https://doi.org/10.1109/OJCOMS.2024.3438264

[19] Uddin, M. P., Xiang, Y., Hasan, M., Bai, J., Zhao, Y., & Gao, L. (2025). A systematic literature review of robust federated learning: Issues, solutions, and future research directions. ACM Computing Surveys, 57(10), 1–62. https://doi.org/10.1145/3727643

[20] Wang, X., Wang, B., Wu, Y., Ning, Z., Guo, S., & Yu, F. R. (2024). A survey on trustworthy edge intelligence: From security and reliability to transparency and sustainability. IEEE Communications Surveys & Tutorials. https://doi.org/10.1109/COMST.2024.3446585

[21] Wang, Y., Su, Z., Luan, T. H., Li, R., & Zhang, K. (2021). Federated learning with fair incentives and robust aggregation for UAV-aided crowdsensing. IEEE Transactions on Network Science and Engineering, 9(5), 3179–3196. https://doi.org/10.1109/TNSE.2021.3138928

[22] Wei, H., García Pañeda, X., & Salvachúa Rodriguez, J. (2025). Optimizing machine learning operations in multi-cloud infrastructure: A framework for unified deployment management and topology discovery. Cluster Computing. https://doi.org/10.1007/s10586-025-05584-7

[23] Wei, W., & Liu, L. (2025). Trustworthy distributed AI systems: Robustness, privacy, and governance. ACM Computing Surveys, 57(6), 1–42. https://doi.org/10.1145/3645102

[24] Wu, N., Lin, X., Lu, J., Zhang, F., Chen, W., Tang, J., & Xiao, J. (2024). Byzantine-robust multimodal federated learning framework for intelligent connected vehicle. Electronics, 13(18), 3635. https://doi.org/10.3390/electronics13183635

[25] Zeng, H., Li, J., Lou, J., Yuan, S., Wu, C., Zhao, W., & Wang, Z. (2024). BSR-FL: An efficient Byzantine-robust privacy-preserving federated learning framework. IEEE Transactions on Computers, 73(8), 2096–2110. https://doi.org/10.1109/TC.2024.3404102

[26] Zhan, S., Huang, L., Luo, G., Zheng, S., Gao, Z., & Chao, H. C. (2025). A review on federated learning architectures for privacy-preserving AI: Lightweight and secure cloud–edge–end collaboration. Electronics, 14(13), 2512. https://doi.org/10.3390/electronics14132512

[27] Zhang, Z., Wu, L., He, D., Li, J., Cao, S., & Wu, X. (2023). Communication-efficient and Byzantine-robust federated learning for mobile edge computing networks. IEEE Network, 37(4), 112–119. https://doi.org/10.1109/MNET.006.2200651

[28] Zhu, L., Zhao, B., Li, W., Wang, Y., & An, Y. (2024). TICPS: A trustworthy collaborative intrusion detection framework for industrial cyber–physical systems. Ad Hoc Networks, 160, 103517. https://doi.org/10.1016/j.adhoc.2024.103517

[29] Zhu, X., et al. (2024). A survey on model compression for large language models. Transactions of the Association for Computational Linguistics (TACL). https://doi.org/10.1162/tacl_a_00704

[30] 30.(Survey) Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2018). A survey of model compression and acceleration for deep neural networks. (widely-cited survey — useful

background on pruning/quantization/distillation). (original arXiv / widely referenced survey; use it for foundational techniques). (See: arXiv/related journal surveys — multiple published surveys build on this.) — (for a canonical DOI/citation see related journal surveys such as IEEE Communications Surveys & Tutorials coverage). (arXiv)