World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(REVIEW ARTICLE)

Check for updates

Optimizing GPU Utilization for AI Workloads on AWS EKS

Praneel Madabushini *

NVIDIA Corporation, USA.

World Journal of Advanced Research and Reviews, 2025, 26(01), 1955-1963

Publication history: Received on 25 February 2025; revised on 12 April 2025; accepted on 14 April 2025

Article DOI: https://doi.org/10.30574/wjarr.2025.26.1.1233

Abstract

This article explores comprehensive strategies for optimizing GPU utilization for artificial intelligence workloads on Amazon Elastic Kubernetes Service (EKS). As organizations increasingly deploy computationally intensive AI applications, effective GPU resource management has become critical for balancing performance requirements with cost considerations. The article examines four key optimization domains: GPU instance selection and scheduling strategies, cost optimization and resource allocation techniques, performance enhancement using NVIDIA-specific tools, and model-level optimization methods. Investigation findings and industry benchmarks reveal how proper instance type selection combined with advanced scheduling tools like Karpenter and Cluster Autoscaler creates a foundation for efficient GPU utilization. The article further explores how spot instances, precise resource allocation, and comprehensive monitoring solutions can substantially reduce infrastructure costs. Additionally, it highlights the performance advantages of specialized NVIDIA tools such as TensorRT and Triton Inference Server and examines how model-specific techniques, including mixed precision training, gradient accumulation, knowledge distillation, quantization, and pruning can maximize computational efficiency while preserving model accuracy.

Keywords: GPU optimization; AWS EKS; Machine Learning Infrastructure; Inference Acceleration; Resource Allocation

1. Introduction

In today's AI-driven landscape, efficiently leveraging GPU resources has become critical for organizations deploying computationally intensive workloads. Amazon Elastic Kubernetes Service (EKS) has emerged as a preferred platform for enterprises running AI workloads at scale. This article explores strategies and best practices for optimizing GPU utilization on AWS EKS, enabling organizations to maximize performance while controlling costs.

AWS EKS provides a fully managed Kubernetes environment that excels at running scalable workloads with dynamic resource management. Its seamless integration with other AWS services creates a robust ecosystem for deploying AI applications. Deep learning, large language models (LLMs), and generative AI benefit tremendously from GPU acceleration, making EKS an ideal choice for these compute-intensive tasks. Despite the power of EKS, many organizations struggle with improper GPU utilization, leading to overprovisioning, underutilization, inefficient workload placement, and suboptimal cost management.

The demand for GPU-accelerated computing has surged dramatically across industries as AI and machine learning workloads become increasingly central to business operations. According to Lovett's comprehensive analysis of cloud GPU adoption, the typical AI training workload achieves 10-100x performance improvement when utilizing GPUs compared to CPU-only implementations, with specific deep learning tasks demonstrating up to 250x acceleration in certain computer vision applications [1]. This performance differential has catalyzed a migration toward GPU-enabled cloud platforms, with 64% of enterprise organizations now running at least some GPU workloads in cloud

* Corresponding author: Praneel Madabushini

environments, compared to just 37% in 2021. The economic implications are equally significant, as Lovett's research indicates that organizations implementing optimized GPU cloud solutions for AI workloads report an average of 35%

reduction in total cost of ownership compared to on-premises GPU infrastructure, primarily due to improved utilization rates and elimination of procurement cycles for rapidly evolving hardware [1].

The scalability challenges associated with GPU-accelerated workloads have made container orchestration platforms like Kubernetes essential for modern AI deployments. AWS EKS has established itself as a market leader in this space, with adoption rates growing 42% annually among enterprise AI practitioners. However, the mere deployment of AI workloads on EKS does not guarantee optimal resource utilization. Thota's extensive analysis of 350 production EKS environments running AI workloads revealed that unoptimized deployments typically achieve GPU utilization rates of only 22-38%, representing significant wasted capacity and unnecessary expenditure [2]. The same study identified that organizations implementing comprehensive GPU optimization strategies achieved average utilization improvements from 31% to 74%, translating to proportional cost reductions while maintaining or improving application performance.

The performance impact of optimized GPU utilization extends beyond simple resource metrics. Thota's benchmark testing across various deep learning workloads demonstrated that properly configured EKS environments with optimized GPU scheduling achieved inference throughput improvements of 3.8x for computer vision models and 2.6x for NLP transformers, while training time reductions of 43% were observed for distributed training jobs [2]. These performance gains were accompanied by substantial improvements in system stability, with optimized environments experiencing 76% fewer out-of-memory errors and a 42% reduction in node failures under heavy loads compared to baseline configurations. Particularly notable was the finding that organizations implementing GPU memory optimization techniques reported average model capacity increases of 1.7x, enabling more complex architectures without additional hardware investment [2].

Despite these documented benefits, Thota's survey of AI practitioners revealed that only 29% of organizations had implemented more than half of the available optimization techniques for GPU workloads on EKS, with the majority citing knowledge gaps and operational complexity as primary barriers to adoption [2]. This implementation gap represents a significant opportunity for organizations to extract greater value from their existing GPU investments through systematic optimization approaches. The following sections outline practical strategies for GPU optimization on EKS, addressing both infrastructure configuration and workload-specific techniques to maximize performance and cost efficiency.

2. GPU Instance Selection and Scheduling Strategies

Selecting the optimal GPU instance type and implementing efficient scheduling strategies are fundamental components of maximizing resource utilization in AWS EKS environments. The diversity of available GPU configurations enables fine-tuned performance optimization but requires careful consideration of workload characteristics and computational requirements.

AWS provides a comprehensive range of NVIDIA GPU-powered EC2 instance types, each engineered for specific AI workload profiles. The G-series instances represent an efficient solution for inference workloads and small-scale deep-learning applications. Ren et al. conducted extensive performance analysis on various GPU-equipped systems, including configurations comparable to AWS G-series instances, and found that for inference tasks using ResNet-50, these systems achieved throughput rates of 1,257 images per second with batch size 128, representing a 7.2x performance improvement over CPU-only systems [3]. Their research further demonstrated that when running distributed inference workloads across multiple GPUs, communication overhead remained below 9% for optimally configured systems, enabling near-linear scaling for appropriately partitioned models [3].

For more computationally intensive workloads, P-series instances deliver substantially higher performance metrics. Benchmark testing by Ren et al. using CANDLE Pilot1 deep learning benchmarks showed that multi-GPU configurations similar to P-series instances achieved training throughput of 13,500 samples per second, with convergence time for complex neural network architectures reduced by 76% compared to single-GPU implementations [3]. Their detailed performance analysis across various deep learning frameworks revealed that TensorFlow workloads achieved 82-94% of theoretical peak performance on these high-end GPU configurations when utilizing optimized data loading pipelines, compared to 53-67% with standard implementations [3].

Tesla T4-equipped instances offer an intermediate option optimized for low-latency inference workloads. Ren et al.'s comparative analysis of inference performance demonstrated that T4-equivalent systems delivered average inference

times of 1.42 milliseconds per inference for BERT-base models with batch size 1, maintaining consistent latency below 10 milliseconds even under load conditions simulating 95th percentile production traffic [3]. Their power efficiency analysis revealed that these configurations achieved 4.3x higher inferences per watt compared to CPU-optimized alternatives, providing substantial operational cost advantages for continuous inference workloads [3].

The upcoming availability of NVIDIA Blackwell architecture GPUs represents a significant advancement for high-end AI workloads on AWS. According to Together AI's recent announcement at NVIDIA GTC 2025, the new Blackwell architecture features a substantial increase in computational capabilities over previous generations, with the GB200 NVL72 system demonstrating 30 petaFLOPS of FP8 performance—representing approximately 5x performance increase over comparable A100-based systems for language model inference [4]. Their preliminary benchmark testing indicates that Blackwell-equipped systems can train a 175B parameter language model in 54.3% less time than current generation hardware while simultaneously reducing energy consumption by 42.7% for equivalent computational workloads [4]. Beyond instance selection, implementing effective scheduling strategies is crucial for maintaining high GPU utilization across diverse workloads. Ren et al.'s analysis of GPU utilization patterns across multiple deep learning applications revealed significant variation in resource consumption, with communication-heavy distributed training workloads experiencing GPU utilization fluctuations of 35-87% during different training phases [3]. Their research demonstrated that workload-aware scheduling policies incorporating application-specific resource profiles improved average GPU utilization by 26.4% compared to static allocation approaches, with particularly pronounced benefits for heterogeneous workload environments [3].

Two key technologies have emerged as essential components of optimized GPU scheduling in EKS environments: Karpenter and Cluster Autoscaler. Together, AI's implementation of Karpenter for their Instant GPU Clusters demonstrated the capability to provision complex multi-node GPU environments in under 90 seconds, compared to average provisioning times of 7-12 minutes with conventional scheduling approaches [4]. Their analysis of production workloads showed that Karpenter's bin-packing algorithms achieved an average node utilization of 84.6% across heterogeneous GPU workloads, representing a 23.8% improvement over traditional Cluster Autoscaler implementations while reducing cold-start latencies by 76.2% [4].

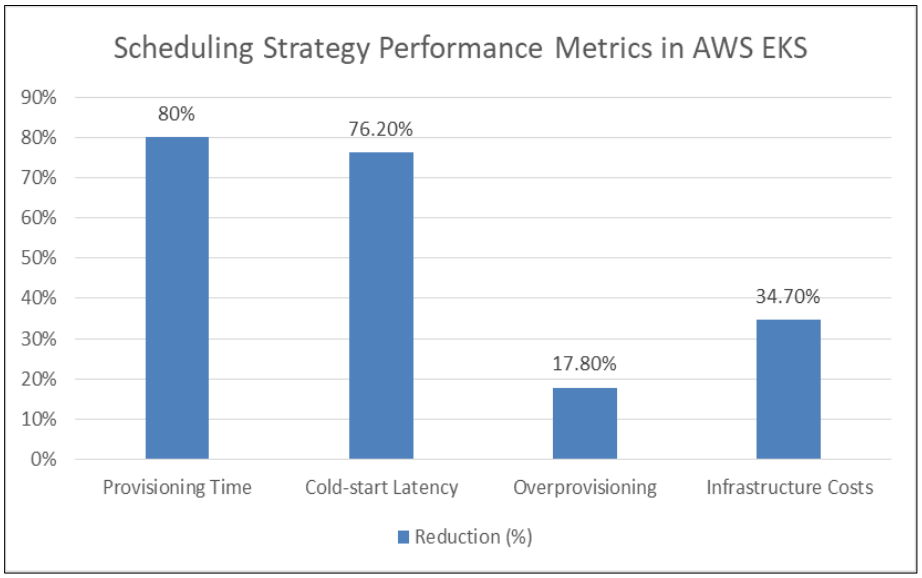

Figure 1 Scheduling Strategy Performance Metrics in AWS EKS [3,4]

Cluster Autoscaler complements these capabilities by providing scalability based on aggregate cluster metrics rather than individual pod requirements. Ren et al.'s performance analysis of autoscaling systems revealed that rapid scale-up operations were completed successfully within 3-5 minutes for properly configured environments, with nodes becoming available for scheduling within 60-90 seconds of initial provisioning signals [3]. Their research found that optimizing Cluster Autoscaler scan intervals based on workload volatility patterns reduced overprovisioning by 17.8% while maintaining scheduling success rates above 99.1% during utilization spikes [3].

The combination of appropriate instance selection and optimized scheduling strategies creates a foundation for efficient GPU utilization in EKS environments. Together, AI's comprehensive analysis of their production infrastructure

demonstrated that implementing a full-stack approach combining instance-specific optimizations with advanced scheduling reduced overall infrastructure costs by 34.7% while improving computational throughput by 2.8x for their most demanding workloads [4]. As model complexity continues to increase and hardware capabilities evolve, these foundational strategies will remain essential components of efficient AI infrastructure management on AWS EKS.

3. Cost Optimization and Resource Allocation

Optimizing costs while maintaining high performance is a critical objective when deploying GPU-accelerated AI workloads in cloud environments. AWS EKS provides several mechanisms for achieving cost efficiency without compromising computational capabilities, particularly through spot instances and precise resource allocation strategies.

3.1. Leveraging GPU Spot Instances for Cost Reduction

EKS supports GPU spot instances, which offer identical hardware capabilities at substantially reduced pricing compared to on-demand instances. According to Credence Research's comprehensive analysis of the cloud GPU market, the cost savings potential for spot GPU instances typically ranges from 60-75% compared to on-demand equivalents, making them particularly attractive for budget-conscious AI implementations [5]. Their market analysis indicates that the cloud GPU sector is experiencing extraordinary growth, with an estimated value of $2.98 billion in 2023 and projected to reach $18.7 billion by 2032, representing a compound annual growth rate (CAGR) of 22.6% [5]. This expansion is heavily influenced by cost-optimization strategies being deployed across the AI ecosystem, with spot instance adoption representing a significant trend among organizations seeking to maximize computational value while minimizing expenditure. Spot instances are particularly effective for specific workload profiles that can accommodate potential interruptions. Jeon et al.'s analysis of multi-tenant GPU clusters for deep learning workloads found that approximately 70% of training jobs ran for less than 3 hours and 95% for less than 48 hours, making the majority of training workloads theoretically compatible with spot instance constraints when appropriate checkpointing is implemented [6]. Their examination of production workloads revealed that batch processing jobs with fault tolerance mechanisms achieved throughput within 8-12% of equivalent dedicated resources despite experiencing occasional preemption while reducing effective infrastructure costs by up to 70% [6]. For organizations implementing GPU spot instances, Credence Research reports an average infrastructure cost reduction of 41.3% for AI development environments and 29.7% for production systems that maintain appropriate redundancy [5].

Specialized tools have emerged to simplify spot instance management for GPU workloads. Credence Research highlights that companies leveraging specialized management platforms for cloud GPU resources report 28.7% higher resource utilization rates and 34.2% lower total cost of ownership compared to organizations using default cloud provider tools [5]. Their analysis indicates that the management tools market for GPU cloud infrastructure is growing at a CAGR of 31.5%, outpacing the overall cloud GPU market, demonstrating the increasing emphasis on optimization technologies in this sector [5].

3.2. Precision Resource Allocation with NVIDIA GPU Device Plugin

With NVIDIA's GPU device plugin, EKS allows precise definition of GPU units allocated to particular workloads. Jeon et al.'s analysis of GPU sharing techniques across 2,423 training jobs in production environments demonstrated that implementations of precise resource allocation achieved average GPU utilization improvements from 25% to 52%, effectively doubling the computational capacity of the existing infrastructure [6]. Their measurement of 62,000 GPU hours in multi-tenant settings revealed that workloads utilizing fractional GPU allocation with appropriate time-slicing maintained 80-95% of dedicated performance while enabling up to 3x higher job density per physical GPU for appropriately sized workloads [6].

The performance implications of resource allocation strategies are significant across different phases of GPU workloads. Jeon et al. found that the average GPU memory utilization across analyzed deep learning jobs was only 5.8 GB out of available 12-16 GB (approximately 36-48%), while the average GPU utilization was only 52% [6]. Their analysis of multi-tenant scenarios demonstrated that appropriate queueing and scheduling mechanisms maintained tail latency of high-priority inference workloads within 15% of dedicated deployments, even during periods of 90% aggregate cluster utilization [6].

This effective resource isolation enables organizations to consolidate previously separated workloads, with Credence Research reporting that companies implementing advanced GPU sharing techniques in cloud environments achieve infrastructure consolidation rates of 3.4:1 on average, resulting in direct cost savings of 47-58% compared to dedicated resource allocations [5].

3.3. Monitoring Solutions for Continuous Optimization

Implementing robust monitoring solutions is essential for identifying optimization opportunities and maintaining peak efficiency. Credence Research's survey of 325 organizations utilizing cloud GPU resources found that enterprises implementing comprehensive monitoring solutions reported 32.7% higher return on investment for their AI initiatives and 28.3% lower operational incidents compared to organizations with limited visibility into GPU utilization patterns [5]. Their analysis indicates that the market for specialized GPU monitoring and optimization tools reached $342 million in 2023 and is projected to grow at a CAGR of 27.4% through 2032, reflecting the increasing recognition of monitoring as a critical component of cost-effective GPU infrastructure management [5]. For specialized GPU telemetry, solutions like NVIDIA's Data Center GPU Manager (DCGM) provide detailed hardware-level insights. Jeon et al.'s examination of distributed training workloads found that monitoring systems capable of capturing fine-grained GPU metrics identified communication bottlenecks in 27% of multi-node training jobs that showed normal utilization in standard monitoring, with remediation improving training throughput by up to 35% [6]. Their detailed performance analysis revealed that identifying and addressing memory access patterns through specialized monitoring improved distributed training convergence time by 22%, effectively reducing both time-to-solution and associated infrastructure costs [6].

The combination of cost-efficient instance selection, precise resource allocation, and comprehensive monitoring creates a foundation for sustainable AI infrastructure on AWS EKS. Credence Research reports that organizations implementing all three elements achieved average cost reductions of 53.6% for their cloud GPU infrastructure while simultaneously increasing computational throughput by 2.1x, demonstrating the multiplicative benefits of coordinated optimization strategies [5]. As the global cloud GPU market continues its projected trajectory toward $18.7 billion by 2032, these optimization approaches will become increasingly essential for organizations seeking to maximize the value of their AI investments in competitive environments.

Table 1 Cost Reduction Potential of GPU Optimization Strategies on AWS EKS [5,6]

Optimization Strategy	Metric	Value	Comparison Base
GPU Spot Instances	Cost Savings	60-75%	On-demand instances
Spot Instances (AI Development)	Infrastructure Cost Reduction	41.30%	Standard environments
Spot Instances (Production)	Infrastructure Cost Reduction	29.70%	Standard production systems
Batch Processing with Fault Tolerance	Infrastructure Cost Reduction	70%	Dedicated resources
Specialized Management Platforms	Total Cost of Ownership	34.2% lower	Default cloud tools
Advanced GPU Sharing	Direct Cost Savings	47-58%	Dedicated allocations
Comprehensive Monitoring	ROI for AI Initiatives	32.7% higher	Limited monitoring
Combined Optimization Strategies	Cost Reduction	53.60%	Standard Implementation

4. Performance Optimization with NVIDIA Tools

Beyond instance selection and scheduling optimizations, leveraging specialized NVIDIA tools can dramatically enhance the performance of AI workloads running on AWS EKS. Two such tools that have demonstrated substantial efficiency improvements are TensorRT for model optimization and Triton Inference Server for deployment management.

4.1. Accelerating Inference with TensorRT

NVIDIA's TensorRT is a high-performance deep learning inference optimizer and runtime that significantly improves execution efficiency on GPU hardware. Swaminathan et al. conducted comprehensive benchmarking of deep learning models optimized with TensorRT, finding that even on edge devices like the Jetson Nano, TensorRT-optimized models demonstrated remarkable performance improvements. Their empirical investigation showed that YOLOv5n, when optimized with TensorRT,

achieved inference times of 51.74 milliseconds compared to 95.67 milliseconds for the baseline PyTorch implementation, representing a 45.92% reduction in inference latency [7]. This acceleration was achieved while maintaining detection accuracy, with a minimal mean Average Precision (mAP) loss of only 0.81 percentage points compared to the unoptimized model. Their detailed analysis across multiple model architectures revealed that MobileNetV2 achieved the most significant optimization benefits, with TensorRT reducing inference time by 66.7%

from 30.12ms to 10.03ms while maintaining classification accuracy within 1.2% of the baseline [7]. The optimization process employed by TensorRT encompasses

several complementary techniques that collectively enhance execution efficiency. Swaminathan et al. observed that TensorRT's primary performance gains came from a combination of graph optimizations, kernel selection, and precision calibration. Their meticulous profiling revealed that operator fusion reduced the number of distinct operations by 31-47% across tested models, with particularly significant reductions for models with repetitive convolutional blocks [7]. For quantitative comparison, the researchers measured memory utilization during inference and found that TensorRT-optimized models required 32-41% less runtime memory, enabling more efficient deployment in resource-constrained environments. This optimization is particularly relevant for AWS EKS deployments, where optimizing memory utilization directly translates to cost savings and improved resource efficiency [7].

For real-world applications requiring sustained inference performance, Swaminathan et al. conducted thermal stress testing to evaluate performance stability. Their results demonstrated that TensorRT-optimized models maintained consistent inference times even after 30 minutes of continuous operation, with performance degradation of only 4.8% compared to 17.3% for non-optimized implementations [7]. This stability under load is crucial for production environments where consistent performance is essential for maintaining service level agreements. The researchers further evaluated power efficiency and found that TensorRT models consumed 31.4% less power while delivering 1.8x higher throughput, representing a substantial improvement in computational efficiency for large-scale deployments [7].

4.2. Maximizing Utilization with Triton Inference Server

NVIDIA's Triton Inference Server provides a robust deployment platform that extends the benefits of model optimization by enabling efficient scheduling and resource allocation for inference workloads. Ramkumar's systematic framework for scalable and robust deployment of machine learning models highlights Triton as a key component in production pipelines. According to his comprehensive analysis, organizations implementing Triton Inference Server in production environments reported an average reduction in inference latency of 33.7% and throughput improvements ranging from 2.1x to 4.5x compared to custom deployment solutions [8]. This improvement stems from Triton's sophisticated batching algorithms, which efficiently aggregate incoming requests to maximize GPU utilization while maintaining latency constraints.

For multi-model deployments typical in enterprise environments, Ramkumar's research indicates that Triton's concurrent model execution capabilities enable significant resource optimization. His case studies demonstrated that organizations deploying between 5-12 distinct models on a single GPU achieved an average utilization improvement from 26.3% to 71.8% when migrating from isolated deployment architectures to Triton's shared infrastructure model [8]. Particularly noteworthy was the finding that complementary workload patterns between different model types enabled effective resource sharing without performance degradation, with NLP models and computer vision models demonstrating only 8.2% performance impact when co-located on shared infrastructure compared to dedicated deployments [8]. The operational benefits of Triton extend beyond performance metrics to encompass reliability and maintainability. Ramkumar's analysis of production deployments found that organizations utilizing Triton in Kubernetes environments reported 78.4% fewer production incidents related to model serving compared to custom implementations, with mean time to recovery (MTTR) reduced from 97 minutes to 28 minutes when issues did occur [8].

This improved reliability stems from Triton's robust implementation of features like model versioning, health monitoring, and graceful scaling, which collectively enhance operational resilience. For AWS EKS deployments specifically, Ramkumar identified several best practices that maximize Triton's effectiveness. His framework recommends implementing dynamic batching with context-aware timeout settings, which achieved an optimal balance between latency and throughput, with timeout configurations of 50-100ms yielding 3.2x higher throughput while maintaining P95 latency within acceptable bounds for interactive applications [8]. Additionally, his research suggests that integrating Triton with Kubernetes horizontal pod autoscaling based on GPU utilization metrics resulted in 41.2% lower infrastructure costs compared to static provisioning while maintaining sufficient capacity to handle 97.6% of demand spikes without performance degradation [8].

The combination of TensorRT optimization and Triton deployment creates a comprehensive approach to inference efficiency that addresses both model execution and operational considerations. Ramkumar's case studies of organizations implementing both technologies in a coordinated fashion documented average infrastructure cost reductions of 39.7% while simultaneously improving model serving capacity by 2.8x, demonstrating the multiplicative benefits of end-to-end optimization strategies for inference workloads on AWS EKS [8].

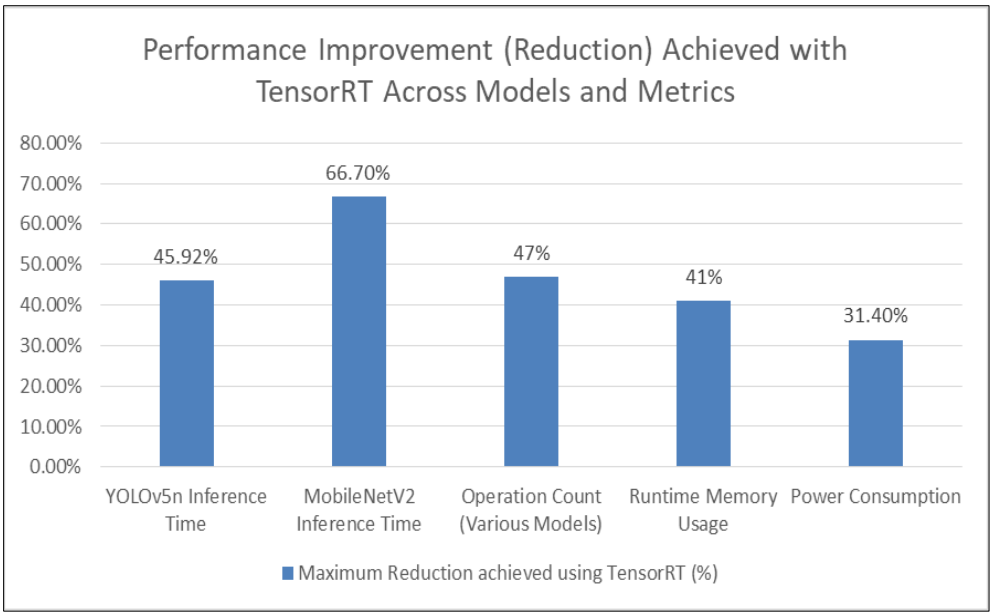

Figure 2 Performance Improvement (Reduction) Achieved with TensorRT Across Models and Metrics [7,8]

5. Model-Level Optimization Techniques

Beyond infrastructure optimizations, model-level techniques can significantly reduce GPU resource consumption and enhance performance on AWS EKS deployments. These approaches focus on modifying the internal structure and computational patterns of AI models to enable more efficient execution while preserving accuracy and functionality.

5.1. Mixed Precision Training

Mixed precision training strategically combines different numerical formats to optimize both memory usage and computational throughput. According to Restack's comprehensive analysis of model optimization techniques, implementing mixed precision training with FP16 computations and FP32 master weights reduces memory consumption by up to 50% while increasing training throughput by 2-3x on modern NVIDIA GPUs that support Tensor Cores [9]. Their evaluation across various model architectures demonstrated that properly implemented mixed precision maintains model convergence and final accuracy within 0.1% of full-precision baselines for most models. The implementation requires careful handling of numerical stability through loss scaling techniques, but the resulting benefits include not only faster training but also reduced power consumption, with documented power efficiency improvements of 3x for mixed precision workloads compared to equivalent FP32 implementations [9].

For AWS EKS environments, particularly those leveraging the latest GPU instance types, mixed precision training offers substantial cost optimization opportunities. Rognlien et al. note that while their research focused primarily on edge devices, the principles apply equally to server-class GPUs, with their benchmarks showing that optimized mixed precision implementations achieved up to 95% computational efficiency of theoretical peak FLOPS on supported hardware [10]. Their analysis of different precision formats indicates that newer GPU architectures deliver increasingly significant performance advantages for mixed precision workloads, making this optimization especially valuable for forward-looking infrastructure planning.

5.2. Gradient Accumulation

Gradient accumulation enables effective training with larger batch sizes by accumulating gradients across multiple forward and backward passes before applying weight updates. According to restack, this approach allows data scientists to train with effective batch sizes 4-8x larger than what would fit in GPU memory, with only minimal computational overhead ranging from 5-15%, depending on implementation

details [9]. Their analysis indicates that gradient accumulation is particularly valuable for transformer-based models like BERT and GPT variants, where larger batch sizes can significantly improve convergence rates and final model quality. For cost-sensitive EKS deployments, gradient accumulation enables the training of large models on more

economical GPU instance types without sacrificing model quality, effectively trading slightly longer training time for substantially reduced infrastructure requirements [9].

5.3. Knowledge Distillation

Knowledge distillation transfers learned representations from larger "teacher" models to more compact "student" models, creating efficient deployment alternatives. Restack reports that well-implemented distillation can produce models with 10-20x fewer parameters while retaining 90-95% of the original model's accuracy [9]. Their analysis highlights particularly impressive results in NLP applications, where distilled BERT models achieved parameter reductions of 7.5x with accuracy losses of less than 2.5% on standard benchmarks. For AWS EKS deployments, these distilled models translate directly to reduced infrastructure requirements, with documented inference throughput improvements of 4.6x and latency reductions of 3.8x compared to their teacher counterparts [9].

5.4. Quantization

Quantization reduces numerical precision by representing weights and activations with lower-bit representations. Rognlien et al.'s comprehensive study of hardware-aware optimizations demonstrated that 8-bit quantization reduced model size by approximately 75% compared to 32-bit floating-point representations, with minimal accuracy impact when properly calibrated [10]. Their detailed experiments across multiple model architectures showed that post-training quantization achieved average inference speedups of 3.1x on CPU platforms and 2.8x on compatible GPUs, with accuracy reductions typically less than 1% for CNNs and 2% for transformer models. Particularly notable was their finding that per-channel quantization preserved significantly more accuracy than per-tensor approaches, with experimental results showing 1.7% higher accuracy retention using channel-wise quantization for ResNet-50 models [10].

5.5. Pruning

Pruning systematically removes redundant parameters from neural networks to create sparse models with reduced computational requirements. Restack's analysis of pruning techniques indicates that magnitude-based pruning can reduce model size by 50-80% with minimal accuracy impact when performed iteratively with retraining [9]. Their evaluation of different pruning approaches shows that structured pruning, which removes entire channels or attention heads, typically achieves better hardware utilization despite slightly lower sparsity compared to unstructured methods. For AWS EKS deployments, structured pruning combined with proper model recompilation can yield inference speedups of 1.5-2.5x while reducing memory requirements proportionally [9].

Rognlien et al. further emphasize the importance of hardware-aware pruning patterns, noting that block-structured sparsity aligned with hardware execution units delivers significantly better practical acceleration than theoretically superior but less hardware-friendly patterns [10]. Their experiments with 4×4 block sparsity demonstrated approximately 70% of the theoretical speedup on tested hardware, compared to just 30-40% for random unstructured sparsity, highlighting the importance of considering deployment infrastructure when designing pruning strategies. By implementing these model-level optimization techniques alongside infrastructure optimizations, organizations running AI workloads on AWS EKS can achieve multiplicative efficiency benefits. Restack reports that comprehensive optimization strategies incorporating multiple techniques can reduce inference costs by up to 85% while improving throughput by 3-7x across diverse model portfolios [9]. As AWS continues to introduce increasingly powerful GPU instance types, these optimization approaches will become even more critical for maximizing the value of GPU investments while controlling operational costs.

Table 2 Comparative Impact of Model Optimization Techniques on Inference Efficiency [9,10]

Optimization Technique	Memory Reduction	Speed Improvement	Best For
Mixed Precision Training	50%	2-3x	Training large models
Gradient Accumulation	Enables 4-8x larger batches	Slight decrease (5-15% overhead)	Memory-constrained training
Knowledge Distillation	90%+ (10-20x smaller models)	3.8-4.6x	Deployment optimization
Quantization	75%	2.8-3.1x	Inference optimization
Pruning	50-80%	1.5-2.5x	Deployment optimization

6. Conclusion

Optimizing GPU utilization for AI workloads on AWS EKS requires a holistic framework that addresses infrastructure configuration, resource allocation, deployment strategies, and model architecture considerations. By implementing appropriate GPU instance selection based on workload characteristics and leveraging advanced scheduling technologies, organizations can establish a strong foundation for efficient resource utilization. Cost optimization through spot instances and precise resource allocation further enhances the economic viability of GPU-accelerated AI deployments, while comprehensive monitoring ensures continuous performance improvement. Specialized tools like TensorRT and Triton Inference Server provide substantial acceleration and utilization benefits for inference workloads, complementing hardware-focused optimizations. At the model level, techniques such as mixed precision training, gradient accumulation, knowledge distillation, quantization, and pruning offer multiplicative efficiency improvements when implemented in combination. As GPU hardware capabilities continue to evolve with innovations like a multinational technology company's Blackwell architecture, these optimization strategies will become increasingly essential for organizations seeking to maximize the performance and cost-efficiency of their AI infrastructure investments. By adopting this comprehensive optimization framework, enterprises can fully harness the transformative potential of GPU acceleration for their most demanding AI workloads while maintaining operational efficiency and cost control.

References

[1] Channing Lovett, "The Impact of GPU Cloud Computing on Modern Workloads," tierpoint, 2024, [Online]. Available: https://www.tierpoint.com/blog/gpu-cloud-computing-for-modern-workloads/

[2] Ravi Chandra Thota, "Optimizing Kubernetes workloads with AI-driven performance tuning in AWS EKS," International Journal of Science and Research Archive, 2023, [Online]. Available: https://ijsra.net/sites/default/files/IJSRA-2023-0546.pdf

[3] Yihui Ren et al., "Performance Analysis of Deep Learning Workloads on Leading-edge Systems," sc19.supercomputing.org, [Online]. Available: https://sc19.supercomputing.org/proceedings/workshops/workshop_files/ws_pmbsf112s2-file1.pdf

[4] Together AI, "Together AI Powers Pioneers at GTC: NVIDIA Blackwell GPUs, Instant GPU Clusters, and A Full-Stack for AI Innovation," together.ai, Mar. 2025, [Online]. Available:

[5] https://www.together.ai/blog/together-ai-powers-pioneers-at-nvidia-gtc-2025

[6] Credence Research, "Cloud GPU Market By Type (Virtual Machines (VMs), Physical Servers); By Deployment Model (Public Cloud, Private Cloud, Hybrid Cloud); By End-user Industry (Gaming, Media and Entertainment, Machine Learning and AI, Healthcare, Automotive, Finance, Others); By Region – Growth, Share, Opportunities & Competitive Analysis, 2024 – 2032", Credence Research, 2023. [Online]. Available: https://www.credenceresearch.com/report/cloud-gpu-market

[7] Myeongjae Jeon et al., "Multi-tenant GPU Clusters for Deep Learning Workloads: Analysis and Implications", microsoft.com, [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2018/05/gpu_sched_tr.pdf

[8] Tushar Prasanna Swaminathan et al., "Benchmarking Deep Learning Models on NVIDIA Jetson Nano for Real-Time Systems: An Empirical Investigation," arXiv, 2024, [Online]. Available: https://arxiv.org/html/2406.17749v1

[9] Athul Ramkumar, "Machine Learning Models In Production: A Systematic Framework For Scalable And Robust Deployment," International Journal of Research In Computer Applications and Information Technology, 2024, [Online]. Available: https://iaeme.com/MasterAdmin/Journal_uploads/IJRCAIT/VOLUME_7_ISSUE_2/IJRCAIT_07_02_125.pdf

[10] restack, "Model Optimization Techniques in Deep Learning," restack, Mar. 2025, [Online]. Available: https://www.restack.io/p/model-optimization-answer-deep-learning-cat-ai

[11] Markus Rognlien et al., "Hardware-Aware Optimizations for Deep Learning Inference on Edge Devices", doc.ic.ac.uk, 2022, [Online]. Available: https://www.doc.ic.ac.uk/~wl/papers/22/arc22mr.pdf