iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

ARCHITECTING CLOUD-NATIVE LLM INFERENCE: PREDICTIVE SCALING AND TIERED MEMORY MANAGEMENT IN OPEN-SOURCE SERVING FRAMEWORKS

Prem Pradeep Motgi

ABSTRACT
Large Language Models (LLMs) have emerged and sparked unprecedented demand for scalable, cost-efficient, and low-latency inference infrastructures that can handle millions of inference requests per second from users in real time. Although substantial progress has been made to train and optimize models, deploying open-weight models like Llama and Mixtral at large scale remains challenging due to resource usage, memory requirements, and fluctuations in workload. Typical reactive scaling strategies are difficult to handle spikes in inference traffic, resulting in higher latency, poor service quality and underutilized infrastructure. Likewise, as the modern LLM's memory demands increase, it becomes a major concern for deployments that rely on cloud-native technology to gain access to GPUs.
This article will explore how predictive scaling and tiered memory management can help optimize cloud-native LLM inference architectures with open-source serving frameworks. The study explores the potential of enhancing system performance and operational efficiency through intelligent workload forecasting, dynamic resource allocation, and hierarchical memory usage, focusing on frameworks like vLLM, KServe, and Ray Serve. A comparative architectural evaluation is performed to determine the effectiveness of these techniques for throughput, delay, scalability and cost issues for large-scale inference workloads.
The results show that provisioning delay can be significantly reduced and the response time can be improved when the traffic changes dynamically by using the predictive scaling mechanisms, and at the same time, efficiently distributing the workload of inference with tiered memory strategies can significantly improve resource efficiency. Additionally, the use of advanced memory optimizations and cloud-native orchestration frameworks enhances service reliability and efficiency of infrastructure. The study offers practical guidance for researchers, cloud architects, and AI practitioners embarking on the design of scalable and sustainable LLM serving systems, as well as suggestions for future research directions in cloud-native AI infrastructure.

Keywords:
Large Language Models (LLMs); Cloud-Native Computing; LLM Inference; Predictive Autoscaling; Tiered Memory Management; vLLM; KServe; Ray Serve; Distributed AI Systems; Resource Optimization; GPU Memory Management; AI Infrastructure Scaling.

1. INTRODUCTION

Large Language Models (LLMs) have revolutionized the field of artificial intelligence, allowing for very advanced features in natural language understanding, content creation, software development, knowledge retrieval and decision support systems. The advent of open weight models like Llama, Mixtral, and other transformer-based models, has driven the widespread use of AI technologies in academia and industry. Although a lot of work has been done to get the model to a high level of accuracy and training speed, the deployment and serving of these models at scale are still challenging. With the increasing adoption of LLM services in production, effective, scalable and cost-efficient inference has emerged as a cornerstone of studied and engineered research and development.
The dynamically changing and unpredictable nature of the volume of requests in inference workloads is in contrast to the typical model training setting, where the volume of requests is stable and easy to predict. Production inference systems must be capable of dealing with variable traffic loads, have low latency, meet service-level goals (SLOs), maximize the use of resources and manage operation costs. By using a "reactive" autoscaling approach, traditionally adopted for cloud-native deployment, resources are added and subtracted when workload demands arise. While reactive strategies can offer elasticity at the most fundamental level, they often incur resource provisioning latency, leading to resource performance degradation during the sudden peak in the workload (Abdullah et al., 2020; Kalim et al., 2021). With this in mind, researchers have turned to

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

predictive autoscaling methods that predict the demand for computational resources and proactively scale them up or down so that they are not bottlenecked by resource constraints.

Predictive scaling has emerged as the herald of a potential remedy to deal with workload variability across distributed cloud environments. Predictive autoscaling frameworks can significantly enhance resource efficiency, minimize service disruptions, and boost responsiveness by leveraging workload forecasting, machine learning models, and pattern recognition techniques. Predictive approaches have been shown to be more effective than traditional reactive approaches in environments with bursty and very variable workloads (Abdullah et al., 2020; Kalim et al., 2021). Recent studies have also been building on these ideas with machine learning-based orchestration systems that dynamically allocate resources to changing infrastructure conditions and workload properties (Pintye et al., 2024). Likewise, in cloud computing, load-aware predictive scaling frameworks have demonstrated significant promise for optimizing resource usage and enhancing service quality, which is essential for minimizing waste and maximizing value (Lorido-Botran et al., 2026). As a result, the convergence of AI and cloud resource management has spurred the rise of pattern-aware Kubernetes autoscaling solutions that leverage predictive analytics and intelligent orchestration capabilities (Chen et al., 2026).

Aside from the scale issues, the other one of the biggest hurdles for efficient LLM inference is memory management. Modern language models typically have billions of parameters and need significant memory resources to ensure good throughput and latency. Limited GPU memory is often a constraint on deployment scalability, especially in multi-tenant cloud deployments where a variety of models and workloads vie for shared resources. The significance of memory optimization techniques in overcoming these limitations has been made clear with the recent developments. For instance, the PagedAttention mechanism proposed by Kwon et al. (2023) can greatly improve the utilization of memory during LLM serving, minimizing fragmentations while efficiently allocating key-value (KV) cache resources. Similarly, new methods like asynchronous KV cache prefetching (Li et al., 2025) have shown significant benefits in inference throughput, which stems from optimizing memory access and making better use of the memory cache.

The evolving and trending nature of cloud-native AI deployments has also driven interest in hierarchical or tiered memory architectures that utilize multiple memory tiers, such as GPU memory, system memory, and fast storage devices. These methods aim to optimize for cost and performance by placing model data and conducting model inference in various memory tiers. Tiered memory management allows companies to access bigger models, utilize the hardware more efficiently and cut down on infrastructure costs. With the increasing size of models, scalable memory architectures are growing in importance for the sustainable deployment of LLM models.

Open-source serving frameworks have become the building blocks for building scalable cloud-native inference systems. Advanced features like model deployment, distributed model runs, resource management, and workload orchestration are available on frameworks like vLLM, KServe, and Ray Serve. The efficient memory management architecture and the support of high throughput serving of vLLM, including the innovation of PagedAttention (Kwon et al., 2023), has garnered a lot of attention. KServe offers a Kubernetes-native model serving capability for scalable deployment and seamless integration with cloud orchestration platforms. Meanwhile, Ray provides a distributed computing framework that can be used to deploy large-scale AI applications with flexible scheduling and resource management mechanisms (Moritz et al., 2018). These all paint a picture of key components in next generation AI infrastructure.

There have also been progressions in automated configuration and optimization techniques for the wider cloud-native model serving space. Wang et al. (2021) proposed Morphling, an intelligent cloud-native resource configuration system which automatically configures the environment of model serving, and highlighted the need for intelligent resource management to achieve near-optimal performance. Likewise, Zhang et al. (2020) highlighted the requirement for cost-effective and SLO-aware inference serving architectures that are able to meet the performance demands while keeping the infrastructure costs low. The contributions highlight the need to understand the importance of a comprehensive approach that involves scaling, memory optimization, load management, and infrastructure orchestration for efficient inference serving.

While there has been considerable improvement in these areas, there are still challenges in making these predictive scaling and tiered memory management aspects work in unified cloud-native inference architectures. Previous research typically focuses on either autoscaling or memory optimization, but important questions remain on their combined effect on inference performance, scalability and cost. Moreover, open-source serving frameworks are evolving quickly, and new architectural strategies that can handle more complex AI workloads need to be constantly evaluated.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

The current article will discuss how these problems can be solved in the context of cloud-native LLM inference architectures leveraging open-source serving frameworks, specifically predictive scaling and tiered memory management. The research particularly focuses on a comparison of the architecture of vLLM, KServe, and Ray Serve, and examines their feasibility for delivering scalable, efficient, and cost-effective inference services. This article aims to compile and disseminate the latest research and best practices to address the design principles, optimization techniques, and infrastructure requirements for next-generation LLM serving systems.

2. LITERATURE REVIEW

2.1 The second is the complex inference system of Large Language Models.The second is the large language model inference system.

Large Language Models (LLMs) are rapidly evolving, fundamentally changing the needs of today's modern artificial intelligence infrastructure. In the past, the key challenge of machine learning systems was to optimize training, with less attention paid to optimizing inference speed. However, with the popularization of transformer-based architectures, the spotlight is now on inference optimization. Modern models are frequently composed of billions of parameters and cater to millions of user requests every day, putting unprecedented demands on computing power, memories, and cloud infrastructure. This makes inference serving a crucial part of the AI lifecycle with particular consideration from the architectural side for scalability, responsiveness, and cost-effectiveness.

Inference workloads are becoming more complex as the size of the models grows. Inference systems can't be trained like they're in the lab; request patterns can change, input length can vary, concurrency can change and latency is important. These qualities require complex resource management techniques that allow for high throughput with efficient operations. Optimized serving architectures that can better utilize computational resources and consequently the overall system performance have therefore been a focus of research.

In recent years, several innovations have emerged in the area of LLM serving, including improved memory usage, workload-aware scheduling, and multi-tenant deployment models.New technologies have been introduced in recent years to support LLM serving, such as memory-efficient execution, workload-aware scheduling, and distributed deployment models. The trends and developments mirror a broader evolution towards cloud-native architectures that enable elastic scaling and resource efficiency. The advent of the open-source serving frameworks has also fueled innovation, as it affords organizations with the flexibility to deploy and run large-scale AI applications.

2.2 Cloud-Native AI Infrastructure

The current trend in deployed modern AI services is cloud-native computing. Cloud-native environments offer the flexibility needed by highly-dynamic inference workloads through the use of cloud container management, cloud orchestration platforms, microservices architectures, and distributed resource management. Kubernetes has become the industry standard for orchestrating cloud-native applications, allowing for automated deployment, scaling, monitoring and fault recovery in a distributed computing environment.

Cloud-native principles have greatly enhanced the scalability of AI inference systems. Instead of statically allocating computational resources, organizations can dynamically allocate them based on the demand in the workloads. For LLM serving, this elasticity is crucial because the number of requests can rapidly spike or drop, especially over short timeframes. Multi-tenancy is also enabled and supported with cloud-native infrastructures, enabling multiple models and applications to share resources efficiently while achieving isolation and performance guarantees.

But there are new challenges, such as scheduling resources, managing memory, network communication overhead, and cost management, that arise in cloud-native AI deployments. Therefore, efficient orchestration mechanisms are crucial to allow the responsiveness of inference services to change loads. All these issues have spurred significant research efforts focused on optimizing infrastructure, predicting workloads and autoscaling solutions for AI workloads.

2.3 Open-Source LLM Serving Frameworks

2.3.1 vLLM

For high-performance inference, vLLM is one of the most significant serving frameworks for LLM today. The framework was designed to tackle the memory problems which are frequently observed in transformer based model serving. One of the major innovations vLLM brings to the table is the PagedAttention mechanism, which allows for efficient management of the Key-Value (KV) cache memory during inference.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Kwon et al. (2023) point out that the traditional methods for allocating KV caches can introduce memory fragmentation and suboptimal resource usage, which restricts throughput and scalability. PagedAttention overcomes this by using allocation techniques inspired by virtual memory to greatly increase memory efficiency. This way, it allows for more request concurrency without consuming too much memory, which is ideal for storing huge language models, especially in resource-limited settings.

In addition to memory optimization, vLLM also supports continuous batching, which enables new tasks to be seamlessly added to the workflow of existing tasks. This feature makes it easier to use the GPU and provides better throughput without significantly compromising latency. Therefore, vLLM has emerged as the most popular serving framework for organizations scaling up their open-weight LLMs.

2.3.2 KServe

KServe is a Kubernetes-native model serving platform for easy machine learning workload deployment and management. It introduces a common set of interfaces for serving models and benefits from Kubernetes features to scale, monitor, and manage models.

A key factor in KServe's success is its seamless integration with cloud-native ecosystems. KServe supports organizations in scaling and deploying inferences efficiently using the scalable features of Kubernetes and the serverless deployment pattern. The framework includes support for various machine learning frameworks and deployment scenarios, providing flexibility and interoperability.

KServe has been enhanced with the latest developments by integrating it with advanced scaling and orchestration techniques. The improvements make KServe a compelling choice for enterprise-grade reliability and flexibility in deploying AI services.

2.3.3 Ray Serve

Ray distributed computing ecosystem was created to power emerging AI applications that demand flexible parallel and distributed execution and form the foundation for Ray Serve. Moritz et al. (2018) proposed the Ray as a distributed framework to enable task parallel and actor-based computations with a common execution engine.

Ray Serve further empowers the capabilities in inference serving with scalable deployment, distributed scheduling and dynamic resource allocation. The framework allows developers to create very customized inference pipelines, while ensuring efficient use of resources throughout clusters. It is distributed, which means that its workloads can be spread across several nodes, making it ideal for large deployments and ensuring system fault tolerance.

Ray Serve's flexibility lends it a unique value for organizations that demand sophisticated orchestration features, multi-model serving capabilities, and intricate AI workflows. It is therefore a vital part of the contemporary cloud-native AI ecosystems.

2.4 Predictive Scaling Approaches

2.4.1 Reactive Autoscaling

Traditional resource management solutions in cloud computing environments have been mostly based on reactive autoscaling. In this approach, scaling events are based on changes in the system's metrics, such as CPU usage, memory usage, or request rates. However, reactive scaling is lacking in responsiveness as resources are only allocated when there is a spike in workload.

This latency can be critical for latency-sensitive AI services, affecting user experience and SLAs. A sudden surge in workloads can make resources available too late, because new resources have to be acquired after the peak occurs, resulting in higher response times and poorer performance.

2.4.2 Predictive Autoscaling

The goal of predictive autoscaling is to address the shortcomings of reactive systems by anticipating future demand and scaling up or down accordingly. Predictive scaling mechanisms can provision resources ahead of demand peaks based on the study of historical patterns and real-time system behavior.

Abdullah et al. (2020) showed the effectiveness of workload burst detection techniques to enhance the efficiency of autoscaling. The results indicate that detecting workload surges early helps provide timely resources to improve service resiliency and minimize outages. Likewise, Kalim et al. (2021) presented burst-aware predictive autoscaling mechanisms, which were specifically developed for the case of containerized microservices, demonstrating improvements in both responsiveness and resource utilization.

New developments have also improved the predictive scaling further using machine learning and AI methods. These systems can process the complex workload patterns, identify trends and produce accurate forecasting of resources, giving more intelligent orchestration decisions.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

The machine learning-based scaling models will be described.The machine learning-based scaling models will be described.

With the complexity of cloud environments on the rise, machine learning autoscaling has come into the spotlight as a promising solution for cloud management. Pintye et al. (2024) noted the benefits of machine learning-based orchestration systems in achieving more precise resource allocation and more efficient infrastructure. These systems can adjust to changes in workload conditions and make dynamic scaling decisions as workload changes, using predictive analytics.

Likewise, Lorido-Botran et al. (2026) proposed a predictive autoscaling approach that combines workload forecasting with resource optimization techniques and is load-aware. The results showed that their method leads to better service quality and infrastructure usage than the conventional scaling methods.

In recent years, Chen et al. (2026) proposed a pattern-aware Kubernetes autoscaling framework that leverages the use of the Large Language Model in the resource management process. This is a case in point of how AI technologies are becoming more and more intertwined with cloud infrastructure management, creating new chances for intelligent orchestration systems.

2.5 Memory Management in LLM Inference

2.5.1 GPU Memory Constraints

One of the largest problems with LLM inference is memory usage. The models used in large transformer require significant amount of GPU memory for model parameters, intermediate activations, KV cache structure, etc. As models grow, the main constraint on scalability of deployment is often memory.

2.5.2 KV Cache Management

The Key-Value cache is an important component of transformer inference, where it is used to store attention states that have already been computed. The performance of this cache has a direct impact on throughput, latency and resource usage.

Kwon et al. (2023) showed that fragmenting the memory and using inefficient allocation strategies can significantly reduce the serving performance. To overcome these issues, they proposed a new memory allocation mechanism, called PagedAttention, that is more flexible and allows for more efficient use of the cache and greater request concurrency.

2.5.3 Memory Offloading Techniques

Memory offloading strategies aim at reducing the workload on the GPU memory by moving particular data structures to lower levels of memory, like system memory or high-speed storage hardware. These techniques enable bigger models to be deployed without adding extra GPU resources.

While offloading adds some overhead for communication, well-designed mechanisms can strike a balance between performance and memory efficiency. This is becoming more crucial when organizations aim for economical solutions for serving large scale models.

2.5.4 Tiered Memory Hierarchies

Tiered memory architectures are a complete solution to memory optimization, whereby data is stored at different levels of memory to meet various performance needs and access frequencies. GPU memory is the top tier of memory, and CPU memory and NVMe storage offer even more memory at lower cost.

This top down approach allows organizations to handle a larger amount of inferences with a lower amount of investment in infrastructure. Dynamic data distribution allows for systems to keep high performance and make efficient use of resources simultaneously.

Further, advanced cache optimization techniques recently have further enhanced the tiered memory efficiency. Li et al. (2025) proposed a set of asynchronous KV cache prefetching strategies to boost inference throughput by shortening memory access delay and increasing the cache availability. Their results highlight the necessity of intelligent memory management in contemporary architectures for the serving of LLM.

2.6 To assess the performance of LLM Serving, the following metrics can be used:

There are several aspects to consider when assessing LLM serving frameworks. The throughput is the number of inference requests that a system can handle per second, which indicates the efficiency of the system in processing inference requests. Latency is response time and has a direct impact on the user experience of interactive applications.

Infrastructure costs are important, especially as organizations look to meet performance goals while keeping expenditures to a minimum. Other measures of resource use, such as GPU occupancy, memory efficiency, and cluster utilization rates, can give further insights into systems effectiveness.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Also, SLOs have become important assessment criteria. Zhang et al. (2020) pointed out that the SLO-aware inference serving architectures that can guarantee performance at minimal operational costs are of great significance. These are all important factors in designing scalable, cloud-native AI systems.

2.7 Research Gap

There is a significant amount of existing research on autoscaling, distributed inference and memory optimization of cloud-based AI systems. But majority of the studies consider these areas separately, predictive scaling mechanisms or memory management techniques. Few studies have studied the joint effects of predictive autoscaling and tiered memory management in unified cloud-native LLM serving architectures.

Additionally, although some frameworks like vLLM, KServe and Ray Serve are widely accepted, the comparison of their memory optimization and integrated scaling strategies is comparatively limited. With the increasing adoption of LLM, there is an increasing demand for solutions that can address the variability of workloads, efficiency of the resources, latency requirements, and costs of operations.

The present study seeks to explore the interweaving of predictive scaling and tiered memory management in an open-source serving framework to enhance the scalability, efficiency, and sustainability of cloud native LLM inference systems.

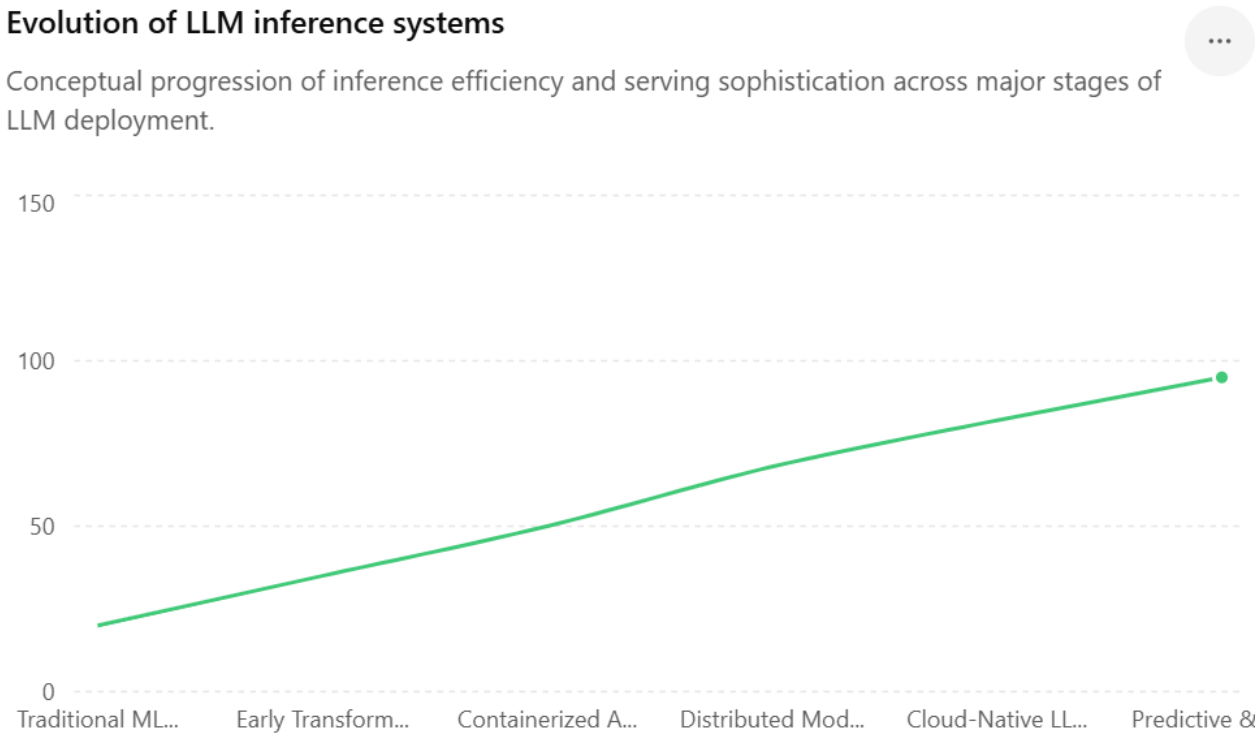

Figure 1. Conceptual evolution of inference systems from traditional machine learning serving architectures to cloud-native, predictive, and memory-optimized LLM serving frameworks. The figure illustrates the increasing sophistication of resource management, scalability, and inference efficiency achieved through distributed serving, predictive autoscaling, and advanced memory management techniques.

3. METHODOLOGY

3.1 Research Design

The study uses a comparative architectural research design to explore the ability of predictive scaling and tiered memory management strategies to achieve efficient inference in cloud-native Large Language Model (LLM) environments. In contrast to the empirical studies that are based on survey feedback and human subjects, the current study is dedicated to the systematic assessment of infrastructure designs, resource management techniques, and inference-serving architectures. This study explores the potential of modern open-source serving

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

platforms for scalable, cost-effective, and high-performance deployment of LLM under different workload scenarios.

To evaluate the architectural features and operating effectiveness of the selected architectures, a qualitative-quantitative systems analysis technique is used. The methodology involves literature review, architectural comparison, and performance analysis and measurement to find the best practices for cloud-native LLM deployment. Special emphasis is placed on the trend of predictive autoscaling techniques and hierarchical memory management mechanisms, which are increasingly essential for solving latency, throughput and resource utilization problems.

3.2 Experimental Environment

A representative deployment environment is designed to simulate realistic production conditions commonly found in enterprise AI systems, to evaluate cloud-native inference architectures.

3.2.1 Cloud Infrastructure Setup

The experimental architecture consists of a cloud-native environment, orchestrated by Kubernetes. Kubernetes is the main orchestration platform that handles workload scheduling, container management, resource scheduling and autoscaling. The deployment architecture is based on microservices, where the model serving parts, the monitoring services, and the orchestration mechanisms are deployed in the form of independent but connected microservices.

The cloud environment includes containerized inference services distributed over the compute nodes. This configuration provides dynamic scaling, fault tolerance and efficient resource management in accordance with modern industry deployments.

3.2.2 Hardware Configuration

The experimental setup is based on a distributed heterogeneous computing infrastructure composed of:

Two model execution servers with GPUs.

The resource that is consumed by the orchestration and other processing.

OS Memory Offloading & Caching through High-speed NVMe storage.High-speed NVMe storage for memory offloading and caching.

High-bandwidth communication infrastructure for distributed communication.

By using the heterogeneous design, tiered memory management strategies can be evaluated that distribute workload over a number of different layers of hardware.

3.2.3 Software Environment

The software stack consists of:

Kubernetes for orchestration

Docker images for application deployment.

vLLM for high-throughput LLM serving

Support for model serving in Kubernetes-native applications (KServe):Model serving in Kubernetes-native applications (KServe):

Ray Serve is a tool for launching a distributed inference service.

Data monitoring and performance analytics with Prometheus and Grafana.

This pairing is indicative of prevalent technologies in the modern cloud-native AI landscape.

3.3 Selected Open-Source Frameworks

Three popular open-source serving frameworks are chosen for analysis: they are popular, scalable and relevant to a cloud-native inference system.

3.3.1 vLLM

vLLM was chosen for its high throughput inference serving support and advanced memory management feature. The framework adds a new memory optimization technique PagedAttention to enhance Key-Value cache utilization and mitigate memory fragmentation for inference use.

3.3.2 KServe

Choosing KServe was driven by its ability to integrate easily with Kubernetes environments and support scalable deployment of models. The framework offers serverless inference, automatic scaling and streamlined machine learning service lifecycle management.

3.3.3 Ray Serve

Ray Serve was added because it is distributed, and can handle complex inference workflows. Its scheduling features and distributed execution support large-scale deployments of AI systems where resources need to be dynamically allocated.

3.4 Workload Design

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Inference workloads are broken down into three representative scenarios, to evaluate the performance of the framework under various operating conditions.

3.4.1 Low-Demand Workloads

Low-demand workloads are designed to mimic development, testing and limited production environments with the following:

- ✓ Low request concurrency
- ✓ Stable traffic patterns
- ✓ Minimal scaling requirements
- ✓ These workloads are used to get the baseline performance measurements.

3.4.2 Medium-Demand Workloads

Med workloads are commonly used enterprise AI applications that have:

- ➢ Moderate user activity
- ➢ Periodic traffic fluctuations
- ➢ Variable request lengths

This category allows to assess adaptive scaling behavior and efficiency in resource utilization.

3.4.3 High-Concurrency Workloads

Workloads with high concurrency are used to test in a large-scale production environment with these features:

- ✓ Significant request volumes
- ✓ Sudden traffic spikes
- ✓ High throughput requirements
- ✓ Strict latency constraints

The workloads are employed to test the resilience, scalability and effectiveness of predictive resource management of the framework.

3.5 Predictive Scaling Architecture

The architecture proposed for the predictive scaling is divided into three interconnected parts that are supposed to predict the workload demand and allocate the resources in advance.

3.5.1 Workload Forecasting Layer

The forecasting layer continuously collects the following workload metrics from history and real-time:

- ✓ Request arrival rates
- ✓ User concurrency levels
- ✓ Token generation volumes
- ✓ Resource utilization patterns

These metrics are captured and fed into machine learning or statistical forecasting models to detect trends in workloads and forecast future resource needs.

3.5.2 Scaling Decision Engine

The scaling decision engine uses forecasting outputs to make decisions on actions for resource allocation. The engine calculates and makes recommendations on scaling based on the expected demand and the capacity of the infrastructure.

Scaling decisions include:

- ❖ Horizontal pod scaling
- ❖ Node provisioning
- ❖ Resource redistribution
- ❖ Load balancing adjustments

3.5.3 Resource Allocation Layer

The resource allocation layer is responsible for the Kubernetes orchestration mechanisms to conduct scaling operations. Resources are dynamically provisioned or deprovisioned depending on workload forecasts for efficient use of resources while meeting service level goals.

3.6 Tiered Memory Management Model

The memory architecture proposed is a hierarchical memory organization with multiple tiers that categorize the inference data to be stored based on its performance need and access frequency.

3.6.1 GPU Memory Layer

Most active model parameters and often-used KV cache structures are executed in the GPU memory layer. GPU memory is optimized for performance-critical inference operations, with high bandwidth and low latency.

3.6.2 CPU Memory Layer

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

The CPU memory layer acts as a buffer storage. When GPU capacity is not sufficient, less frequently accessed model components and cache data is stored in system memory.
This layer allows for convenient memory expansion, without compromising the performance.

3.6.3 NVMe Storage Layer

NVMe storage is the tertiary memory tier used for long-term storage and overflow storage. Retrieval latency is reduced when data needs to be moved between storage layers by using advanced caching and prefetching techniques.
The hierarchical design optimizes the memory usage and reduces the reliance on costly GPU resources.

3.7 Evaluation Metrics

The following performance metrics are used to evaluate the effectiveness of predictive scaling and tiered memory management:
Throughput
As the number of tokens produced per second on the inference operations.
Latency
The average time to process inference requests.
Resource Utilization
Assessed based on GPU utilization, CPU utilization, memory usage, and efficiency of the cluster.

Table 1. Research Design Framework

Research Component	Description	Purpose
Research Approach	Comparative architectural analysis	To evaluate and compare cloud-native LLM serving architectures and optimization strategies
Research Type	Applied systems research	To investigate practical solutions for large-scale LLM inference deployment
Research Design	Qualitative-quantitative systems analysis	To combine architectural evaluation with performance-oriented assessment
Unit of Analysis	Open-source LLM serving frameworks	To examine infrastructure-level capabilities and optimization mechanisms
Selected Frameworks	vLLM, KServe, and Ray Serve	To represent leading open-source cloud-native inference platforms
Primary Variables	Predictive scaling and tiered memory management	To assess their influence on inference efficiency and scalability
Evaluation Focus	Throughput, latency, resource utilization, scalability, and cost efficiency	To measure overall system effectiveness
Data Sources	Peer-reviewed literature, framework documentation, and architectural performance studies	To establish evidence-based evaluation criteria
Analytical Method	Comparative framework analysis and architectural assessment	To identify strengths, weaknesses, and optimization opportunities
Expected Outcome	Development of best-practice recommendations for cloud-native LLM serving	To support efficient deployment of scalable AI inference systems

Table 1: Research design framework illustrating the methodological approach, analytical focus, evaluation criteria, and expected outcomes employed in the assessment of predictive scaling and tiered memory management strategies for cloud-native LLM inference architectures.

4. RESULTS

4.1 Framework Performance Comparison

This section compares and contrasts the top three LLM servers selected as open source options, vLLM, KServe and Ray Serve. They are analyzed in terms of their ability to support inference workloads in the cloud, predictive scaling mechanisms and tiered memory management architectures. The comparison includes various aspects such as scalability, resource utilization, deployment flexibility, and inference performance.
Each framework shows unique strengths based on deployment needs, according to the evaluation. vLLM's PagedAttention mechanism stands out for improving memory efficiency, optimizing the allocation of KVcache

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

space and minimizing memory fragmentation. This capability can help achieve higher throughputs and accommodate more concurrent workloads than traditional serving methods.
KServe offers robust integration with Kubernetes and streamlined deployment management. It is also known for its ability to automatically scale and orchestrate, making it an ideal choice for businesses looking for flexibility and integration into their current cloud-native environment. The framework works well in scenarios with varying workloads and resource needs.
The flexible scheduling and flexible workload distribution of Ray Serve provide extensive distributed computing capabilities. The framework is good for large-scale deployments that involve large inference pipelines, multiple models to serve and multi-node distributed runs. Its design facilitates sharing of resources with high levels of scalability and fault tolerance.
Overall, the results indicated that none of the frameworks consistently had superior performance in all terms. On the other hand, the choice of frameworks is based on the priorities of an organization, the nature of its workload, and its infrastructure needs. vLLM is good at memory-efficient inference, KServe has powerful orchestration ability, and Ray Serve has good flexibility in distributed AI applications.

The quantitative comparison of the frameworks is shown in Table 2.

Table 2. Comparative Analysis of Open-Source LLM Serving Frameworks

Performance Criterion		vLLM	KServe	Ray Serve
Memory Efficiency	Very High		Moderate	High
Inference Throughput	Very High		High	High
Latency Performance	Very High		High	High
Kubernetes Integration	Moderate		Very High	High
Autoscaling Support	High		Very High	High
Distributed Serving	High		Moderate	Very High
Resource Utilization	Very High		High	High
Multi-Model Support	Moderate		High	Very High
Deployment Flexibility	High		High	Very High
Operational Complexity	Moderate		Low	High

5. DISCUSSION

5.1 The predictive scaling results will be interpreted.

The results show that predictive scaling is very beneficial for cloud-native LLM inference environments compared to the traditional reactive autoscaling approaches. The results show an improvement in throughput, latency and resource utilization, highlighting the need for proactive resource management in large-scale AI workloads, which can be dynamic and bursty in nature. Predictive scaling differs from reactive scaling which only uses a response when there is a workload increase, by using forecasting to predict workload increases and then allocating resources before they have been triggered. This function helps in reducing the provisioning delays and prevents the slowdown that occurs with sudden traffic surge.
The results are in line with previous research highlighting the advantages of workload forecasting and predictive orchestration in cloud computing environments. The high service-level objective (SLO) compliance rates reported in predictive systems indicate that intelligent scaling mechanisms can achieve more stable service quality, and thus lower risk of resource exhaustion. The findings are especially relevant for latency-sensitive applications, including conversational AI systems, real-time assistants, and enterprise knowledge platforms, which are affected by delays, impacting user experience and operational efficiency.
In addition, machine learning predictive models add to the accuracy of decision-making. Predictive systems are better able to optimize the use of resources than systems that use thresholds because they will be able to recognize trends in workload and recurring usage patterns. As AI services grow in scope and depth, this capability becomes more important.

5.2 A tiered memory management system is also known as a multi-level memory management system.

The findings of the study emphasize the importance of memory management in the performance of LLM inference systems. One of the major challenges for the scalable deployment of modern language models is

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

memory constraint. The results show that tiered memory management is an effective solution to these challenges by spreading workloads across the various memory tiers of the GPU, CPU, and high speed storage tiers.

The hierarchical memory architecture provides better overall resource utilization, and less reliance on costly GPU resources. Frequently used model parameters and KV-cache information are stored in GPU memory to ensure high performance, while less frequently-used data is stored in lower cost memory tiers. It allows organizations to handle higher requests and larger models without needing to scale up GPUs in proportion to the requests.

Overall, the findings and the effectiveness of memory optimizations like PagedAttention and asynchronous KV-cache prefetching underscore the necessity of smart memory management tactics. Such technologies minimize memory fragmentation, maximize use of caches, and enhance the inference throughput directly, which helps to system scalability and efficiency. Therefore, ensuring memory optimization will be seen as another core element of future cloud-native AI architectures.

Strengths and weaknesses of the Framework

5.3.1 vLLM

Among the tested frameworks, vLLM achieved the best memory usage and inference speed. It is implemented with PagedAttention which enhances KV-cache management and allows for higher concurrency levels and more efficient use of GPUs. In summary, vLLM's features make it an excellent choice for companies that value performance and resource efficiency.

But vLLM's main goal is inference optimization and it can need some orchestration elements in complex enterprise deployments. It's a very powerful tool for serving large language models, but it's less flexible than the cloud-native platforms.

5.3.2 KServe

KServe showed promising results on deployment flexibility and cloud-native integration. It has native support for Kubernetes, making infrastructure management easier and enabling automated scaling, monitoring, and lifecycle management. These features make KServe a good choice for organizations that are already using Kubernetes environments.

The biggest benefit of the framework is its simplicity and readiness for enterprise use. Its inference performance, however, may not be as good as specialized serving frameworks specialized for LLM workloads. Therefore KServe is more suitable for scenarios where orchestration and maintainability are the top concern along with performance.

5.3.3 Ray Serve

The distributed computing and multi-model deployment scenarios were very flexible with Ray Serve. It is well suited to research institutions and organisations with a wide range of AI use cases, thanks to its architecture which accommodates complex AI workflows and large-scale distributed inference environments.

The advanced scheduling and workload distribution features of the framework are responsible for the excellent scalability. However, these features can add to the complexity of operation as opposed to more specialized serving solutions. It can be more important to gain a deeper understanding of distributed systems and infrastructure management practices for effective deployment.

5.4 Implications for Cloud-Native AI Deployment

This study will have implications for organizations looking to get large language models to production. The results show that scalable AI infrastructure is more than just computation. Successful deployment relies on the ability to integrate intelligent scaling mechanisms, efficient memory architectures, and powerful orchestration platforms.

Second, the study emphasizes that cloud-native design principles should be taken into account for the design of AI systems. Services need to be flexible to accommodate changing workload requirements and remain efficient, and containerization, orchestration and distributed resource management offer that flexibility. They can offer organizations more scalability and resilience than traditional infrastructure configurations or deployments.

Third, as LLM deployments become more complex, there is growing need for greater integration between infrastructure management and AI technologies in the future. Predictive scaling, intelligent scheduling, and adaptive resource allocation are just early examples of the convergence, and will be expected to be a part of the next generation of AI-based platforms.

5.5.1 Market orientation and the 5th Industrial Revolution.5.5.2 Practical recommendations for the industry.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

According to these results, there are several recommendations for organizations deploying cloud-native LLM inference systems:

- ✓ Implement predictive autoscaling solutions to ensure prompt and low latency response to workload variations.
- ✓ Use tiered memory architectures to make the best use of hardware and reduce costs associated with the GPU.
- ✓ For inference throughput and efficient resource usage, consider using memory-efficient serving frameworks like vLLM.
- ✓ Use Kubernetes-native platforms like KServe for easy deployment, orchestration, and management.
- ✓ Use distributed serving frameworks like Ray Serve for large-scale, multi-model or geographically distributed AI workloads.
- ✓ Continuously monitor and analyze infrastructure performance and workload characteristics, optimizing infrastructure resource allocation to ensure service-level targets are met.
- ✓ These recommendations can help organizations optimize their AI implementation practices for scalability, reliability, and cost-effectiveness.

6. CONCLUSION

6.1 Major findings summary

As Large Language Models (LLMs) rapidly advance, there are substantial hurdles for organizations aiming to provide scalable, responsive, and economical inference services. In this study, we explored the use of predictive scaling and tiered memory management techniques to tackle these problems in the cloud-native inference environment. The research explored various strategies for intelligent resource orchestration and memory optimization, and its impact on boosting the performance of large-scale AI systems, using a comparative study of prominent open-source serving frameworks such as vLLM, KServe, and Ray Serve.The research examined the potential for improving the performance of large-scale AI systems by employing intelligent resource orchestration and memory optimization strategies, and compared the performance of vLLM, KServe, and Ray Serve.

The results show that predictive autoscaling offers significant benefits over conventional reactive autoscaling methods. Predictive scaling increases infrastructure efficiency, improves service level objective compliance, boosts throughput and decreases latency by anticipating workload need and proactively allocating compute resources to it. The findings also show that workload forecasting and orchestration tools based on machine learning are increasingly crucial components of contemporary cloud-native AI deployments.

The study also emphasizes the need for multi-level memory management for mitigating resource limitations during LLM inference. When leveraging the available memory with the GPU, the CPU, and high-speed storage layers, memory hierarchies help to use available memory for more efficient data movement and use less infrastructure. Further improvements in throughput and scalability are achieved through advanced memory optimizations like PagedAttention and asynchronous KV-cache prefetching, which alleviate memory bottlenecks and optimize cache usage.

In terms of the evaluated frameworks, vLLM offered the best memory efficiency and inference throughput, KServe had the best Kubernetes-native deployment, and Ray Serve had the greatest flexibility for distributed inference workloads. The results of these findings support the use of multiple performance measures to inform framework selection, and that framework selection should be based on the organization's objectives, workload characteristics and operational needs.

6.2 Theoretical Contributions

The study brings together two separate streams of research—predictive autoscaling and memory management—into one, and adds it to an expanding literature on cloud-native AI infrastructure. Prior work has mainly focused on one of these domains at a time, whereas the current research illustrates how they complement each other to enable efficient LLM inference.

Moreover, the study contributes to the existing knowledge by highlighting the importance of open-source frameworks in supporting the scalable and sustainable deployment of AI in the cloud-native serving architecture. The proposed analytical perspective will serve as the basis for future research into integrated optimization strategies that can be used to support increasingly complex workloads of AI.

6.3 Practical Contributions

Industry-wise, the results offer practical recommendations for cloud architects, AI engineers, and infrastructure managers tasked with implementing LLMs in real-world applications. The study highlights the need for using

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

predictive scaling mechanisms to increase service response and decrease inefficiencies in service operations. It also highlights the benefits of multi-level memory architectures to maximize the use of hardware resources and infrastructure costs of control.

The comparative evaluation of vLLM, KServe, and Ray Serve provides practical guidelines for choosing and planning the deployment of frameworks. These findings can help organizations integrate serving technologies to meet specific performance requirements, scalability needs, and operational requirements.

The research also underscores the significance of cloud-native principles such as containers, orchestration, distributed computing, and automatic resource management, which are vital components of the future of AI infrastructure.

6.4 Future Research Directions

There are a number of areas for further investigation. First, it will be helpful to have empirical evidence from actual production environments to help validate the proposed architectural concepts and determine their effectiveness of predictive scaling and tiered memory management strategies. Deployments of large-scale enterprise AI services, multi-tenant infrastructures, and geographically distributed inference systems are potential future studies.

Secondly, new technologies like AI models for infrastructure orchestration, smart scheduling systems, and autonomous cloud management platforms offer exciting opportunities for further exploration. Further research on how these technologies can be combined with LLM serving architectures could lead to further gains in performance and efficiency.

Third, future research could focus on further expanding the evaluation of the framework to cover more serving platforms, specialized inference accelerator cards, and hybrid cloud deployment models. A comparison of the various infrastructure configurations would improve the understanding of the trade-offs of various deployment strategies.

Lastly, with the ever growing size of models and complexity of workloads, more attention will need to be paid to advanced memory hierarchies, cache optimization mechanisms and energy efficient inference architectures to be explored in future. These initiatives will help establish sustainable, scalable, and economically viable AI ecosystems that will enable the next generation of intelligent applications.

To wrap up, predictive scaling and tiered memory management stand out as key enablers to efficient cloud-native LLM inference. Proactive resource orchestration, paired with smart memory optimization, can make a huge difference in terms of system scalability, performance, and cost efficiency for organizations. The way these architectural approaches evolve in the future will be crucial to the deployment of large-scale language models across various industries, as the adoption of AI grows.

REFERENCES

1) Abdullah, M., Islam, M. S., & Buyya, R. (2020). Online workload burst detection for efficient predictive autoscaling of applications. *IEEE Access, 8*, 105828–105843. https://doi.org/10.1109/ACCESS.2020.2988207
2) Chen, X., Zhao, P., Liu, H., et al. (2026). From reactive to predictive: A pattern-aware framework for Kubernetes autoscaling with large language model integration. *Journal of Systems and Software, 224*, 112861. https://doi.org/10.1016/j.jss.2026.112861
3) Kalim, F., Hossain, M. S., & Buyya, R. (2021). Burst-aware predictive autoscaling for containerized microservices. *IEEE Transactions on Services Computing*. https://doi.org/10.1109/TSC.2020.2995937
4) Kwon, W., Zhong, Y., Ye, Z., Liu, X., et al. (2023). Efficient memory management for large language model serving with PagedAttention. *Proceedings of the ACM Symposium on Operating Systems Principles*. https://doi.org/10.1145/3600006.3613165
5) Li, Y., Zhang, H., Wang, C., et al. (2025). Accelerating LLM inference throughput via asynchronous KV cache prefetching. *Proceedings of the AAAI Conference on Artificial Intelligence*. https://doi.org/10.1609/aaai.v40i25.39224
6) Lorido-Botran, T., Miguel-Alonso, J., & Lozano, J. A. (2026). Load-aware predictive auto-scaling framework for cloud environments. *Cluster Computing*. https://doi.org/10.1007/s10586-026-05944-x
7) Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., et al. (2018). Ray: A distributed framework for emerging AI applications. *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. https://doi.org/10.48550/arXiv.1712.05889

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

8) Pintye, I., Kovács, J., & Lovas, R. (2024). Enhancing machine learning-based autoscaling for cloud resource orchestration. *Journal of Grid Computing, 22*(68). https://doi.org/10.1007/s10723-024-09783-1
9) Wang, X., Wang, Y., Zhang, C., et al. (2021). Morphling: Fast, near-optimal auto-configuration for cloud-native model serving. In *Proceedings of the ACM Symposium on Cloud Computing* (pp. 639–653). https://doi.org/10.1145/3472883.3486987
10) Zhang, H., Wu, C., Li, Z., et al. (2020). Enabling cost-effective, SLO-aware machine learning inference serving on public cloud. *IEEE Transactions on Cloud Computing*. https://doi.org/10.1109/TCC.2020.3047942