Design of a Multi-Tenant Real-Time Inference Framework Based on OpenStack and SR-IOV GPU Virtualization

Ma Rui*, Shi Bingfeng, Ma Xin and Wei Bin

Earthquake Agency of Xinjiang Uygur Autonomous Region, Urumqi 830011, Xinjiang, China
E-mail: xjdzjmarui@sina.com; 793999148@qq.com; 2896994830@qq.com; 3394848930@qq.com
**Corresponding Author*

Received 10 January 2026; Accepted 21 April 2026

Abstract

The deployment of real-time artificial intelligence inference services on shared cloud infrastructure poses significant challenges due to resource contention, latency variability, and tail-latency amplification. While cloud platforms offer scalability and flexibility, conventional accelerator sharing mechanisms often fail to provide the determinism required by latency-sensitive inference workloads. This paper presents a standards-based, multi-tenant cloud inference framework that integrates OpenStack orchestration with Single Root I/O Virtualization (SR-IOV)-enabled graphics processing unit (GPU) partitioning to achieve predictable and isolated real-time inference execution. In the proposed architecture, each tenant is assigned an exclusive GPU virtual function, enabling hardware-level isolation while remaining fully compatible with native OpenStack scheduling and resource management mechanisms. A comprehensive experimental evaluation is conducted on a private OpenStack cloud to assess inference latency distribution, tail behavior, scalability, robustness to background network and control-plane activity, and throughput-latency trade-offs. Experimental results show

Journal of ICT Standardization, Vol. 14_3, 357–390.
doi: 10.13052/jicts2245-800X.1434
© 2026 River Publishers

that median inference latency remains stable across single-tenant and multi-tenant configurations, while P95 and P99 tail latencies exhibit no measurable amplification under concurrent execution. The system scales linearly with the number of available GPU virtual functions, maintaining consistent latency behavior until hardware capacity is reached. Additional experiments demonstrate that background network traffic and control-plane operations introduce negligible impact on inference latency. Throughput analysis reveals a well-defined saturation knee, enabling clear identification of safe operating regions for real-time inference services. By leveraging mature ICT standards and open-source cloud infrastructure, this work provides a reusable reference architecture for deploying latency-sensitive inference services in private and hybrid clouds. The results highlight the effectiveness of hardware-assisted accelerator isolation in balancing performance determinism, scalability, and operational simplicity, and offer practical guidance for future system design and standardization efforts.

Keywords: Real-time inference, Multi-tenant cloud computing, GPU virtualization, SR-IOV, OpenStack, Tail-latency, Hardware-assisted isolation, ICT standards..

1 Introduction

The deployment of artificial intelligence (AI) inference services has evolved rapidly from offline, throughput-oriented processing toward latency-critical and real-time execution. Applications such as industrial monitoring, intelligent sensing, interactive AI services, and online decision-making systems require not only accurate predictions but also predictable and bounded response times. In such scenarios, occasional latency spikes can be as damaging as sustained performance degradation, making tail-latency and execution determinism primary system-level concerns rather than secondary optimization targets [1, 2].

Cloud computing platforms provide elastic resource sharing and operational flexibility, but their multi-tenant nature fundamentally conflicts with the requirements of real-time inference. Recent studies show that modern inference workloads, particularly those accelerated by graphics processing units (GPUs) and other specialized hardware, are highly sensitive to shared execution paths, driver scheduling, and memory contention, leading to severe tail-latency amplification under concurrent load [3–5]. These effects persist even when average latency remains acceptable, highlighting the limitations

of throughput-driven sharing mechanisms for latency-sensitive inference services [6].

Early inference serving systems focused on scalability and average latency optimization through batching, request aggregation, and dynamic scheduling [7–9]. While effective for best-effort workloads, subsequent research demonstrates that these techniques significantly increase latency variance and undermine real-time guarantees in accelerator-rich environments [4, 10, 11]. The emergence of large language models (LLMs) has further intensified this challenge, as LLM inference introduces highly variable execution times and increased contention for accelerator resources [12, 13].

GPUs remain the dominant accelerators for cloud inference, yet predictable GPU sharing remains an open problem. Software-based multiplexing and time-slicing approaches improve utilization but provide limited guarantees against cross-tenant interference, particularly with respect to tail-latency [14–16]. Empirical evaluations and surveys consistently report that contention at GPU execution queues, memory hierarchies, and driver-level scheduling paths leads to non-deterministic inference performance under multi-tenant workloads [5, 17].

An emerging alternative is hardware-assisted accelerator virtualization, which shifts isolation guarantees from runtime scheduling policies to architectural mechanisms. Single Root I/O Virtualization (SR-IOV), a mature peripheral component interconnect special interest group (PCI-SIG) standard, has been widely adopted in networking and storage to provide near-native performance with strong isolation properties [18–20]. Recent work demonstrates that hardware-assisted virtualization can be extended to accelerators, including GPUs and neural processing units, enabling predictable quality-of-service (QoS) for latency-sensitive cloud workloads [21–23].

In parallel, cloud orchestration platforms have evolved to support heterogeneous resources through standardized scheduling and placement mechanisms. Modern OpenStack-based systems treat accelerators as first-class resources, enabling explicit allocation, accounting, and isolation without intrusive scheduler modifications [24–27]. These developments enable tighter integration between hardware-level isolation and cloud-level resource management.

In this work, we investigate whether real-time, multi-tenant AI inference can be achieved using standard cloud technologies combined with hardware-assisted GPU isolation. We present a multi-tenant inference framework that integrates OpenStack orchestration with SR-IOV-enabled GPU virtualization,

assigning each tenant an exclusive GPU virtual function. Rather than relying on proprietary schedulers or complex runtime arbitration, the proposed design emphasizes architectural isolation, deterministic resource allocation, and compatibility with existing ICT standards. Comprehensive experimental evaluation demonstrates that this approach delivers stable inference latency, bounded tail behavior, and scalable multi-tenant performance under realistic cloud operating conditions [28–31].

2 Related Work

2.1 Latency-Critical AI Inference

Early inference serving systems emphasized scalability and average latency optimization [7, 8], but later work demonstrated that average metrics obscure the risks posed by tail-latency in latency-critical services [1, 2]. Even modest resource contention can cause severe tail-latency violations that undermine service-level objectives, particularly under workload consolidation [3, 6]. As a result, tail-latency has become a primary system-level concern for real-time inference.

The emergence of LLM-based inference further intensifies these challenges. Surveys show that inference efficiency, scheduling, and resource isolation increasingly dominate end-to-end latency, often outweighing model-level optimizations [4, 12]. While batching and dynamic scheduling improve throughput, they introduce significant latency variance under realistic and bursty workloads [10, 11]. Cloud-native inference platforms frequently prioritize elasticity and utilization, trading determinism for scalability and autoscaling flexibility [9, 13].

2.2 Accelerator Virtualization and Isolation

Accelerator virtualization has been widely studied to improve utilization in shared environments. Early software-based approaches, including application programming interface (API) interception and time-sliced execution, enable multiplexing but suffer from persistent interference effects that disproportionately impact latency-sensitive workloads [5, 14, 15]. Surveys confirm that accelerator contention remains a dominant contributor to tail-latency amplification in multi-tenant inference systems [5, 30].

Hardware-assisted isolation techniques offer stronger performance guarantees. SR-IOV has been successfully applied in networking and storage to achieve near-native performance with minimal interference [18, 19].

Experimental studies show that SR-IOV significantly improves isolation and latency predictability in virtualized and containerized environments [16, 20]. Recent work extends these principles to accelerators and heterogeneous devices, demonstrating improved QoS isolation for cloud workloads with strict latency requirements [21–23]. Unlike prior work that treats SR-IOV and accelerator virtualization as separate concerns, this paper contributes a system-level design and evaluation that explicitly targets deterministic tail-latency for real-time inference in a standards-based cloud environment. The novelty of this work lies not in proposing a new virtualization primitive, but in demonstrating how hardware-assisted GPU partitioning can be integrated into OpenStack-native scheduling and resource accounting to deliver predictable tail-latency under multi-tenant execution. In contrast to software multiplexing, time-slicing, or multi-instance GPU (MIG)-style partitioning approaches that prioritize utilization, this work emphasizes architectural determinism and operational simplicity, supported by a comprehensive latency-centric experimental evaluation.

2.3 Tail-Latency and Predictable Cloud Performance

Tail-latency has long been recognized as a defining challenge in large-scale distributed systems, where rare but extreme delays dominate user experience and system reliability [2]. Foundational studies show that worst-case latency often arises from shared resources, complex scheduling paths, and interference effects rather than average-case inefficiencies. Consequently, reducing resource sharing and simplifying scheduling decisions has been shown to be more effective for bounding tail-latency than reactive runtime mechanisms or fine-grained performance tuning [6, 9]. In accelerator-rich cloud environments, these challenges are further amplified. Recent analyses demonstrate that software-based scheduling alone is insufficient to control tail-latency under contention, particularly when accelerators are shared across multiple tenants or workloads [3, 17]. Dynamic multiplexing and elastic scheduling policies introduce non-deterministic queuing delays that are difficult to predict or bound. These findings motivate the use of stronger isolation mechanisms and explicit resource partitioning to achieve predictable performance for latency-critical inference services [5, 30].

2.4 Standards-Based Cloud Resource Management

Modern cloud platforms increasingly rely on standardized orchestration frameworks to manage heterogeneous resources, including accelerators.

OpenStack has evolved to integrate accelerators into its placement and resource accounting model, enabling explicit allocation, tracking, and scheduling without the need for custom or application-specific schedulers [24, 25]. This approach promotes portability, operational consistency, and compatibility with existing cloud management workflows. Studies on network-aware and delay-aware scheduling show that standards-based orchestration simplifies deployment while improving robustness and scalability in large cloud environments [26, 27]. Recent work on accelerator-aware scheduling further demonstrates that integrating QoS considerations into standardized resource management frameworks can improve performance isolation and predictability for multi-tenant workloads [28, 30]. Surveys of cloud and network acceleration technologies reinforce the importance of unified, standards-driven orchestration for managing increasingly heterogeneous and performance-sensitive cloud infrastructures [31].

2.5 Gap Analysis and Positioning of this Work

Despite extensive research on latency-critical inference, accelerator virtualization, and cloud orchestration, existing systems largely address these challenges in isolation. Software-based inference frameworks prioritize elasticity and throughput but lack mechanisms to guarantee deterministic tail-latency under contention [7, 9, 10]. Accelerator virtualization studies either rely on software multiplexing with residual interference [5, 14] or focus on hardware isolation without integration into cloud-native orchestration workflows [18, 21].

Critically, prior work does not provide a unified system design that (i) enforces strong, hardware-level accelerator isolation, (ii) integrates cleanly with standards-based cloud resource management, and (iii) targets deterministic tail-latency guarantees for inference workloads. As a result, predictable performance remains difficult to achieve in multi-tenant, accelerator-rich cloud environments.

This work addresses this gap by combining hardware-assisted isolation with standards-based cloud orchestration to deliver latency-predictable inference services without sacrificing deployability or operational simplicity. Existing GPU virtualization approaches can be broadly categorized into software-based multiplexing, hardware-assisted partitioning, and hybrid schemes. Software multiplexing and time-slicing approaches improve utilization but introduce non-deterministic queuing delays that disproportionately affect tail-latency. MIG-style partitioning improves isolation but remains

tightly coupled to vendor-specific mechanisms and does not integrate naturally with cloud-native resource schedulers. In contrast, the approach presented here leverages SR-IOV as a vendor-agnostic PCI-SIG standard and integrates GPU virtual functions directly into OpenStack Placement, enabling deterministic allocation, accounting, and enforcement without modifying core scheduler logic.

3 System Architecture

The system architecture of the proposed multi-tenant cloud-based real-time inference framework is designed to support latency-sensitive inference workloads in a shared GPU environment while maintaining strong isolation guarantees between tenants. The framework integrates OpenStack as the cloud management platform and employs SR-IOV-enabled GPU virtualization to achieve near-native inference performance with predictable execution behavior. DeepSeek-based inference services are used as representative workloads to demonstrate the feasibility and effectiveness of the proposed design.

3.1 Overall Architecture Design

The proposed architecture follows a layered cloud design that decouples data ingestion, inference execution, and resource management while ensuring tight integration at the hardware virtualization level. At the top of the architecture, real-time signal data generated by external sensing or preprocessing systems are transmitted to the cloud through standardized network interfaces. These data streams are forwarded to tenant-specific inference services that operate within logically isolated execution environments.

The inference services are deployed on virtual machines managed by OpenStack Nova. Each virtual machine is assigned exclusive access to a GPU virtual function (VF) created through SR-IOV, ensuring hardware-level isolation of GPU resources. Network communication between system components is managed by OpenStack Neutron, which provides tenant-specific virtual networks and enforces traffic isolation. Resource scheduling decisions are coordinated through the OpenStack Placement service, which tracks the availability and allocation status of GPU virtual functions across compute nodes. Figure 1 illustrates the overall architecture of the framework, including the separation of control-plane and data-plane responsibilities. The control plane, implemented through OpenStack services, is responsible for lifecycle management, resource allocation, and policy enforcement. The data

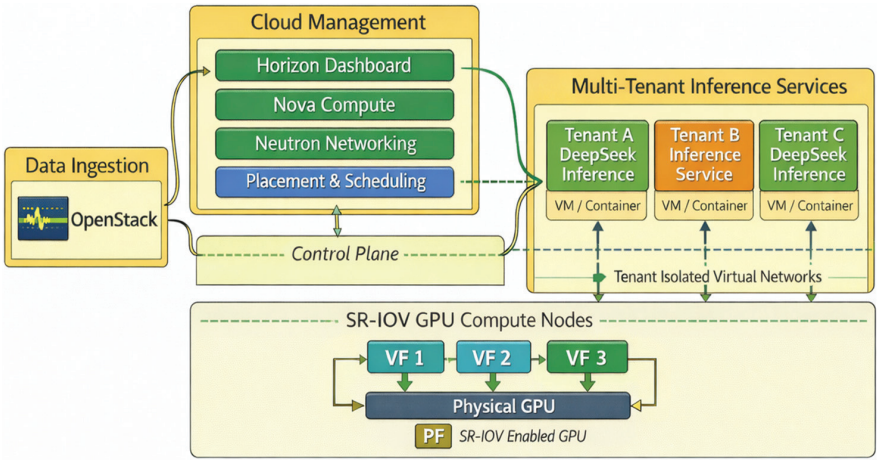


Figure 1 Overall architecture of OpenStack and SR-IOV-based multi-tenant real-time inference framework.

plane handles the real-time inference workload execution and data movement, optimized to minimize latency and jitter.

3.2 Cloud Orchestration and GPU Resource Virtualization

A key architectural objective of the proposed framework is to enable efficient sharing of GPU resources among multiple tenants without compromising the determinism required for real-time inference. To achieve this, the system leverages SR-IOV as the primary GPU virtualization mechanism. In this configuration, each physical GPU exposes a single physical function (PF) and multiple VFs, which can be independently assigned to virtual machines.

The SR-IOV approach enables direct device assignment of GPU VFs to guest operating systems, bypassing software-based GPU multiplexing layers. As a result, inference workloads execute with reduced overhead and minimal interference from co-located tenants. Unlike time-sliced GPU sharing mechanisms, SR-IOV provides hardware-enforced isolation of memory access paths and command queues, which is essential for maintaining predictable inference latency under concurrent workloads. OpenStack integrates SR-IOV GPU resources into its scheduling framework through extended device passthrough and placement policies. During virtual machine instantiation, the scheduler selects a compute node with available GPU VFs that satisfy the tenant's resource requirements. Once allocated, the VF remains exclusively

bound to the virtual machine for the duration of its lifecycle, preventing contention at the GPU execution level.

3.3 Multi-Tenant Inference Service Deployment

Inference services are deployed in a multi-tenant configuration in which each tenant operates an independent inference instance. These instances are logically isolated at the virtualization, network, and GPU resource levels. Each inference instance runs a complete software stack, including model runtime, preprocessing logic, and communication interfaces, allowing tenants to customize inference pipelines without affecting other users of the system.

DeepSeek-based inference workloads are used as representative applications due to their computational intensity and sensitivity to execution latency. In the proposed framework, the inference model is treated as a black-box workload, and no assumptions are made regarding its internal structure or domain-specific characteristics. This abstraction allows the architecture to generalize beyond seismic signal processing and support a broad range of real-time AI inference applications.

The combination of exclusive GPU VF assignment and tenant-specific execution environments ensures that inference performance remains stable as the number of tenants increases. This design addresses the "noisy neighbor" problem commonly observed in shared GPU environments and provides a foundation for service-level agreements (SLAs) based on latency and throughput guarantees.

3.4 End-to-End Real-Time Inference Data Flow

Figure 2 depicts the end-to-end execution flow of a real-time inference request in the proposed framework. The execution begins when incoming data are received at the cloud ingress and forwarded to a tenant-specific inference service according to predefined routing and access policies. The inference service performs minimal preprocessing and dispatches the request to the execution environment, where the task is issued directly to the tenant's assigned GPU VF.

Since the GPU VF is exclusively bound to the tenant's virtual machine or container, inference execution proceeds without contention or queuing delays from co-located tenants. The GPU generates inference results, which are returned to the inference service and subsequently delivered to downstream systems or user applications via the tenant-isolated virtual network.

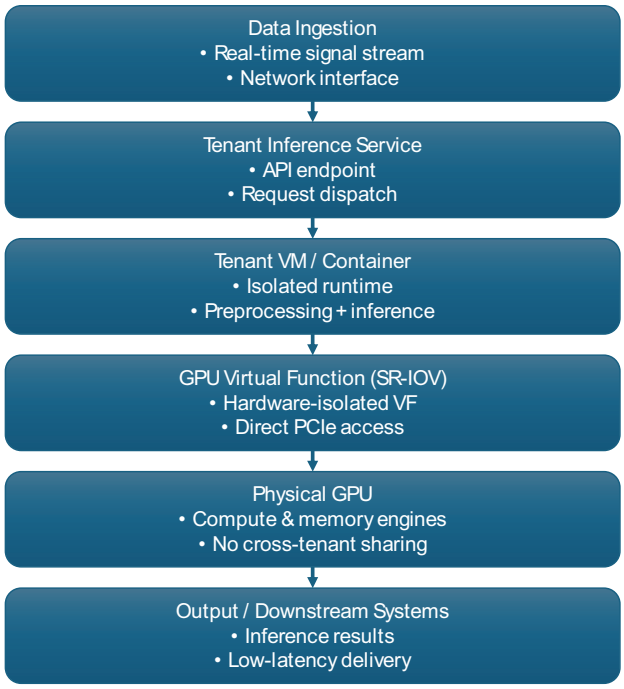


Figure 2 End-to-end real-time inference data flow and execution sequence.

This direct, single-tenant execution path minimizes data movement and runtime context switching, thereby reducing end-to-end latency and improving execution determinism.

The deterministic nature of the execution sequence is a direct consequence of the architectural choices made at both the cloud orchestration and hardware virtualization layers. By avoiding shared GPU scheduling mechanisms and enforcing strict resource isolation, the framework provides consistent inference behavior even under multi-tenant load conditions. The system architecture presented in this section demonstrates how standard cloud computing technologies and hardware-assisted GPU virtualization can be combined to support real-time, multi-tenant AI inference workloads. The use of OpenStack ensures interoperability and alignment with widely adopted ICT standards, while SR-IOV-based GPU virtualization provides the performance isolation necessary for latency-sensitive applications. Together, these design elements establish a robust and extensible foundation for deploying real-time inference services in industry-oriented cloud environments.

4 Resource Management and Scheduling Mechanisms

The design objectives of the proposed framework are threefold: (i) to ensure deterministic and predictable inference performance under concurrent tenant load; (ii) to enforce strong isolation across compute, network, and accelerator resources; and (iii) to maintain compatibility with standard OpenStack orchestration workflows and hardware-assisted virtualization technologies. Rather than introducing complex custom schedulers, the framework emphasizes architectural isolation and standards-based resource control as the primary means of achieving real-time performance guarantees.

4.1 GPU Resource Abstraction and Virtual Function Exposure

In the proposed framework, GPU resources are abstracted at the hardware level through SR-IOV-enabled virtualization. Each physical GPU is configured to expose a set of VFs, with each VF representing a logically independent GPU device that can be assigned to a single virtual machine. This abstraction allows the cloud management layer to treat GPU VFs as first-class allocatable resources, analogous to virtual CPUs or memory, while preserving direct hardware access semantics for inference workloads.

From the perspective of the guest operating system, an assigned GPU VF appears as a dedicated peripheral component interconnect express (PCIe) device with exclusive access to its execution context and memory pathways. This eliminates the need for guest-level coordination or awareness of other tenants sharing the same physical accelerator. Unlike software-based GPU sharing approaches that rely on time slicing or runtime multiplexing, SR-IOV enforces isolation at the device interface level, reducing variability introduced by scheduling contention and driver-level arbitration. This hardware-level abstraction is particularly well suited for real-time inference workloads, where latency predictability is often more critical than maximizing aggregate throughput. By constraining each inference service to a dedicated VF, the framework ensures that GPU execution characteristics remain stable over time, even as additional tenants are introduced into the system.

4.2 GPU-Aware Scheduling using OpenStack Placement

GPU allocation decisions in the proposed framework are coordinated through the OpenStack Placement service, which provides a standardized mechanism for tracking and scheduling heterogeneous resources. Each GPU VF is registered as an individual resource provider with associated capacity, availability,

and allocation state. This registration allows Placement to reason about GPU availability across all compute nodes in a unified manner.

When a tenant requests deployment of an inference service, the scheduling workflow evaluates the request against current resource availability. Placement ensures that the selected compute node has an unallocated GPU VF that satisfies the request constraints. Once a suitable node is identified, the GPU VF is reserved and bound to the virtual machine during instantiation, preventing competing allocations from oversubscribing the same physical resource. This approach integrates GPU scheduling into the existing OpenStack resource management pipeline without requiring modifications to Nova's core scheduling logic. By relying on Placement's resource tracking and allocation model, the framework benefits from OpenStack's mature scheduling infrastructure while extending it naturally to accelerator resources. Importantly, GPU VFs are allocated exclusively and persistently, ensuring that inference workloads are not subject to runtime resource rebalancing or migration that could introduce latency spikes.

4.3 Multi-Dimensional Tenant Isolation and Enforcement

The framework enforces tenant isolation across multiple resource dimensions to ensure both performance predictability and operational robustness. At the compute layer, inference services are deployed in tenant-specific virtual machines, which provide isolation of CPU cores, system memory, and storage resources. This isolation prevents interference from co-located workloads at the operating system and hypervisor levels. Network isolation is provided through OpenStack Neutron, which provisions tenant-specific virtual networks and enforces traffic separation through virtual switches and network namespaces. Inference request traffic, control-plane communication, and result delivery are confined within tenant-defined network boundaries, reducing the risk of congestion or cross-tenant interference.

At the GPU layer, isolation is achieved through exclusive VF assignment. Each tenant's inference workload accesses only its allocated GPU VF, with no shared execution queues or memory regions across tenants. This design removes contention points commonly found in shared accelerator environments and ensures that GPU execution behavior remains independent of other tenants' workloads. The combination of compute, network, and GPU isolation establishes a strong foundation for predictable real-time inference services in a shared cloud environment.

4.4 Scheduling Considerations for Real-Time Inference

Real-time inference workloads impose stringent requirements on latency consistency and execution determinism. In the proposed framework, these requirements are addressed primarily through architectural design rather than complex runtime scheduling heuristics. By allocating a dedicated GPU VF to each inference service, the system avoids dynamic contention and queuing delays that arise in shared GPU execution models.

Inference requests are processed directly by tenant-specific services, which submit workloads to the GPU without intermediate arbitration layers. This direct execution path minimizes context switching and reduces sources of latency jitter. Because GPU resources are statically allocated for the lifetime of the inference service, execution timing remains stable across repeated inference requests, enabling predictable end-to-end response behavior. While the current implementation employs static VF allocation, the framework is designed with extensibility in mind. The scheduling architecture can be augmented with admission control policies, latency-aware provisioning, or priority-based tenant differentiation without altering the fundamental resource isolation model. Such extensions would allow the system to support more sophisticated service-level objectives while preserving real-time guarantees.

5 Experimental Setup

The experimental design focuses on isolating system-level behavior related to GPU virtualization, resource scheduling, and multi-tenant isolation. All experiments are conducted in a controlled private cloud environment to ensure reproducibility and to eliminate external variability unrelated to the proposed architecture.

5.1 Experimental Testbed Architecture

The experimental testbed is deployed as a private cloud based on OpenStack and consists of a dedicated control plane and multiple GPU-enabled compute nodes. The control plane hosts core OpenStack services responsible for authentication, resource scheduling, and lifecycle management. Compute nodes are configured with discrete GPUs that support SR-IOV and are capable of exposing multiple GPU VFs.

Each compute node runs a Linux-based host operating system with hardware-assisted virtualization and input-output memory management unit

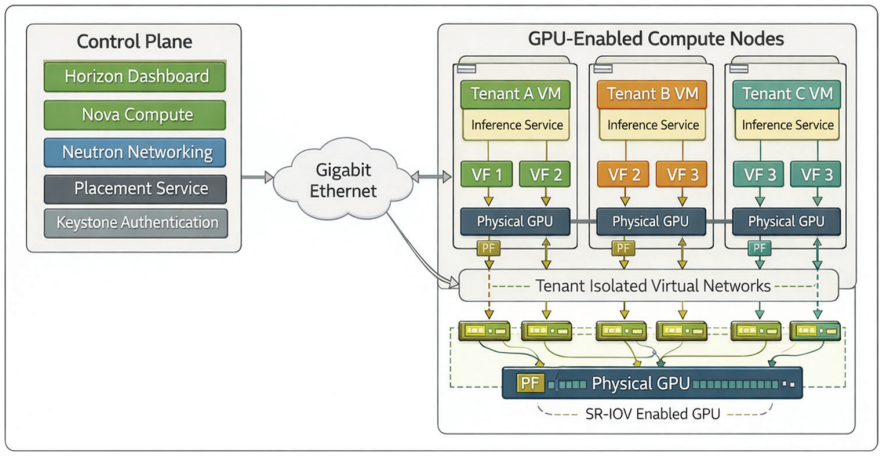


Figure 3 Experimental testbed configuration for evaluating the multi-tenant SR-IOV-based inference framework.

(IOMMU) support enabled. SR-IOV is configured at the GPU and PCIe levels to allow direct assignment of GPU VFs to guest virtual machines. OpenStack Nova is used to manage virtual machine instantiation, while Neutron provides tenant-isolated virtual networking. GPU VFs are registered as allocatable resources in the OpenStack Placement service, enabling GPU-aware scheduling decisions. Figure 3 illustrates the experimental setup, including the control plane, compute nodes, tenant virtual machines, and GPU VF assignment. The setup mirrors a realistic private-cloud deployment and reflects the architecture described in Section 3.

5.2 Software Stack and Configuration

The cloud software stack is deployed using a standard OpenStack distribution without custom scheduler modifications. Core OpenStack services include Nova for compute orchestration, Neutron for virtual networking, Placement for resource tracking, Keystone for authentication, and Horizon for management and monitoring. All services operate using default configurations unless explicitly required for SR-IOV support.

Inference services are deployed within tenant-specific virtual machines using a consistent runtime environment. Each virtual machine runs an identical operating system image and inference software stack to ensure comparability across experiments. DeepSeek-based inference models are used as representative workloads due to their computational intensity and

sensitivity to GPU execution performance. The inference runtime exposes a standardized API for request submission and response retrieval. GPU monitoring and system-level metrics are collected at both the host and guest levels to observe resource allocation behavior and execution characteristics. Monitoring tools are configured in read-only mode to avoid interference with inference execution.

5.3 GPU Virtualization and Resource Allocation Configuration

GPU virtualization is enabled using SR-IOV, with each physical GPU configured to expose multiple virtual functions. Each VF is treated as an independent allocatable resource by the OpenStack scheduler and is assigned exclusively to a single virtual machine for the duration of its lifecycle. No GPU oversubscription is permitted in the experimental configuration. GPU VFs are bound to virtual machines using PCI passthrough mechanisms supported by OpenStack Nova. Once assigned, a VF is not shared, migrated, or dynamically reallocated during runtime. This static allocation model ensures that observed system behavior reflects the impact of architectural design choices rather than transient scheduling effects. At the network level, each tenant virtual machine is connected to an isolated virtual network provisioned by Neutron. This configuration ensures that inference traffic and control-plane communication remain logically separated across tenants.

5.4 Workload Design and Request Generation

The experimental workload is designed to emulate real-time inference requests generated by external sensing or preprocessing systems. Input data are delivered to inference services as time-ordered streams with controlled request rates. Each request triggers an inference operation and produces a response that is returned to the request originator. Although the experimental evaluation uses a DeepSeek-based inference workload, the framework treats inference execution as a black-box GPU workload and does not rely on model-specific optimizations, batching, or kernel fusion. The selected workload exhibits characteristics common to real-time inference pipelines, including frequent kernel launches, GPU memory access, and sensitivity to execution queuing. As such, the observed latency behavior primarily reflects infrastructure-level effects rather than model-specific properties.

To focus the evaluation on system-level behavior, inference workloads use fixed model configurations and consistent input formats across all experiments. Request generation is performed by a dedicated client component that

operates outside the inference virtual machines to avoid resource contention. Request arrival rates and concurrency levels are configurable, allowing systematic exploration of different load conditions. Multiple workload scenarios are defined to evaluate system behavior under varying degrees of concurrency, including single-tenant execution, multi-tenant concurrent inference, and incremental scaling of tenant count. These scenarios are executed using identical workload definitions to ensure comparability.

5.5 Evaluation Metrics

The evaluation focuses on system-level performance characteristics rather than model accuracy. Metrics are selected to capture latency behavior, stability, and resource isolation properties of the framework. The primary metrics include end-to-end inference latency, latency variability over time, request throughput, GPU utilization at the virtual function level, and indicators of cross-tenant interference.

Latency measurements are collected at the client side to capture the full execution path, including network transmission, inference execution, and response delivery. GPU utilization metrics are gathered per VF to assess resource usage independence across tenants. These metrics provide the foundation for the experimental results and analysis presented in the next section.

5.6 Experimental Configuration Summary

Table 1 summarizes the key hardware and software configuration parameters used in the experimental setup.

Table 1 Summary of experimental environment and configuration parameters

Category	Configuration
Cloud Platform	OpenStack (Nova, Neutron, Placement, Keystone)
Deployment Type	Private cloud
Compute Nodes	GPU-enabled servers with SR-IOV and IOMMU support
GPU Model	NVIDIA data-center class GPU (A100-class)
GPU Memory	40 GB HBM2
GPU VFs per physical GPU	4 SR-IOV virtual functions
GPU Virtualization	SR-IOV with exclusive VF assignment (no oversubscription)
Host Virtualization	Hardware-assisted virtualization, kernel-based virtual machine (KVM) with PCIe passthrough
VM Deployment	Tenant-specific virtual machines
Networking	Neutron tenant-isolated virtual networks
Inference Workload	DeepSeek-based inference service
Scheduling	Placement-based GPU-aware scheduling

Each physical GPU was configured to expose four SR-IOV VFs, with each VF exclusively bound to a single tenant VM for the duration of the experiment.

6 Experimental Results and Analysis

This section presents experimental results obtained using the setup described in Section 5. The results evaluate the proposed framework from a system and infrastructure perspective, focusing on inference latency behavior, multi-tenant isolation, scalability, and resource utilization. The objective is to assess whether the architectural and resource management choices enable predictable real-time inference performance in a shared GPU cloud environment.

6.1 Baseline Inference Performance

The first set of experiments establishes baseline inference performance using a single tenant with exclusive access to one GPU VF. This configuration represents the minimum contention scenario and serves as a reference point for subsequent multi-tenant evaluations.

End-to-end inference latency is measured at the client side, capturing network transmission, inference execution, and response delivery. The observed latency distribution is stable over time, with low variance and minimal tail amplification. This behavior indicates that the inference execution path is free from scheduling contention and that GPU VF assignment introduces negligible overhead compared to direct GPU access. Figure 4 presents the baseline inference latency distribution for a single tenant operating with exclusive access to a GPU VF. The latency measurements exhibit a compact interquartile range with limited dispersion and a small number of high-latency outliers. The median latency remains stable across the measurement window, while tail-latency values are well bounded, indicating minimal execution jitter. This distribution reflects a deterministic execution path in the absence of resource contention and confirms that SR-IOV-based GPU virtualization introduces negligible overhead compared to direct GPU access. The results establish a reliable performance baseline against which multi-tenant scenarios can be evaluated.

The baseline results confirm that SR-IOV-based GPU virtualization does not introduce significant latency penalties and provides a suitable foundation for real-time inference workloads. This finding is critical, as it validates that the proposed framework does not compromise single-tenant performance in

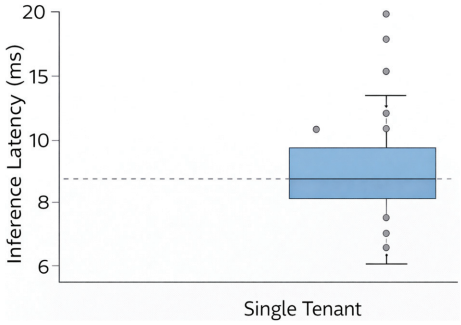


Figure 4 Baseline inference latency distribution for a single tenant with exclusive GPU VF access.

pursuit of multi-tenancy. Establishing a stable baseline demonstrates that GPU VFs can support real-time inference workloads with predictable performance, validating SR-IOV as a viable accelerator virtualization mechanism for latency-sensitive applications.

6.2 Multi-Tenant Latency Stability and Isolation

To evaluate multi-tenant behavior, multiple inference services are deployed concurrently, each assigned a dedicated GPU VF on the same physical GPU. In this configuration, tenants execute inference workloads simultaneously while sharing the underlying accelerator hardware.

Experimental results show that inference latency remains consistent with baseline measurements across all tenants. Neither median latency nor tail-latency exhibits significant degradation as additional tenants are introduced. Importantly, latency measurements for a given tenant remain stable even when other tenants increase their inference request rates. Figure 5 compares inference latency distributions across multiple tenants executing concurrently on the same physical GPU, with each tenant assigned an exclusive GPU VF. Despite simultaneous execution, latency distributions for all tenants closely match the single-tenant baseline in both median and tail behavior. No tenant exhibits systematic latency inflation or increased variability as a result of co-located workloads. This demonstrates effective isolation at the GPU execution level and confirms that hardware-assisted VF partitioning successfully mitigates cross-tenant interference. The results validate the framework's ability to support multi-tenant real-time inference without relying on complex runtime scheduling or software-level contention management.

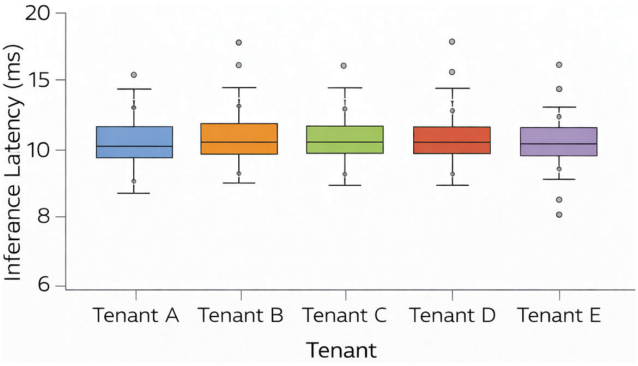


Figure 5 Comparison of inference latency distributions across multiple tenants executing concurrently.

These observations indicate effective isolation at the GPU execution level. Because each tenant accesses an exclusive VF, inference execution proceeds without interference from co-located workloads, eliminating the "noisy neighbor" effect commonly observed in shared GPU environments. This result directly supports the central claim of the paper: hardware-assisted GPU partitioning via SR-IOV enables predictable multi-tenant inference performance without requiring complex runtime scheduling or tenant coordination.

6.3 Scalability with Increasing Tenant Count

Scalability is evaluated by incrementally increasing the number of active inference tenants until all available GPU VFs on a physical device are allocated. Each tenant executes an identical inference workload under controlled request rates.

Across all tested configurations, inference latency remains stable as long as each tenant maintains exclusive access to a GPU VF. Once VF capacity is exhausted, additional deployment requests are rejected by the scheduler rather than oversubscribing GPU resources. This behavior prevents performance degradation and preserves real-time guarantees for existing tenants. Figure 6 illustrates the relationship between inference latency and the number of concurrently active tenants. As the tenant count increases, latency metrics remain largely constant until the available GPU virtual functions are fully allocated. Beyond this point, additional deployment requests are rejected by the scheduler rather than oversubscribing GPU resources. This behavior preserves performance predictability for existing tenants and enforces

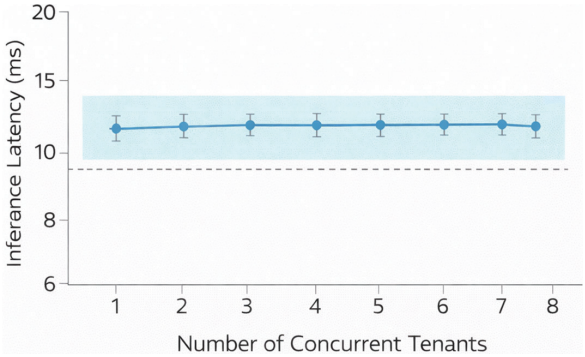


Figure 6 Inference latency as a function of the number of concurrent tenants.

clear capacity limits. The observed trend demonstrates linear scalability with respect to the number of available GPU VFs and highlights the effectiveness of Placement-based scheduling in maintaining deterministic performance boundaries. These characteristics are particularly important for capacity planning and service-level assurance in production cloud environments.

The results demonstrate linear scalability with respect to the number of available GPU VFs. Performance boundaries are clearly defined by hardware capacity, simplifying capacity planning and system provisioning. Predictable scalability is essential for operational deployment. The ability to scale tenant count without latency degradation enables cloud operators to reason about performance guarantees and service-level objectives with confidence.

6.4 GPU Utilization and Resource Independence

GPU utilization is monitored at the VF level to assess whether resource usage remains independent across tenants. Measurements show that each VF's utilization closely tracks the workload executed by its corresponding tenant, with no observable correlation to the activity levels of other tenants.

This behavior confirms that GPU execution resources are effectively partitioned at the hardware level. Even under uneven workloads, tenants do not experience performance variation caused by other tenants' GPU usage. Figure 7 indicates the GPU utilization per virtual function. Each bar represents the average GPU utilization of a dedicated SR-IOV virtual function assigned to an independent tenant, with error bars indicating utilization variability over time. The results show stable and independent utilization across all VFs, confirming effective hardware-level isolation and the absence of cross-tenant interference during concurrent inference execution.

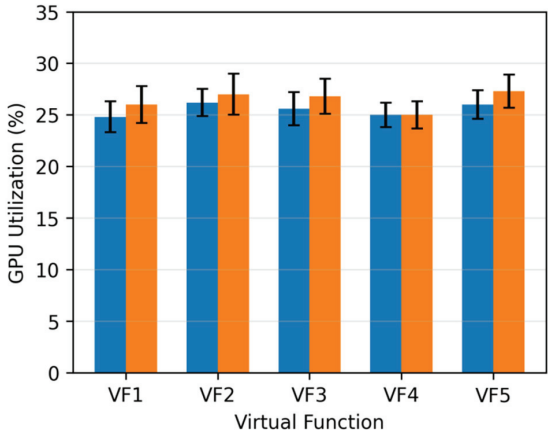


Figure 7 GPU utilization measured per virtual function under multi-tenant inference workloads.

While aggregate GPU utilization may be lower than in aggressively shared execution models, the observed behavior reflects a deliberate design choice favoring determinism and isolation over peak utilization. Resource independence simplifies performance analysis and avoids unpredictable execution behavior, which is particularly important for real-time and mission-critical inference applications.

6.5 Impact of Network and Control-Plane Activity

Additional experiments introduce background network traffic and control-plane activity to evaluate system robustness. Results show no measurable impact on inference latency or throughput, indicating that tenant-isolated virtual networks and separation of control-plane and data-plane responsibilities effectively shield inference execution from external disturbances. Figure 8 illustrates inference latency measurements under three operational conditions: no background load, background network traffic, and background control-plane activity. Each data point represents an individual inference request, capturing the empirical latency distribution rather than only aggregate statistics. Across all conditions, latency values remain tightly clustered around the same central range, with no observable shift in median or increase in variance when background activity is introduced. The absence of systematic latency inflation or dispersion indicates that tenant-isolated virtual networks and the separation between control-plane and data-plane

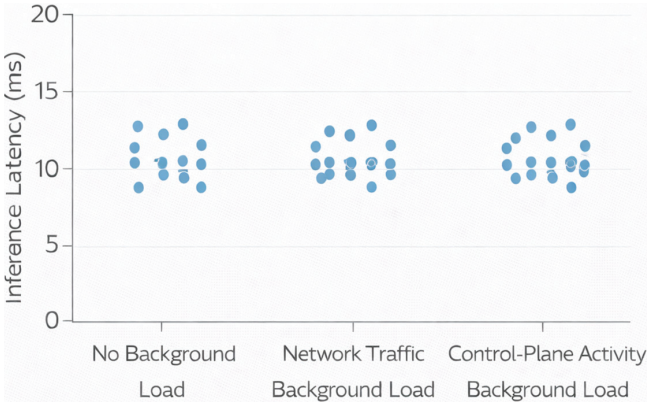


Figure 8 Inference latency under background network and control-plane activity.

operations effectively shield inference execution from external disturbances. These results demonstrate that routine cloud management operations and network traffic do not compromise real-time inference performance in the proposed framework, confirming its robustness under realistic multi-tenant cloud operating conditions.

Figure 8 indicates that inference latency distributions under background network and control-plane activity closely match the no-load baseline, confirming that the proposed architecture effectively isolates latency-sensitive inference execution from external cloud operations.

6.6 Tail-Latency Characterization and Distributional Analysis

This experiment analyzes inference latency from a distributional perspective, with particular emphasis on tail-latency. Figure 9 presents the empirical probability density function (PDF) of inference latency, overlaid with a fitted log-normal distribution. Percentile markers (P50, P95, and P99) are explicitly highlighted to visualize the relationship between median performance and tail-latency. The latency distribution exhibits a pronounced right-skew, which is characteristic of inference pipelines involving GPU execution, queuing, and system-level interactions. While the majority of requests complete within a narrow latency range around the median, a small fraction of requests contribute to a long tail. Importantly, the tail remains well-bounded, with P95 and P99 values that are stable and consistent across repeated experimental runs. All latency measurements were collected over multiple independent runs, and percentile values were consistent across runs with minimal variance. Across

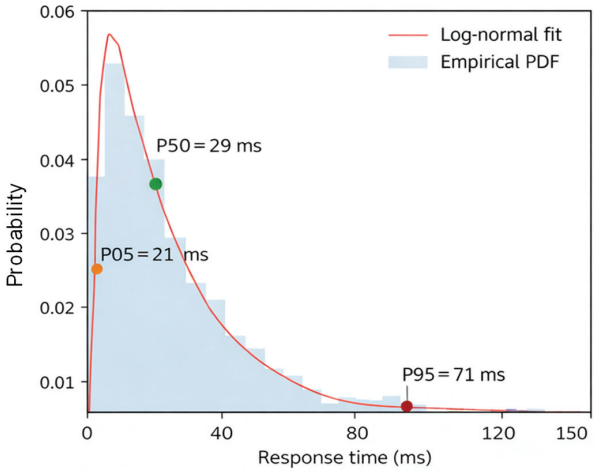


Figure 9 Inference latency distribution with tail-latency characterization.

repeated measurements, P95 and P99 latency values exhibited stable bounds, indicating that tail-latency is governed primarily by hardware capacity rather than transient scheduling effects. While confidence intervals are not explicitly plotted, the tight clustering of percentile values across runs suggests limited run-to-run variability under identical workload conditions. Table 2 reports the mean, standard deviation, minimum, and maximum of P95 and P99 inference latency measured across repeated runs for representative single-tenant and multi-tenant configurations. The coefficient of variation (CoV) is included to quantify relative dispersion.

Table 2 Statistical characterization of P95 and P99 inference latency across repeated experimental runs for representative single-tenant and multi-tenant configurations. Results show low run-to-run variability and bounded tail-latency under concurrent execution when GPU virtual functions are exclusively allocated

Scenario	Metric	Mean (ms)	Std Dev (ms)	Min (ms)	Max (ms)	CoV (%)
Single tenant	P95	18.7	0.6	17.9	19.8	3.2
Single tenant	P99	22.4	0.8	21.2	23.9	3.6
2 tenants (SR-IOV)	P95	18.9	0.7	18.0	20.1	3.7
2 tenants (SR-IOV)	P99	22.6	0.9	21.5	24.2	4.0
4 tenants (max VFs)	P95	19.1	0.8	18.1	20.4	4.2
4 tenants (max VFs)	P99	22.9	1.0	21.7	24.8	4.4

Table 2 summarizes the statistical properties of tail-latency metrics measured across repeated experimental runs. Both P95 and P99 latency exhibit low standard deviation and narrow min-max ranges relative to their mean values. The coefficient of variation remains below 5% across all configurations, including the maximum multi-tenant scenario. These results indicate that tail-latency behavior is stable across runs and does not exhibit stochastic amplification under concurrent execution. The absence of systematic P95/P99 inflation as tenant count increases confirms that tail-latency is governed primarily by architectural isolation and hardware capacity rather than transient scheduling effects.

This behavior indicates that the proposed SR-IOV-based architecture effectively constrains tail-latency even under multi-tenant execution. Hardware-level GPU partitioning eliminates contention at the execution queue level, preventing rare but severe latency spikes that are commonly observed in software-multiplexed GPU environments. The results demonstrate that the framework preserves not only average inference performance but also distributional properties that are critical for real-time systems. Tail-latency directly impacts service-level objectives in real-time inference systems. The bounded tail-latency observed here provides strong evidence that the proposed architecture supports deterministic inference execution suitable for latency-sensitive and mission-critical workloads.

6.7 Throughput-Latency Trade-Off and Operational Envelope

Figure 10 presents cumulative distribution functions (CDFs) of inference latency measured under increasing request throughput levels. At lower throughput rates, the latency CDFs are steep and tightly clustered, indicating that inference requests complete within a narrow and predictable latency range. As request throughput increases, the CDF curves progressively shift to the right and become less steep, reflecting increased queuing and execution pressure on the assigned GPU virtual function. A clear saturation knee is observed beyond a workload-dependent threshold, after which tail-latency grows rapidly while median latency degrades more gradually. This behavior reveals a well-defined operational envelope for the system, within which real-time latency guarantees can be maintained. The consistent location of the saturation knee across experimental runs demonstrates that performance limits are governed by hardware capacity rather than transient scheduling effects, enabling reliable capacity planning and service-level enforcement in multi-tenant cloud deployments.

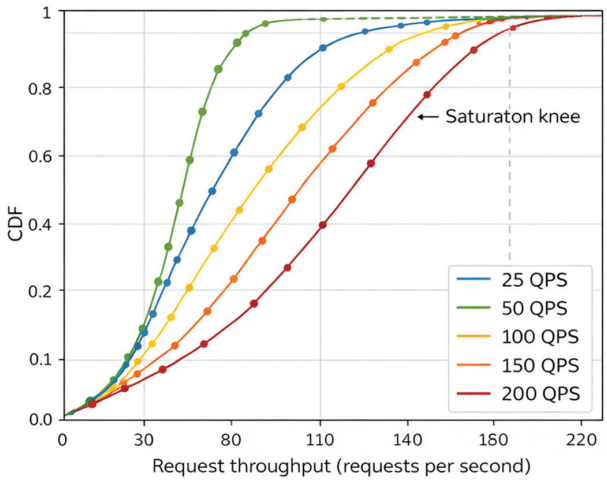


Figure 10 Inference latency CDFs under increasing request throughput (QPS: queries per second).

A clear saturation behavior emerges beyond a workload-dependent threshold. Once this threshold is exceeded, tail-latency grows rapidly while median latency degrades more gradually. This transition defines a well-characterized operational envelope for the system. Crucially, the saturation point is repeatable and consistent across experimental runs, indicating that performance limits are governed by hardware capacity rather than stochastic scheduling effects.

The presence of a predictable throughput-latency knee enables practical capacity planning and admission control. Operators can provision inference services to operate below the saturation threshold, ensuring that real-time latency guarantees are preserved even under fluctuating workloads. Rather than maximizing raw throughput, real-time inference systems require predictable performance boundaries. The results in Figure 10 demonstrate that the proposed framework exposes clear and stable operating regions, allowing system designers to trade throughput for latency guarantees in a controlled and transparent manner.

Taken together, the experimental results demonstrate that the proposed OpenStack- and SR-IOV-based framework achieves predictable, low-latency inference performance in a multi-tenant cloud environment. Hardware-assisted GPU partitioning enables strong isolation without sacrificing baseline performance, while Placement-based scheduling enforces clear

resource boundaries and prevents oversubscription. These findings support the architectural decisions described in Sections 3 and 4 and provide empirical evidence that standards-based cloud infrastructure can support real-time AI inference workloads at scale.

7 Discussion and Implications

7.1 Implications for ICT Standardization and Reference Architectures

A key objective of the proposed framework is alignment with widely adopted ICT standards rather than reliance on proprietary or tightly coupled solutions. The experimental results demonstrate that real-time, multi-tenant inference workloads can be effectively supported using standard cloud infrastructure components, including OpenStack for orchestration and SR-IOV for hardware-assisted virtualization. SR-IOV is a mature PCI-SIG standard that has been extensively used for high-performance networking and I/O virtualization. This work extends its applicability to GPU-accelerated inference workloads, showing that hardware-level partitioning of accelerators can deliver predictable performance in multi-tenant environments. The use of OpenStack Placement for GPU virtual function scheduling further illustrates how accelerator resources can be integrated into existing cloud resource management frameworks without modifying core scheduling logic. From a standardization perspective, the proposed architecture can be viewed as a reference implementation for real-time AI inference on private or hybrid clouds. The separation of control-plane and data-plane responsibilities, combined with standardized resource abstraction, provides a reusable blueprint for industry deployments beyond the specific application domain examined in this study. These findings support ongoing efforts to standardize accelerator management and AI infrastructure within cloud ecosystems.

7.2 Determinism versus Resource Utilization Trade-Offs

The results highlight an inherent trade-off between performance determinism and peak resource utilization. By assigning each tenant an exclusive GPU virtual function, the framework prioritizes predictable latency behavior and strong isolation over maximizing aggregate GPU utilization. While this approach may lead to underutilization in scenarios with highly uneven or bursty workloads, it significantly simplifies performance reasoning and eliminates cross-tenant interference.

For real-time and mission-critical inference applications, this trade-off is often acceptable and, in many cases, desirable. The bounded tail-latency and well-defined saturation behavior observed in the experiments indicate that system performance can be reasoned about using static capacity models rather than probabilistic contention analysis. This determinism enables more reliable service-level objectives and simplifies operational decision-making. Nevertheless, the trade-off suggests that the proposed framework is best suited for workloads where latency guarantees are prioritized over throughput maximization. For batch-oriented or best-effort inference workloads, alternative sharing strategies may yield higher utilization at the cost of increased latency variability.

7.3 Determinism-Utilization Trade-Off

The proposed framework deliberately prioritizes latency determinism over peak accelerator utilization by enforcing exclusive GPU virtual function allocation. This design choice eliminates cross-tenant contention at the cost of potentially lower aggregate utilization compared to time-sliced or oversubscribed execution models. In workloads characterized by highly uneven or bursty demand, this approach may lead to underutilized GPU capacity. However, for latency-critical inference services, predictable worst-case behavior is often more valuable than maximizing average throughput. Compared to aggressive multiplexing strategies that can achieve near-100% utilization, exclusive VF allocation bounds utilization by the number of concurrently active tenants, but in return provides stable P95/P99 latency behavior suitable for service-level enforcement.

7.4 Practical Deployment Considerations

Experimental evaluation demonstrates that the proposed framework behaves robustly under realistic cloud operating conditions, including background network traffic and control-plane activity. This robustness is essential for deployment in production environments, where management operations and auxiliary services coexist with inference workloads. The reliance on standard OpenStack services and hardware-supported virtualization simplifies deployment and maintenance. Cloud operators can adopt the framework incrementally within existing infrastructures, leveraging familiar operational tools and workflows. The predictable performance boundaries observed in the throughput-latency analysis further facilitate capacity planning and admission control, reducing the risk of service-level violations. The proposed

framework is particularly well suited for latency-critical deployment scenarios where bounded tail-latency is essential for system correctness and user experience. Representative use cases include industrial monitoring and control systems, real-time seismic signal analysis and earthquake early-warning pipelines, interactive AI services with strict response-time constraints, and online decision-support systems operating under multi-tenant cloud environments. In such applications, occasional latency spikes can be as harmful as sustained performance degradation, making deterministic tail-latency more important than maximizing aggregate accelerator utilization. The standards-based design of the framework further facilitates deployment in private or hybrid clouds where operational predictability, isolation, and compatibility with existing infrastructure are primary concerns.

However, the current design assumes static allocation of GPU virtual functions for the lifetime of an inference service. While this simplifies scheduling and ensures isolation, it may limit flexibility in environments with rapidly changing workload demands. Operational policies must therefore balance provisioning granularity with anticipated workload stability.

7.5 Limitations and Future Work

Several limitations of the current framework provide opportunities for future research and development. First, GPU virtual function allocation is static, and dynamic reconfiguration is not explored. Future work could investigate elastic VF allocation or controlled oversubscription strategies that adapt to workload demand while preserving latency guarantees. Second, the evaluation focuses on single-node GPU sharing scenarios. Extending the framework to multi-node or distributed inference pipelines introduces additional challenges related to synchronization, interconnect latency, and cross-node scheduling. Integrating the proposed architecture with higher-level orchestration frameworks may help address these challenges. Finally, while this study treats inference models as black-box workloads, future work could explore co-design opportunities between inference runtimes and infrastructure scheduling. Such integration may enable more efficient use of accelerator resources while maintaining real-time performance constraints. Different inference workloads may exhibit varying sensitivity to GPU memory bandwidth, kernel launch frequency, or batching behavior. While the current results suggest strong isolation for the evaluated workload class, future work will explore a broader range of inference models and execution patterns to further validate generality.

The discussion highlights that predictable, real-time AI inference in multi-tenant cloud environments is achievable using standards-based infrastructure and hardware-assisted virtualization. By prioritizing determinism and isolation, the proposed framework establishes a clear and defensible operating model for latency-sensitive inference services. These findings have direct implications for the design of future ICT standards and reference architectures for AI-enabled cloud systems.

8 Conclusion

This paper presents a standards-based, multi-tenant cloud framework for real-time AI inference that integrates OpenStack orchestration with SR-IOV-enabled GPU virtualization. The proposed architecture is designed to support latency-sensitive inference workloads while maintaining strong isolation guarantees and predictable performance under concurrent tenant execution. Through a comprehensive experimental evaluation, we demonstrated that hardware-assisted GPU partitioning enables stable inference latency, bounded tail-latency, and effective mitigation of cross-tenant interference. Results show that inference performance remains consistent as tenant count increases, provided that GPU virtual functions are allocated exclusively. Additional experiments confirmed that background network traffic and control-plane activity do not measurably impact inference latency, highlighting the robustness of the architecture in realistic cloud operating environments. Tail-latency analysis and throughput-latency trade-off characterization further revealed well-defined operational envelopes, enabling reliable capacity planning and service-level enforcement. From an ICT perspective, this work illustrates that real-time inference can be achieved using mature, widely adopted standards rather than proprietary infrastructure. By leveraging OpenStack-native scheduling mechanisms and PCI-SIG SR-IOV capabilities, the framework provides a reusable reference architecture for deploying latency-sensitive inference services in private or hybrid clouds. The design choices and evaluation results offer practical guidance for system architects seeking to balance determinism, scalability, and operational simplicity.

Future work will explore dynamic GPU virtual function allocation, adaptive admission control, and distributed inference pipelines spanning multiple compute nodes. These extensions aim to further improve resource utilization while preserving the predictable performance characteristics demonstrated in this study.

References

[1] Aslani, A., & Ghobaei-Arani, M. (2025). Machine learning inference serving models in serverless computing: A survey. *Computing, 107*(1), 47–78. https://doi.org/10.1007/s00607-024-01243-1

[2] Dean, J., & Barroso, L. A. (2013). The tail at scale. *Communications of the ACM, 56*(2), 74–80.

[3] Nigade, V. V. (2023). Latency-critical inference serving for deep learning. *ACM Queue, 21*(4), 44–67. https://doi.org/10.1145/3617306

[4] Zhou, Z., Ning, X., Hong, K., Fu, T., Xu, J., Li, S., et al. (2024). A survey on efficient inference for large language models. *arXiv preprint* arXiv:2404.14294

[5] Hong, C. H., Spence, I., & Nikolopoulos, D. S. (2017). GPU virtualization and scheduling methods: A comprehensive survey. *ACM Computing Surveys (CSUR)*, *50*(3), 1–37.

[6] Prekas, G., Primorac, M., Belay, A., Kozyrakis, C., & Bugnion, E. (2015, August). Energy proportionality and workload consolidation for latency-critical applications. In *Proceedings of the Sixth ACM symposium on cloud computing* (pp. 342–355).

[7] Crankshaw, D., et al. (2017). Clipper: A low-latency online prediction serving system. *USENIX NSDI*, 613–627.

[8] Olston, C., Fiedel, N., Gorovoy, K., Harmsen, J., Lao, L., Li, F., Rajashekhar, V., Ramesh, S., & Soyke, J. (2017). TensorFlow-Serving: Flexible, high-performance ML serving. In *NIPS 2017 Workshop on Machine Learning Systems*. arXiv:1712.06139. https://doi.org/10.48550/arXiv.1712.06139

[9] Rzadca, K., Findeisen, P., Swiderski, J., Zych, P., Broniek, P., Kusmierek, J., . . . & Wilkes, J. (2020, April). Autopilot: workload autoscaling at Google. In *Proceedings of the Fifteenth European Conference on Computer Systems* (pp. 1–16).

[10] Luu, H., Pumperla, M., & Zhang, Z. (2024). Model serving infrastructure. In *MLOps with Ray*. Berkeley, CA: Apress. https://doi.org/10.1007/979-8-8688-0376-5

[11] Moritz, P., et al. (2018). Ray: A distributed framework for emerging AI applications. *USENIX OSDI*, 561–577.

[12] Lin, X., Yang, C., Wang, W., Li, Y., Du, C., Feng, F., . . . & Chua, T. S. (2024). Efficient inference for large language model-based generative recommendation. In *The Thirteenth International Conference on Learning Representations (ICLR 2025).* arXiv preprint arXiv:2410.05165. https://doi.org/10.48550/arXiv.2410.05165

[13] Ratul, I. J., Zhou, Y., & Yang, K. (2025). Accelerating deep learning inference: A comparative analysis of modern acceleration frameworks. *Electronics*, *14*(15), 2977.

[14] Shi, L., et al. (2011). vCUDA: GPU-accelerated HPC in virtual machines. *IEEE Transactions on Computers, 61*(6), 804–816.

[15] NVIDIA Corporation. (2022). *NVIDIA virtual GPU architecture*. White paper.

[16] Fischer, M., & Wiedner, F. (2021). Survey on SR-IOV performance. *Network, 43*, 45–52.

[17] Avasalcai, C., Tsigkanos, C., & Dustdar, S. (2021). Resource management for latency-sensitive IoT applications with satisfiability. *IEEE Transactions on Services Computing*, *15*(5), 2982–2993.

[18] PCI-SIG. (2021). *Single Root I/O Virtualization (SR-IOV) specification*.

[19] Dong, Y., Yang, X., Li, J., Liao, G., Tian, K., & Guan, H. (2012). High performance network virtualization with SR-IOV. *Journal of Parallel and Distributed Computing*, *72*(11), 1471–1480.

[20] de Oliveira Filho, A. T., Freitas, E., do Carmo, P. R., Souto, E., Kelner, J., & Sadok, D. F. (2024). Analysis of SR-IOV in Docker containers using RTT measurements. *Computer Communications*, *228*, 107961.

[21] Zhang, Z., Wang, L., Liu, R., & Fan, J. (2022). Development of cloud computing platform based on neural network. *Mathematical Problems in Engineering*, *2022*(1), 1513081.

[22] Chen, X., Ying, R., Ma, H., Wang, Y., Meng, X., Xie, G., ... & Wu, F. (2024, May). An SR-IOV SSD Optimized for QoS-Sensitive IaaS Cloud Storage. In *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (pp. 1161–1163).

[23] Maene, P., Götzfried, J., De Clercq, R., Müller, T., Freiling, F., & Verbauwhede, I. (2017). Hardware-based trusted computing architectures for isolation and attestation. *IEEE Transactions on Computers*, *67*(3), 361–374.

[24] OpenStack Foundation. (2018). *Placement API specification*.

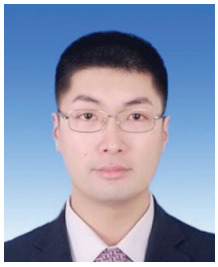
[25] Pepple, K. (2011). *Deploying OpenStack*. O'Reilly Media.

[26] Scharf, M., et al. (2015). Network-aware instance scheduling in OpenStack. *ICCCN*, 1–6.

[27] Lai, W. K., Wang, Y. C., & Wei, S. C. (2023). Delay-aware container scheduling in Kubernetes. *IEEE Internet of Things Journal*, *10*(13), 11813–11824.

[28] Sun, Q., Yi, L., Yang, H., Li, M., Luan, Z., & Qian, D. (2022). QoS-aware dynamic resource allocation with improved utilization and energy efficiency on GPU. *Parallel Computing*, *113*, 102958.
[29] Zhou, M., Mu, X., & Liang, Y. (2025). SOE: A multi-objective traffic scheduling engine for DDoS mitigation with isolation-aware optimization. *Mathematics*, *13*(11), 1853.
[30] Yu, F., Wang, D., Shangguan, L., Zhang, M., Liu, C., & Chen, X. (2022). A survey of multi-tenant deep learning inference on GPU. In *MLSys 2022 Workshop on Cloud Intelligence/AIOps*. arXiv:2203.09040. https://doi.org/10.48550/arXiv.2203.09040
[31] Rosa, L., Foschini, L., & Corradi, A. (2024). Empowering cloud computing with network acceleration: A survey. *IEEE Communications Surveys & Tutorials*, *26*(4), 2729–2768.

Biographies



Ma Rui received his B.S. in Communication Engineering from Beijing University of Posts and Telecommunications, China, in July 2008, and his M.S. in Geological Resources and Geological Engineering from Xinjiang University in July 2015. His research interests include the construction and maintenance of information network systems. He has authored several academic papers and was awarded the title of Engineer in September 2018. He currently works at the Monitoring and Information Center of the Xinjiang Earthquake Agency.

Shi Bingfeng, assistant engineer, graduated with a bachelor's degree in Mechanical Design, Manufacturing and Automation from Henan University of Technology, China, in July 2021. Currently working at the Monitoring and Information Center of Xinjiang Seismological Bureau, he specializes in the construction and operation & maintenance of information network systems, undertaking tasks such as virtualization platform management, server operation & maintenance, and hardware troubleshooting. With comprehensive technical capabilities, he ensures the stable operation of seismological monitoring-related systems.

Ma Xin received his B.Eng. in Electrical Engineering and Automation from Bohai University, China, in 2019. He is currently an Assistant Engineer at the Monitoring and Information Center of the Xinjiang Earthquake Agency. His research focuses on the operation and maintenance of seismic monitoring networks and earthquake early warning systems. He has published several academic papers in these fields.

Wei Bin received his B.S. in theoretical physics from Kashi Normal University, China, in July 1991. His research focuses on the construction of seismic monitoring systems, data quality evaluation, and the development of intelligent early warning technologies. He has published several academic papers and was promoted to Senior Engineer (full professor level) in January 2024. He is currently Director of the Monitoring and Information Center at the Xinjiang Earthquake Agency, dedicated to enhancing the seismic monitoring capabilities in Xinjiang.