cognizant®

Optimizing resource utilization and reducing costs through GPU segmentation

Contents

Executive summary 03

Optimizing GPU usage: Segmentation and multitenancy for scalable AI 04

The growing cost-efficiency gap in AI infrastructure 04

GPU segmentation: Concepts and architectural principle 05

- NVIDIA MIG 06
- Time-slicing and GPU sharing via schedulers 07
- Container-based GPU isolation and orchestration 08
- Orchestration and policy-based GPU allocation 09

Enterprise GPU segmentation and multitenancy requirements 10

Our approach towards multitenancy 10

- Segmentation and isolation strategy with dynamic fractional GPUs 11
- Scheduling, monitoring and fair-share allocation 12
- Tenant isolation, access control and resource governance 12

Business impact and outcomes 13

GPU segmentation and multitenancy for a scalable AI edge 14

Executive summary

Graphics processing units (GPUs) have become one of the most constrained and expensive resources in modern AI infrastructure. As enterprises accelerate machine learning and generative AI (gen AI) initiatives, challenges such as GPU capacity constraints, longer procurement lead times and low utilization—often between 30% and 60%--drive costs upward and obstruct productivity. Inefficient allocation methods such as dedicating entire GPUs to single workloads, result in significant underutilization and wasted capital.

GPU segmentation offers a powerful solution by partitioning GPUs into smaller, manageable units that multiple teams and workloads can use simultaneously. This enables isolation, fairness and governance while improving utilization significantly. In this white paper, we will look at the four primary segmentation techniques—NVIDIA Multi-Instance GPU (MIG), time-slicing, container-based isolation and orchestration, and policy-based orchestration—and how each of them offer unique benefits and trade-offs.

Cognizant's AI Factory builds on these foundations with a platform-centric multi-tenancy approach that integrates segmentation strategies with intelligent scheduling, governance and observability. This combination provides a scalable, flexible and security-first framework for enterprise GPU utilization, delivering up to 30%–40% improvements in utilization, faster experimentation cycles and substantial infrastructure cost savings.

Optimizing GPU usage: Segmentation and multitenancy for scalable AI

GPUs now sit at the center of AI driven innovation, and their scarcity, cost and operational complexity make efficient utilization a strategic necessity rather than a technical preference. And organizations should pay attention because the gap between GPU supply and demand is widening, and the cost of inefficiency—both financial and operational—continues to rise as AI adoption accelerates.

This white paper outlines the core factors driving GPU underutilization and provides a view of the four dominant GPU segmentation methodologies used across modern enterprise environments. It also details how Cognizant's platform driven multitenancy approach integrates segmentation, orchestration, governance and observability to help organizations securely and efficiently scale shared GPU infrastructure. Readers will gain an understanding of why segmentation matters, how different techniques compare and what platform level capabilities are required to achieve sustainable, high efficiency GPU operations at scale.

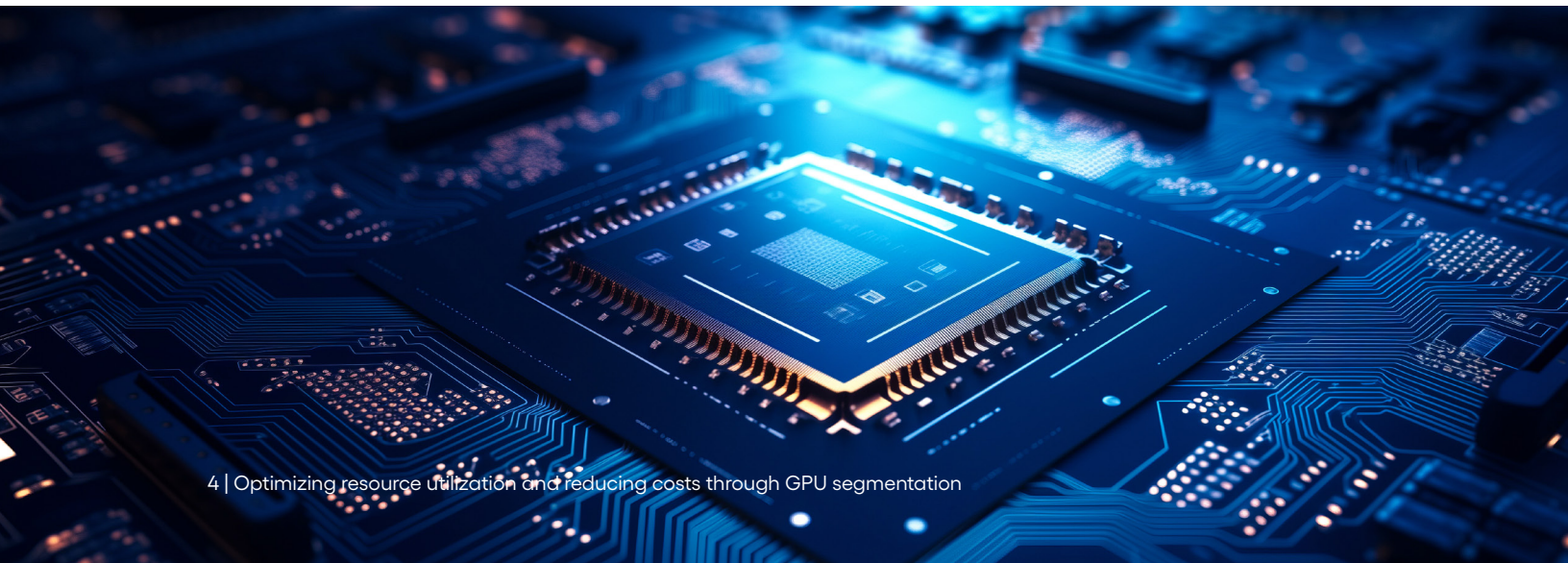
The growing cost-efficiency gap in AI infrastructure

Modern AI infrastructure is facing a severe cost-efficiency problem driven by GPU scarcity and underutilization. Yet industry data shows average GPU utilization hovers around 40%, leaving 60% of capacity idle, consuming power, rack space and cooling without delivering computation. For a 10-GPU cluster of NVIDIA H200s, even 50% utilization translates to over US $20,000 per month in idle computational capacity. This inefficiency translates into thousands in wasted spend over a typical lifecycle.

Dedicated GPU allocation gives teams exclusive access to entire devices, even if workloads use only a fraction of resources, causing fragmentation and waste. Conversely, uncontrolled sharing without governance leads to resource contention, unpredictable performance and security gaps.

Both extremes fail to provide a middle ground of safe, observable and governed sharing, which are essential for scaling enterprise AI workloads efficiently.

The ripple effects go beyond hardware costs. Low utilization slows training cycles, forcing teams to overprovision GPUs to meet SLAs and inflating budgets further. Lack of orchestration demands specialized staff for manual scheduling and troubleshooting, adding months of engineering overhead. Governance gaps expose enterprises to compliance risks and data leakage. Ultimately, the true cost of GPU infrastructure includes not just hardware and energy but also human and opportunity costs, making intelligent resource management critical for sustainable AI operations.



GPU segmentation: Concepts and architectural principle

Segmentation decouples physical resource ownership from logical resource allocation. Instead of assigning entire GPUs to individual teams or projects, segmentation enables a many-to-one relationship; dozens of workloads can logically own segments of a single physical GPU, each with its own memory, segment and scheduling guarantees.

Segmentation can be done through hardware controls, driver implementations or orchestration utilities and tools. But effective multiuser group segmentation requires coordination across layers. A hardware-isolated MIG instance still requires intelligent scheduling to avoid fragmentation. A time-sliced GPU shared by multiple containers requires driver-level memory enforcement to prevent one container from exhausting GPU memory at the expense of others.

Four key methodologies for GPU segmentation

1. NVIDIA multi-instance GPU (MIG)

- Hardware-based partitioning into up to 7 isolated instances. silicon-level isolation
- **Best for:** Cloud providers, regulated industries, multi-tenant with strict SLAs
- **Pros:** Hardware isolation, predictable performance, fault containment
- **Cons:** Limited GPU support, rigid partitioning, licensing costs

2. Time-slicing & GPU sharing

- Software-based rapid context switching between workloads. Driver maintains seperate execution contexts. Round-robin scheduling
- **Best for:** Development teams, research clusters, burst workloads
- **Pros:** Hardware agnostic, dynamic flexibility, simple implementation
- **Cons:** No isolation, performance overhead, unpredictable latency

3. Container-based GPU isolation

- OS-level virtualization via Kubernetes + namespaces/cgroups. GPU device plugin exposes resources, container runtime enforces limits.
- **Best for:** Internal multi-team environments, ML development platforms
- **Pros:** Universal compatibility, Kubernetes-native, granular policy control
- **Cons:** Weak isolation, no QoS guarantees, reactive memory enforcement

4. Orchestration & policy-based

- Intelligent scheduling and governance at platform layer. Job queue evaluation: quotas, priorities, bin packing, dynamic reallocation.
- **Best for:** Enterprise AI, cloud services, academic institutions
- **Pros:** Fairness/governance, universal applicability, operational control
- **Cons:** Complex implementation, policy overhead, scheduling policy

By enabling multiple smaller workloads to share a single GPU, segmentation directly increases the fraction of GPU capacity devoted to useful computation. Rather than leaving 60% of a GPU idle while a single job runs, segmentation allows three or four additional workloads to execute simultaneously. With segmentation, you get flexibility to dynamically adjust resource allocation in response to changing demand.

It also ensures complete tenant separation by preventing memory access, execution observation and resource contention, enabling secure and performant multitenant workloads on shared physical hardware. With key focus to reduce reliance of infrastructure engineers, segmentation ensures predictable performance and SLA compliance by allocating guaranteed GPU resources, while abstracting resource management for users enabling automation, self-service and policy-driven governance without manual intervention.

The four most common GPU segmentation methodologies:

NVIDIA MIG

MIG is a hardware-based feature available on NVIDIA GPUs that enables a single physical GPU to be partitioned into as many as seven fully isolated instances. Each instance behaves like an independent GPU from the perspective of a software, while sharing the underlying silicon resources.

MIG provides isolation at the architectural level. In a conventional GPU, all streaming multiprocessors (SMs) and memory resources are shared across workloads. With MIG enabled, SMs are divided into disjoint groups, each exclusively assigned to a single instance. Similarly, the memory subsystems—such as L2 cache banks, memory controllers and DRAM address buses—are partitioned so that each instance has dedicated memory paths inaccessible to others.

This design creates a strong isolation boundary, ensuring on-chip interconnects, caches and memory controllers are physically distinct across instances. As a result, one instance cannot access another's memory, interfere with its cache or exploit timing-based side-channel vulnerabilities. MIG is typically used by cloud service providers to partition GPUs into fixed-size instances that are leased to customers with guaranteed isolation and SLAs., or shared development and production inferencing at scale Multiple inference models run in parallel, each with its own GPU instance.

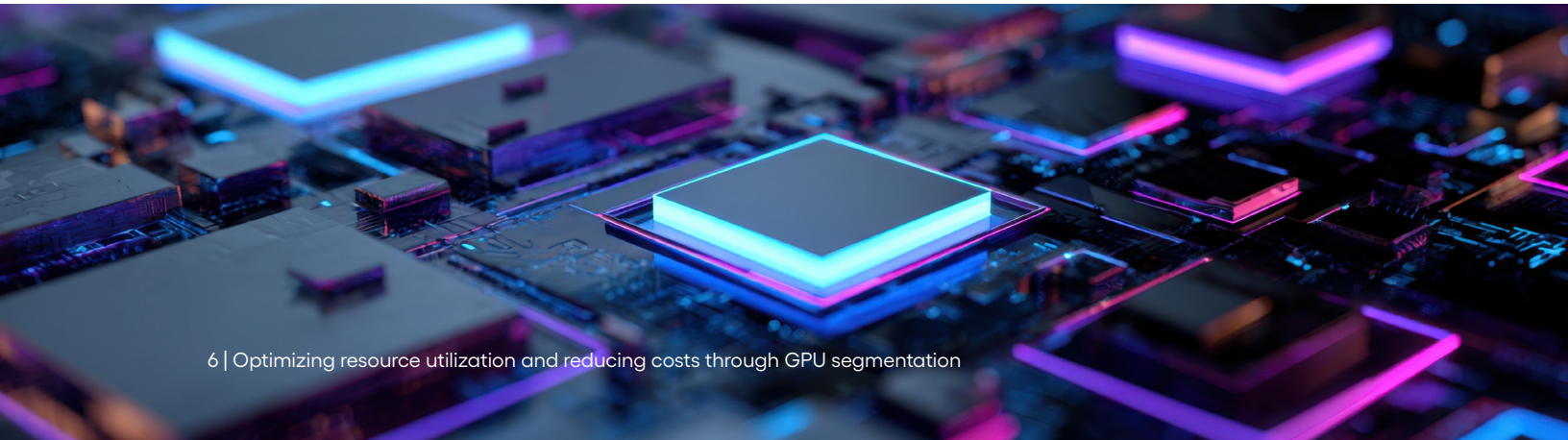
Advantages:

- Hardware-level Isolation and security: MIG enforces silicon-level separation with cryptographic boundaries, ensuring secure multitenant workloads and compliance for regulated industries
- Predictable performance: Dedicated compute, memory and cache per instance eliminate resource contention, delivering stable latency and throughput without noisy neighbor effects
- Fault containment and compatibility: Errors remain isolated within an instance, and full CUDA compatibility allows applications to run without modification

Limitations:

- Licensing constraints: MIG is supported only on select NVIDIA GPUs (A100, H100), requiring vGPU licensing in virtualized environments, which adds cost and complexity
- Rigid partitioning and scheduling overhead: Fixed profiles cannot be dynamically resized, leading to fragmentation and underutilization, while scheduling algorithms must address bin-packing challenges to avoid 10%–20% efficiency loss
- Memory allocation trade-offs: Partitioning reduces available memory per instance, limiting large-memory workloads even when other instances are idle

MIG is excellent for multitenant scenarios where isolation requirements are paramount and workload sizes are relatively uniform. Cloud providers and regulated enterprises benefit most from MIG's hardware isolation and predictable performance. However, organizations with highly variable workload sizes (some requiring 20 GB, others 2 GB) may experience fragmentation and underutilization. In such cases, combining MIG with other segmentation techniques can provide additional flexibility.



Time-slicing and GPU sharing via schedulers

GPU sharing via scheduler is a software-based technique that enables multiple workloads to share a single physical GPU by rapidly switching execution between them. Instead of partitioning hardware resources, the GPU's compute engines are allocated to one workload for a short times interval (a time slice, typically microseconds to milliseconds), then preempted and switched to another workload in a round-robin or priority-based order.

The GPU driver maintains separate execution contexts for each workload, including kernel command buffers, register state and memory allocations. When switching, the driver saves the current workload state and restores the next one. For applications using standard CUDA APIs, the application perceives access to the full GPU, hence resolving limitations that MIG faces.

SLURM is a popular option. The scheduler treats the GPUs as standard resources, while the plugin ensures fair time-sharing across workloads. Typically used when we need to run multiple experiments and models concurrently without blocking each other.

Advantages:

- **Hardware agnostic:** Works on any GPU, including older models without MIG support
- **Dynamic and flexible:** Logical GPU ratios can be adjusted without hardware changes or rigid profiles with flexible oversubscription and no memory fragmentation
- **Simple implementation:** Enabled via standard GPU device plugin and utilities with no special licenses or complex setup

Limitations:

- **No isolation:** Memory and fault isolation are absent; one workload can impact others
- **Performance overhead:** Context switching and cache eviction reduce throughput, especially under heavy sharing
- **Unpredictable latency:** SLA guarantees are hard to maintain without advanced priority-aware scheduling

Time-slicing is ideal for environments prioritizing flexibility and cost efficiency over strict isolation. Development teams, research clusters and organizations with burst or heterogeneous workloads benefit most. However, for regulated industries or latency-critical applications, MIG or hybrid approaches may be preferable.

Container-based GPU isolation and orchestration

Container-based GPU sharing leverages operating system-level virtualization and Kubernetes policies to manage GPU access without hardware partitioning or time-slicing. Multiple containers can request GPU resources from a Kubernetes cluster. The scheduler assigns them to nodes with available GPUs, and the container runtime enforces isolation using namespaces and cgroups.

The GPU device plugin exposes GPUs as a standard resource (nvidia.com/gpu) to Kubernetes. When a pod requests GPUs, the scheduler places it on a node with sufficient availability, and the Kubernetes services configures limits and device paths for the container. GPU memory and compute can be constrained using resource limits, enforced by the Linux kernel and optionally enhanced by CUDA MPS for multiprocess sharing.

Unlike MIG or time-slicing, this approach requires no special hardware features or licenses. Isolation is provided by OS and Kubernetes abstractions, making it operationally simple and widely compatible.

Container-based GPU sharing maximizes flexibility and supports heterogeneous workload clusters, where diverse jobs share GPUs with tailored resource quotas. It also enables development platforms, allowing multiple ML containers to run concurrently on shared GPU infrastructure. Additionally, multiteam environments benefit from namespace-level policies that ensure fair access and governance across a shared cluster.

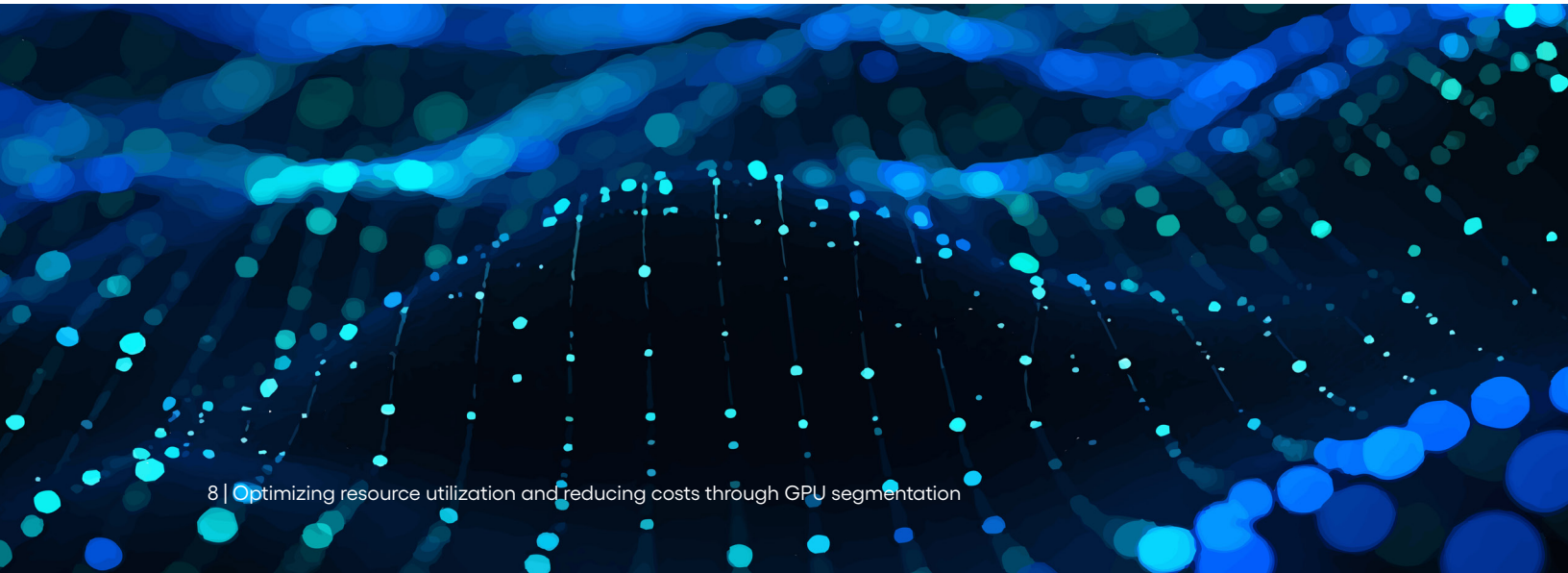
Advantages:

- Universal compatibility: Works with any NVIDIA and AMD GPU and driver version, with no special hardware or licensing required
- Kubernetes-native integration: Uses standard Kubernetes constructs (quotas, namespaces, RBAC), aligning with existing infrastructure practices
- Granular policy control: Resource limits and quotas can be applied at container, pod or namespace-level for fine-grained multitenancy

Limitations:

- Weak isolation: Relies on OS-level boundaries; lacks GPU-level isolation, making it unsuitable for untrusted workloads
- No QoS guarantees: Performance and latency vary based on colocated workloads, and SLAs are hard to enforce
- Reactive memory enforcement: Memory limits are enforced after allocation pressure, risking GPU state corruption before termination

Container-based isolation is suitable for internal, trusted multitenant scenarios within a single organization. It provides good operational flexibility and cost efficiency, but lacks the isolation strength required for untrusted external workloads. Organizations prioritizing operational simplicity and leveraging existing Kubernetes infrastructure benefit most.



Orchestration and policy-based GPU allocation

Orchestration-based GPU allocation treats GPU sharing as a scheduling and governance problem rather than a hardware segmentation challenge. Instead of partitioning GPUs or time-slicing at the driver level, this approach uses intelligent scheduling algorithms and policy enforcement at the platform layer to maximize utilization, ensure fairness and uphold organizational constraints.

Workloads are submitted to a job queue, and the orchestration platform evaluates resource requests, cluster state, team quotas and priority policies before making allocation decisions. It can enforce fair-share allocation, implement priority-based scheduling and apply bin-packing algorithms to minimize fragmentation. Advanced platforms also support dynamic reallocation, reclaiming underutilized GPUs and redistributing them to waiting jobs, while maintaining governance rules such as preventing resource monopolization.

This methodology is typically implemented as a control-plane service integrated with Kubernetes, Slurm or similar schedulers. It operates independently of the underlying GPU segmentation method, making it compatible with dedicated GPUs, MIG instances, time-sliced GPUs or container-based sharing.

Orchestration-based allocation is best fit for enterprise AI where multiple teams share a GPU cluster under quota-based policies. Cloud AI services leverage this approach to enforce fair-share allocation and enable preemption for premium customers. Academic and research institutions also adopt policy-driven scheduling to ensure equitable access across projects while prioritizing time-sensitive workloads.

Advantages:

- Fairness and governance: Enforces quotas and priorities to prevent monopolization and ensure equitable resource distribution
- Universal applicability: Works across all GPU segmentation methods—dedicated, MIG, time-slicing or container-based
- Operational control: Enables organizations to define business logic (priorities, quotas, preemption) without modifying applications

Limitations:

- Complex implementation: Requires sophisticated scheduling algorithms, state management and integration with cluster schedulers
- Policy definition overhead: Misconfigured quotas or priorities can lead to inefficiency or unfairness while policies need ongoing review
- Scheduling latency: Computing optimal allocation decisions can introduce delays, especially in large clusters

Policy-based orchestration is ideal for large multitenant environments such as enterprises with dozens of teams, cloud providers with hundreds of customers or academic institutions with complex allocation requirements. It provides the operational control, fairness and accountability necessary for large-scale shared infrastructure. Combined with hardware isolation (MIG) or container isolation, it creates a complete multitenancy solution.

Enterprise GPU segmentation and multitenancy requirements

True multitenancy, defined as the ability to safely and securely share GPU infrastructure across different teams, projects or customers with different trust levels, requires more than hardware or software partitioning. It requires a comprehensive platform that addresses four key dimensions: isolation, fairness, observability and governance.

Isolation

- Enforced across hardware (NVIDIA MIG), virtualization (vGPU/PCI passthrough), OS (namespaces, cgroups), and network (VPC/VLAN)
- Defense-in-depth is mandatory; no single layer is sufficient

Fairness

- Prevents monopolization of shared GPU capacity
- Fair-share scheduling ensures long-term balanced allocation using quotas and weights
- Enables predictable access and aligns with cost models

Observability

- Full visibility into GPU allocation, usage and workload performance.
- Tracks MIG instance/GPU slice mapping, compute and memory metrics, and cost attribution
- End-to-end instrumentation ensures actionable insights and anomaly detection

Governance

- Includes IAM, resource quotas, network controls, storage and secret isolation, and audit logging
- Ensures secure, policy-driven operations
- MIG + containers alone are not enough; platform-level governance is required for fairness, transparency and compliance

While hardware segmentation (MIG), driver-level segmentation (time-slicing) and OS-level isolation (containers) are necessary, they are not sufficient for multitenancy. A cluster might be MIG-enabled and containerized, but without platform-level orchestration and governance, it still suffers from a lack of fair allocation, centralized governance and observability.

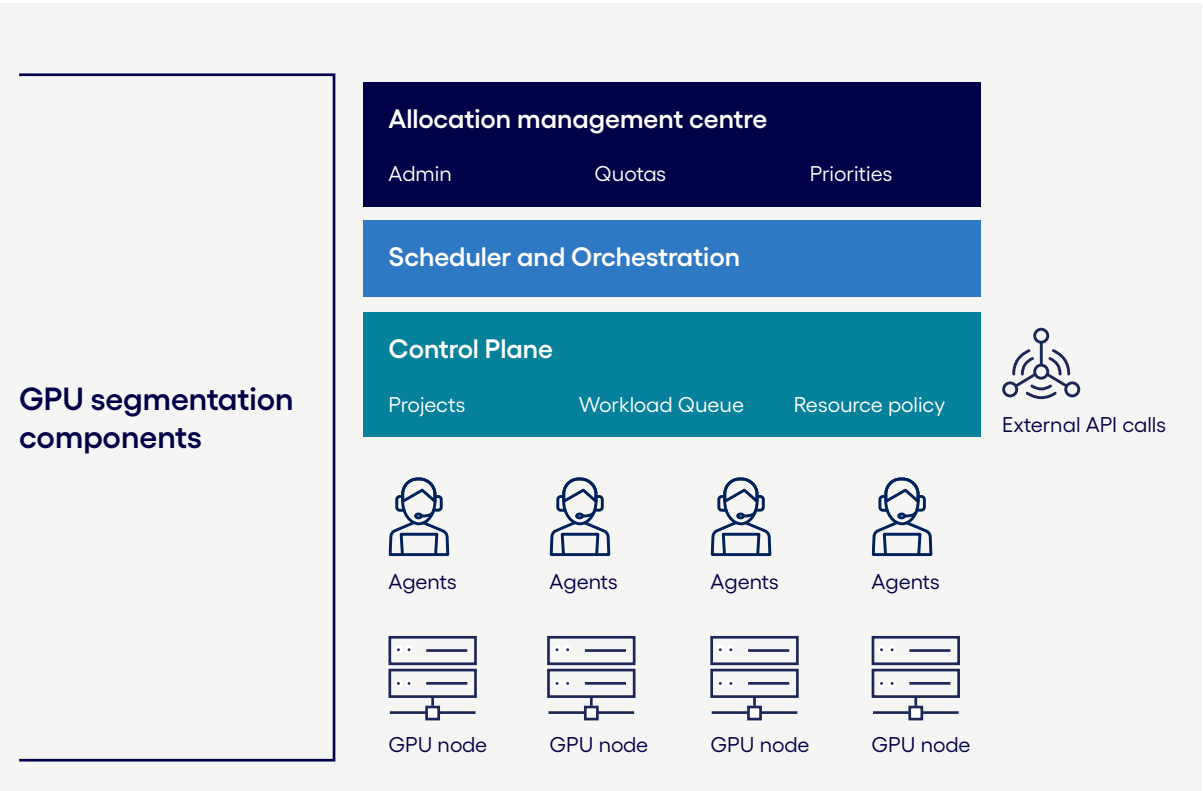
Our approach towards multitenancy

A complete multitenancy solution requires a platform approach that sits above hardware and OS-level isolation mechanisms and provides orchestration, governance, observability and policy enforcement. This is where our approach differentiates, abstracting segmentation complexity and providing a turnkey multitenancy solution.

Cognizant AI Factory approach, which combines platform with OS-level segmentation and time-slicing, to deliver GPU resource optimization and multitenant infrastructure.

Key components of our GPU management include:

- **Control plane:** A centralized service that maintains state about the cluster, teams, projects, resource policies and workload queues. The server exposes APIs for submitting jobs, defining policies and querying cluster status
- **Compute nodes management through agents:** Lightweight agents deployed in the compute environment (on-premises, cloud instances, Kubernetes pods) that listen for jobs from queues and execute them. Agents report resource metrics, utilization and job status back to the server.
- **Scheduler and orchestrator:** The core component that receives job submissions, evaluates cluster state and organizational policies, and makes allocation decisions. It can integrate with Kubernetes or manage bare-metal clusters directly
- **Resource policies and allocation management center:** A control interface where administrators define GPU segmentation strategies, quotas, priorities and preemption policies.



Segmentation and isolation strategy with dynamic fractional GPUs

Dynamic fractional GPUs is implemented through a combination of driver-level memory isolation and policy-based scheduling.

- Driver-level memory monitoring: Agents monitor GPU memory utilization at the driver level, tracking how much memory each workload allocates and enforces preconfigured memory limits per job. If a job approaches its limit, it is either throttled or preempted, preventing out-of-memory conditions.
- Time-slicing integration: Uses NVIDIA GPU time-slicing via Kubernetes to transparently run multiple workloads on the same GPU.
- MIG support (enterprise): Directly manage MIG instances for hardware-level isolation, allocating them to workloads based on resource requirements and policies.
- Policy-based orchestration: Scheduler evaluates job requirements, checks cluster state, applies quotas and priorities, and allocates GPU segments accordingly.

Scheduling, monitoring and fair-share allocation

Our platform approach uses a queue-based scheduling system to manage GPU resources efficiently across multiple teams and workloads. Administrators define resource queues that represent specific classes of resources such as high-memory GPUs or inference-only nodes. Each queue is configured with quotas, priorities and GPU preferences, ensuring predictable access to resources while supporting organizational policies. Jobs submitted by users or automated workflows are placed into these queues.

The scheduler continuously monitors cluster state, comparing each queue's current GPU allocation against its configured quota. When resources are available, they are allocated to queues that are under their quota, ensuring fairness across teams. High priority jobs are given preference.

The scheduler aims to minimize fragmentation and optimize throughput by balancing workloads across GPU types and nodes. By considering job requirements, queue priorities and available capacity, the scheduler ensures that resources are allocated intelligently.

Tenant isolation, access control and resource governance

Our platform approach enforces multitenancy through a layered approach that combines access control, resource isolation and policy enforcement to ensure security, fairness and compliance in shared GPU environments.

Role-based access control (RBAC) forms the foundation of governance. Administrative roles such as cluster admins, team leads and users are assigned distinct permissions, ensuring that individuals can only access projects and data within their designated scope. This prevents unauthorized access and maintains strict boundaries between teams.

We also enforce per-tenant resource quotas, defining limits on concurrent jobs, total GPU hours and maximum GPUs per job. Jobs exceeding these quotas are either queued or rejected, preventing resource monopolization. Additionally, network and storage isolation is implemented via Kubernetes network policies and scoped storage buckets, ensuring that tenants cannot access each other's traffic or data.

Finally, built-in audit and compliance logging provides full transparency. Every resource allocation, job submission and policy enforcement action is recorded, enabling teams to track usage and verify fairness. This governance framework not only strengthens security but also supports compliance with organizational and regulatory standards, delivering a robust platform solution for enterprise-scale multitenancy.

Business impact and outcomes

Reduced infrastructure costs

GPU segmentation and multitenant allocation deliver significant financial benefits by improving utilization and reducing hardware requirements. For example, increasing utilization from 40% to 75% effectively multiplies available capacity by 1.875x without additional GPUs. This translates into substantial savings: fewer GPUs purchased (up to $1.6M for 40 GPUs at $30K–$40K each), lower power consumption, reduced cooling and facility costs, and minimized depreciation over a five-year lifecycle. For large enterprises operating multiple clusters, these savings scale dramatically, freeing capital for innovation and reducing monthly cloud spend by up to 40%.

Improved utilization and operational visibility

Segmentation enhances resource efficiency while providing real-time visibility into GPU usage. Platforms such as ClearML offer dashboards that track memory, compute utilization and power draw at workload, team and cluster levels. This transparency enables accurate cost attribution and chargeback models, incentivizing teams to optimize usage. Additionally, SLA monitoring ensures that critical workloads meet performance targets, with alerts for latency or contention issues, improving reliability and accountability.

Accelerated model development and experimentation

By reducing queue delays and enabling fractional GPU allocation, segmentation accelerates AI development cycles. Jobs that previously waited days for dedicated GPUs can now start within hours, enabling rapid iteration and parallel experimentation. Teams can run multiple hyperparameter sweeps concurrently, significantly shortening time-to-market. Self-service resource provisioning further reduces operational friction, allowing researchers to focus on innovation rather than infrastructure bottlenecks.

Scalability across teams and projects

GPU segmentation decouples infrastructure growth from organizational expansion. A cluster that once supported 10 dedicated teams can now accommodate 20–30 teams through shared resources and fair scheduling. Flexible allocation adapts to fluctuating demand, while gradual scaling allows new teams to start small and grow incrementally. This approach ensures capital efficiency and operational agility, making it ideal for enterprises planning large-scale AI initiatives.

GPU segmentation and multitenancy for a scalable AI edge

GPU segmentation, implemented through hardware partitioning (MIG), driver-level time-slicing, container orchestration and intelligent scheduling, represents a fundamental shift in how enterprises can manage GPU infrastructure. Rather than allocating entire GPUs to individual teams or projects, segmentation enables safe, fair and efficient sharing of expensive GPU hardware.

True multitenancy requires integration across these techniques and a platform layer that provides orchestration, governance, observability and policy enforcement. Without such a platform, the operational complexity of managing segmented GPUs becomes overwhelming.

Our platform approach demonstrates how a comprehensive MLOps platform can abstract this complexity and deliver measurable benefits of GPU, including utilization improvements of 30%–40%, capital cost reductions, faster model development cycles and the ability to scale AI initiatives across teams without proportional infrastructure investment.

For AI leaders, the strategic imperative remains clear in a GPU-constrained environment where every percentage of utilization improvement translates to millions of dollars of capital savings. GPU segmentation and intelligent orchestration are no longer optional. They are core competencies for organizations competing in AI-driven markets.

The transition requires thoughtful planning on selecting segmentation approaches that match workload characteristics and isolation requirements—implementing fair-share policies that respect organizational structure and instrumenting the infrastructure for complete observability. But the return on this investment is substantial—infrastructure that scales with organizational growth, teams that develop AI applications faster and capital that is deployed toward innovation rather than idle compute.

Organizations that prioritize GPU segmentation and multitenancy establish a competitive advantage in deploying and scaling AI applications at lower cost and faster velocity than competitors still managing GPU infrastructure through dedicated allocation and manual intervention.

References

- NVIDIA. Multi-Instance GPU (MIG) Programming and Deployment Guide
- NVIDIA. GPU Operator and Time-Slicing Configuration Documentation
- Kubernetes Authors. Kubernetes GPU Resource Management and Device Plugin Architecture
- NVIDIA. Cloud Partner Reference Architecture for Multi-Tenant Clouds.
- ClearML. MLOps Platform Deployment Guides.
- Fair-Share Scheduling for Deep Learning Workloads

Authors

Arun Kumar, arun.skumar@cognizant.com

Hemant Patade, Hemant.Patade@cognizant.com

Kshitij Chaudhary, kshitij.chaudhary@cognizant.com

Cognizant (Nasdaq-100: CTSH) engineers modern businesses. We help our clients modernize technology, reimagine processes and transform experiences so they can stay ahead in our fast-changing world. Together, we're improving everyday life. See how at www.cognizant.com or follow us **@Cognizant.**

World Headquarters
300 Frank W. Burr Blvd.
Suite 36, 6th Floor
Teaneck, NJ 07666 USA
Phone: +1 201 801 0233

European Headquarters
280 Bishopsgate
London
EC2M 4AG
England
Tel: +44 (01) 020 7297 7600

India Corporate Office
Siruseri-Software Technology Park of India (STPI)
SDB Block—Ground Floor North Wing
Plot No H4, SIPCOT IT Park
Chengalpattu District
Chennai 603103, Tamil Nadu
Tel: 1800 208 6999

APAC Headquarters
1 Fusionopolis Link,
Level 5 NEXUS@One-North,
North Tower Singapore 138542
Phone: + 65 6812 4000

© Copyright 2025–2027, Cognizant. All rights reserved. No part of this document may be reproduced, stored in a retrieval system, transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the express written permission of Cognizant. The information contained herein is subject to change without notice. All other trademarks mentioned herein are the property of their respective owners.

WF4413901