Unified vs Discrete Memory in the Age of Confidential AI

Why the CPU↔GPU memory boundary is becoming the hardest part of "secure acceleration" — and what unified/coherent memory can (and can't) fix.

Executive summary

Modern AI workloads are increasingly **memory-bound** (model weights, KV cache, large context windows, retrieval corpora) and increasingly **security-bound** (regulated data, proprietary models, multi-tenant clouds). When you combine those two trends, the "classic" architecture—**CPU DRAM and GPU VRAM as separate banks**—creates a painful triangle:

- **Performance**: you pay for copies and data movement, often across relatively slower interconnects than GPU HBM.
- **Security**: confidential VMs (CVMs) encrypt guest memory, which complicates device DMA and forces "shared buffers" or encryption workarounds.
- **Scalability**: when a single GPU's VRAM is insufficient, you need more GPUs — but passing through many GPUs to one CVM, and attesting them all, becomes operationally complex.

A true **unified memory architecture** (one physical pool shared by CPU and GPU) can dramatically simplify both programming and confidentiality, but it's still uncommon in enterprise datacenters. Apple's SoCs are the mainstream example of a CPU+GPU unified memory model. In enterprise, what's spreading faster is a spectrum of "almost unified" designs: **coherent CPU↔GPU memory models and large coherent memory pools** (NVIDIA Grace Hopper/Blackwell) and **CPU+GPU packages with shared HBM** (AMD MI300A).

---

1) First, clarify the terms: "unified memory" is overloaded

When people say "unified memory," they often mean one of three different things:

A. True unified physical memory (one pool)

CPU + GPU share the **same physical DRAM pool** (one addressable memory fabric). Apple GPUs explicitly follow a unified memory model where CPU and GPU share system memory.

This is the "Apple-style UMA" most people point to.

B. Coherent heterogeneous memory (multiple pools, coherent access)

CPU memory and GPU memory may remain physically distinct (e.g., LPDDR on CPU + HBM on GPU), but the platform provides a **coherent memory model** and a **single address space** where the GPU can directly

access CPU memory at high bandwidth. NVIDIA's Grace Hopper and Grace Blackwell platforms are explicitly positioned around a CPU+GPU coherent memory model enabled by NVLink-C2C.

C. Unified virtual memory (a programming model)

Some ecosystems use "unified memory" to mean a software model where the runtime migrates pages between CPU and GPU for you (e.g., "unified memory" in CUDA). This can be hugely helpful, but it does **not** necessarily mean the hardware has a single physical pool.

This article focuses on **A** and **B** because they change the security + scalability story for confidential computing most dramatically.

---

2) Discrete memory architecture: the default CPU DRAM + GPU VRAM model

What it is

- CPU has its own DRAM (DDR5/LPDDR, etc.)
- GPU has its own VRAM (HBM/GDDR)
- Data moves between them via PCIe (or PCIe + GPU interconnects like NVLink)

This is still the most common design for:

- high-end gaming/workstations
- datacenter AI training clusters
- most cloud GPU instances

Why it became dominant

GPU VRAM is engineered for bandwidth, not just capacity. HBM delivers enormous throughput to feed tensor cores and large matrix math. Discrete GPUs also preserve modularity: upgrade GPU without changing CPU.

Pros (discrete CPU+GPU memory)

- **Peak GPU bandwidth**: dedicated HBM/GDDR is purpose-built for GPU workloads.
- **Modularity**: mix-and-match CPUs/GPUs; swap components independently.
- **Scale-out-friendly**: attach many GPUs across multiple hosts; well-established ecosystem.
- **Isolation**: it's easier to conceptually treat GPU as a separate trust domain (sometimes desirable).

Cons (discrete CPU+GPU memory)

- **Explicit data movement**: performance often bottlenecks on host↔device transfers.
- **Memory duplication**: weights/data may be stored in both CPU DRAM and GPU VRAM.
- **GPU memory ceiling**: VRAM per GPU is finite, forcing multi-GPU sharding for large models.

- **Complexity under encryption/TEEs**: confidential computing makes the memory boundary much more expensive (next section).

---

3) Unified memory architecture: why it's attractive, and why it's not everywhere

True unified physical memory (Apple-style UMA)

Apple's unified memory model removes the CPU↔GPU copy boundary because CPU and GPU share system memory.

Benefits:

- **Zero-copy by default**: fewer redundant copies of weights/tensors.
- **Better utilization of total RAM**: no "VRAM stranded" problem.
- **Simpler programming and orchestration**: fewer explicit transfers.

Costs / constraints:

- **Bandwidth contention**: CPU and GPU share the same pool; you can get interference.
- **Packaging and upgrade tradeoffs**: unified memory is typically tightly coupled to the SoC; you often can't upgrade VRAM independently.
- **Ecosystem lock-in**: many enterprise AI stacks target NVIDIA CUDA + datacenter GPUs.

Coherent CPU↔GPU memory (enterprise "almost unified")

NVIDIA's Grace Hopper architecture emphasizes that NVLink-C2C enables the GPU to access Grace CPU memory directly at high bandwidth and even "oversubscribe" GPU memory by using CPU memory. NVIDIA's GB200 systems similarly emphasize large coherent memory pools at rack scale.

This is not a single DRAM pool, but it *can feel like one* from the perspective of addressability and coherence.

---

4) Confidential computing makes "discrete memory" hurt (a lot)

Confidential computing's core promise is **data-in-use protection**: guest memory is encrypted and protected from a potentially malicious hypervisor and other tenants.

The key friction point: devices and DMA

Confidential VM designs generally distinguish between:

- **Private memory**: encrypted/protected pages
- **Shared memory**: pages intentionally exposed to the host or devices for I/O

This shows up explicitly in both AMD and Intel designs:

- For AMD memory encryption (SEV), DMA operations in a guest must be performed on shared memory.
- Intel TDX base behavior treats PCIe devices as untrusted and does **not** permit devices to access TD private memory directly; I/O must go through shared buffers.

That's the fundamental reason discrete CPU↔GPU memory becomes so painful in CVMs: **a GPU is a PCIe device that wants to DMA into guest memory**.

What this forces in practice

When a CVM must keep tensors/weights confidential, but the GPU can't DMA into private memory, you end up with combinations of:

- **Bounce buffers** (copy data into shared pages for device I/O, then copy back)
- **Software encryption layers** on I/O pathways
- **Hardware link encryption and trusted-device protocols** (IDE/TDISP), when supported

Each of these adds cost and complexity.

---

5) The "re-encryption wall" between CPU and GPU memory banks

In a discrete architecture, confidential computing introduces a recurring pattern:

1. Data exists in **CPU-encrypted** guest DRAM (private pages).
2. GPU can't safely read it via DMA unless:
3. the page is shared (weakens confidentiality), or
4. there is a trusted I/O path (device attested + protected link + allowed access)
5. Data is moved into GPU memory (which may itself be encrypted by the GPU in "confidential GPU" modes)

This is why you see both **bandwidth bottlenecks** and what feels like **re-encryption overhead** between memory banks.

A concrete example from the NVIDIA ecosystem: the ACM Communications article on confidential GPUs notes that the H100 GPU includes a DMA engine supporting AES-GCM-256 for transfers between CPU and GPU. That's a powerful mitigation, but it's still additional machinery in the path that wouldn't exist in a single-pool UMA design.

---

6) “Unified memory” is especially compelling for confidential AI

If you could design a CVM where CPU and GPU share *one* encrypted memory pool (or a coherently protected pool), the confidential computing story becomes cleaner:

- **No shared-page gymnastics** for host↔device DMA
- **Less copying** (or copying becomes an optimization, not a requirement)
- **A simpler TEE boundary**: fewer transitions between “CPU private memory” and “device-visible shared buffers”
- **Better scaling of memory capacity** without instantly demanding more GPUs

This is the architectural *intuition* behind why unified/coherent memory shines for confidential AI.

But the catch is equally important:

> To get the full benefit, you need the GPU to be inside the trust story (attested, isolated, and protected in use) — otherwise “unified” just means “one big place to leak data.”

That’s why confidential GPU capabilities and trusted I/O protocols matter as much as memory topology.

---

7) Current enterprise reality: confidential GPU is possible, but multi-GPU CVMs are constrained

NVIDIA confidential GPU: real support, real limits

NVIDIA’s own Secure AI whitepaper describes “Multiple GPU Pass-Through” modes, but also calls out sharp constraints:

- Protected PCIe in Hopper is **exclusively supported** in specific HGX Hopper 8-GPU / 4-NVSwitch systems, and all eight GPUs are passed through to a single CVM; CPU↔GPU traffic uses bounce buffers for encryption, and (in that mode) NVLink traffic between GPUs may remain unencrypted.
- Another mode encrypts the NVLink pathway too and still supports up to **eight** GPUs passed through to a CVM, again involving software encryption with bounce buffers (with IDE/TDISP as an alternative path in certain Blackwell/CPU configurations).

This aligns closely with the user pain points:

- encryption + bounce buffers drive bandwidth overhead,
- scaling a single CVM beyond “one node’s worth” becomes hard,
- and the supported topology list is narrow.

NVIDIA also publicly announced Confidential Computing general availability on H100 and describes its use in virtualized environments.

Cloud offerings show the “scalability cliff”

In practice, some confidential GPU offerings are still limited. For example, an Azure Q&A thread (July 2025) notes that *only a single NVIDIA H100 NVL GPU* was being offered per node in a specific region with confidential computing enabled, and asks about multi-GPU options.

Meanwhile, Google Cloud’s A3 High family explicitly offers H100 machine types with **1, 2, 4, and 8 GPUs** in general availability for non-confidential use. And Canonical announced Ubuntu Confidential VMs available on Google Cloud A3 with NVIDIA H100 GPUs (confidential AI architecture combining SEV-SNP + H100, encrypted PCIe, and attestation).

The practical takeaway: **multi-GPU is common in general**, but **multi-GPU + CVM** is still rolling out unevenly and often under special constraints.

---

8) Why “many GPUs per one CVM” becomes operationally brutal

Even ignoring security, multi-GPU passthrough has classic headaches: IOMMU grouping, VFIO quirks, reset behavior, and the VM needing enough PCIe slots.

Confidential computing adds several layers:

A. PCIe topology limits show up fast in VMs

A PCIe bus has finite addressing space. QEMU’s own PCIe documentation notes that a PCI Express root bus supports up to **32 devices**. Enterprise virtualization docs describe the same 32-slot-per-bus structure and how bridges expand it (theoretical scaling), but real VMs consume slots for “built-in” devices too.

In other words: yes, you *can* build a big virtual PCIe hierarchy, but it’s increasingly brittle as you add devices (GPUs, NVSwitch/NVLink bridges, NICs, DPUs, storage controllers, etc.).

B. Devices are untrusted by default in TEEs

Intel’s TDX Connect materials explicitly say that by default PCIe devices are untrusted and cannot access TD private memory directly; I/O uses shared buffers. Linux kernel documentation for TDX and SEV describes the practical reality: shared vs private pages, and shared memory used for DMA allocations (e.g., via SWIOTLB in TDX).

So with **each additional GPU**, you multiply:

- the number of DMA channels that must be made safe,
- the number of shared buffer paths (or trusted DMA paths) to manage,
- and the blast radius of mistakes (one misconfigured shared region can leak).

C. Attestation scales poorly when "everything is a TCB member"

Confidential computing fundamentally relies on **attestation**. Canonical's description of confidential AI explicitly calls out CPU and GPU attestation as a pillar.

For confidential GPUs, attestation and secure sessions commonly use protocols like SPDM. NVIDIA staff have described SPDM sessions being established between the GPU and the driver in a CVM.

When you go from 1 GPU → 8 GPUs, you don't just multiply hardware — you multiply:

- certificate management and verification
- measurement and policy checks
- failure modes (one GPU fails attestation → what happens to the whole CVM?)

This is one reason "confidential + many GPUs" quickly turns into "secure, but complex."

---

9) NVLink vs PCIe: why interconnect matters (especially for memory pooling)

PCIe: ubiquitous, but not built for GPU memory pooling

PCIe is the universal attachment fabric, but multi-GPU memory access over PCIe is not what it's optimized for. It's also where TEE-related device security (IDE/TDISP) must operate to protect in-flight data between components.

NVLink/NVSwitch: built for GPU↔GPU bandwidth and pooled memory domains

NVIDIA positions NVLink Switch as a rack-level switch capable of supporting up to **576 fully connected GPUs** in a non-blocking fabric and enabling a large NVLink domain.

NVIDIA also describes rack-scale systems like GB200 NVL72, where 72 Blackwell GPUs form a large NVLink domain intended to behave like a single massive GPU at rack scale.

Confidential twist: you must secure the GPU↔GPU fabric too

The Secure AI whitepaper explicitly calls out cases where CPU↔GPU traffic is protected via bounce buffers while GPU↔GPU NVLink traffic may be unencrypted in some modes, and describes an alternative mode where NVLink is also encrypted (still up to eight GPUs per CVM).

So NVLink can solve the *performance* side of scaling, but confidential computing requires explicit answers for the *security* side of that fabric.

---

10) What's available today (late 2025): unified/coherent memory in consumer vs enterprise

Consumer / end-user market

1) Apple Silicon (true unified memory model)
Apple GPUs use a unified memory model with CPU and GPU sharing system memory.

2) Integrated GPUs on x86 (UMA, but not "datacenter UMA")
Intel documentation notes that integrated graphics uses Unified Memory Architecture (UMA) where the GPU uses system memory. This is unified *in the integrated-GPU sense*, but it doesn't deliver datacenter-grade GPU compute, VRAM bandwidth, or ecosystem parity with H100-class systems.

3) Discrete GPUs remain dominant for high-end AI on PCs
Workstations still largely rely on discrete GPU VRAM because bandwidth and compute density matter.

Enterprise market

1) AMD Instinct MI300A (CPU+GPU with unified HBM)
AMD's MI300A documentation explicitly describes CPU and GPU sharing a unified address space and a unified HBM3 memory capacity (128GB) per socket.

This is one of the clearest "enterprise-class" examples of CPU+GPU moving closer to true unified memory.

2) NVIDIA Grace Hopper / Grace Blackwell (coherent CPU↔GPU memory models)
NVIDIA describes Grace Hopper as delivering a CPU+GPU coherent memory model via NVLink-C2C, including the ability for the GPU to access Grace CPU memory at high bandwidth and oversubscribe GPU memory. NVIDIA's GB200 systems extend this to rack-scale coherent systems (NVL72).

3) Intel: integrated UMA exists, but enterprise CPU+GPU unified-memory products are not broadly shipping
Intel has strong integrated-GPU UMA on the client side, but on the datacenter accelerator side it remains mostly discrete (e.g., Gaudi 3 with HBM). Intel's Falcon Shores, which was often discussed as a future hybrid, was reported as not coming to market (kept as an internal test vehicle), with attention shifting to a successor.

4) CXL memory expansion/pooling: a different "unification" path
CXL aims to extend coherent memory concepts beyond a single package. Industry and academic discussions describe CXL-attached memory expansion/pooling as a way to treat device memory as part of a larger memory pool (with important caveats about latency/coherence).

This matters because a lot of "unified memory" pain is fundamentally "I need more memory close to acceleration."

---

11) Pros & cons comparison (with confidential computing as the lens)

Dimension	Unified / coherent memory (UMA / coherent CPU↔GPU)	Discrete CPU DRAM + GPU VRAM
Data movement	Fewer explicit copies; can reduce host↔device transfer overhead (especially if coherent access exists).	Frequent host↔device copies; can become the dominant bottleneck for large models and streaming pipelines.
Total "usable" memory for models	Can feel like a single large pool (Apple UMA; or GPU can access CPU memory in coherent platforms).	VRAM is a hard ceiling per GPU; exceeding it pushes you toward multi-GPU sharding or host paging.
Peak GPU bandwidth	Often lower than pure HBM-per-GPU designs if you rely on system DRAM; coherent designs still keep HBM for GPU hot data.	Best-in-class bandwidth (HBM) tightly coupled to GPU.
Upgrade / modularity	Typically lower (SoC/package coupling).	High: GPUs can be swapped, scaled, or upgraded independently.
Confidential computing simplicity	Potentially much simpler: fewer shared-buffer paths and fewer "private vs shared" gymnastics if the accelerator is inside the TEE story.	Harder: TEEs treat PCIe devices as untrusted by default, requiring shared buffers or trusted I/O; encryption/ bounce buffers add overhead.
Multi-GPU scaling	Coherent memory helps reduce "VRAM cliff," but large jobs still need many GPUs; orchestration depends heavily on fabric (NVLink/NVSwitch).	Mature multi-GPU ecosystem, but CVM passthrough and attestation complexity grows quickly with GPU count.

12) The current "passthrough restrictions" story in one picture

Discrete world (today): secure acceleration is a patchwork

- CPU TEE protects **guest DRAM** (private pages).
- GPU is a PCIe device; by default it can only DMA through **shared buffers** in many TEE models.
- To regain confidentiality + performance, you add:
- GPU confidential computing mode (GPU memory encryption + GPU attestation)
- protected links / trusted device protocols (IDE/TDISP/TDX Connect / SEV-TIO)
- and you still often pay for bounce buffers in some modes.

Unified/coherent world (emerging): fewer boundaries, but higher integration

- If CPU+GPU share memory (or have coherent access), the system can avoid a big class of CPU↔GPU transfer and page-sharing issues.

- But you increasingly depend on **platform-level integration** (firmware, drivers, attestation, link encryption) being correct end-to-end.

---

13) Practical implications for architects designing confidential AI systems

If you must run AI inside CVMs today

- Prefer **platforms explicitly designed for confidential GPUs** rather than DIY passthrough wherever possible (because support matrices can be very specific).
- Treat device I/O as a first-class design constraint: TDX and SEV models make shared buffers fundamental for many devices.
- Expect **multi-GPU to be harder than you think**: vendor docs show real caps (e.g., 8 GPUs in certain protected passthrough modes), and cloud availability may lag.

If your pain is "VRAM is too small"

- Coherent CPU↔GPU designs (Grace Hopper/Blackwell) are explicitly positioned to let the GPU access large CPU memory pools at high bandwidth to reduce the "VRAM cliff."
- CPU+GPU packages with shared HBM (MI300A) remove a lot of host↔device transfer overhead in principle by placing CPU and GPU around a unified memory system.

If your pain is "I need many GPUs per CVM"

- Check three independent ceilings: 1) **Security ceiling**: what's supported for confidential multi-GPU pass-through and how traffic is protected. 2) **Virtual PCIe ceiling**: QEMU/PCIe topology and device count (root bus 32 devices; bridges expand but add complexity). 3) **Operational ceiling**: attestation for each GPU, certificate lifecycle, and failure handling.

---

14) Where this is headed: trusted I/O + coherent memory + rack-scale fabrics

Several trends are converging:

- **Trusted device + link protection standards**: PCIe IDE and TDISP are explicitly aimed at securing device interfaces in TEE contexts.
- **CPU TEEs evolving toward trusted I/O**: Intel TDX Connect and AMD SEV-TIO describe models for bringing devices into the trust boundary and protecting data paths.
- **Coherent memory pools**: Grace Hopper/Blackwell emphasize coherent memory models; CXL aims to expand memory beyond the CPU socket.
- **Rack-scale GPU fabrics**: NVLink Switch fabric scales toward hundreds of GPUs in a single NVLink domain.

If these mature together, the industry moves toward something that *behaves like* unified memory at datacenter scale—without requiring every workload to live on a consumer SoC.

Closing thought

The “unified vs discrete memory” debate used to be mostly about **performance and programmability**. Confidential computing turns it into a three-way systems problem:

- **performance**
- **security guarantees**
- **operational scalability**

Discrete memory architectures can absolutely support confidential AI — but today they often do so with **tight topology constraints, bounce buffers, and complex attestation plumbing**. Unified/coherent memory architectures reduce the number of boundaries you need to cross, which is why they’re so appealing — but enterprise-grade availability is still emerging and tied to a few platform families rather than being the default everywhere.