Unified vs Discrete Memory for Confidential AI

Why the "simple" act of giving a Confidential VM a GPU still turns into bounce buffers, bandwidth walls, and a lot of attestation plumbing — and how unified/coherent memory architectures change the game.

TL;DR

Modern AI stacks want three things at once:

1. **A lot of memory** (weights + KV cache + activations),
2. **A lot of bandwidth** (tokens/sec is largely a memory game), and
3. **A strong TEE story** (confidential compute / CVMs / regulated data).

Discrete CPU+GPU memory architectures are great at (1) and (2) when you're *not* confidential — but in confidential computing they introduce an extra tax: **moving data between two protected memory domains** (CPU DRAM and GPU HBM/VRAM) while maintaining end-to-end confidentiality.

Unified/coherent memory architectures (true UMA SoCs, and "near-UMA" superchips like CPU+GPU packages with coherent address space) reduce copies, reduce encryption boundary crossings, and can dramatically simplify the security + performance story — **but** they're not yet broadly available across mainstream enterprise deployments in the same way Apple's UMA is ubiquitous for consumer devices.

---

1) Terminology: "Unified Memory" means different things

People say "unified memory" and can mean three distinct things:

A. Hardware Unified Memory Architecture (UMA)

- **One physical memory pool** shared by CPU and GPU.
- Often on a **single SoC package**.
- Typically includes **cache coherency** or a coherent interconnect.

Examples: Apple Silicon Macs/iPads; integrated GPUs on many laptops/desktops.

B. Coherent CPU↔GPU memory systems (near-UMA superchips)

- CPU and GPU may still have distinct physical memory types (e.g., LPDDR + HBM),
- But present a **single coherent address space** with *very* high bandwidth between CPU and GPU.

Examples: NVIDIA Grace Hopper / Grace Blackwell class systems; AMD MI300A-style CPU+GPU packages.

C. Software unified memory (unified virtual address space + page migration)

- A programming/runtime model: one pointer/address space, but data may migrate between CPU and GPU memory.
- Great for productivity, but performance and security depend heavily on the platform.

Examples: CUDA “Unified Memory” / UVM; Intel oneAPI USM.

This article focuses on **A and B** as “unified/coherent memory architectures,” because they change the *hardware boundary* story that matters most for confidential computing.

---

2) Discrete memory architecture (CPU DRAM + GPU VRAM): why it dominates

The architecture

A typical datacenter AI node today looks like:

- **CPU** with large DRAM (and sometimes extra memory tiers),
- **Discrete GPU(s)** with their own high-bandwidth memory (HBM/GDDR),
- An interconnect between them: **PCIe**, and/or **NVLink / Infinity Fabric**.

Why it’s great (outside TEEs)

- **Peak bandwidth**: GPU HBM is designed for massive throughput.
- **Independent scaling**: add more GPUs to add more compute and VRAM.
- **Vendor ecosystem maturity**: CUDA, ROCm, drivers, debuggers, schedulers.
- **Operational flexibility**: swap GPUs, upgrade GPUs, multi-node fabrics.

The core downside

Discrete memory means **data duplication and movement**: - Host pre/post-processing in CPU memory, - Model compute in GPU memory, - Explicit copies or page migration between the two.

Outside confidential computing, this is an engineering problem. Inside confidential computing, it becomes a **security and architecture problem**.

---

3) Hardware UMA / coherent memory: what changes

The big difference

If CPU and GPU share **one coherent memory domain**, then:

- The workload can (in principle) keep data **in one place**,

- The security boundary can be (in principle) **one encryption domain**,
- The system can avoid "copy → re-encrypt → DMA → decrypt" loops.

Why Apple feels "magical" for local AI

On Apple Silicon, the CPU and GPU share a unified memory pool. This is why a single machine can sometimes run models that would otherwise require careful multi-GPU choreography: you're not fighting the DRAM↔VRAM boundary every time you touch tensors.

That said, consumer UMA is not automatically "enterprise confidential compute." A strong cloud CVM story also needs: verifiable attestation, tenant isolation, secure I/O, fleet management, and support for multi-tenant scheduling.

---

4) A practical comparison

Dimension	Discrete CPU+GPU memory	UMA / coherent CPU↔GPU memory
Data movement	Frequent CPU↔GPU transfers	Often reduced or eliminated
Programming model	Explicit copies, pinned buffers, stream mgmt	Feels simpler; fewer copies
Peak GPU bandwidth	Excellent (HBM)	Can be excellent (HBM) but shared/ contended
Capacity scaling	Add GPUs to add VRAM	Capacity fixed by package (unless multi-package fabric)
Cost / upgrade path	Modular	More integrated; upgrades are "whole platform"
Confidential compute friction	High (two protected domains + I/O)	Potentially lower (fewer boundary crossings)
Multi-GPU scaling	Strong in bare metal; complex in CVMs	Depends on fabric; can be excellent in integrated rack-scale systems

---

5) The confidential computing pain: why CPU+GPU separation hurts

Confidential computing wants: **data is encrypted in use**, and a remote party can **attest** the environment.

When you introduce a discrete GPU, you get two fundamental issues:

5.1 Two different "places data can live"

- **CPU private memory** (encrypted with the CVM's keying model),

- **GPU protected memory** (if the GPU is in confidential mode, it has its own protection/attestation model).

To do useful work, tensors must cross this boundary.

5.2 “Untrusted device” I/O model → shared buffers and bounce buffers

Many confidential VM designs assume that **PCIe devices are untrusted by default** and cannot directly DMA into “private” guest memory.

So the common pattern becomes:

1. App holds plaintext in private guest memory.
2. Data is copied into a **shared** buffer (accessible to the host / devices).
3. Data is **encrypted/authenticated** for the GPU.
4. GPU copies it into its protected memory and decrypts/validates.

This is where you pay: - **Extra copies**, - **Extra encryption work**, - **Extra synchronization**, - And most importantly: **a bandwidth wall** if your workload is transfer-heavy.

5.3 Why “memory re-encryption between banks” becomes a real bottleneck

In a normal VM, staging through host memory is already overhead. In a CVM, you also must preserve confidentiality across: - Hypervisor-visible shared memory, - Device DMA, - GPU memory.

Even if compute overhead is small, the *end-to-end* pipeline can be dominated by transfers.

5.4 “TEE memory migration” gets awkward

Many modern AI runtimes rely on: - Unified virtual addressing, - Paging, - Memory oversubscription, - Runtime migration between CPU and GPU.

But confidential computing changes what is legal: - Pages may need to be explicitly marked **private vs shared**. - Devices may be forbidden from touching private pages. - The hypervisor becomes a participant in page-state transitions.

Result: features that are easy in non-confidential settings become constrained, slower, or operationally brittle.

---

6) The scaling cliff: multi-GPU inside a CVM is harder than it looks

Running a big model securely usually pushes you toward **multiple GPUs** (capacity) and/or **multiple GPUs with fast interconnect** (bandwidth, tensor parallelism, KV cache sharding).

In a confidential VM environment, this collides with several constraints.

6.1 Passthrough policy constraints (the “1 GPU per CVM” reality)

Even when confidential GPU support exists, many stacks start with a conservative security envelope:

- **Single GPU passthrough per confidential VM** is a common early limitation.
- GPU partitioning (e.g., MIG) may be supported, but that’s still “one physical GPU device.”

This creates a blunt scalability problem: - If your workload needs 4–8 GPUs for capacity, but the platform supports 1 GPU/CVM, you’re forced into **distributed designs** (more nodes, more orchestration, more attack surface, more cost).

6.2 KVM/QEMU + VFIO: PCIe topology and practical limits

On paper you *can* attach many PCIe devices to a VM. In practice, multi-GPU passthrough gets messy because:

- A PCIe root bus has a limited number of device slots; scaling requires bridges/root ports.
- Each passthrough device must be isolated cleanly via **IOMMU groups**.
- Some platforms require ACS/ARI features to avoid grouping multiple devices together.

Even if you can model hundreds of devices in theory, the operational complexity of doing “8–16 GPUs per VM” with clean isolation is non-trivial.

6.3 PCIe bandwidth and the “host staging” problem

If the TEE model forces staging through shared memory, your effective CPU↔GPU bandwidth is constrained by:

- PCIe bandwidth,
- Additional copies,
- Crypto overhead,
- Synchronization.

This is exactly the kind of pipeline UMA tries to avoid.

6.4 NVLink advantage: fewer PCIe limits, more of a single system

NVLink/NVSwitch fabrics can connect large GPU groups into a single high-speed domain. In the extreme, rack-scale systems can present dozens of GPUs as one tightly coupled accelerator domain.

This matters for two reasons:

1. **Performance:** tensor parallel collectives and memory sharing are dramatically faster.
2. **Systems design:** you can reduce the “PCIe device explosion” problem.

However, in confidential settings you still have to solve: - Attesting each device, - Establishing secure channels, - Managing keys and trust across the fabric, - Ensuring the hypervisor cannot spoof or reroute traffic.

7) Multi-GPU attestation: the “trust graph” gets complicated fast

Confidential AI isn’t just “attest the CPU.” It becomes:

- Attest the **CPU TEE** (CVM),
- Attest the **GPU TEE mode**,
- Bind them via a secure channel,
- Ensure the **driver stack** is consistent with the attested measurements,
- Repeat for *N* GPUs,
- Potentially attest and secure **interconnect components** (switches, firmware, NICs), depending on threat model.

As you scale GPU count, the trust graph grows: - More devices, - More firmware, - More keys, - More failure modes.

This is why early deployments often prefer “one GPU per CVM” and scale out rather than scale up.

---

8) Current market reality (consumer vs enterprise)

8.1 Consumer: UMA is common — but not in the way datacenters want

- **Apple Silicon**: hardware UMA, high bandwidth, strong on-device security model.
- **Intel/AMD integrated GPUs**: share system memory (UMA-ish), but usually not aimed at massive datacenter AI.
- **Discrete GPUs**: still dominate for peak performance and gaming/AI workstations.

Takeaway: consumers see UMA daily; enterprises still mostly run discrete GPU stacks.

8.2 Enterprise: coherent superchips exist, but adoption is uneven

You can now buy platforms where CPU and GPU are tightly coupled with coherent memory behavior, but:

- They are **premium**,
- Often targeted at **HPC/AI supercomputing**,
- Require new server designs and supply chains,
- And may not fit every enterprise procurement model.

8.3 Intel: software-first “unified memory,” hardware remains mostly discrete

Intel’s “unified shared memory” terminology largely refers to programming models and runtime abstractions. In datacenter accelerators, Intel’s current direction has emphasized discrete accelerators and platform integration, rather than shipping a mainstream CPU+GPU UMA superchip in broad volume.

---

9) Why unified/coherent memory architectures shine for confidential AI

If your goal is **Confidential AI with fewer sharp edges**, UMA/coherent memory helps in four big ways.

9.1 Fewer boundary crossings → fewer places to leak

Every time data crosses: - Private guest memory → shared buffer, - CPU → device DMA, - device → GPU protected memory,

you expand your attack surface.

UMA reduces the number of explicit crossings.

9.2 Less bounce-buffer plumbing

If CPU and GPU can operate over the same protected memory domain (or a coherently secured domain), the system can reduce: - extra copies, - staging buffers, - and encryption handoffs.

9.3 Better latency predictability

One of the worst properties of staging + migration is variability: - page faults, - staged copies, - contention in shared buffers.

Coherent memory can make the performance profile smoother.

9.4 A clearer story for “whole workload is confidential”

Enterprises want a simple claim:

> “My prompts, weights, intermediates, and outputs were protected end-to-end — and I can prove it.”

When CPU and GPU are separate memory banks, you have to prove that across two TEEs, two key hierarchies, and a protected transport.

UMA-like designs can collapse that into a more unified story (assuming vendors extend attestation and protection across the whole package).

---

10) The honest downsides of UMA / coherent memory

UMA is not a silver bullet.

10.1 Memory contention

A single memory pool means: - CPU and GPU contend for bandwidth, - One noisy neighbor can affect another.

10.2 Capacity is “what you bought”

With discrete GPUs you can: - Add more VRAM by adding GPUs.

With UMA, you’re bounded by: - The package’s memory capacity, - And the vendor’s SKU choices.

10.3 Upgradability and fleet homogeneity

UMA platforms tend to be more integrated, which can be operationally harder: - Replace/upgrade is often an entire node refresh.

10.4 Ecosystem and tooling maturity

Apple UMA is great, but the enterprise ecosystem for large-scale deployment (drivers, schedulers, observability, multi-tenant guarantees, regulated compliance) has historically been richer in NVIDIA-dGPU-centric stacks.

10.5 Confidential I/O is still the hard problem

Even with unified memory, a cloud-grade confidential story must handle: - Secure networking, - Secure storage, - Secure device assignment, - Side-channel risk, - Firmware supply chain.

UMA helps the *CPU↔GPU memory* problem, but not the entire confidential platform story.

---

11) What’s actually available today (a grounded snapshot)

Below is a reality-based snapshot of “unified/coherent memory” options and “confidential GPU” options that exist in practice today.

11.1 Hardware UMA (mainstream)

- **Apple Silicon**: consumer/pro workstation UMA with large unified memory configurations.

11.2 Coherent CPU↔GPU superchips (enterprise)

- **AMD MI300A-class** designs: CPU+GPU in one package with unified memory address space.
- **NVIDIA Grace Hopper / Grace Blackwell-class** designs: coherent CPU↔GPU coupling with extremely high bandwidth interconnect and large combined memory domains.

11.3 Confidential GPU in the cloud (selected offerings)

- Some hyperscalers provide **confidential VMs with NVIDIA H100-class GPUs**, typically with explicit constraints (often single-GPU configurations, special provisioning models, and a specific CPU TEE like Intel TDX).

The key point: confidential GPU exists, but it's still early enough that **scale-up** (many GPUs per CVM) is not "the default path."

---

12) Design patterns that work *right now* (discrete GPUs + CVMs)

If you must ship confidential AI today on discrete GPUs, the winning patterns are the ones that minimize boundary crossings.

Pattern 1: Keep large tensors resident on the GPU

- Treat CPU memory as control-plane + small buffers.
- Avoid repeated host↔device transfers.

Pattern 2: Fuse preprocessing/postprocessing where possible

- Push tokenization, sampling, light postprocessing onto the GPU.
- Reduce round-trips.

Pattern 3: Use GPU partitioning instead of multi-GPU CVMs (when supported)

- If the platform supports secure MIG-like partitioning, it can be operationally simpler than multi-GPU passthrough inside a CVM.

Pattern 4: Accept scale-out, but make it "CVM-native"

If you can't have 8 GPUs in one CVM: - Run many single-GPU CVMs, - Use a secure, attested service mesh, - Treat orchestration as part of your TCB design.

Pattern 5: Make attestation a first-class primitive

- Don't treat attestation as "startup ceremony."
- Integrate it into scheduling, key release, and continuous monitoring.

---

13) Where the industry is heading

13.1 Trusted I/O (TDISP / PCIe IDE / "secure VFIO")

The industry is converging on the idea that: - Devices need a standard way to be *trusted* by a TEE, - So they can DMA into protected memory safely, - Without exposing sensitive traffic to the host.

This is the missing link that would make discrete accelerators much more TEE-friendly.

13.2 More coherent CPU↔GPU platforms

We're likely to see continued push toward: - CPU+GPU superchips, - Coherent fabrics, - Rack-scale GPU domains.

13.3 Memory pooling and composability (CXL era)

Even if we don't get "one monolithic UMA everywhere," we're moving toward: - More coherent memory pooling, - More composable accelerators, - More standard security primitives for devices.

Conclusion

Discrete CPU+GPU memory architectures are the default because they scale compute and VRAM beautifully — but **confidential computing exposes their weakest point**: every transfer between CPU memory and GPU memory becomes both a performance and security event.

Unified/coherent memory architectures reduce the number of those events. That's why UMA feels so good on consumer Apple Silicon — and why coherent enterprise superchips are strategically important for the future of confidential AI.

In the meantime, the practical reality is:

- Confidential GPU is real,
- But **multi-GPU confidential passthrough is still constrained**,
- And the industry is actively building the missing pieces (trusted I/O, secure VFIO, standardized device attestation) that will make "CVM + many GPUs" feel as natural as it does on bare metal.