Contribution ID: **346** Type: **not specified**

"Parallel Paths to High-Bandwidth Memory for ML/AI: Specific Purpose Memory vs. Driver-Managed Approaches"

Background and Motivation

High-bandwidth memory (HBM) has become a critical resource for modern machine-learning and AI workloads, offering orders-of-magnitude improvements in bandwidth and latency compared to traditional DDR DRAM. As HBM adoption grows—whether on GPU accelerators like AMD's MI200 series or NVIDIA's Grace Hopper/Blackwell architectures—platform firmware and operating-system drivers must evolve to present these new memory types effectively to applications. Two parallel streams have emerged for surfacing HBM to ML/AI developers:

1. Specific Purpose Memory (SPM), as defined in UEFI 2.8 (EFI_MEMORY_SP), which marks regions of firmware-reserved memory for special use and allows the operating system to treat them distinctly; and
2. Driver-Managed RAM, wherein device drivers (e.g., CUDA on NVIDIA) hot-plug HBM into the system's RAM pool as a managed NUMA domain, exposing it transparently to malloc() and other allocators.

This paper compares these two paradigms, explores their interdependencies, and highlights how implementation details—particularly Intel's DAX integration—create some issues for the SPM approach.

Specific Purpose Memory (SPM) Overview

With UEFI 2.8, the EFI_MEMORY_SP attribute enables firmware to tag physical memory regions as Specific Purpose Memory. Unlike generic reservations, SPM communicates performance intent—such as low latency or high bandwidth—to the OS, allowing finer-grained allocation strategies.

• Firmware Tagging: During early boot, UEFI reports SPM regions via the EFI memory map.
• Kernel Interpretation: On x86, efi_init() parses this map, converts EFI_MEMORY_SP into the E820_TYPE_SOFT_RESERVED type, and invokes memblock_reserve() with the IORES_DESC_SOFT_RESERVED flag.
• Driver Claiming: With CONFIG_EFI_SOFT_RESERVE enabled, Linux's DAX (device-dax) and HMEM drivers automatically claim SPM regions. At runtime, the efi=nosoftreserve parameter or daxctl tools can override this "soft reservation."

Common uses of SPM include:

• DAX/HMEM integration for persistent memory (Intel Optane DC, CXL-attached memory)
• CXL hot-plug recommendations (EFI_CONVENTIONAL_MEMORY + EFI_MEMORY_SP)
• AMD GPU's A+A (Accelerator + Addressable) use of HBM

Driver-Managed HBM Exposure

Parallel to SPM, some vendors bypass firmware reservation entirely and manage HBM purely in driver space:

• NVIDIA CUDA on Grace Hopper/Blackwell:
○ The CUDA driver invokes add_memory_driver_managed(), hot-plugs HBM as system RAM.
○ Exposed as a new NUMA domain with IORESOURCE_SYSRAM_DRIVER_MANAGED.
○ Absent from /sys/firmware/memmap, ensuring no conflict with firmware lists.
○ Standard allocators (malloc(), numa_alloc_onnode()) can target HBM directly.

This approach treats HBM just like another RAM device but under driver control, streamlining integration with existing memory allocators and avoiding BIOS dependencies.

Interdependencies and Implementation Details

1. Kernel Configurations
○ CONFIG_EFI_SOFT_RESERVE: Enables SPM reservation; implies CONFIG_DEV_DAX_HMEM.
○ HMEM Driver (v5.5+): Interfaces with EFI/ACPI to classify memory by performance tier; crucial for Intel Optane and CXL memory.

2. Memory Types in Linux
○ EFI_MEMORY_SP: Firmware-provided hint.
○ E820_TYPE_SOFT_RESERVED: Kernel's interpretation of SPM.
○ MEMORY_DEVICE_COHERENT: Zone-DEVICE memory type for coherent device memory (e.g., HBM on AMD).
○ IORESOURCE_SYSRAM_DRIVER_MANAGED: Marks driver-added memory (e.g., NVIDIA HBM) as system RAM.
3. AMD GPU's Use of SPM
○ AMD marks HBM as SPM, then KFD maps it as MEMORY_DEVICE_COHERENT for heterogeneous memory management (HMM/SVM).
○ Because AMD HBM is neither hot-pluggable nor a NUMA domain, SPM seemed the simplest route—though arguably unnecessary for HMM coherence.
4. NVIDIA's Driver-Managed Choice
By hot-plugging HBM as a driver-managed NUMA domain, CUDA avoids BIOS requirements and unlocks standard allocation paths.

Conclusion

As ML/AI workloads continue to push the boundaries of memory bandwidth and latency, platform architects must balance flexibility, performance, and ease of use. The UEFI-driven SPM model and driver-managed hot-plug approach both offer valuable paths to HBM integration, but each has trade-offs in firmware dependency, allocator compatibility, and runtime configurability. By refining firmware tagging and kernel arbitration—particularly in light of Intel's DAX-centric approach—we can forge a more cohesive memory-exposure paradigm that meets the diverse demands of next-generation AI systems.

Primary author: Mr BHARDWAJ, Rajneesh (AMD)

Co-author: Mr BLINZER, Paul (AMD)

Presenter: Mr BHARDWAJ, Rajneesh (AMD)

Session Classification: Device and Specific Purpose Memory MC

Track Classification: Device and Specific Purpose Memory MC