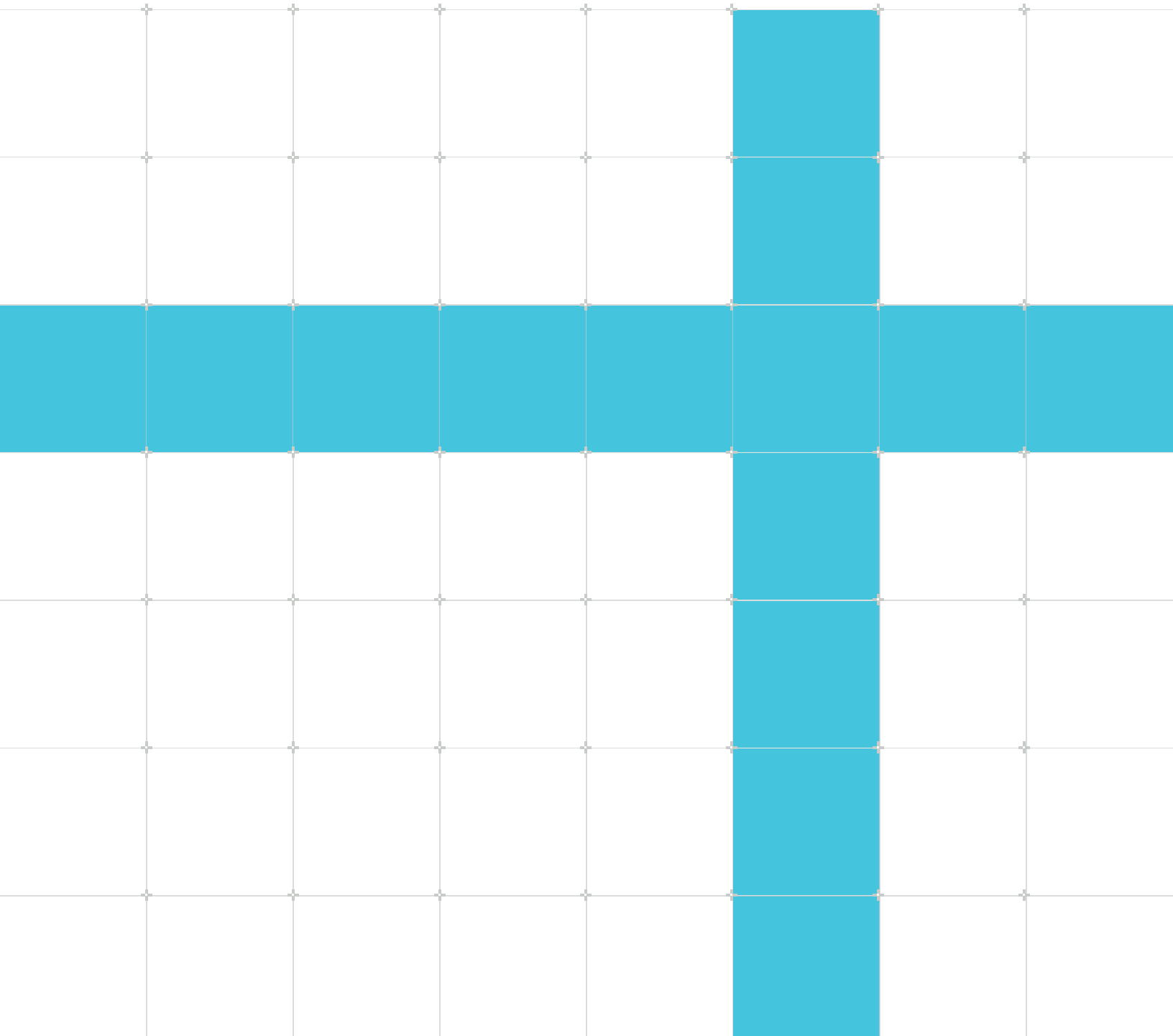
arm

Learn the architecture - Memory System Resource Partitioning and Monitoring (MPAM) Overview

Version 1.0

Issue 01
107768_0100_01_en



Learn the architecture - Memory System Resource Partitioning and Monitoring (MPAM) Overview

Release information

Document history

Issue	Date	Confidentiality	Change
0100-01	27 February 2023	Non-Confidential	Initial release

Proprietary Notice

and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm® welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. Overview

This guide introduces the Memory System Resource Partitioning and Monitoring (MPAM), an optional addition to the Arm architecture to support memory system partitioning. MPAM is documented in the Memory System Resource Partitioning and Monitoring (MPAM), for A-profile architecture Arm Architecture Reference Manual Supplement.

1.1 Before you begin

This guide assumes that you are familiar with Arm Exception levels, memory management, and Security states. If you are unfamiliar with any of the previous topics, read the following guides before continuing with this guide:

- AArch64 exception model: Introduces the Exception and privilege model in AArch64.
- AArch64 memory management: Introduces the MMU, which controls virtual to physical address translation.

2. Background

Many of today's computing needs are satisfied by shared-memory computer systems, where multiple applications, or multiple virtual machines (VMs) run concurrently. Shared-memory compute systems enable applications or multiple virtual machines to simultaneously share memory resources. Sharing memory resources reduce design costs and design footprints, enable higher performance, and greatly reduce redundant data copies.

In an ideal world, each shared application or virtual machine would utilize an equal or proportionate amount of the shared resources within the system. By doing so, an inherent level of fairness is present within the system, ensuring that applications or virtual machines do not disproportionately utilize the system resources. This could otherwise impact bandwidth and performance of the neighboring shared applications or virtual machines. However, in the real world, each application or virtual machine competes for resources, interfering with each other.

Memory-system resource partitioning and usage monitoring (MPAM) helps mitigate this effect by providing memory resource partitioning and associated monitoring features. This guide will introduce MPAM, its concepts, use cases, and terminology.

3. Arm Memory System Resource Partitioning and Monitoring (MPAM) Extension

Many aspects of a shared-memory compute system are inherently managed as part of the fundamental operation of the OS or hypervisor. For example, the scheduler performs CPU time management, and the memory allocator manages memory allocation.

Memory System Resource Partitioning and Monitoring (MPAM) extends the ability for software to co-manage runtime resource allocation of memory system components such as caches, interconnects, and memory controllers. These are resources that otherwise would not be controllable by software at this granularity.

Such resource control requires runtime reaction and continuous tracking. These requirements are due to the interchanging of requests from different PEs, Security states, and Exception levels that could be running different VMs or applications.

MPAM achieves this by enabling supervisory software such as an OS or Hypervisor to assign a unique partition identifier to instruction accesses and data memory accesses for each VM or application. These uniquely assigned partition identifiers accompany the memory accesses throughout their lifetime in the memory system. Memory system components use partition identifiers to configure the allocation of resources to a particular VM or application.

MPAM has two functions: Memory-system resource partitioning and Memory-system resource usage monitoring. These functions provide software with the capabilities to control and monitor how the allocation of resource is divided between VMs or applications. Software can then provide an appropriate level of responsiveness, fairness, or purposeful unfairness.

3.1 Memory-system resource partitioning

The performance of programs running on a computer system can be affected by the memory-system performance. Memory system performance can be affected by several shared resources within the memory-system. These may include caches, interconnects, and memory controllers.

In a memory system shared by multiple VMs, OSs, and applications, the resources available to each software environment may vary, depending on which other programs are also running. This is because those other programs may consume more or less of an uncontrolled memory-system resource.

Memory-system resource partitioning provides controls on the limits and use of previously uncontrolled memory-system resources.

One example of memory-system resource partitioning would be the partitioning of cache, such that allocations into the cache from a particular VM, OS, or application are limited to a subset of the total cache resource.

3.2 Memory-system resource usage monitoring

A memory-system resource usage monitor measures the level of resource usage or accumulates resource usage events, depending on the type of resource. Here are some examples of how software could utilize memory-system resource usage monitoring:

- Dynamically utilizing data gathered by the memory-system resource usage monitor to influence the level of memory-system resource partitioning for a particular VM, OS, or application.
- Logging data gathered by the memory-system resource usage monitor for applications, VMs, or systems for offline analysis.
- Statically utilizing data gathered by the memory-system resource usage monitor to determine appropriate resource partitioning.
- Monitoring to detect anomalous behavior and to apply limits to misbehaving software.

4. Why use MPAM?

Memory-system resource partitioning and usage monitoring can reduce memory-system interference. This section describes some example scenarios.

4.1 Independent software sharing a memory system

With faster processors, it is often less expensive to integrate into a single computer system the functions previously performed by two or more systems. If any of these previously separate systems was real-time or otherwise performance-sensitive, it may be necessary to isolate the performance of that function from others in the integrated system.

Memory system performance can be monitored, and the measured usage can guide optimization of system partitioning.

Partitioning is often statically determined by the system developer. Partitions may be given non-shared resource allocations to reduce the indeterminism caused by shared resources interference. The number of partitions required could be small, similar to the number of previously separate systems.

4.2 Service-level provisioning in multi-tenant VM servers

MPAM can be used to reduce the “noisy neighbor” effect on a shared tenancy virtual machine server, so that no tenant should be able to over-consume resources. This allows a software-controlled, hardware-enforced implementation of a mechanism for isolation for fairness. The noisy neighbor effect occurs when an application or VM uses more of the available resources than it was intended to, often leading to performance issues for others on the shared infrastructure.

When a server runs multiple VMs for different users, it is necessary to prevent one VM from using more resource than allowed by its Service Level Agreement (SLA).

MPAM partitions provide a means to regulate the memory-system resources used by a VM. While there need only be a few service levels provisioned onto a server, each VM needs a separate PARTID so that resource-usage controls can be separately responsive to the resource demands of that VM.

4.3 Foreground and background job optimization

When foreground and background jobs are run on the same system, the foreground job’s response time should not be compromised, and the background job’s throughput should be optimized. The performance of the foreground and background jobs can be monitored, and the resource

allocations can be changed dynamically to track system loading while optimizing foreground response time and background throughput.

An example of this approach is proposed in Heracles: Improving Resource Efficiency at Scale. This paper describes a system that requires only two partitions, one for web-facing applications and another for best-effort applications. The Heracles approach measures the service-level objective of tail latency for web service and adjusts the division of resources between the two partitions. Resource-usage monitoring is also used to tune resource allocation for particular resources.

5. How does MPAM work?

This section describes key concepts that are required to understand how MPAM can be used to implement Memory-system resource partitioning and Memory-system resource usage monitoring.

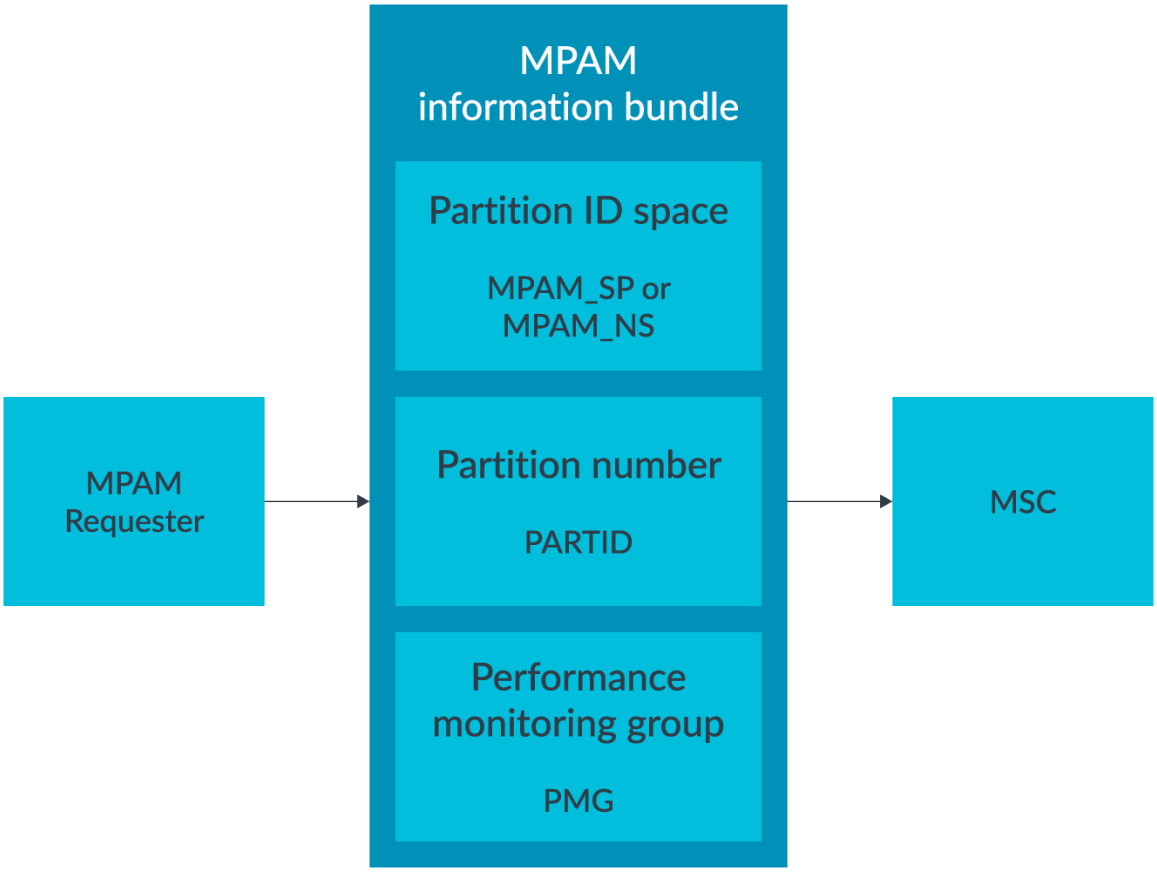
5.1 MPAM Requester

An MPAM Requester is a Requester capable of transmitting an MPAM information bundle. An Arm PE which implements FEAT_MPAM is an MPAM Requester. An example Arm PE that implements FEAT_MPAM is the Cortex-A710.

5.2 MPAM information bundle

MPAM relies on a mechanism for attaching partition identifiers to executing software, such as VMs, OSs, or applications. MPAM implements this by propagating an MPAM information bundle to all downstream Memory-System Component (MSC). An MPAM information bundle consists of a Partition ID Space, a Partition Number (PARTID), and a Performance Monitoring Group (PMG). Together the partition ID space and partition number are used to uniquely identify an MPAM resource partition within an MSC.

Figure 5-1: MPAM information bundle



5.3 Partition number (PARTID)

A partition number (PARTID) references a particular partition within a Partition ID Space. The numerical value of a partition number has no inherent meaning. It is expected that a unique partition number (PARTID) will be assigned to different VMs, OSs, or applications by the relevant PARTID Space Manager.

The architectural maximum width of a PARTID field is 16 bits. This being said, a PE implementation is permitted to support less than the architectural maximum. For example, the Cortex-A710 supports a 6-bit PARTID.

Arm strongly recommends that the PARTID be the same size in all PEs and MSCs in a system. Software can only use the smallest PARTID size supported by the system.

5.4 Partition ID space

MPAM supports multiple partition ID (PARTID) spaces. The Security state of the MPAM Requester determines the partition ID space that is sent within the MPAM information bundle. The Arm architecture defines two Security states:

- Non-secure state
- Secure state

When a PE is executing in a Secure state, the MPAM information bundle indicates the Secure partition ID (PARTID) space (MPAM_NS == `0x0`). When a PE is executing in a Non-secure state, the MPAM information bundle indicates the Non-secure partition ID (PARTID) space (MPAM_NS == `0x1`).

This guide does not cover FEAT_RME and its effects on MPAM. MPAM for RME systems will be discussed in a future guide.

5.5 Performance Monitoring Group (PMG)

The Performance Monitoring Group (PMG) provides an additional filter for an MPAM resource usage monitor. The PMG extends the PARTID monitoring capability and can be used in two ways:

- Monitors can be made sensitive only to PMG, thus achieving grouping: they monitor and aggregate measurements related to all PARTIDs sharing the same PMG value.
- Monitors can be made sensitive to both PARTID and PMG, to achieve finer grain monitoring. For example, 2 VMs could be assigned the same partition of an MSC, each VM would be assigned a unique PARTID and a unique PMG. The MPAM resource usage monitor would allow the monitoring of each VMs usage of the assigned partition.

The architectural maximum width of a PMG field is 8 bits. This being said, a PE implementation is permitted to support less than the architectural maximum. For example, the Cortex-A710 supports a 1-bit PMG.

5.6 PARTID Space Manager

The PARTID Space Manager is a software subsystem running as part the OS, Hypervisor, or Secure Monitor. The function of the PARTID Space Manager is not defined by the MPAM architecture, but is an integral part of any MPAM-aware software stack.

It is expected that a PARTID Space Manager would be responsible for assigning unique partition numbers (PARTID) and PMGs to applications, VMs, or OSs. The MPAM architecture supports multiple partition ID (PARTID) spaces, this means that in most cases, coordination between different PARTID Space Managers is not required.

For example, the PARTID Space Manager in EL3 (Secure Monitor) can assign the partition number (PARTID) `0x55` to a Secure Pprtition and this will be part of the Secure partition ID (PARTID) space. The PARTID Space Manager in Non-secure EL1 (OS) can also assign partition number (PARTID) `0x55` to an application, this will belong to the Non-secure partition ID (PARTID) space. Each partition ID (PARTID) can be allocated to a different partition of an MSC.

5.7 Memory-System Component (MSC)

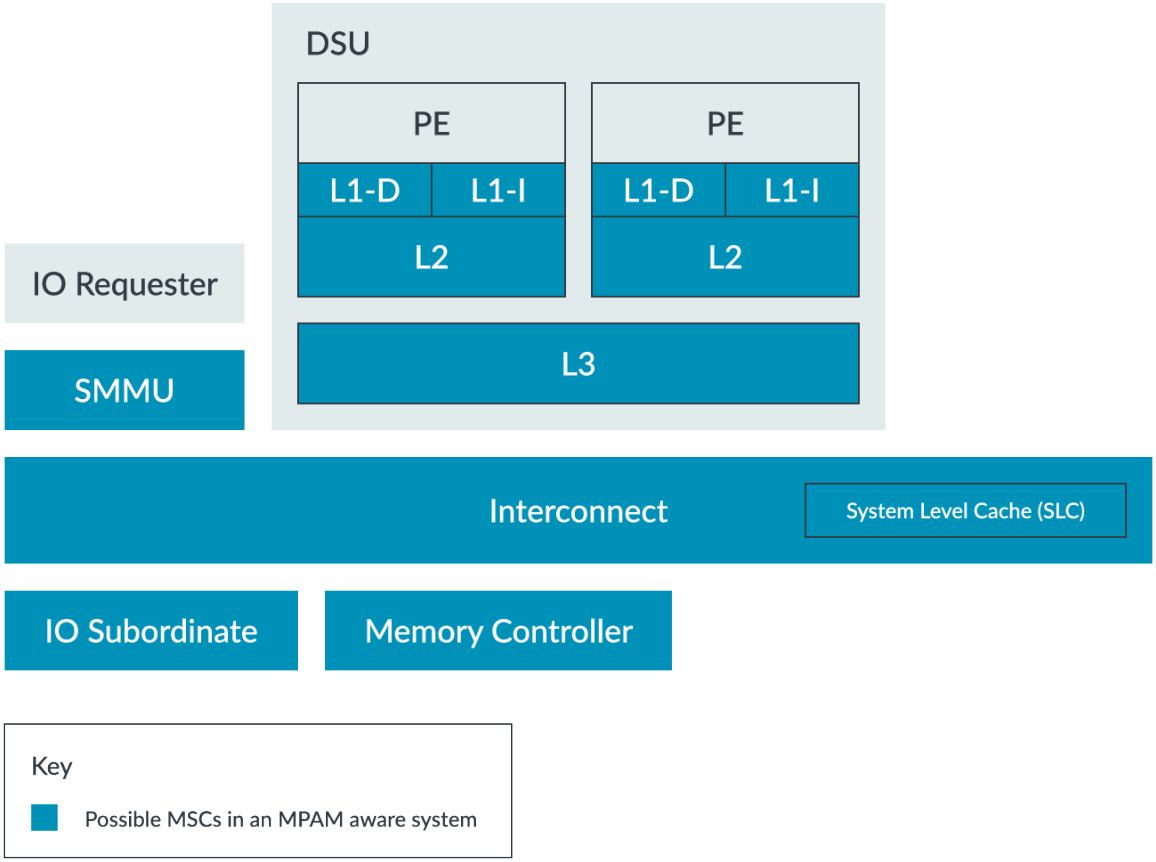
An MPAM-aware system relies on MPAM aware Memory System Components (MSC). MSCs handle memory requests issued by any MPAM Requester. An MSC can have partitionable resources. MSCs can include but are not limited to the following:

- Cache memories
- Interconnects
- System Memory Management Units (SMMU)
- Memory controllers

A PARTID Space Manager configures an MSC to partition its resources. A partitionable resource within the MSC is partitioned according to the programming of the specific resource partitioning controls. For each memory system request, the MSC uses information provided in the MPAM information bundle along with the MSCs resource partitioning settings to determine the amount of partitionable resource a particular VM, or application can utilize.

The following diagram demonstrates a shared memory system containing several MSCs:

Figure 5-2: MSCs



5.8 MPAM resource usage monitor

MPAM resource usage monitors can provide the PARTID Space Manager with measurements of the resource-type usage that can be partitioned by MPAM. Each type of monitor measures the usage by memory transactions of a particular PARTID and optionally, PMG.

5.8.1 Memory-bandwidth usage monitor

A memory-bandwidth usage monitor counts payload bytes meeting specified filter criteria.

5.8.2 Cache-storage usage monitors

A cache-storage usage monitor uses the partition number (PARTID), partition ID space and PMG to filter the cache-storage usage. A cache-storage usage register reports the amount of storage currently present within the cache allocated by the filter for an MPAM resource partition.

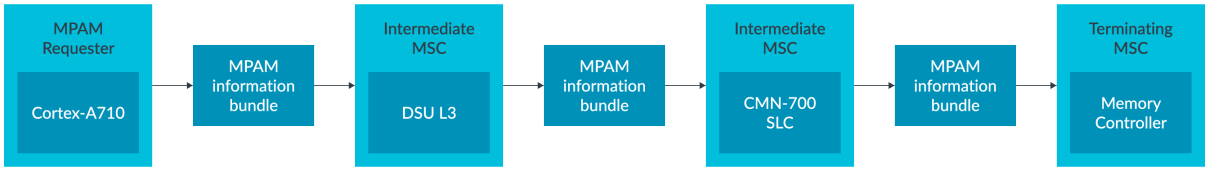
5.9 Propagating MPAM information

The MPAM information bundle needs to be propagated through the memory system from the MPAM Requester to the MSCs. The path through the memory system may consist of one or more MSCs. These MSCs will either be referred to as an intermediate MSC or a terminating MSC.

An intermediate MSC must pass the unaltered MPAM information bundle downstream.

A terminating MSC does not forward the MPAM information bundle from a request. An MPAM aware system requires an interconnect that can support transporting the MPAM information bundle between the MPAM Requesters and the MSCs.

Figure 5-3: Propagation of an MPAM information bundle



6. MPAM Usage Model

It is the responsibility of the PARTID Space Managers to assign VMs and applications to a partition. The hypervisor PARTID Space Manager assigns VMs to partitions, and the Operating System PARTID Space Manager assigns applications to partitions.

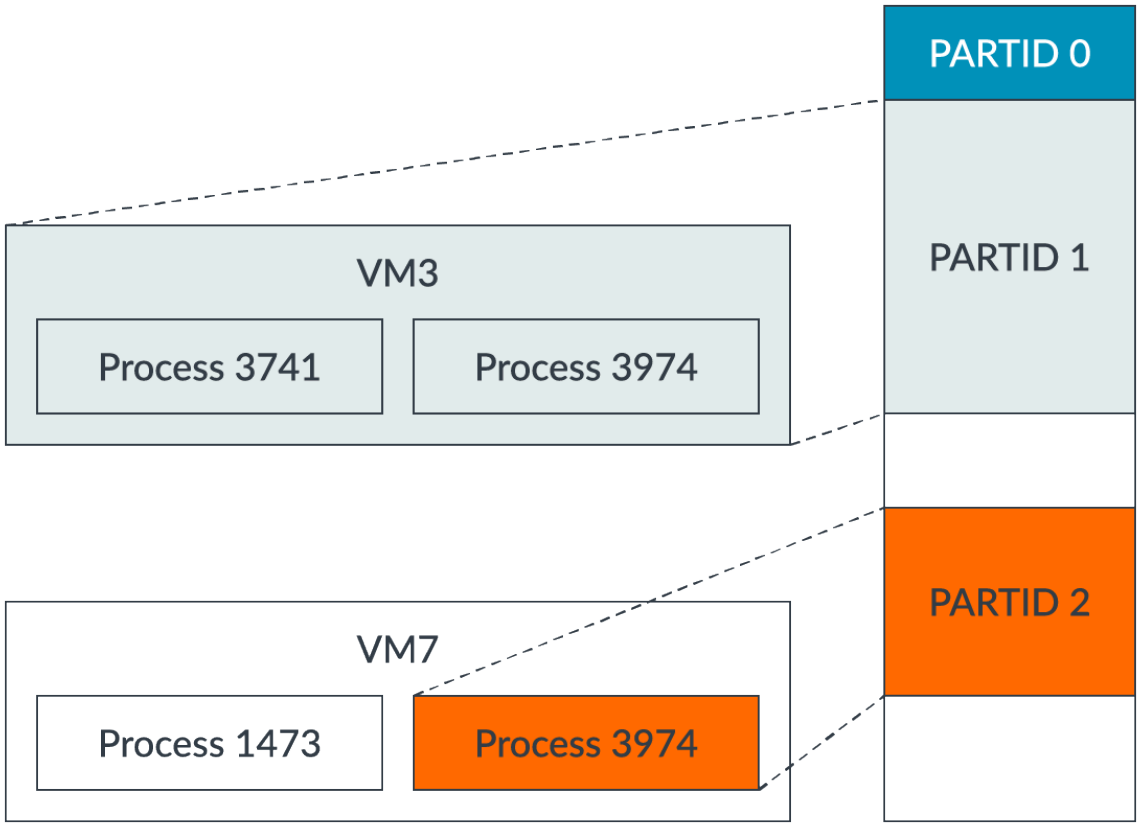
A memory-system partition is associated with a software environment on a PE by loading the corresponding MPAMn_ELx register with PARTID_I and PARTID_D.

Each Exception level has its own MPAMn_ELx register. This means that when taking or returning from an exception, the PE automatically attaches the relevant partition identifiers to memory accesses that are made while executing at that Exception level.

The Hypervisor and OS are responsible for manually saving the context of the relevant MPAMn_ELx registers when switching between VMs or applications.

In an symmetric multiprocessing (SMP) environment, it is common for a VM or application to run across multiple PEs at the same time. In such cases, we would expect the unique partition ID specified in the MPAMn_ELx registers, which are assigned to the VM or application, are consistent across the PEs that software is running on.

Figure 6-1: Partitioning, VMs, and OS processes



Once a memory system request is sent with an MPAM information bundle, it flows through one of more MSCs. MSCs will implement one or more partitioning control interfaces which configure how the resource is partitioned.

6.1 Control interfaces for memory-system resources

The MPAM architecture defines the following standard types of control interfaces for memory-system resources:

- Cache-portion partitioning.
- Cache maximum-capacity partitioning.
- Cache maximum associativity partitioning.
- Memory-bandwidth portion partitioning.
- Memory-bandwidth minimum and maximum partitioning.
- Memory-bandwidth proportional-stride partitioning.

- Priority partitioning.

These control interfaces will be discussed in detail in a future Learn the Architecture MPAM guide.

6.2 Example MPAM Usage

When a PE with MPAM configured issues a memory request, the requests are sent with an MPAM information bundle.

When enabled, the MPAM information bundle is automatically generated by the PE for every request using the current Security state and the MPAM register for the current Exception level. For example, if the current Exception level is EL0, then the MPAM0_EL1 register is used.

The MPAMn_ELx register is part of the context of the software that runs in the ELn. So, an application running at EL0 has its MPAM state in MPAM0_EL1 and that register is saved and restored by the operating system when it context switches between applications.

The MPAM information bundle will be assigned by the associated PARTID Space Manager. It would be expected that the PARTID space manager is not part of this context switching process. Its purpose is to give an unused PARTID from the PARTID space when one is requested by the OS or the hypervisor. This is likely a fairly rare occurrence, for example when starting a new VM.

In the following example, the EL1 OS PARTID Space Manager configures the MPAM information bundle associated with the currently scheduled in application running at Non-secure EL0:

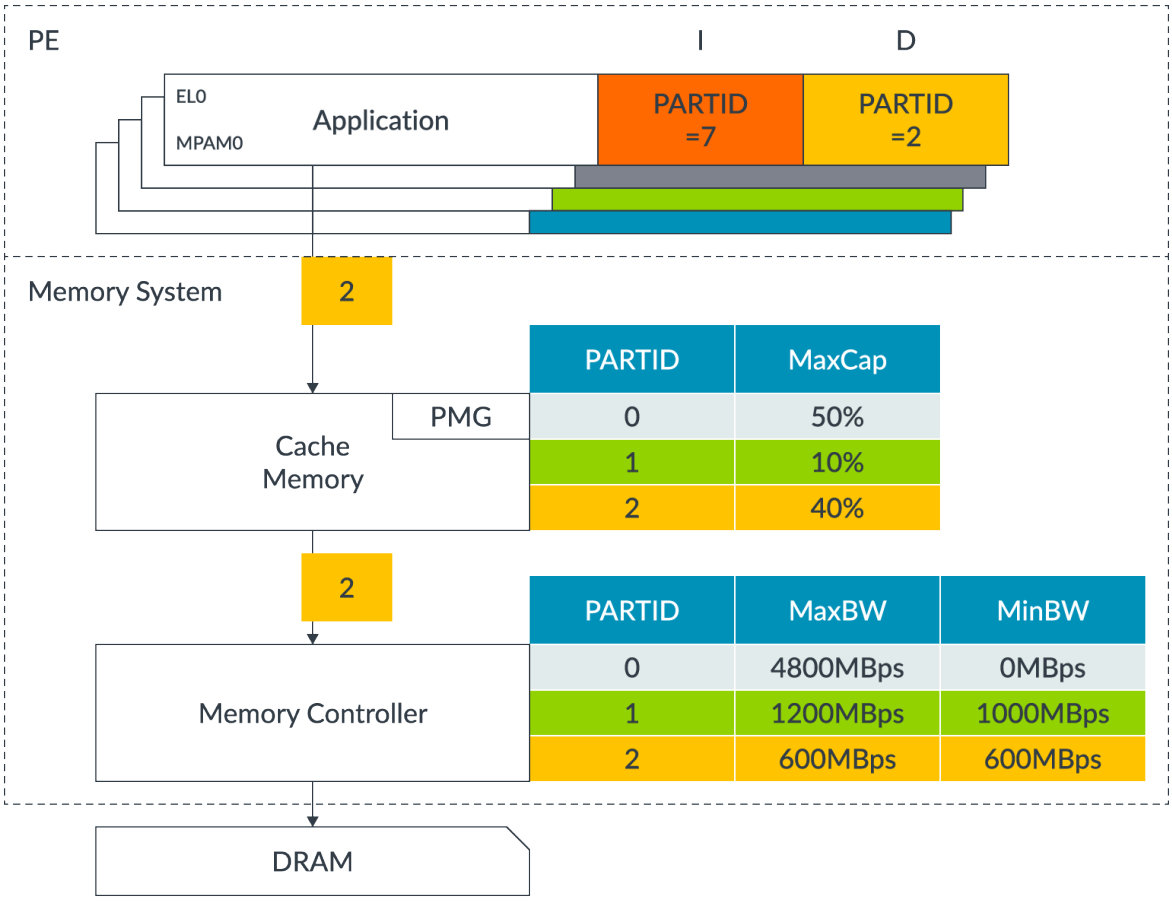
Non-secure EL1 PARTID Space Manager:

```
MRS    x0, xzr
ORR    x0, x0, #(0x2<<16)    // EL1 PARTID_D == 0x2
ORR    x0, x0, #(0x7<<0)     // EL1 PARTID_I == 0x6
MSR    MPAM0_EL1, x0
ISB
```

While executing at EL0, any memory accesses will be sent with the above configured MPAM information bundle.

- MPAM_NS == `0x1`
- PARTID_D == `0x2` / PARTID_I == `0x7`
- PMG_I == `0x0` / PMG_D == `0x0`

Figure 6-2: Example MPAM Usage



Here, the system shows two MSCs, a system cache and a memory controller:

- The system cache uses the MPAM-defined cache maximum-capacity partitioning control interface, which allows the MPAM Manager to specify the amount of cache storage capacity a particular MPAM resource partition may allocate in the cache.
- The memory controller uses the MPAM- defined memory-bandwidth minimum and maximum partitioning control interface, which allows the MPAM Mangaer to specify the amount of bandwidth available to a particular MPAM resource partition.

6.3 Virtualizing MPAM

As previously discussed, it is expected that the different PARTID Space Managers will not coordinate when assigning an MPAM resource partition. To prevent the OS from assigning

an MPAM resource partition already assigned by the Hypervisor, the Hypervisor needs some mechanism to virtualize the MPAM resource partitions assigned by a Guest OS.

The way MPAM handles this situation is to allow the hypervisor to tell the VM's OS through a simulated MPAMIDR_EL1 register that the VM has some small number of PARTIDs that it may use. The small number of PARTIDs are preallocated by the hypervisor's PARTID space manager and stored in the MPAMVPMn_EL2 registers that map virtual PARTIDs to the "real" (we say "physical") PARTIDs preallocated by the hypervisor. The VM's OS can manage the use of the virtual PARTIDs without trapping to the hypervisor.

Without the MPAMVPMn virtual PARTID mapping registers, at EL1 and EL0, all accesses to MPAM1_EL1 and MPAM0_EL1 would need to be trapped.

Accesses from a Guest OS to the configuration registers of all MSCs, and to the System registers that configure the PE MSCs, may be emulated by the host hypervisor. This allows virtual PARTID mapping to be emulated and hypervisor policies governing resource partitioning to be applied. To achieve this, accesses to the MPAM memory mapped registers of the MSC would be trapped using Stage 2 translations, allowing the hypervisor to trap and emulate the MSC access.

Figure 6-3: MPAM Virtualization