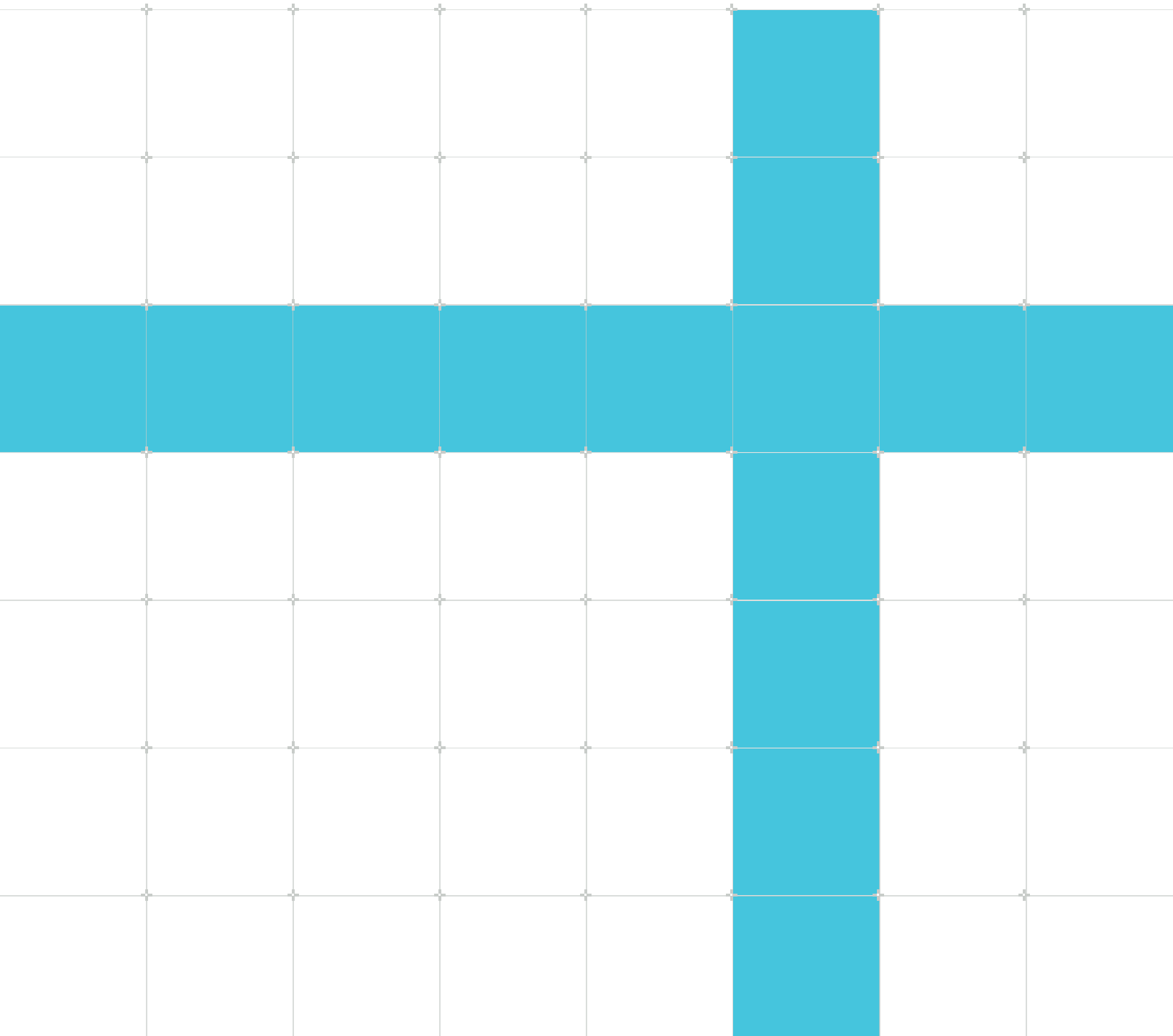
Learn the architecture - Memory System Resource Partitioning and Monitoring (MPAM) Software Guide

Version 1.0

Issue 01
108032_0100_01_en



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

Release information

Document history

Issue	Date	Confidentiality	Change
0100-01	17 July 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 describes the firmware and software that are part of the Memory System Resource Partitioning and Monitoring (MPAM).

MPAM is described 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 model, memory management, and fundamentals of MPAM. If you are unfamiliar with any of the these 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 Memory Management Unit (MMU), which controls virtual to physical address translation.
- Memory System Resource Partitioning and Monitoring (MPAM), for A-profile architecture: Arm Architecture Reference Manual Supplement
- Memory System Resource Partitioning and Monitoring Overview: Introduces the MPAM architecture.

2. Problems of shared resource contention

This section describes an example scenario that highlights the reasons for using MPAM mechanism. Memory System Resource Partitioning and Monitoring Overview introduces the background of contention problems over shared memory-system resources.

In current new processors the number of cores is continuously increasing, especially in large-scale usage models like webservers and datacenters. Increasing cores increases the number of threads or workloads that can be run simultaneously. One of the main economic reasons to run applications in a cloud environment is the ability to distribute the cost of shared resources among multiple customers. Unfortunately, when multi-threaded applications, VMs, or workloads run concurrently they compete for shared resources including Last Level Cache(LLC) or memory bandwidth. Running mixed workloads can pose resource challenges on busy systems.

Physical isolation at the server level is still the way to ensure enough resources for critical workloads today. However this leads to under-utilization of system resource. MPAM enables software to better manage this utilization.

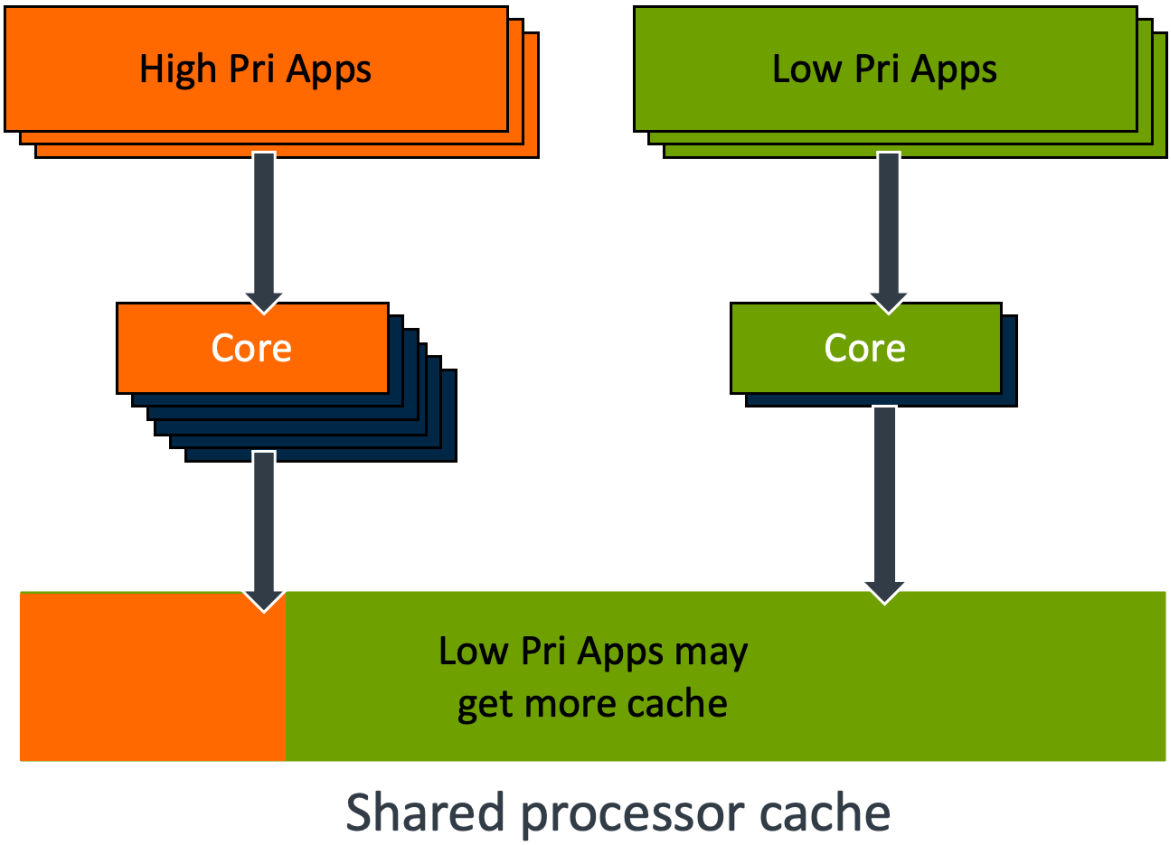
A neighbor task can affect another task's cache performance by using more of the cache than it needs. This cache contention reduces the performance or responsiveness of more important applications. If a task consumes more cache than is best for the system-level goals, it can even affect the overall performance of the entire system. This is referred to as a "noisy neighbor" problem.

2.1 Noisy neighbor

The noisy neighbor effect is when a shared resources multi-tenant systems where one Virtual Machine (VM) or applications within the VM consume more resources than it should have on a physical machine. This can lead to an unwanted performance degradation for other tenants sharing the same physical machine.

The following figure shows an example of Noisy neighbor problem.

Figure 2-1: Noisy-neighbor



For example, in the figure above, processor cores can run many applications in parallel. As a result, the contents of the shared LLC can quickly become overwritten with new data as it is requested from memory by other cores. Also, high-priority applications need to be able to use the shared LLC but they might be blocked by the lower priority applications with high LLC usage.

The noisy neighbor, green in the figure consumes excessive LLC. This is usually allocated on an LRU, fair share, or more smart re-reference prediction cache replacement algorithms. Generally, an application that makes many accesses to memory in a complex pattern with frequent reuse of the cache lines allocates a lot of cache. In this case, the higher-priority application, orange in the figure, might not get optimal cache occupancy so degrading the performance.

The shared LLC has been a performance bottleneck in multi-core systems for a long time.

2.2 How do we resolve this type of problem?

Software mitigation for noisy neighbor problem

- Reduce low-priority tasks CPU time to throttle memory access

- CPU pinning and isolation for high-priority tasks
- Cache coloring memory allocation. Cache coloring is a clever software-only approach to cache partitioning

Although system administrators can control over CPU time, system memory, network bandwidth, or combinations of these resources on modern Operating System or Hypervisor. However existing mechanisms are too coarse. System performance would benefit from additional control mechanisms at different levels of the memory hierarchy such as:

- Shared cache partitioning
- Bandwidth control
- Resource utilization profiling, applications memory resource monitoring

We need more fine-grained control over previously uncontrolled shared memory-system resources to improve system stability, performance, and resource utilization.

2.3 MPAM

The features of MPAM Extensions for A-profile address the challenge of using shared system resources including CPU caches, Last Level Cache, System Level Cache, memory bandwidth and IOMMU resources.

The functionality of MPAM includes both system resource partitioning and system resource usage monitoring. These technologies enable tracking and controlling shared resources such as LLC and main memory (DRAM) bandwidth, used by multiple applications, containers, or VMs running concurrently on the platform. MPAM can help detect noisy neighbors and reduce performance interference. MPAM can also enable system administrators to improve resource efficiency to maintain the performance of critical workloads in complex environments.

There are several usage scenario where MPAM can be beneficial:

- Increase hardware utilization, make hardware less expensive
- Minimize CPU latency
- Meet real-time task deadlines

2.3.1 Main functionality

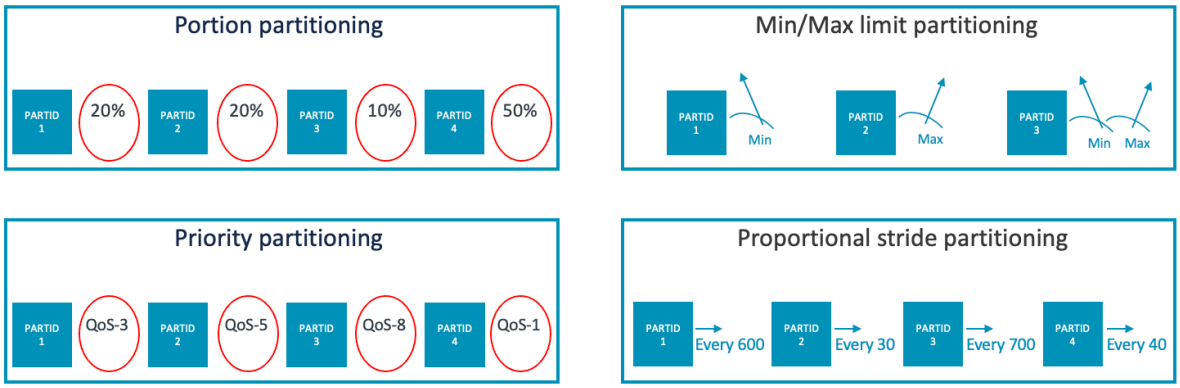
The MPAM architecture defines standard partitioning control interfaces. Each of these standard control interfaces is optional at each Memory-System Component (MSC). An MSC can implement several controls or none.

- MPAM Cache control:
  - Cache portion partitioning
  - Cache capacity partitioning

- MPAM Bandwidth control:
  - Portion partitioning
  - Min/Max limit partitioning
  - Priority partitioning
  - Proportional stride partitioning

The following figure shows MPAM main bandwidth partitioning control functionalities.

Figure 2-2: mpam main functionality



- MPAM resource usage monitoring:
  - Cache-storage usage monitors
  - Memory-bandwidth usage monitors

3. Enable MPAM in firmware and software stack

Enabling MPAM on the platform involves enabling MPAM EL1/EL2 register access from EL3 firmware, which is Trusted Firmware-A. MPAM needs enabling at the highest implemented Exception level (EL3), otherwise the register accesses will trap to EL3.

3.1 Trusted Firmware-A

Trusted Firmware-A needs to be built to enable MPAM support for lower Exception levels. This is achieved by setting ENABLE_MPAM_FOR_LOWER_ELS to 1 in the build arguments. Also it adds support for MPAM EL2 context management if we have a Secure hypervisor alongside non-Secure hypervisor.

3.2 SCP-firmware

System Control Processor (SCP)is a dedicated processor that is used to abstract power and system management tasks away from application processors. SCP-firmware initializes the system and provides runtime services.

MPAM support in some ARM IPs need to be enabled in SCP-firmware.

For example, Arm CoreLink CMN-700 Coherent Mesh Network MPAM support need to be enabled in CMN-700 initialization in SCP-firmware. SCP-firmware enables MPAM Cache Portion Partitioning (CPOR) and Cache Capacity Partitioning (CCAP) for SLCs in CMN-700 por_hnf_{cfg_ctl, aux_ctl} registers.

3.3 UEFI/edk2

UEFI/edk2 firmware provides MPAM firmware data by ACPI table.

3.4 U-boot

U-boot provides MPAM firmware data by device tree.

3.5 Linux kernel

Linux kernel needs to be built with CONFIG_MPAM=y to enable MPAM support. It will discover CPU support for MPAM. MPAM has an extra ID register that describes the extra MPAM support, like MPAM virtualization.

3.6 Discovery of MPAM

In a system containing MPAM, you must provide MPAM aware software to discover the memory system topology and some implementation parameters of MPAM controls and monitors via firmware data.

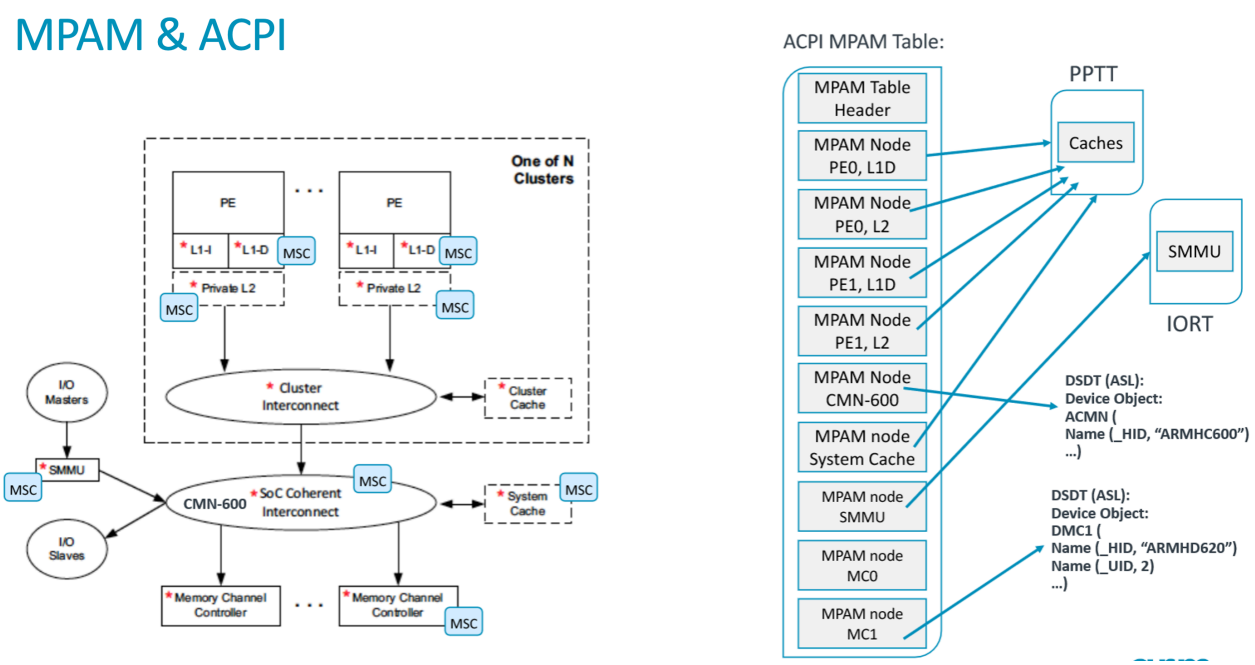
MSCs topology is passed to Linux kernel by firmware data such as Device Tree or ACPI interface. MPAM controls and parameters of MSCs are discoverable in memory-mapped ID registers. Linux MPAM driver will decide the width of memory system partitioning and monitoring values communicated through the system according to the needs of the system.

3.6.1 ACPI table

The ACPI MPAM table describes the MPAM capabilities in a system. Memory requests from requesters are handled by Memory-System Components or MSCs. SMMUs, caches, TLBs, and memory channel controllers are examples of an MSC. The MPAM table describes the properties of MSCs in the system. The table also describes the topological arrangement of MSCs and resources in the system. The topology is expressed in terms of the location of resources within the system and the relation between an MSC and the resources that it is manages.

The following figure shows an example system with MPAM support implemented in the form of MSCs distributed across various system memory components.

Figure 3-1: MPAM ACPI

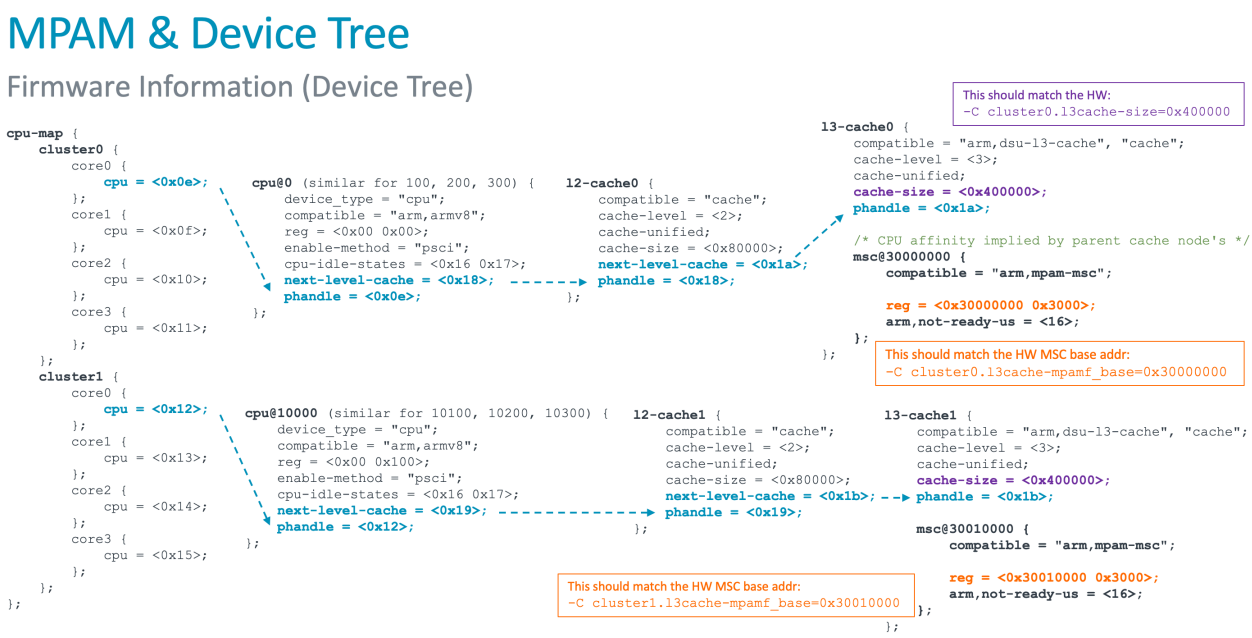


3.6.2 Device tree

The MPAM dt-binding design assumes that an MSC is a sub-block of something else such as a memory controller, cache controller, or IOMMU.

The following figure shows an example of the MPAM device tree firmware data interface.

Figure 3-2: MPAM device tree



The device tree describes the memory system topology and properties of MSCs in the system. There are 2 DSU clusters, each with 4 CPUs in the system. Each MSC node is embedded in the DSU L3 cache node.

MPAM Device Tree support is still an experimental implementation. The Device Tree Binding will evolve as the spec changes.

4. A closer look at MPAM software

MPAM software use the Memory-Mapped Registers (MMRs) to discover and configure the MPAM behavior of an MSC. All MPAM MMRs are located on one of the MPAM feature pages. An MPAM feature page is a block of addresses that contains all of the MPAM MSC MMRs in that address space. Each MPAM feature page base address must be:

- Aligned to a 4KB boundary
- Completely contained with a single 64KB-aligned block

This enables the MPAM feature page base address to be placed within a single 64KB page.

MPAM MMRs prefixes are:

- MPAMF prefix : MPAM IDR registers
- MPAMCFG prefix : registers configures MPAM resource controls
- MSMON prefix : resource monitor configuration and readout registers

An MSC's MPAM feature page is located from information about the device. It is possibly provided via firmware data such as device tree or ACPI. See Appendix B MSC Firmware Data in Memory System Resource Partitioning and Monitoring (MPAM), for A-profile architecture: Arm Architecture Reference Manual Supplement.

4.1 Programming configuration of MPAM settings

A partitionable resource can be partially allocated to a partition according to the programming of the MPAM resource control or controls for that resource. MPAMCFG resource control settings are selected by the MPAMCFG_PART_SEL register:

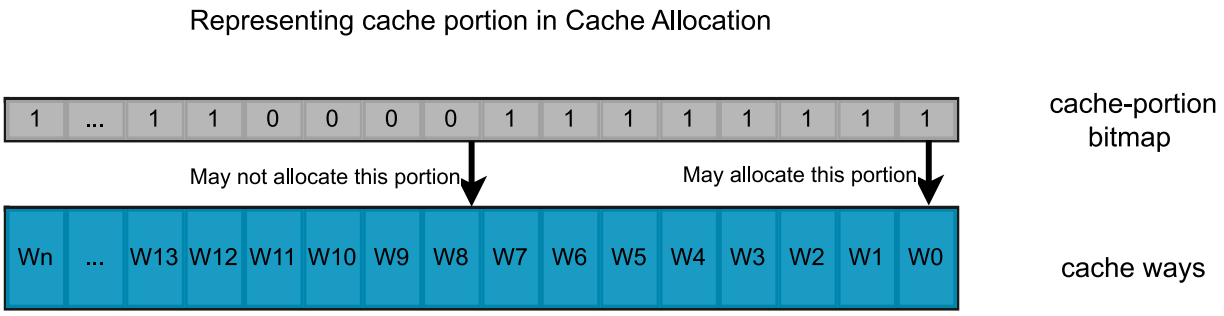
- The RIS field selects the resource instance
- The PARTID_SEL field selects the PARTID of the resource control setting accessed

4.1.1 Generic portion bitmap

In portion resource controls, the control setting is a bitmap in which each bit corresponds to a particular portion of the resource, for example cache-portion partitioning.

The following figure shows MPAM cache portion bitmap.

Figure 4-1: MPAM cache_portion_interface

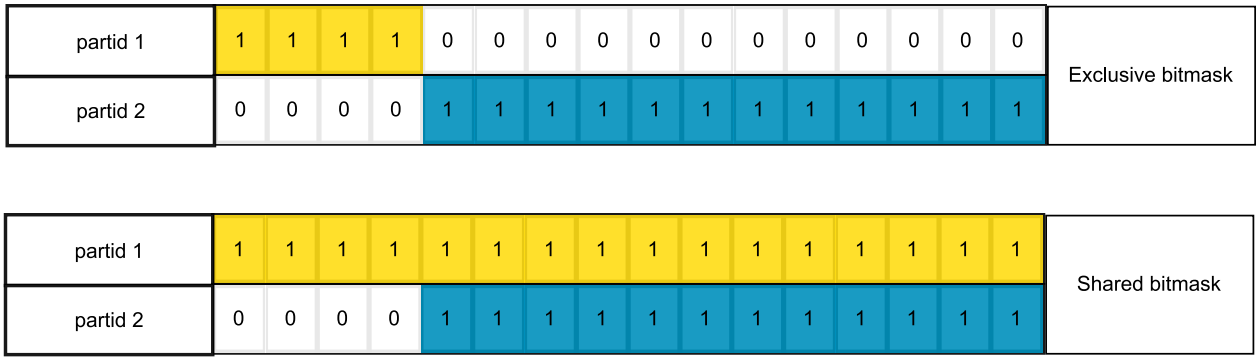


4.1.2 Cache portion shared and exclusive allocations

You can allocate cache portions for exclusive use of a partition or shared between two or more partitions. The mode of the resource group dictates the sharing of its allocations. A shareable resource group enables sharing of its allocations while an exclusive resource group does not.

The following figure shows MPAM cache portion bitmap exclusive and shared allocation examples.

Figure 4-2: MPAM cache_portion_allocation



In the figure above:

- Exclusive bitmap example. Partid1 may allocate 4 cache portions while partid2 may allocate the other 12 cache portions exclusively
- Shared bitmap example. Partid1 might allocate all the 16 cache portions and share 12 cache portions with partid2.

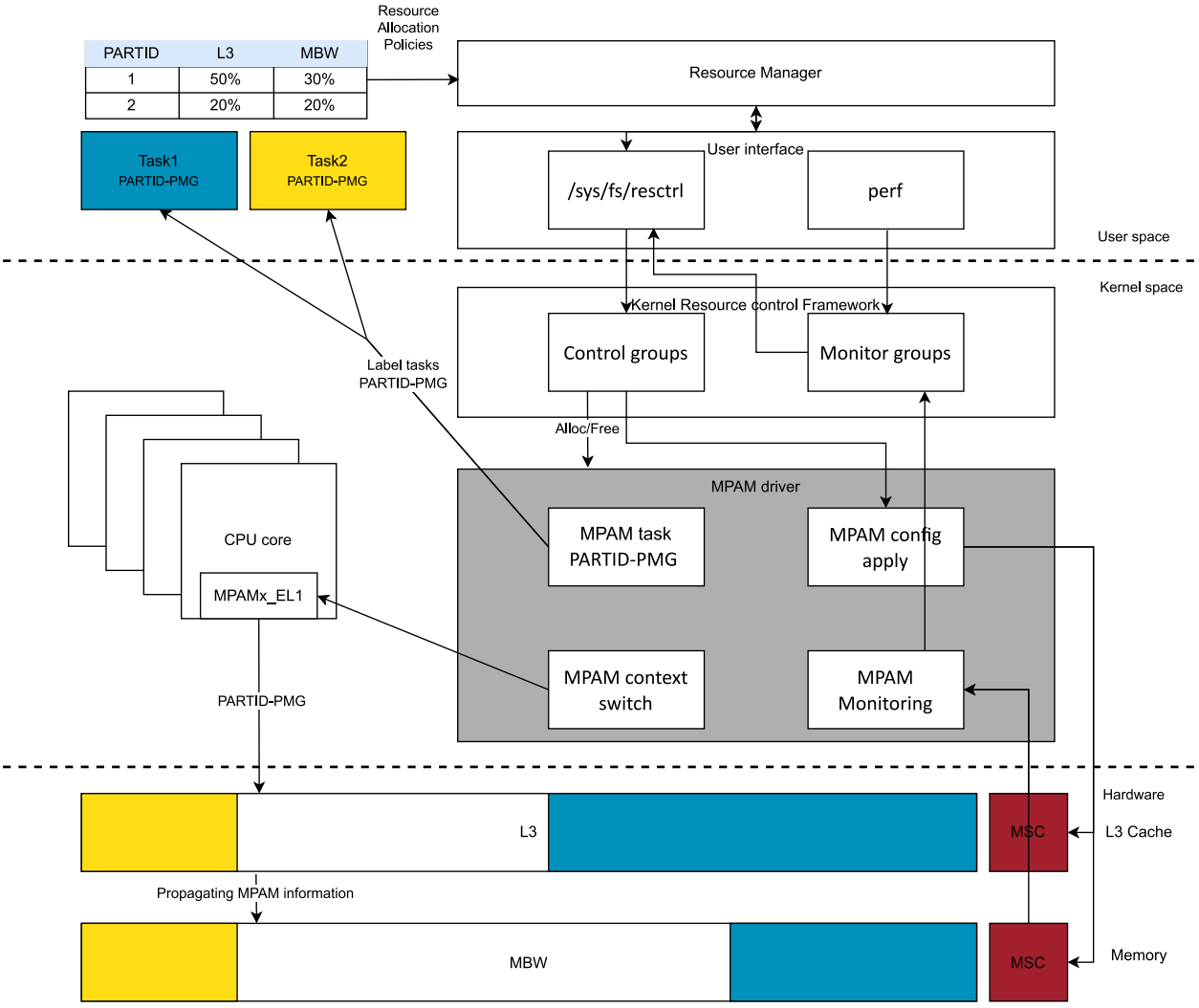
4.2 Linux MPAM overview

Linux is the primary supported operating system at the moment.

resctrl is a Linux kernel feature by which Arm's MPAM and Intel's RDT can be configured and controlled. resctrl exposes MPAM capabilities and configuration options via a file-system interface. On the latest kernel source tree, users would find resctrl adapted for X86 RDT. Arm is working on changing resctrl into a generic resource partitioning interface. For Arm64 architecture, you can configure MPAM using resctrl. The steps by which MPAM could be configured via resctrl are described in the subsequent section.

The following figure shows linux kernel resource control framework and MPAM driver overview.

Figure 4-3: MPAM Software



The MPAM driver is designed so that the default configuration uses a single PARTID, PARTID 0 with the default maximum partition configuration for the MSCs. This is done in the early stages of Linux kernel boot up.

4.2.1 Resource control and monitor groups

Resource groups are represented as directories in the resctrl file system. The default group is the root directory which, immediately after mounting, owns all the tasks and CPUs in the system. The root directory can make full use of all resources.


Note

CPUs can generate traffic with a range of PARTID and PMG values. However each MSC can have its own maximum size for these fields. Before MPAM can be used, the driver needs to probe each MSC, to find the system-wide smallest value that can be used.

4.2.2 Exploring resctrl file-system

resctrl is enabled if MPAM driver finds MSCs it can support on the platform.

Run the following command to mount the resctrl file-system.

```
# mount -t resctrl resctrl /sys/fs/resctrl
```

When the resctrl file-system has been mounted, change directory to /sys/fs/resctrl and list the files.

```
# cd /sys/fs/resctrl
/sys/fs/resctrl# ls

cpus        info     mon_data      schemata    tasks
cpus_list   mode    mon_groups    size
```

4.2.3 Configuring MPAM via resctrl file-system

On a system with MPAM features, you can create more directories in the root directory that specify different amounts of each resource.

The administrator can create a resource allocation schema for a group of tasks or CPUs. The administrator can modify schemas at runtime, and assign tasks/CPUs to use the schemas.

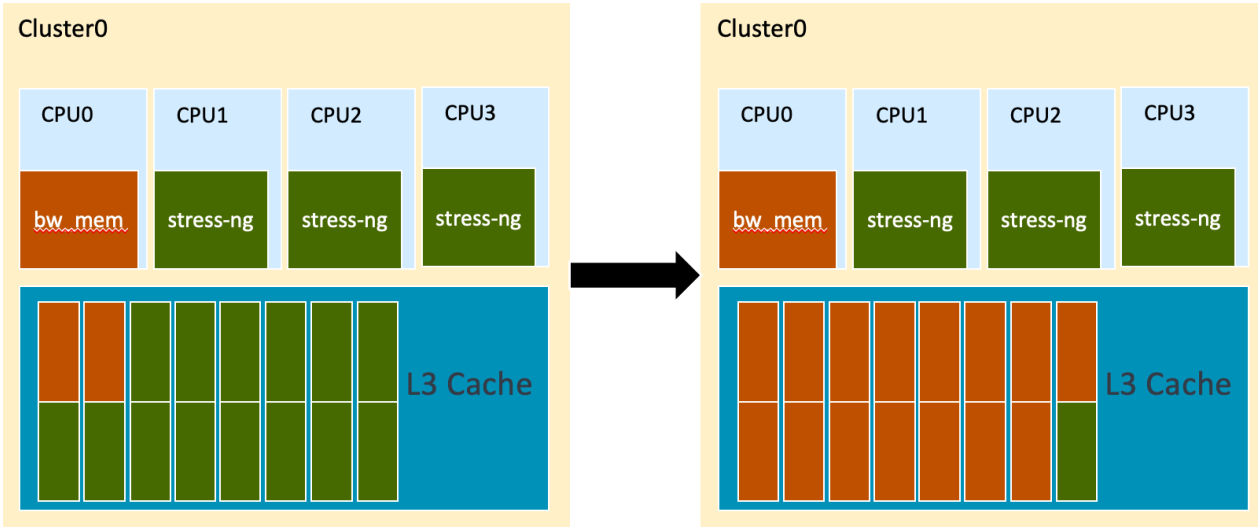
```
cd /sys/fs/resctrl
mkdir low_priority
echo "L3:0=1" > low_priority/schemata
echo "e" > low_priority/cpus
echo <low priority task pids> > low_priority/tasks
```

```
mkdir high_priority
echo "L3:0=fffe" > low_priority/schemata
echo ”1" > high_priority/cpus
echo <high priority task pids> > high_priority/tasks
```

The following example shows how to create 2 resource groups, a high-priority group and low-priority group. The kernel resource control framework internally allocates 2 new PARTIDs and associates them with the 2 new resctrl groups. 15/16 L3 cache portions in cluster0 are assigned to high-priority resctrl group. We also add dedicated CPU0 in cluster0 to high-priority resctrl group for running high-priority task. For the low-priority group we assign 1/16 L3 cache portions in cluster0 exclusively. Update the cache allocation policy by writing the schemata files with corresponding cache portion bit masks.

The following figure shows a simple example of resolve the noisy neighbor problem in previous section Noisy neighbor.

Figure 4-4: MPAM Example



This example shows how to:

- Restrict the noisy neighbor
- Anticipate cache allocation to resolve cache contention

4.3 Xen MPAM overview

The Arm MPAM extension is enabled in the Arm automotive Xen solution. MPAM support enables Xen domains to be assigned with dedicated SLC portions so that cache contention with multiple domains can be mitigated.

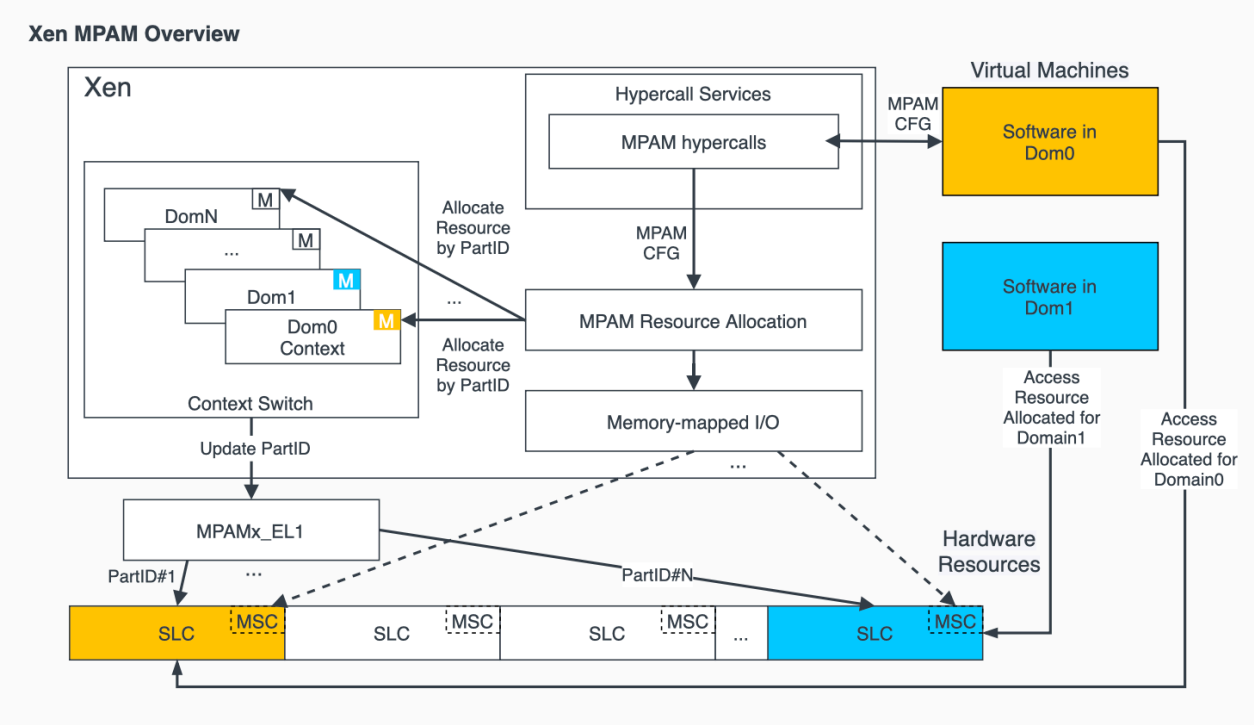
4.3.1 Xen MPAM extension

The current implementation of Xen MPAM extension support includes:

- Discover MPAM CPU feature
- Initialize MPAM at Xen boot time
- Support MPAM in Xen tools to apply the domain MPAM configuration in userspace at runtime

The following figure shows Xen MPAM support overview.

Figure 4-5: Xen MPAM Overview



4.3.2 Configure MPAM for domains

The Xen stack offers several ways for users to configure MPAM for domains:

- You can use the Xen command-line parameter dom0_mpam to configure the cache portion bit mask for Dom0
- There is a set of sub-commands in xen tool to allow users to use MPAM at runtime

See Xen MPAM Overview: In The Arm Neoverse N2 Automotive Reference Stack

5. Overall system-level dynamic resource controller based on MPAM

Most cloud-based solutions are composed of compute resources, such as:

- Web and Application Tiers: These compute resources are responsible for hosting and serving web applications, APIs, and other online services. They handle incoming user requests, process logic, and generate responses.
- Batch Processors: Batch processors are used for executing computationally intensive tasks or processing large volumes of data in batches. They are commonly employed for data processing, data transformation, and analytics workloads.
- Scheduled Jobs: Scheduled jobs involve running specific tasks or scripts at predetermined times or intervals. These jobs automate routine operations, such as data backups, database maintenance, report generation, and system monitoring.
- Specialized Resources: Cloud providers offer specialized compute resources to address specific needs. Graphics Processing Units (GPUs) are utilized for parallel processing and accelerating tasks related to graphics rendering, machine learning, and scientific simulations.
- High-performance compute instances offer increased processing power, memory, and networking capabilities for demanding workloads.

Workloads running in cloud data centers exhibit varied behaviors in processing user requests and the resulting CPU utilizations. Latency Critical (LC) workloads, for example, e-commerce and online search, are highly sensitive to fulfilling query latency requirements. Best Effort (BE) workloads, such as big data analytics, occupy hardware resources in their best efforts to improve system throughput. Current cloud data centers usually co-locate LC and BE workloads in the same server. This improves server CPU utilization and also reduce machine costs.

However, co-locate can lead to resource contention between tasks of different priorities. Without isolation controls, tasks of different priorities might compete for resources such as LLC and memory bandwidth. This might hinder the allocation of sufficient resources to LC tasks and affect the service quality of LC applications.

MPAM provides resource partitioning control capabilities for applications of different priorities. By limiting the usage of LLC and memory bandwidth resources by BE-type applications, we can improve total throughput in mixed deployment scenarios without violating service quality of LC applications.

With an MPAM-enabled system, a system administrator or orchestrator can monitor and allocate memory bandwidth or cache partition, per application, container, Virtual Machine (VM), or even per-thread if necessary.

For an example of this approach, see Heracles: Improving Resource Efficiency at Scale.

Entities that can be controlled with MPAM resource control include:

- Process/tasks

- VMs which transfer all PIDs of VM to one resource group
- Containers which put the entire container into one resource group


Note

The number of entities being monitored is limited by the PARTID+PMG range of the platform.

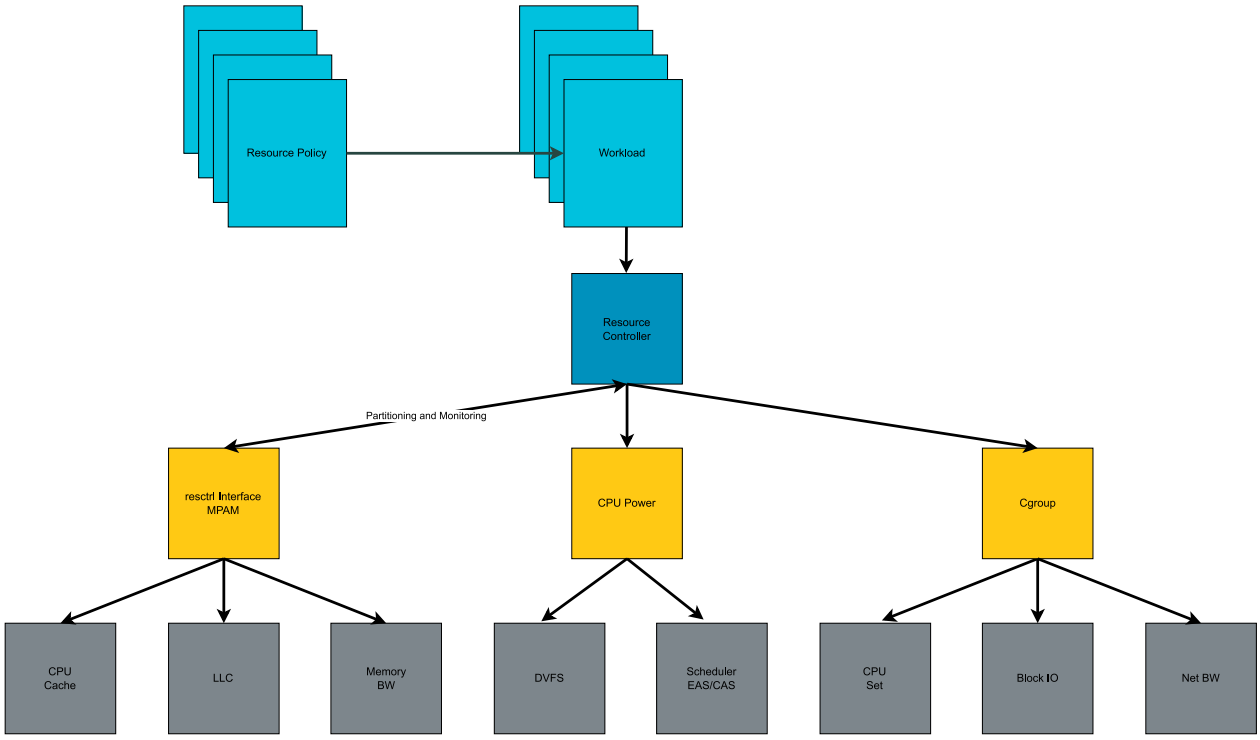
5.1 System-level resource control example

Although MPAM provides resource partitioning control capabilities, you need a systematic approach to achieve high-quality resource QoS for cloud workload co-locations. The key requirement for workload co-location is to improve the BE workload's throughput as much as possible while not violating the LC workload's Service-Level Objectives (SLOs).

Service providers can:

- Monitor the resource usage in the system
- Apply resource governance. Consider applying policies that avoid a single tenant overwhelming the system and reducing the capacity available to others.
- Continuously monitor and adjust, rebalance resources
  - Make sure latency-sensitive workloads get enough share of resources, CPU, cache, memory, I/O bandwidth
  - Improve the BE workload's throughput

Figure 5-1: System-level resource controller



System-level resource controller can achieve high utilization and energy efficiency by:

- Combining MPAM control over cache and memory bandwidth
- Efficient task-packing and CPU capacity/energy aware scheduling and CPU power allocation
- Cgroup control over CPU set
- Block IO
- Network bandwidth isolation

Some of the largest cloud providers collectively might have more than 2 million servers worldwide. With such large numbers, even small improvements have a large impact. With high-quality MPAM based solution it can achieve significant improvements on average CPU utilization and translates to an energy efficiency gain.