東京 2025

LINUX PLUMBERS CONFERENCE

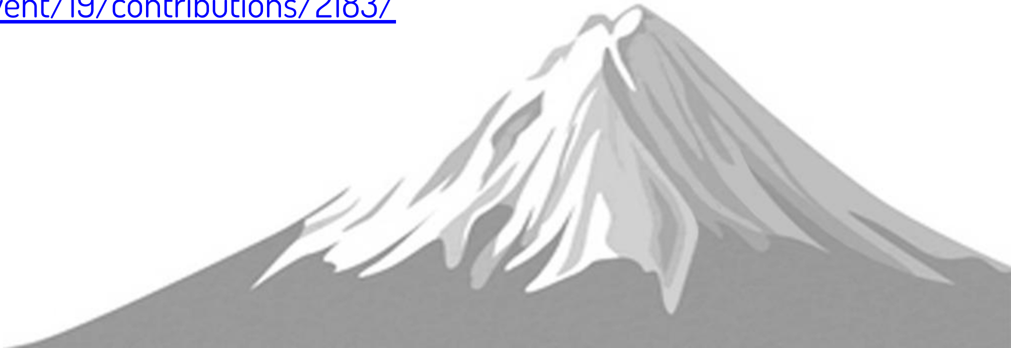
TOKYO, JAPAN / DECEMBER 11-13, 2025

RISC-V QoS:

Ssqoid, CBQRI, RQSC and resctrl

Drew Fustini (Tenstorrent)
fustini@kernel.org / dfustini@tenstorrent.com

https://lpc.events/event/19/contributions/2183/



東京 2025
LINUX
PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

Ssqosid extension: Quality-of-Service Identifiers

- Ssqosid v1.0 was ratified in 2024. A proof-of-concept back in 2023
- QoS in this context is concerned with shared resources on an SoC such as cache capacity and memory bandwidth.
  - Intel and AMD already have QoS features on x86 and ARM has MPAM
- Ssqoisd adds a new CSR named **srmcfg** which configures a hart with two identifiers:
  - Resource Control ID (RCID) for resource allocation control
  - Monitoring Counter ID (MCID) for monitoring of resource usage
- These IDs accompany each request issued by the hart to shared resource controllers like a cache or memory controllers

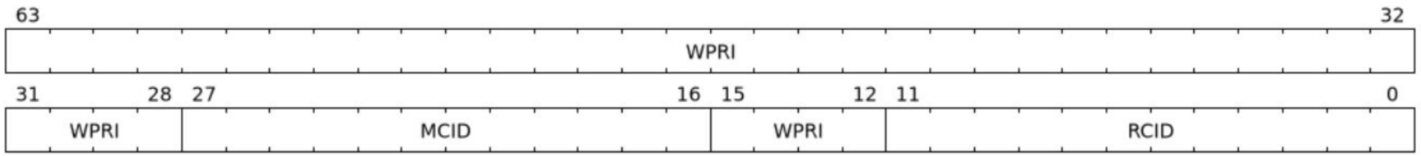


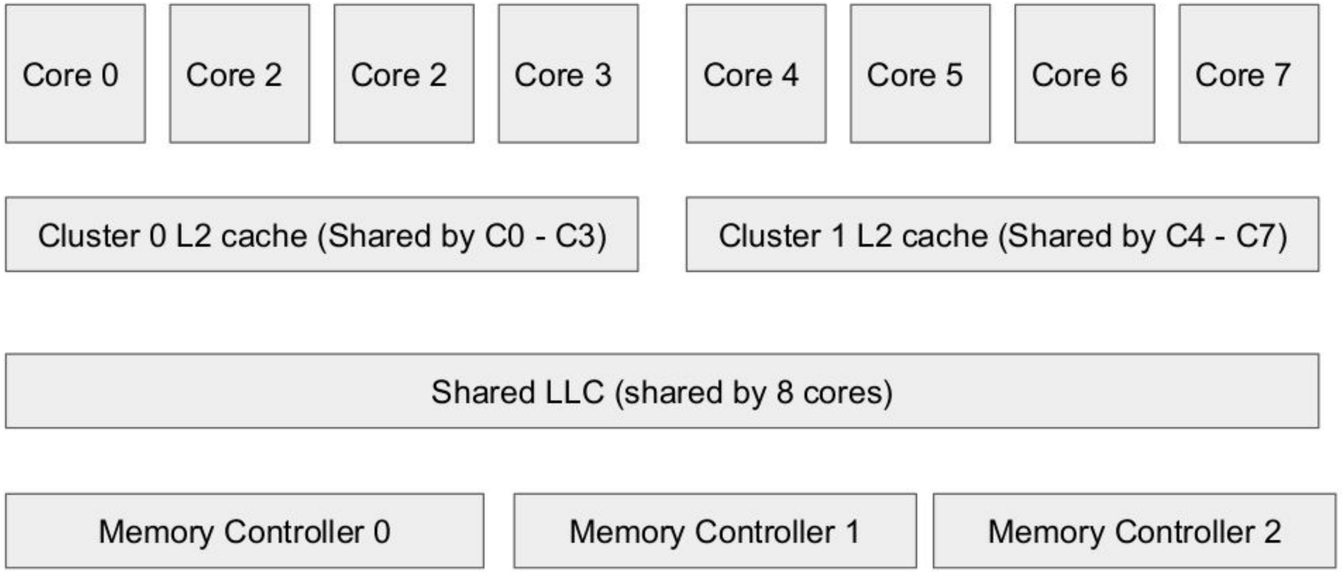
Figure 1. Supervisor Resource Management Configuration (`srmcfg`) register for SXLEN=64

東京 2025 LINUX PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

CBQRI: Capacity and Bandwidth Controller QoS Register Interface

- CBQRI v1.0 was ratified in 2024 after a proof of concept developed in 2023.
- Capacity-controller QoS Register Interface: capacity allocation and usage monitoring
  - NCBLKS field holds the total number of allocatable unitless **capacity blocks** in the controller
  - The capacity represented by capacity block not specified by the spec
  - Example:, a cache controller may define a 16-way cache to have NCBLKS of 16
- Bandwidth-controller QoS Register Interface: bandwidth allocation and bandwidth usage monitoring
  - NBWBLKS field reports the total number of available **bandwidth blocks** in the controller
  - The bandwidth represented by each bandwidth block not specified by the spec
  - MRBWB field reports the maximum number of bandwidth blocks that can be reserved

Example SoC


Core 0
Core 2
Core 2
Core 3
Core 4
Core 5
Core 6
Core 7
Cluster 0 L2 cache (Shared by C0 - C3)
Cluster 1 L2 cache (Shared by C4 - C7)
Shared LLC (shared by 8 cores)
Memory Controller 0
Memory Controller 1
Memory Controller 2

- Cluster 0 L2 cache is shared by 4 cores labeled Core 0 to Core 3
- Cluster 1 L2 cache is shared by 4 cores labeled Core 4 to Core 8
- Shared LLC (e.g. L3) is shared by all 8 cores
- Shared LLC is connected to 3 memory controllers labeled 0 to 2

Example SoC

- Number of RCIDs: 64 *(for allocation control)*
- Number of MCIDs: 256 *(for monitoring)*
- Capacity: L2 cache controller
  - NCBLKS: 12 *(e.g. number of ways)*
  - Number of Access Types: 2 (code and data)
  - Usage monitoring not supported
  - Allocation operations: CONFIG_LIMIT, READ_LIMIT
- Capacity: LLC controller
  - NCBLKS: 16
  - Number of Access Types: 2 (code and data)
  - Usage monitoring operations: CONFIG_EVENT, READ_COUNTER
  - Event IDs supported: None, Occupancy
  - Capacity allocation ops: CONFIG_LIMIT, READ_LIMIT, FLUSH_RCID
- Bandwidth: memory controllers
  - NBWBLKS: 1024
  - MRBWB: 80 (80%)
  - Usage monitoring operations
    - CONFIG_EVENT, READ_COUNTER
  - Event IDs supported
    - Total read/write byte count, Total read byte count, Total write byte count
  - Bandwidth allocation operations
    - CONFIG_LIMIT, READ_LIMIT
  - Number of access types
    - 1 (no code/data differentiation)

RQSC

- ACPI RQSC table specification describes:
    - the properties of the QoS controllers in the system
    - the topological arrangement of the QoS controllers and resources
- RQSC slides from RVI Platform Runtime Services (PRS) TG meeting on March 27, 2025.
- Software proof-of-concept with Qemu and Linux

resctrl

- resctrl is a virtual filesystem in Linux that is the user interface for cache and memory QoS features
  - All processes belong to the default resource group which defaults to full access to all the resources
  - New resource groups can be created, and tasks are assigned to groups by their PID
- Example:. L3 cache with 16 wayswould be represented by cache bitmask is `0xffff`
  - Write to the "schemata" file to modify the default group to only use half of the L3 cache (0x00ff) on domain 0:

```
# cd /sys/fs/resctrl; echo -e "L3:0=ff;1=ffff" > schemata
```

  - Create a new resource group "p1" and give it access to the top half of the L3 cache (0xff00) on domain 0:

```
# mkdir p1; echo -e "L3:0=ff00" > p1/schemata
```

  - Move an important task (PID 1234) to resource group "p1" and pin it to a core in domain 0:

```
# echo 1234 > p1/tasks; taskset -cp 1 1234
```

Discussion: new resource types in resctrl?

- RISC-V CBQRI and ARM MPAM are much more flexible than the existing AMD and Intel QoS capabilities.
  - For example, resctrl MB resource implicitly assumes that memory bandwidth is the same as L3.
  - Current approach is to upstream support for just the aspects of MPAM and CBQRI that 'look like a Xeon'
  - The CBQRI [proof-of-concept]() was awkward because extra L3 domains had to be created for each memory controller just so that the bandwidth can be monitored.
- **Should resource types be added to resctrl to better fit CBQRI bandwidth controllers?**
- **Who cares about resources beyond cache occupancy and memory bandwidth?**
  - PCI bus, interconnects, TLB, ...

Discussion: Device Tree bindings

- The CBQRI proof-of-concept was based on device tree as the ACPI RQSC table was defined later
- riscv,cbqri.yaml defined properties for cache and memory controller nodes:
  - `riscv,cbqri-rcid` for max number of RCIDs
  - `riscv,cbqri-rcid` for max number of MCIDs
- The rest can be discovered by reading the CC and BC capabilities through CBQRI operations during probe
- **Does anyone who is implementing CBQRI care about Device Tree?**
- **Should we just focus on ACPI RQSC table and not upstream any DT bindings?**

東京 2025

LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025