Contribution ID: **378** Type: **not specified**

RISC-V QoS: CBQRI, RQSC and resctrl

QoS Background

Some of the next generation of RISC-V SoCs are expected to have QoS (Quality-of Service) functionality to control and monitor the usage of resources such as cache capacity and memory bandwidth. The RISC-V Quality-of-Service Identifiers (Ssqosid) extension [1] adds the srmcfg CSR to configure a hart with two identifiers: a Resource Control ID (RCID) and a Monitoring Counter ID (MCID). These identifiers accompany each request issued by the hart to shared resource controllers. RISC-V Capacity and Bandwidth Controller QoS Register Interface (CBQRI) specification [2] allows resource allocation to be controlled and monitored.

Intel and AMD already have QoS features on x86 for many years. There is an existing user interface in Linux the resctrl virtual filesystem [3]. There was a discussion session on resctrl at LPC 2023 led by Peter Newman [4][5].

ARM has also introduced a similar QoS specification called MPAM (Memory System Resource Partitioning and Monitoring) [6] and it is now present in some ARM64 server chips. The resctrl had the historical problem of having been implemented in arch/x86. ARM kernel developer James Morse led a multi-year effort to decouple resctrl from x86 and move it into fs. The final move happened in the 6.16 when fs/resctrl was recreated. James is now working on upstreaming support for MPAM [7].

CBQRI and 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. Both James and I have tried to take the path of only implementing the aspects of MPAM and CBQRI that 'look like a Xeon' [7]. This did present an awkward situation for the CBQRI proof of concept [8] where extra L3 domains are created for each memory controller so that the bandwidth can be monitored.

Open items to discuss

Resource types in resctrl
- Should a new resource type be added to resctrl to better fit CBQRI bandwidth controllers?
- Anything beyond cache and memory bandwidth? PCI bus, etc?

DT Bindings
- What should the DT bindings look like for capacity controllers and bandwidth controllers?
- The proof of concept had bindings that allowed a cache controller driver and memory controller as a temporary measure to allow the CBQRI code to be exercised
- Does anyone who is implementing CBQRI have any input on what DT bindings should be?
- Bigger question: do people actually care about DT?
- There is now support for the ACPI RQSC [9] table so it may be the case that everyone expects to use that instead of DT?

References

1. https://github.com/riscv/riscv-ssqosid/releases/tag/v1.0
2. https://github.com/riscv-non-isa/riscv-cbqri/releases/tag/v1.0
3. https://docs.kernel.org/filesystems/resctrl.html
4. https://lpc.events/event/17/contributions/1567/
5. https://www.youtube.com/watch?v=j6SB_-CeFHo
6. https://developer.arm.com/documentation/107768/0100/Overview
7. https://lore.kernel.org/all/ee08ba7e-2669-447f-ae04-5a6b00a16e77@arm.com/

8. https://lore.kernel.org/linux-riscv/20230419111111.477118-1-dfustini@baylibre.com/
9. https://lf-rise.atlassian.net/wiki/spaces/HOME/pages/433291272/ACPI+RQSC+Proof+of+Concept

Primary author: FUSTINI, Drew (Tenstorrent)

Co-author: PATRA, ATISH (Rivos)

Presenter: FUSTINI, Drew (Tenstorrent)

Session Classification: RISC-V MC

Track Classification: RISC-V MC