UMaine VMW Group Tech Report
UMAINE-VMW-TR-UNCORE-OFFCORE-2015-10

System-wide Performance Counter Measurements: Offcore, Uncore, and Northbridge Performance Events in Modern Processors

Vincent M. Weaver
University of Maine
vincent.weaver@maine.edu

July 9, 2017

UMaine VMW Group Tech Report
UMAINE-VMW-TR-UNCORE-OFFCORE-2015-10

Abstract

Modern processors often have many processing cores in one package (or socket). Traditional hardware performance counters measure only values on a single core. A chip package has many resources which are package-wide and thus need a separate performance reporting mechanism. The values for these shared and off-core resources are reported as “offcore”, “uncore” or “northbridge” events.

1 Introduction

Modern processors often have many processing cores in one package (or socket). Traditional hardware performance counters measure only values on a single core. A chip package has many resources which are package-wide and thus need a separate performance reporting mechanism.

The shared socket-wide values are called “uncore” by Intel and “northbridge” by AMD. For simplicity in this paper both will be referred to as “uncore”.

Offcore events are per-core values on their way to the uncore. These are thus more like traditional events, but they are programmed by a slightly different interface and interact directly with the uncore values so we will investigate them too.

The names and terminology for these counters vary by vendor and processor release. This document intends to give a rough overview of the capabilities of the various common systems that support such uncore.

Most of this information was gathered from various architectural manuals, as well as the Linux source code. The Intel Linux drivers were written by Intel employees and typically their knowledge matches actual hardware better than the manuals do.

2 Uncore Implementations

Uncore supports varies by vendor and processor model. This is a quick overview of uncores found on recent processors. For actual lists of events supported, see the referenced architectural manuals.

2.1 Intel

Intel chips can have elaborate uncore support. Usually the server models of the chips have much more extensive uncore support.

There are two kinds of access modes for uncore. One is to use model specific registers (MSRs) which are chip-specific special registers that can be read by privileged `rdmsr` instructions. This is also how traditional hardware performance counters are implemented. The other access method is via the PCI config space. This is different than MSR access, and requires a different set of tools to read. If your operating system has uncore support it will abstract away the differences between these interfaces.

Here is a summary of the MSR access uncores available on various Intel CPUs. Not all of them are available on all CPUs.

- B-Box (Coherency and Memory Ordering)
- C-Box (last level cache)
- M-Box (Integrated Memory Controllers)
- P-Box (Physical QPI interconnect)
- R-Box (Crossbar Router Box)
- S-Box (QPI)
- U-Box (System Config Controller)
- W-Box (Power Controller)
- PCU (Power Control) (how different from w-box?)
- ARB (arbitration?)

Here are some uncores available that are accessible via the PCI device config space. Again, not all are available on all CPUs.

- HA Home Agent (Coherence Protocols)
- iMC (Integrated Memory Controller)
- QPI (Quick Path Interconnect)
- R2PCIe (Ring to PCIe)
- R3QPI (Ring to QPI)
- IRP (IIO Coherency)

The number of uncores (sometimes called boxes) varies based on the uncore type and the processor type. They also have a varying amount of counters.

Some of the counters are generic, that is any event can be programmed into any of the counters. Some are fixed, in that a counter can only report a specific value. Others have constraints, in that a counter can be programmed to report some, but not all, possible events.

Another feature found is packet matching or filtering. A counter can be programmed to only report events that match a filter value.

In theory the uncores can report an interrupt on overflow, allowing sampling. On Linux this possibly is not turned on due to buggy behavior in the hardware? I need to check on that.

The MSR interfaces can be programmed from any of the cores, although is is recommended (though not required) that you program core0, although interrupts can be delivered to any.

2.1.1 Nehalem and Westmere (not -EX)

The desktop (non-server) Nehalem (Model 26,30) and Westmere (37,44) chips were the first uncore implementations and it is simpler than the ones that come later. It is described in section 18.7.2 in the Intel Software Devel Manual Volume 3B [1]. A summary is shown in Table 1.

Table 1: Nehalem and Westmere (not -EX) uncore boxes.

Name	Boxes	Counters	Generic	Filtering	Size	Access	libpfm4	Linux Support
Uncore	1	8	?	Y	48-bit	MSR	Y	3.6

2.1.2 Nehalem-EX (Xeon 7500)

The Nehalem-EX server (model 46) has a complex uncore which you can find described in a standalone document [2]. A summary is shown in Table 2.

Table 2: Nehalem-EX uncore boxes.

Name	Boxes	Counters	Generic	Filtering	Size	Access	libpfm4	Linux Support
C-Box	8	6	Y	N	48-bit	MSR	N	3.6
S-Box	2	4	Y	Y	48-bit	MSR	N	3.6
B-Box	2	4	N	Y	48-bit	MSR	N	3.6
M-Box	2	6	N	N	48-bit	MSR	N	3.6
R-Box	1 (L/R)	16 (2/port)	N	Y	48-bit	MSR	N	3.6
U-Box	1	1	Y	N	48-bit	MSR	N	3.6
W-Box	1	4	Y	N	48-bit	MSR	N	3.6

U-Box is global control which also has a small set of events (16). Some have odd names `BUF_VALID_DOOR_BELL`. On overflow of any Box counters can be frozen and trigger an interrupt which is handled by the U-Box.

The C-Box is the interface between core and last-level cache. There are 8 of them. Even if no cache hooked up to a C-Box it still is there and tracks snoops, etc. All C-Box accesses go through an S-Box.

The B-Box reports coherency and memory ordering for the M-Box.

The S-Box reports on the QPI connection, which is the interface between the Last level cache and system, as well as the C and R/B Boxes.

The R-Box is the crossbar router box.

The M-Box reports on the memory controllers. It converts accesses to the Intel SMI (Scalable Memory Interface) which handles FB-DIMMs

The W-Box is the power controller, with events such as cycles in Turbo mode. It also has a fixed cycle counter useful for profiling.

2.1.3 Westmere-EX (Xeon E7)

The Westmere-EX (model 47) has a complex uncore [3]. It is similar to the Nehalem-EX interface, with the main difference being that there are 10 C-Boxes.

A summary is shown in Table 3.

Table 3: Westmere-EX uncore boxes.

Name	Boxes	Counters	Generic	Filtering	Size	Access	libpfm4	Linux Support
C-Box	10	6	Y	N	48-bit	MSR	N	3.6
S-Box	2	6	Y	Y	48-bit	MSR	N	3.6
B-Box	2	4	N	Y	48-bit	MSR	N	3.6
M-Box	2	6	N	N	48-bit	MSR	N	3.6
R-Box	1 (L/R)	16 (2/port)	N	Y	48-bit	MSR	N	3.6
U-Box	1	1	Y	N	48-bit	MSR	N	3.6
W-Box	1	4	Y	N	48-bit	MSR	N	3.6

2.2 SandyBridge, IvyBridge, Haswell (non-server)

The desktop (non-server) (non-EP) versions of the SandyBridge (model 42), Ivy-Bridge (model 58), and Haswell (models 60 and 69) chips have similar uncore interfaces. This is described in section 18.9.6 of the Intel Vol3b documentation [1]. A summary is shown in Table 4.

Table 4: Non-server SandyBridge, IvyBridge, and Haswell uncore boxes.

Name	Boxes	Counters	Filtering	Size	Access	libpfm4	Linux Support
C-Box	varies	2	Y	44-bit	MSR	Y (except HSW)	3.6 (SNB) 3.10 (IVB) n/a (HS
ARB	1	2	?	44-bit	MSR	N	n/a
iMC	4	4gen/1fix	?	48-bit	PCI	N	3.15

The L3-cache has multiple slices, each is given a C-Box (coherence). It has event constraints.

2.3 SandyBridge-EP (Xeon E5-16xx and E5-26xx)

The SandyBridge-EP (model 45) ("efficient performance") has a different uncore [4] than previous server models. A summary is given in Table5.

Table 5: SandyBridge-EP uncore boxes.

Name	Boxes	Counters	Filtering	Size	Access	libpfm4	Linux Support
U-Box	1	2	N	44-bit	MSR	Y	3.6
C-Box	8-14	4	Y	44-bit	MSR	Y	3.6
HA	1	4	Y	48-bit	PCI	Y	3.6
iMC	4	4gen/1fix	?	48-bit	PCI	Y	3.6
IRP	2	4	?	48-bit	PCI	N	3.6
PCU	1	4	Y	48-bit	MSR	Y	3.6
QPI	1	4	Y	48-bit	PCI	Y	3.6, 3.12 (Filter)
R2PCIE	1	4 w/ Constraints	?	44-bit	PCI	Y	3.6
R3QPI	1	3 w/ Constraints	?	44-bit	PCI	Y	3.6

The U-Box still holds global config, but only has 4 events.

C-Box (or CBo) still handles last-level cache. It supports filtering.

HA is the Home agent.

iMC is the Integrated Memory controller. It is the interface between DRAM and the ring controller through the Home Agent. Values can increase by up to 8b per cycle.

The IRP handles IIO (?) coherency. Values can increase by up to 8b per cycle.

The PCU is the Power-control interface. Values can increase by up to 4b per cycle. There are two additional C-state residence registers that can be read via a different interface.

The QPI is the QPI (quick patch interconnect?) link layer.

The R2PCIe measures Ring to PCIe events. Values can increase by up to 5 per cycle. Only counter 0 measures occupancy events, 2/3 for ring utilization.

The R3QPI measures Ring to QPI events.

2.4 IvyBridge-EP Ivytown (Xeon E7-8xxx)

The IvyTown (model 62) Uncore is similar to that of the SandyBridge-EP. (cite?) A summary is shown in Table 6.

Table 6: IvyBridge-EP (IvyTown) uncore boxes.

Name	Boxes	Counters	Filtering	Size	Access	libpfm4	Linux Support
U-Box	1	2	N	44-bit	MSR	Y	3.10
C-Box	14	4	Y	44-bit	MSR	Y	3.10
HA	2	4	Y	48-bit	PCI	Y	3.10
iMC	4	4gen/1fix	?	48-bit	PCI	Y	3.10
IRP	2	4	?	48-bit	PCI	Y	3.13
PCU	1	4	Y	48-bit	MSR	Y	3.10
QPI	1	4	Y	48-bit	PCI	Y	3.10, 3.12 (filter)
R2PCIE	1	4 w/ Constraints	?	44-bit	PCI	Y	3.10
R3QPI	1	3 w/ Constraints	?	44-bit	PCI	Y	3.10

2.4.1 Xeon Phi

The Xeon Phi has uncore events but the documentation for them is not public.

It seems like each memory controller (up to 8 of them) has its own box, and each box possibly has 4 counters.

A little bit of info can be found on slides posted on various websites.

2.5 AMD

AMD calls their version of uncore Northbridge events. This dates back to earlier times where motherboard chipsets had two helper chips, the Southbridge and Northbridge. The Northbridge handled access to main memory, hence the terminology.

2.5.1 Shared Northbridge (Families 10h,11h,12h,14h)

Older AMD systems have Northbridge events that share the traditional performance counter MSR interface. Northbridge events are programmed just like regular events using the same interface and counters. Events available include those dealing with the memory controller, crossbar, and HyperTransport events.

Despite looking like a "normal" performance event, northbridge events have some special behavior. Once configured, the counter results are shared by all cores in a package. All other cores can see the counts and other cores are blocked from changing it.

For Family 10h processors (Barcelona, Shanghai, Istanbul, Magny Cours) more details can be found in the Fam 10h BIOS and Kernel Developer's guide [5] section 3.12.

For Family 11h processors (Turion) more details can be found in the Fam 11h BIOS and Kernel Developer's guide [6] section 3.14.

For Family 12h processors (Llano) more details can be found in the Fam 12h BIOS and Kernel Developer's guide [7] section 3.24 (specifically 3.24.7 for NB events).

For Family 14h processors (Bobcat) more details can be found in the Fam 12h BIOS and Kernel Developer's guide [8] section 3.24 (specifically 3.24.2 for NB events).

Linux support for AMD NB events of this type dates to the introduction of the perf_event interface in 2.6.31, but proper constraints for the NB events were not implemented until 2.6.34.

A summary can be found in Table 7.

Table 7: Fam10h Northbridge counters.

Name	Counters	Filtering	Size	Access	libpfm4	Linux Support
Fam10h NB	4 (shared with regular)	Y	48-bit	MSR	Y	2.6.34
Fam11h NB	4 (shared with regular)	Y	48-bit	MSR	Y	2.6.34
Fam12h NB	4 (shared with regular)	Y	48-bit	MSR	Y	2.6.34
Fam14h NB	4 (shared with regular)	Y	48-bit	MSR	Y	2.6.34

2.5.2 Separate Northbridge (Families 15h, 16h)

Unlike earlier processors, Fam15h and newer have a separate Northbridge PMU interface that uses a separate set of MSRs.

Family 15h includes Bulldozer, Piledriver, and Excavator although it is unclear how the model numbers map to CPU family type (the documentation is confusing). Information can be found in section 2.7.2 of the fam15h model00h BIOS and Kernel Developer Guide [9] or section 2.6.1.2 of the fam15h model30h BIOS and Kernel Developer Guide [10].

Family 16h includes Jaguar. Information can be found in section 2.6.1.3 of the fam16h model00h BIOS and Kernel Developer Guide [11] or section 2.6.1.2 of the fam16h model30h BIOS and Kernel Developer Guide [12].

A summary can be found in Table 8.

Initial Linux support was with a Fam10h-compatible interface in 3.9, but the ABI was broken and a completely different uncore-like interface was added in 3.10. Relevant files can be found under `/sys/devices/amd_nb/`

There is no support for generating interrupts. All four counters are generic. All events only increment once per clock.

Table 8: Fam15h Northbridge counters.

Name	Counters	Filtering	Size	Access	libpfm4	Linux Support
Fam15h NB	4	Y	48-bit	MSR	Y	3.9 / 3.10
Fam16h NB	4	Y	48-bit	MSR	N	3.10

2.5.3 L2I (Family 16h)

The L2I events can measure detailed L2 cache statistics. These are package-wide unless specifically masked.

L2I support was initially found in the Family 16h Jaguar processors.

Information can be found in section 2.6.1.2 of the fam16h model00h BIOS and Kernel Developer Guide [11] or section 2.6.1.2 of the fam15h model30h BIOS and Kernel Developer Guide [12].

A summary can be found in Table 9.

Initial Linux support was added in 3.10. The PMU can be found under `/sys/devices/amd_l2/`.

There is no support for generating interrupts. All four counters are generic. All events only increment once per clock.

Table 9: L2I counters.

Name	Counters	Filtering	Size	Access	libpfm4	Linux Support
Fam16h L2I	4	Y	48-bit	MSR	N	3.10

2.5.4 IOMMU

Linux has a driver for IOMMU performance events treated as an uncore-type counter. I have not been able to determine which processors support this feature.

2.6 IBM Power

TODO. I know they have support.

3 Offcore Implementations

Offcore events are found on Intel processors. They allow measuring events that go off-core. Unlike traditional performance events, these require programming an extra offcore register that holds filtering information. So to program an event requires two config settings: one for a regular counter slot, and one for an additional offcore response config register. Some chips also support average latency measurement. Older models share the offcore registers between siblings when hyperthreading. More details can be found in Vol3b [1]. A summary of support can be found in Table 10.

Initial support for Offcore was in 2.6.39 but raw event support did not appear until 3.3. Raw event support was removed at the last minute before the 2.6.39 release and programming the values was silently ignored so it is not possible to detect whether raw offcore support is possible. Even when raw support was disabled, the offcore registers held the value of the last generic offcore-related event programmed, so checking for a 0 result was not sufficient.

Table 10: Offcore event support.

Chip	Offcore Response Registers	Latency	Shared when HT	libpfm4	Linux Support
Nehalem	1	?	Y	Y	2.6.39
Westmere	2	?	Y	Y	2.6.39
Sandybridge	2	?	N	Y	3.1
Ivybridge	2	?	N	Y	3.6
Haswell	2	?	N	Y	3.11
Atom Silvermont	2	Y	?	Y	3.12

4 Power Measurement

Also per-socket rather than per-core.

4.1 Intel RAPL

4.2 AMD APM

4.3 Xeon Phi

4.4 ARM

ARM chips have uncore supporton ARMv8 SoC PMU. Support was added in Linux 4.12.

5 Tool Support

5.1 perf

Linux perf_event exports some uncore and offcore events as generic events found under the /sys (where?).

You can also program using the raw event interface.

5.1.1 Measuring Uncore Events

For uncore support you'll want to be sure that in the `/sys/bus/event_source/devices/` directory there are some uncore PMUs.

The syntax for doing an uncore call is something like this: `./perf stat -a -e "uncore_imc_0/event=0xff,umask=0x00/" /bin/ls`

You may need root privileges to access uncore events.

5.1.2 Measuring Offcore Events

You need to find the values for the events you are interested in. libpfm4 can help with this. It will look something like:
`perf stat -e cpu/cmask=1,event=2,umask=3,offcore_rsp=0x3f80408fff/ /bin/ls`

Another way to do things is to find the values using libpfm4 check_events program as such:

```
$ check_events OFFCORE_RESPONSE_0:ANY_DATA:REMOTE_DRAM
Supported PMU models:
.........
Codes : 0x5301b7 0x2033
```

And you can then use the results in perf like this:
`perf stat -e cpu/config=0x5301b7,config1=0x2033,name=Remote_DRAM_Accesses/ ls`

5.2 PAPI

6 Evaluation

Validation?

7 Related Work

Likwid supports reading uncore events.

RCI tool supports reading uncore(?).

8 Conclusion

Uncore and offcore are the key to exploring the memory wall on modern processors.

References

[1] Intel Corporation, *Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3: System Programming Guide*, Feb. 2014.

[2] Intel Corporation, *Intel® Xeon® Processor 7500 Series Uncore Programming Guide*, Mar. 2010.

[3] Intel Corporation, *Intel® Xeon® Processor E7 Family Uncore Performance Monitoring Programming Guide*, Apr. 2011.

[4] Intel Corporation, *Intel® Xeon® Processor E5 v2 and E7 v2 Product Families Uncore Performance Monitoring Reference Manual*, Feb. 2014.

[5] Advanced Micro Devices, *BIOS and Kernel Developers Guide (BKDG) For AMD Family 10h Processors*, Jan. 2013.

[6] Advanced Micro Devices, *BIOS and Kernel Developers Guide (BKDG) For AMD Family 11h Processors*, July 2008.

[7] Advanced Micro Devices, *BIOS and Kernel Developers Guide (BKDG) For AMD Family 12h Processors*, Oct. 2011.

[8] Advanced Micro Devices, *BIOS and Kernel Developers Guide (BKDG) For AMD Family 14h Models 00h-0Fh Processors*, Feb. 2012.

[9] Advanced Micro Devices, *BIOS and Kernel Developers Guide (BKDG) For AMD Family 15h Models 00h-0Fh Processors*, Jan. 2013.

[10] Advanced Micro Devices, *BIOS and Kernel Developers Guide (BKDG) For AMD Family 15h Models 30h-3Fh Processors*, Mar. 2014.

[11] Advanced Micro Devices, *BIOS and Kernel Developers Guide (BKDG) For AMD Family 16h Models 00h-0Fh Processors*, Feb. 2015.

[12] Advanced Micro Devices, *BIOS and Kernel Developers Guide (BKDG) For AMD Family 16h Models 30h-3Fh Processors*, Mar. 2015.