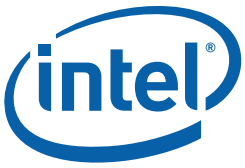

Secure Access of Performance Monitoring Unit by User Space Profilers

White Paper

This paper a proposes a software mechanism targeting performance profilers which would run at user space privilege to access performance monitoring hardware, the latter requires privileged access in kernel mode, in a secure manner without causing unintended interference to the software stack.

June 2016

Revision 1.0

Document Number: 334467-001

Notice: This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure. Intel does not assume any liability for lost or stolen data or systems or any damages resulting from such losses.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel, the Intel logo, and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Contents

Tables

Revision History

Document Number	Revision Number	Description	Date
334467-001	1.0	• Initial release of the document.	June 2016

1 Introduction

Performance monitoring units (PMUs) are present in all modern Intel processor generations, allowing profiling utilities to characterize the interaction between software and CPU resources using a rich set of performance metrics. Profilers are critical tools for software to harvest optimal performance out of the CPU hardware.

The programming interfaces that profiling utilities use to access PMUs or related hardware resources consist of:

- A set of instructions (some require privilege access available only in kernel mode, like RDMSR, WRMSR).
- PMU configuration resources: these are typically Model Specific Registers (MSRs).
- Counter register resources: these can include performance counters in the PMU as well as other counter registers accessible as MSRs.

Traditionally, profiling utilities employ special device drivers operating with ring 0 privilege to configure the PMU, access counter registers, and handle interrupts if the profiler supports sampling (i.e. capture samples of incremental data at fine-grain intervals).

Some OS, such as Linux, provide API access for root privileged user programs to access privileged resources (such as MSRs). When a user program's profiling needs can be served by counting of events (without the need to capture incremental samples), it is often possible and desirable to implement the profiler as a ring 3 application to make use of these privileged APIs. This simplifies development and deployment of the profiler compared to the traditional approaches of a kernel based driver solution with a command line front-end parser.

For security reasons in multi-user OS, the OS only allows access to privileged resources by root users. This implies that the monitoring tool would run with full root rights and have rights to operate privileged resources (as permitted by those API) beyond just monitoring performance events.

To configure and use the PMU, read and write accesses to some PMU MSRs are needed by a user-space profiler. However, having full write access to the entire set of MSRs in a CPU can compromise the OS. Thus, full root rights and write access to full set MSRs should be selectively provisioned to a user-space profiler. On secured shared server systems or securely booted clients with secured kernels full MSR access is usually not available.

The goal of this white paper is to define a subset of MSRs and mechanism with the following in mind:

- Writes to the subset of MSRs are to configure performance metric selection and conduct monitoring of the counter registers, without changing any non-PMU states.

- Define write masks that are applicable to the subset of MSRs to ensure the user-space profiler operates within the intended monitoring mode (i.e. counting).
- A bridge between the OS-API requirement of full root rights and the desired non-root permission for user-space applications.
- Allow collecting performance metrics of the whole system, but do not modify any other state.

A specialized MSR access layer can then give the monitoring tool only access to this safe "monitoring only" subset of MSRs and allow it to run the monitoring as non-root, without risking compromising the system.

Note that monitoring access is still opt-in by the administrator and cannot be done without an explicit configuration change.

1.1 Scope

The scope is largely focused on monitoring for the processor core PMU. Intel platforms have additional PMUs outside the processor core such as the uncore or the chipset. Those are not covered by this white paper.

§

2 Implementation

2.1 Security Model

This white paper define a new “global monitoring only” privilege level for an application. The administrator has to explicitly grant this privilege level to an application. The privilege allows monitoring performance events on all processes of the complete system, but does not change any global state not accessible by an unprivileged application.

The privilege level gives read and write access to a limited number of MSRs in the logical processor and the physical package. Filtering of input settings specified by the application is written to the MSRs by a privileged software layer (kernel driver or a special secure access layer). The active settings of the MSRs reflect the configuration of the performance monitoring hardware.

Input from the non-root application to change any of the secured monitoring registers does not allow:

- Reading or writing any data in memory or in data registers.
- Triggering interrupts.
- Changing state of processes outside the monitoring tool.
- In general, the expectation of performance impact to the target system due to enabling monitoring hardware and the software layer is minimal.

Input from the non-root application permit the following changes to the secured monitoring registers:

- Selection of performance monitoring counter events which are supported by the PMU, as well as (optionally) conditioning of performance counter results (e.g. thresholding, edge triggering).
- This includes the ability to monitor events such as cache misses, branch mispredictions and other architectural and micro architectural events.

The administrator can choose whether ring 0 (kernel) or only all user mode can be monitored.

2.2 Access Layer Requirements

The secure access layer should implement the following functionality:

- Allow specific software access without requiring the software to run with full administrator rights.
- Allowing access to specific white listed MSR registers, as documented in this document.

- Enforce that some registers are read only and that some registers have bits write protected.
- Catching #GP General Protection faults when accessing MSRs and return an error.

2.3 Sharing Model

Write access to the PMU registers by one global monitoring software process can disturb other monitoring tools operating under the same system executive. To allow sharing between different monitoring tools the tool should follow the protocol specified in the Intel Performance Monitoring unit sharing guide (www.intel.com/sdm or https://software.intel.com/file/30388).

Generally this means checking enable bits for programmable counters and not changing the configuration if the counter is already running. Free running counters can be always shared, but should not be written to.

2.4 Architectural Perfmon vs. Model Specific

Some registers are architectural and can be discovered through the CPUID instruction. Other registers are model specific.

2.5 Counter Wrapping

With the secure access restrictions it is not possible to get an interrupt on counter overflow. Software instead needs to poll the counter registers in sufficiently short time intervals to accumulate values before they overflow.

2.6 List of Registers for Secure Access by User-Space Profilers

MSR registers available in Intel processors for user-space profilers via a secure access layer are listed below. Availability of a given MSR in an Intel processor is enumerated either by CPUID feature information or by model-specific signatures reported in Display_Family, Display_Model values returned in CPUID instruction leaf 1 function.

In general, only Intel processors with DisplayFamily = 0x6 are applicable targets of this paper. MSR information applicable to DisplayModel values of 0x1E, 0x1F, 0x1A, 0x2F, 0x25, 0x2C, 0x2E, 0x37, 0x4D, 0x4C, 0x1C, 0x26, 0x27, 0x36, 0x35, 0x2A, 0x2D, 0x3A, 0x3E, 0x3C, 0x45, 0x46, 0x3C, 0x3F, 0x3D, 0x47, 0x56, 0x4E, 0x5E, 0x57 are summarized by category.

Unless otherwise marked all bits in the register can be securely accessed.

Note: For more details on the individual registers, see the Intel® 64 and IA-32 Architectures Software Developer Manuals (www.intel.com/sdm).

Table 2-1. Configuration Registers for PMU and Non-PMU Counters

Name	Access	Address	Description	Scope	Comments
IA32_PERF_EVENTSELx	R/W	0x186+x, x = 0, n-1; n = CPUID.10: EAX[15:8]	Select performance monitoring events and associated configurations.	Thread	Ring 0 access mask 0xffa7ffff, otherwise 0xffa5ffff
IA32_FIXED_CTR_CTRL	R/W	0x38d	Configure fixed counters.	Thread	Ring 0 access mask 0x333, otherwise 0x111
IA32_PERF_GLOBAL_CTRL	R/W	0x38f	Global control to enable/disable fixed counters and performance counters.	Thread	Access mask 0xff000000ff
MSR_OFFCORE_RSP_0	R/W	0x1a6	Configure event-specific mask for OFFCORE_RSP_0 event.	Varies	Writing reserved bits may #GP;
MSR_OFFCORE_RSP_1	R/W	0x1a7	Configure event-specific mask for OFFCORE_RSP_1 event.	Varies	Writing reserved bits may #GP
IA32_PERF_CAPABILITIES	R/O	0x345	Enumerate Perfmon capability.	Thread	
MSR_RAPL_POWER_UNIT	R/O	0x606	Enumerate Granularity of RAPL Energy Status Counters.	Package	Not available to DisplayModels=0x1E, 0x1F, 0x1A, 0x2E, 0x2F, 0x25, 0x2C, 0x1C, 0x26, 0x27, 0x35, 0x36

Table 2-2. PMU Counter Registers

Name	Access	MSR Number	Description	Scope	Comments
IA32_PERFCTRx	R/W	0xc1+x, x = 0, n-1; n = CPUID.10:EAX[15:8]	Value of counter x associated with configured performance event.	Thread	
IA32_PMCx	R/W	0x4c1+x, x = 0, n-1; n = CPUID.10:EAX[15:8]	Full-width=writable counter x.	Thread	
IA32_FIXED_CTRx	R/W	0x309+x	Value of fixed counter x	Thread	

Table 2-3. Other Counter Registers[1]

Name[2]	Access	Address	Description	Scope	Comment
MSR_PKG_Cx_RESIDENCY	R/O	Varies by Available Cx	Duration in applicable package C states.	Package	See Chapter 35 of the Intel® 64 and IA-32 Architectures Software Developer Manual, Volume 3C (www.intel.com/sdm)
MSR_CORE_C1_RESIDENCY	R/O	0x660	Duration in core C1 state.	Core	Only in DisplayModels= 0x37, 0x4D, 0x4A, 0x5A, 0x5D, 0x4C, 0x5C, 0x5F
MSR_CORE_C3_RESIDENCY	R/O	0x3fc	Duration in core C3 states.	Core	Not available to DisplayModels=0x1E, 0x1F, 0x1A, 0x2E, 0x2F, 0x25, 0x2C, 0x1C, 0x26, 0x27, 0x35, 0x36, 0x37, 0x4D, 0x4A, 0x5A, 0x5D
MSR_CORE_C6_RESIDENCY	R/O	0x3fd	Duration in core C6 states.	Core	Not available to DisplayModels=0x1E, 0x1F, 0x1A, 0x2E, 0x2F, 0x25, 0x2C, 0x1C, 0x26, 0x27, 0x35, 0x36, 0x37, 0x4D, 0x4A, 0x5A, 0x5D, 0x4C, 0x5C, 0x5F
IA32_APERF	R/O[3]	0xe8	Actual performance clock count.	Thread	
IA32_MPERF	R/O[3]	0xe7	TSC clock count.	Thread	
MSR_PPERF	R/O	0x64e	Productive performance count.	Thread	Only on DisplayModels= 0x4E, 0x5E
MSR_PKG_ENERGY_STATUS	R/O	0x611	RAPL energy of the package.	Package	Not available to DisplayModels=0x1E, 0x1F, 0x1A, 0x2E, 0x2F, 0x25, 0x2C, 0x1C, 0x26, 0x27, 0x35, 0x36
MSR_SMI_COUNT	R/O	0x34	System management interrupt count.	Thread	Not available to DisplayModels= 0x1C, 0x26, 0x27, 0x35, 0x36

NOTES:

1. A machine readable version of this table can be downloaded from https://download.01.org/perfmon/secure-pmu-access-1.0.csv.
2. The availability and address of some MSRs listed in this table may vary by DisplayFamily_DisplayModel signatures; refer to Chapter 35 of the Intel® 64 and IA-32 Architectures Software Developer Manual, Volume 3C (www.intel.com/sdm).
3. Recommended access layer to enforce read-only for better sharing, however allowing write access does not compromise security.