Libmsr Extensions: Enabling User Monitoring & Control of Hardware Features

Stephanie Labasan, Barry Rountree *Lawrence Livermore National Laboratory, Center for Applied Scientific Computing*

Model specific registers (MSRs) allow privileged access to architecture-level features and controls. These features are encoded in a 64-bit value, the definition of which is platform-specific, making it difficult on the user to determine which definition to use. Libmsr translates from raw MSR bits to various architecture-level features and controls of Intel processors and vice versa. We extend Libmsr to target additional platforms and to leverage their new features, giving the user more control over the underlying hardware.

Another Performance Tool?

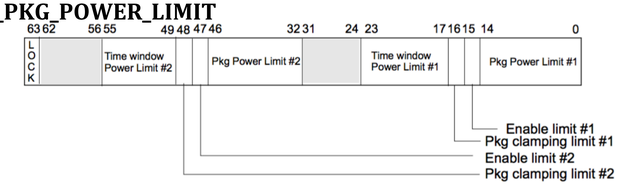
Model specific registers (MSRs) provide privileged access to platform feature through a standard set of instructions known as rdmsr/wrmsr. The 64-bit register value encodes a set of human-readable values based on defined bit fields. Unfortunately, the MSR definitions are architecture specific, making it difficult for the user to fully leverage the interface. Libmsr decodes the registers based on the architecture, and has been extended to leverage new processor features with fine-grained control.

MSRs Expose Several HW Features

- Measure processor-level energy usage
- Perform simultaneous reads/writes, minimizing perturbation to application
- Count the number of instructions retired for each protection ring
- Enforce a power limit on the processor, DRAM, uncore
- Set the operating frequency of a processor or core
- Enable/disable Intel Turbo Boost
- If a thermal event is triggered, determine which core on which processor asserted the event

Libmsr Enables New Research Directions

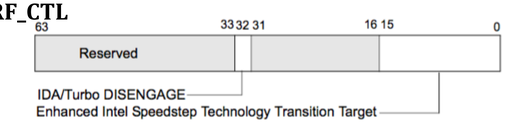
MSR_PKG_POWER_LIMIT



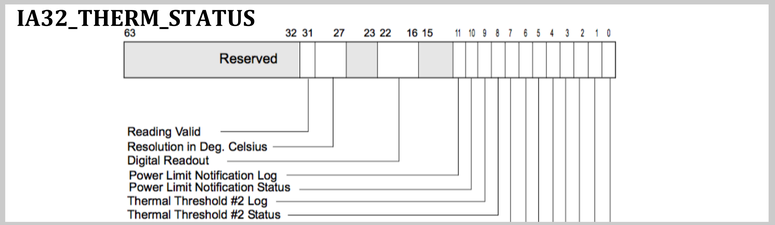
Can we manage application load imbalances by enabling software to dynamically change operating frequencies at the core-level and socket-level? ←

→ At Exascale, we assume node-to-node behavior will vary due to manufacturing variations of the processors contained within. Can we minimize this heterogeneity by enforcing different socket-level power limits?

IA32_PERF_CTL



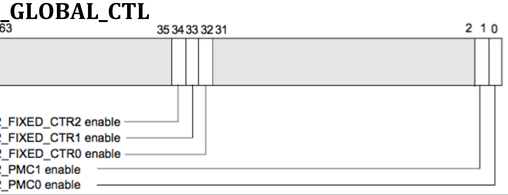
IA32_THERM_STATUS



→ Is the processor running at a high temperature, resulting in it being throttled? Is there a particular core on the processor that is causing high temperatures? Is there sufficient thermal headroom to be granted Turbo Boost frequencies, if they are enabled?

How compute-bound or memory-bound is the application on a per-core basis? What is the memory latency at the core-level? ←

IA32_PERF_GLOBAL_CTL



Libmsr Current State

- Support for Intel Xeon platforms Sandy Bridge, Ivy Bridge, Haswell
- Translate between raw bits to bit fields to human-readable values
- Read/write various MSRs Integration with msr-safe, enabling rdmsr/wrmsr instructions in ring 3 (user mode)
- Access to data from PCI, *e.g.*, integrated memory controller (IMC), QPI, and PCU

Libmsr Future Work

- Additional support for other Intel Xeon (Broadwell, Skylake) and Intel Xeon Phi (Knights Corner, Knights Landing) platforms
- Enable interface for other manufacturers, *e.g.*, AMD, Nvidia
- Refactoring functions to be more general-purpose
- Increasing granularity of control where necessary

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Figures courtesy of Intel's® 64 & IA-32 Architectures SW Developer's Manual

LLNL-POST-698628