Whitelisting MSRs with msr-safe

Kathleen Shoga, Barry Rountree,
Martin Schulz, Jeff Shafer

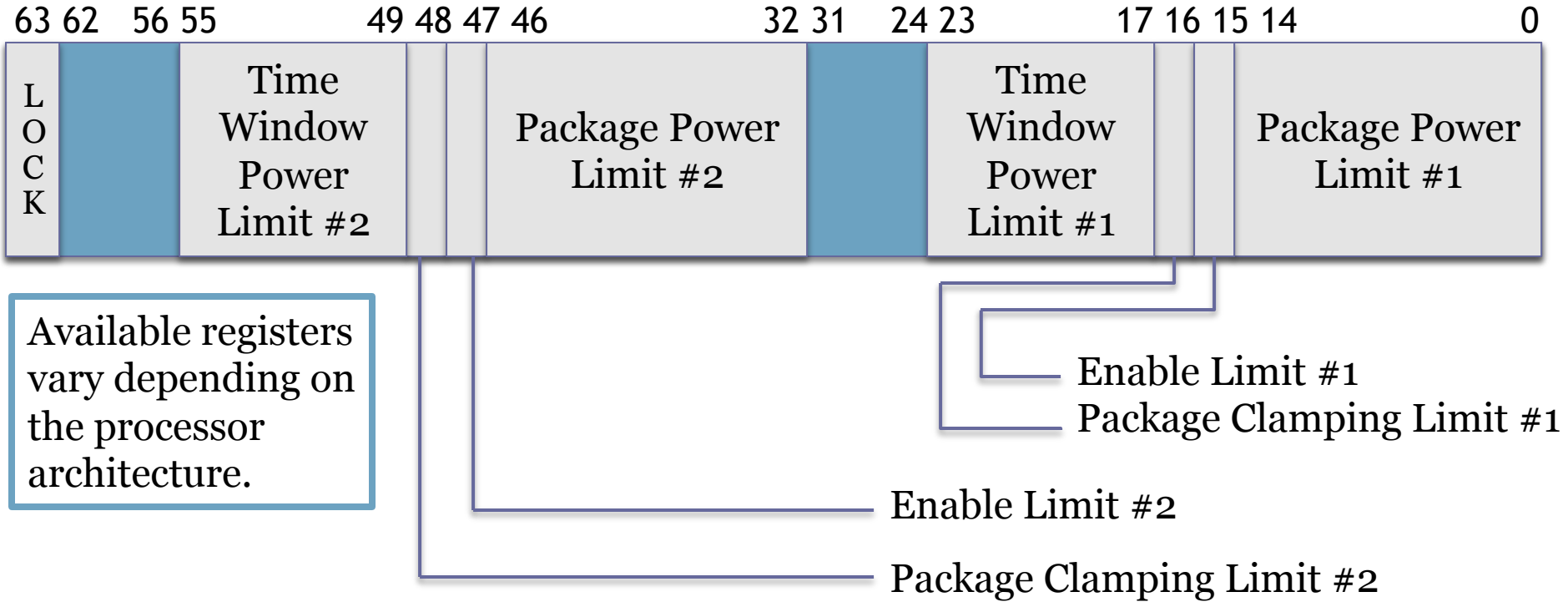
Email: shoga1@llnl.gov

LLNL-PRES-663879

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

MSRs

- Model Specific Registers
- Intel Architectures supported by msr-safe: Sandy Bridge, Ivy Bridge, Haswell…



http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-3b-part-2-manual.pdf

Access to MSRs is Critical

- Processors provide low-level access to critical information and settings via MSRs
  - Power – package (socket) and dram power
  - Thermal – core, package in deg C
  - Performance Counters –
    - Effective frequency
    - Instructions retired
- Enables studies on:
  - Advance performance measurements
  - Power measurements
  - Control for over-provisioned systems

Accessing MSR Data

- Special instructions in kernel space:
  - rdmsr, wrmsr
- User level access through msr kernel module
  - Provides filesystem interface to all of the MSRs through /dev hierarchy
  - No finer-grained permissions

Problem to solve

Site-specific policy

- No access/control for regular users in existing interfaces due to:
  - Security Concerns
    - Full access to MSRs could allow you to “root” the machine
      - Pointer to the vector of hardware interrupt handlers is held in an MSR
  - Permissions
    - All or nothing access
  - Complexity in Registers
    - Error prone

Our Initial Solution

- MSR kernel module + file permissions
- Only allow “trusted” users to have access

Problem

- Updated kernel module required “capability” check for SYS_RAW_IO (**not** MSR specific)
  - However users/binaries with SYS_RAW_IO could also:
    - Perform I/O port operations
    - Create memory mappings below value specified by /proc/sys/vm/mmap_min_addr

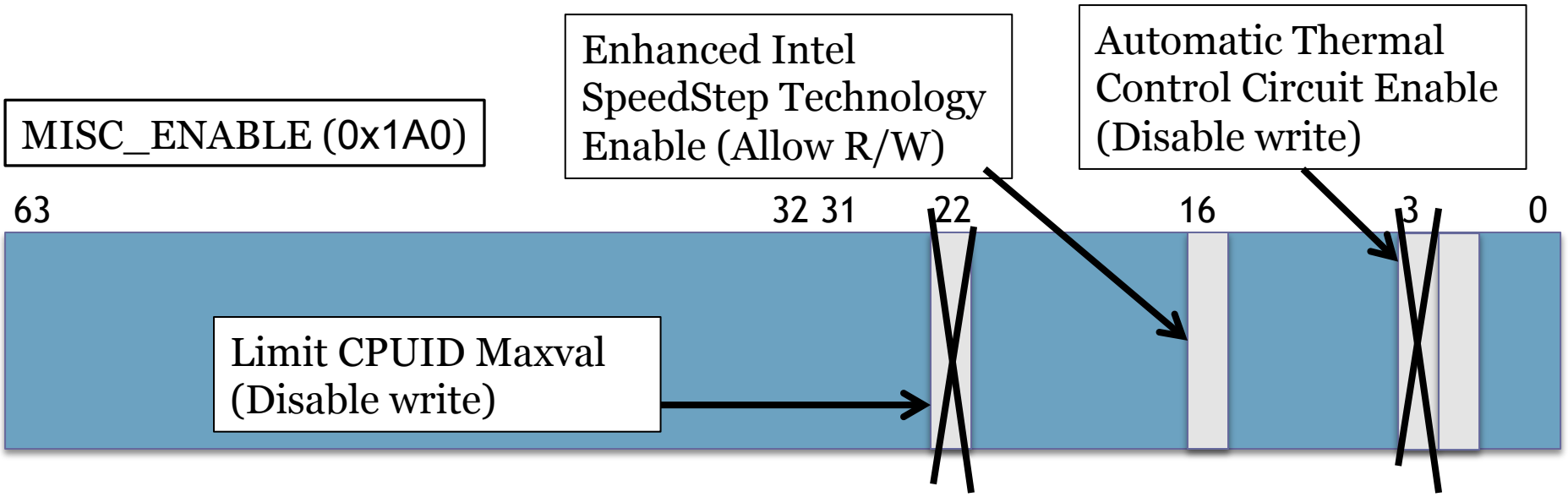
Our New Solution Part 1

msr-safe kernel module + whitelist

- msr-safe kernel
  - Same underlying structure as generic msr kernel module
  - No capabilities check
  - Use whitelist instead
  - Access through /dev/cpu/#/msr_safe

Our New Solution Part 2

- Whitelist **instead** of capabilities check
  - Bit level granularity
  - Access to power, thermal, and performance counters/controls
  - Formatted with tables to match Intel manuals (relatively easy to add new registers)



API

- Compile and insert the module
  - Processor architecture is detected at compile time
  - Whitelist created based off of architecture
- Version number exported to
  - /sys/class/smsrs/version
- List of available registers in the whitelist
  - /sys/class/smsrs/avail

Currently Whitelisted (Ivy Bridge)

0x618 SMSR_DRAM_POWER_LIMIT
0x619 SMSR_DRAM_ENERGY_STATUS
0x61B SMSR_DRAM_PERF_STATUS
0x61C SMSR_DRAM_POWER_
0x010 SMSR_TIME_STAMP_C
0x017 SMSR_PLATFORM_ID
0x0C1 SMSR_PMC0
0x0C2 SMSR_PMC1
0x0C3 SMSR_PMC2
0x0C4 SMSR_PMC3
0x0C5 SMSR_PMC4
0x0C6 SMSR_PMC5
0x0C7 SMSR_PMC6
0x0C8 SMSR_PMC7
0x0E7 SMSR_MPERF
0x0E8 SMSR_APERF
0x186 SMSR_PERFEVTSEL0
0x187 SMSR_PERFEVTSEL1
0x188 SMSR_PERFEVTSEL2
0x189 SMSR_PERFEVTSEL3
0x18A SMSR_PERFEVTSEL4
0x18B SMSR_PERFEVTSEL5
0x18C SMSR_PERFEVTSEL6
0x18D SMSR_PERFEVTSEL7
0x198 SMSR_PERF_STATUS
0x199 SMSR_PERF_CTL

0x19A SMSR_CLOCK_MODULATION
0x19B SMSR_THERM_INTERRUPT
0x19C SMSR_THERM_STATUS
MSR_MISC_ENABLE
MSR_TEMPERATURE_TARGET
MSR_OFFCORE_RSP_0
MSR_OFFCORE_RSP_1
MSR_ENERGY_PERF_BIAS
MSR_PACKAGE_THERM_STATUS
MSR_PACKAGE_THERM_INTERR
SMSR_FIXED_CTR0
0x30A SMSR_FIXED_CTR1
0x30B SMSR_FIXED_CTR2
0x345 SMSR_PERF_CAPABILITIES
0x38D SMSR_FIXED_CTR_CTRL
MSR_PERF_GLOBAL_STATUS
MSR_PERF_GLOBAL_CTRL
MSR_PERF_GLOBAL_OVF_CTRL
MSR_PEBS_ENABLE
MSR_PEBS_LD_LAT
MSR_RAPL_POWER_UNIT
MSR_PKG_POWER_LIMIT
MSR_PKG_ENERGY_STATUS
0x614 SMSR_PKG_POWER_INFO
0x638 SMSR_PP0_POWER_LIMIT
0x639 SMSR_PP0_ENERGY_STATUS

Using MPERF and APERF, you can calculate effective frequency

THERM_STATUS can give thermal information per core

FIXED_CTR0 provides number of instructions retired

Using POWER_UNIT an POWER_LIMIT, you can set power limits on a per package (socket) level

- List can easily be changed before compiling

Convenient access through libmsr

- Companion library developed at LLNL
  - Call high level library functions such as:
    - dump_thermal_terse()
    - dump_rapl_limit(…)
  - Build your own with easy to use:
    - Structs
    - Lower level functions
  - The library will do:
    - Error Checking
    - Low Level Work

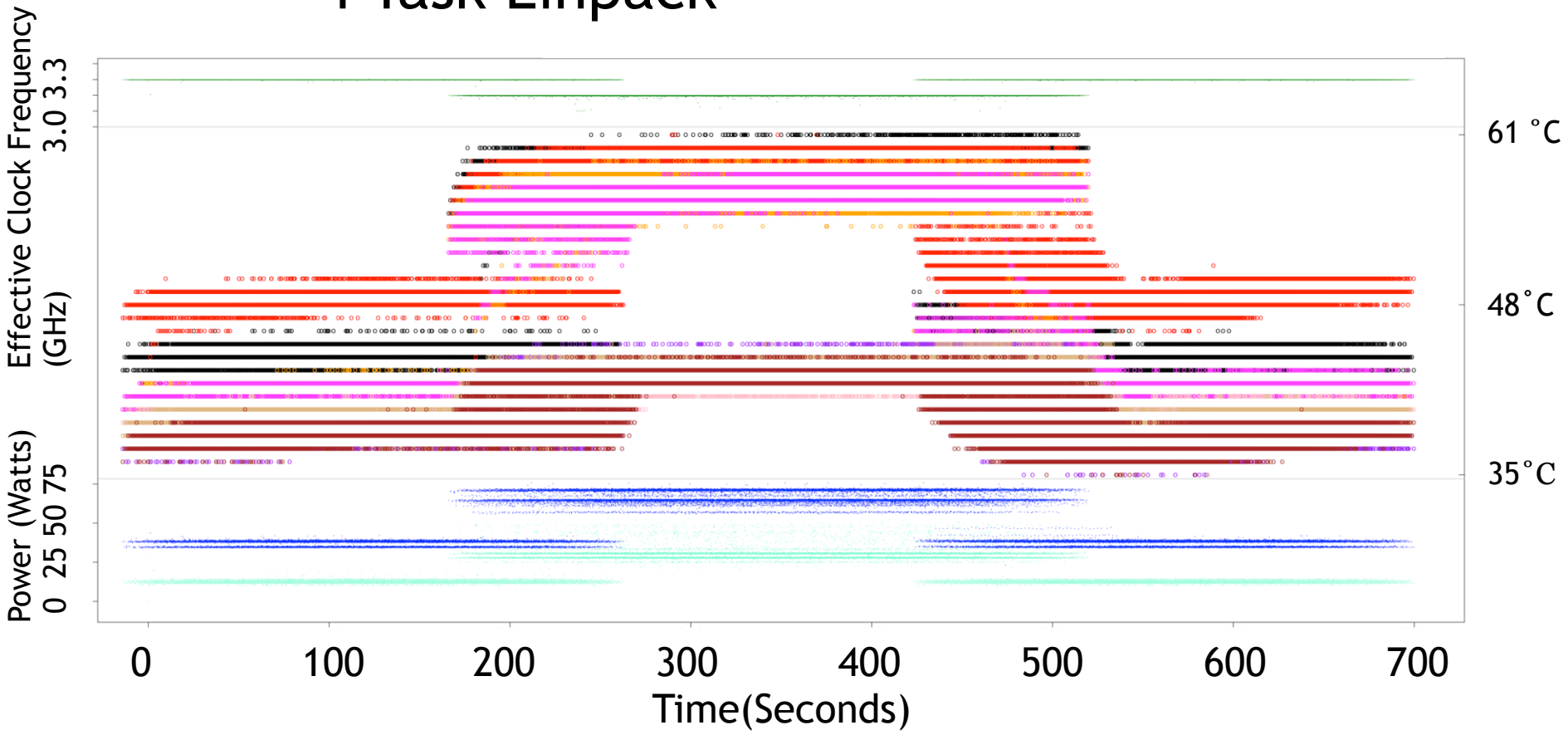
Successes in Deployment

- Production machines: Cab (at LLNL)
  - Intel Xeon E5-2670 Processors (Sandy Bridge)
  - 1,296 nodes
  - 16 cores per node
- In TOSS (Tri-Lab Operating System Stack)
- On LANL TLCC2 machines
  - Tri-Lab Linux Capacity Cluster 2

Case Study: Thermal Measurement/Data

Green: Effective Frequency
Blue: Power
Package- Dark
DRAM- Light
Other: Core Temperatures

4 Task Linpack



Future Work

- Add registers to the whitelist
  - Some registers have unreliable bits
  - Find which MSRs could expose security risks
- Update register tables as new processors become available
  - i.e. Haswell
- Integration with PAPI (In progress)

Summary

- Access to MSRs is critical for:
  - Power and Performance measurements
  - Power capping
- The msr-safe kernel + whitelist enables:
  - Safe use of MSRs for regular users
  - Easy to use API
  - Bit level control for security

Open Source

https://github.com/scalability-llnl/msr-safe
https://github.com/scalability-llnl/libmsr