Out of Band Performance Monitoring of Server Workloads

Leveraging RESTful API to monitor compute resource utilization and performance related metrics for server performance analysis.

Scott Faasse
Server Power and Performance
Hewlett Packard Enterprise
Houston, TX USA
scott.faasse@hpe.com

James Bucek
Performance Engineering
Hewlett Packard Enterprise
Houston, TX USA
james.bucek@hpe.com

David Schmidt
Performance Engineering
Hewlett Packard Enterprise
Houston, TX USA
d.schmidt@hpe.com

ABSTRACT

Performance monitoring is a useful tool to leverage when additional insight is needed or warranted when evaluating the performance results of a compute solution or benchmark. More often it seems, the use of performance monitoring is an afterthought and is utilized only when unexpected results are encountered. Given that most methods for implementing performance monitoring require running additional applications or kernel code in parallel with the application or benchmark itself, it is understandable that there may be a bias against or a reluctance to leverage performance monitoring capabilities throughout the performance evaluation period as well as extending its use into a production environment.

In our paper, we'll introduce a performance monitoring architecture that leverages an out of band (OOB) approach for measuring key server resource performance metrics. We will demonstrate that this approach has zero impact to the performance of the workload running on the server itself and is suitable for use in a production environment. Although the out of band approach inherently has limits to the amount of information that can be gathered for performance analysis, we will demonstrate the usefulness of the information that is available today in debugging performance related issues.

CCS CONCEPTS

• General and reference → Measurement, Performance

KEYWORDS

Benchmarking; RESTful API; Performance Monitoring; Out of Band; Performance Analysis; Servers

ICPE '20, April 20–24, 2020, Edmonton, AB, Canada.

ACM ISBN 978-1-4503-6991-6/20/04...$15.00.
DOI: http://dx.doi.org/10.1145/3358960.3375795

ACM Reference format:

Scott Faasse, James Bucek, and David Schmidt. 2019. Out of Band Performance Monitoring of Server Workloads: Leveraging RESTful API to monitor compute resource utilization and performance related metrics for server performance analysis. In *Proceedings of ACM/SPEC Int. Conference on Performance Engineering (ICPE'20), April 20 – 24, 2020, Edmonton, Canada.* ACM, New York, NY, USA, 8 pages. http://dx.doi.org/10.1145/3358960.3375795

1 Introduction

Resource monitoring is not a new concept in the performance analysis and benchmarking community. The concepts behind monitoring hardware resource utilization and software interaction is well documented and beyond the scope of this paper. Until recently however, most approaches to performance monitoring have been accomplished by leveraging tools that have been written to run as applications or kernel modules designed for a given operating system and hardware architecture. Examples include *top, sar,* and *turbostat* on Linux based OSes as well as *perfmon* and *Intel® VTune* on Microsoft based OSes, to name just a few. These popular tools must be run in parallel with the benchmark/workload and analysis. These tools, while useful, also come with their own performance tax. Since they are executed within the same environment, the tools unavoidably use the same compute resources (CPU time, memory bandwidth, etc) as the workloads being analyzed. This arrangement has the potential to skew the performance results (usually negatively) and as a result, are often not employed in production, limiting their use as a first approach to performance analysis.

Over the past several years, server hardware vendors have been gradually including basic performance monitoring capabilities for their out of band management solutions. As far back as the mid to late 2000s, HP Servers began offering methods for their customers to access basic CPU utilization and average frequency. Although it offered out of band access to CPU metrics, the information was gathered using host side methodologies and was not completely non-intrusive with applications running on the server itself.

More recently, with the release of servers supporting Intel Xeon Scalable Processors in 2017, several server vendors began offering

a more robust approach to out of band performance monitoring that included expanded metrics that provided at least CPU utilization, I/O Utilization, and Memory Bus utilization metrics. These metrics are not only delivered out band through the base server management controller, but they are also measured outside the scope of the processor's execution pipeline, meaning that they are nearly 100% non-interfering.

One such server provider, HPE, offers the ability for their users to gather performance monitoring data from their servers leveraging an open standard set of APIs [1]. In this paper, we will utilize this capability to explore the following:

1) Obtain OOB performance telemetry by leveraging open standard APIs
2) Demonstrate "do no harm" (DNH) testing for compute intensive workloads
3) Demonstrate DNH testing for power efficiency workloads
4) Provide example performance analysis results from looking at various benchmark workload examples.

2 Accessing Performance Monitoring Data

Accessing performance monitoring data out of band, on a server, requires management access to the server's base management controller (BMC.) The BMC is responsible for accessing server component (CPU, Memory, Fans, I/O devices) sensor telemetry if and when it is made available by the component vendor. When telemetry does exist, the BMC can optionally provide that data to customers through various closed or open interfaces.

Since the BMC executes separately from the host (CPU, Memory, I/O, Operating System), requests to the BMC should not interfere with workloads running on the host – specifically when requests come via the external network. We will test/prove this theory in Section 4.

2.1 Redfish (RESTful API)

The DTMF forum (an industry member working group) publishes several open standard interfaces for managing information technologies. The Redfish [2] specification provides an open standard interface for interacting with a server's BMC via simple HTTP style communication (RESTful API). The specification outlines the methods for interacting with BIOS configuration, plug-in devices configuration, basic server features, as well as server telemetry.

We will leverage the telemetry services functionality that is defined in the Redfish specification and is implemented on the latest server offerings provided by HPE. The telemetry provided by HPE ProLiant and Synergy servers [3] support access to the following performance related sensors (metrics) in 10 minute, 1 hour, 24 hour, and 1 week reports:

1. CPU Utilization
2. Memory Bus Utilization
3. I/O Bus Utilization
4. CPU Interconnect Utilization
5. Average CPU frequency
6. Average CPU power
7. Processor Jitter (frequency changes)

2.2 Alternative methods.

Beyond the Redfish interface, there exist other methods to access OOB performance data. Several server vendors, including HPE, also provide a graphical user interface via a web page (Web GUI) hosted by the BMC itself. These interfaces typically provide current performance status as well as the ability to view performance graphs over selected time frames. While these interfaces simply provide another access mechanism to the data, leveraging the interface typically doesn't scale well as it requires human interaction (i.e. not scriptable.)

Redfish and the RESTful API is not the only server programmable interface to obtain management data. The Intelligent Platform Management Interface (IPMI) [8] also provides a hardware/software management architecture that could be used to achieve similar results. This specification however lacks clear standard definitions for reporting performance metrics. Implementations that leverage IPMI for telemetry are likely to be OEM specific and vary between vendors.

Regardless of which programming interface is used, it is important to distinguish the hardware interface that will be leveraged. For instance, with Redfish, you can access the data from the host side or out of band. While the data is still collected by the BMC via out of band mechanisms, there are allowances for the tools to run on the server host itself. It should be obvious that when accessing data from a host side tool, some of the advantages will be lost. Specifically the host side performance penalty when comparing to existing OS level tools that gather similar data. For our research, we will focus on using a remote connection (i.e. network – TCP/IP) to collect performance data.

3 Experimental Setup

To demonstrate the usefulness of leveraging the OOB performance monitoring, we will look at a setup in Figure [1], where a client running on hardware separate from the server under test, will request telemetry data from a server that supports reporting performance telemetry via the Redfish interface. The client setup can be as simple or as complex as the solution dictates (monitoring a single server or scaled out to monitoring 1000s of servers). The data is stored on the remote client and reports can be generated for a particular server over a given time-frame.

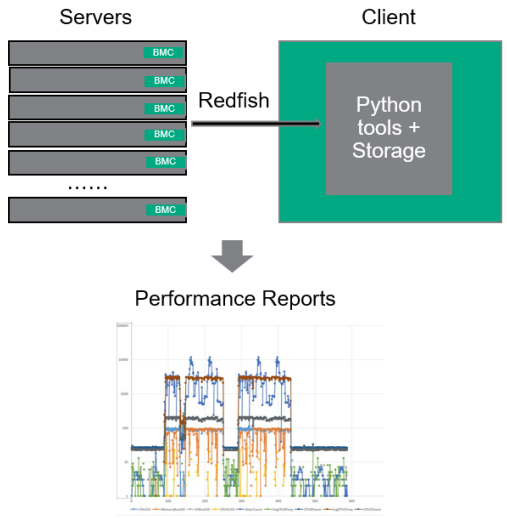

Figure 1. Experimental setup for gathering server performance data

Our DNH testing leveraged a Windows based client with Python 3.4 installed, with the *requests* library installed in order to leverage the HTTP nature of RESTful API accesses. The client was set up to attempt to create a denial of service type of test. Thousands of back to back requests (occurring every X {timeframe}) for telemetry were initiated to ensure that the server's host side performance wasn't negatively impacted.

For scale-out testing, we leveraged a server running Linux, with Python 3.7, and a structured database to store telemetry for dozens of managed servers in a lab that is setup to mimic a datacenter-like environment. The managed servers were running a variety of benchmarks including SPEC CPU® 2017, SPECjbb®2015, SPECvirt_sc® 2013, etc. Given that these servers are used for publication of benchmark results for HPE, this is considered a "production" level environment where any impact to solution performance would have a negative impact to our business. Equally important in benchmark results analysis, the information gleaned from the performance reports would potentially be considered important and business critical.

It is important to note that for 10 minute monitoring reports, the data is averaged at 20 second intervals. For 1 hour, 24 hour, and 1 week reports, the data is averaged at 20 second, 5 minute, and 30 minute averages respectively. For DNH testing, we requested reports from the test server for all 4 intervals. For scale out testing, only 24 hour data was captured to evaluate the usefulness of leveraging the longer 5 minute averages for the metrics.

4 Evaluation and Analysis

We break down our evaluation into two categories: DNH measurement verification, and workload profile characterization. For both categories, the target server's configuration was unchanged during the measurements, except where noted.

4.1 Do No Harm Verification

For the first DNH test, SPEC CPU 2017 was run on a 2-processor HPE ProLiant DL380 Gen10 server running SLES 15 SP1. A single iteration of SPECrate®2017_int_base and SPECrate®2017_fp_base were run and an estimated overall metric was calculated from the results. A second measurement was then started. While the second run was executing, 7000 consecutive queries were made to the RESTful API to collect performance monitoring data. An estimated result was calculated from the second measurement's results. Table [1] shows the SPEC CPU 2017 metric estimates for each test. The results shown in the table are pulled from an actual SPEC CPU 2017 run. They are labeled as metrics since they show more detail than the standard FDR and are not published results.

SPEC CPU 2017 metrics	Server under no external monitoring	Server under high external monitoring
SPECrate2017_int_base (est.)	299.33	300.40
SPECrate2017_fp_base (est.)	255.20	255.54

Table 1. Do no harm validation results using SPEC CPU 2017 as workload

The performance difference between the unmonitored and monitored measurements was less than 0.5%. This difference is well within the run-to-run variation of the benchmark.

The second DNH test was run using the SPECpower_ssj2008 benchmark. In this test case, both server power and workload performance are captured. A 2-processor HPE ProLiant DL380 Gen10 server running Windows Server 2012 R2 Datacenter was used for the test. As with the first test, an initial measurement was made by running the benchmark with no external monitoring of the server. A second measurement was performed while 1000 consecutive queries were made to the RESTful API. Table [2] shows the results for both measurements.

SPECpower_ssj2008 metrics	Server under no external monitoring	Server under high external monitoring
SPECpower_ssj2008 result (overall ssj_ops/watt)	11,022	11,052
Server power at 100% load (W)	465	465
ssj_ops at 100% load	5,304,235	5,307,336
Server power at idle (W)	55.9	55.9

Table 2. Do no harm validation results using SPECpower_ssj2008 as workload

Table [2] shows that the difference in power and performance between the unmonitored and monitored measurements was less than 0.5%, within the run-to-run variation of the benchmark.

Both of the DNH test cases demonstrate that the impact of accessing the OOB performance monitoring data is negligible to the system performance. Additionally, the SPECpower_ssj2008 test case shows the effect of OOB performance monitoring to the system power is likewise negligible.

4.2 SPEC CPU 2017 run analysis

In this section we will show OOB performance measurements gathered while running the SPECrate2017 benchmark suites. The benchmarks were run on the same 2-processor server running SLES15 SP1 as was used in the DNH tests. The SPEC CPU 2017 benchmark harness was configured on the server as was the compiler needed to create the benchmark suites' workload binaries. SPECrate2017_int_base was run first, which required the compilation of the workload binaries. Immediately following the benchmark's completion, SPECrate2017_fp_base was run; its workload binaries were also compiled at the start of the measurement. Finally, both the integer and floating point suites were then immediately run again. For all 4 measurements, a single execution of each base workload module was performed. Figures [2] and [3] show the plotted data gathered from the OOB performance monitor during the runs. Figure [2] shows the utilization statistics and Figure [3] shows the processor characteristics. Note that the vertical axis for Figure [3] is logarithmic in order to clearly see all plots.

The first SPECrate2017_int_base run started at 20:15:19 and completed at 22:00:26. The first SPECrate_fp_base run started immediately afterward and completed at 02:58:11. The second sets of runs completed at 04:25:51 and 08:22:55, respectively.

The periods of lower CPU utilization coincide with the compilation phases for the first executions of the SPECrate2017_{int,fp}_base suites. The difference in the run profile of the benchmarks run with un-compiled and precompiled executables is highlighted in Figure [4], which shows the side-by-side comparison of the CPU utilization during each run.

Examining the performance counter data reveals a number of interesting insights into the benchmarks' utilization patterns. It is clear that neither benchmark heavily taxes the IO subsystem, as its utilization rarely shows any significant usage. Likewise little inter-socket IO traffic is seen, as represented by the CPU Int Con utilization graph in Figure [2], This is indicative of highly NUMA-aware workloads, in this case a well-tuned SPEC CPU 2017 configuration. Patterns can also be seen between the two sets of benchmark runs. Similar peaks and dips can be seen at the same point during each set of runs in the CPU, memory, and CPU interconnect utilizations, as well as the Jitter Count and CPU0 Power consumption. These peaks and valleys represent the differences in resource utilization of the individual workload modules within the benchmark suites. Correlating the usage patterns to the specific workload modules can provide insight into the modules' bottlenecks and provide guidance to optimization efforts. Some of the counters seen in Figures [2] – [5] could be obtained from other methods such as in band performance monitoring software within an operating system. Other metrics, such as Jitter Count, are less easily obtained through alternative methods and may require tools that impact the performance being measured.

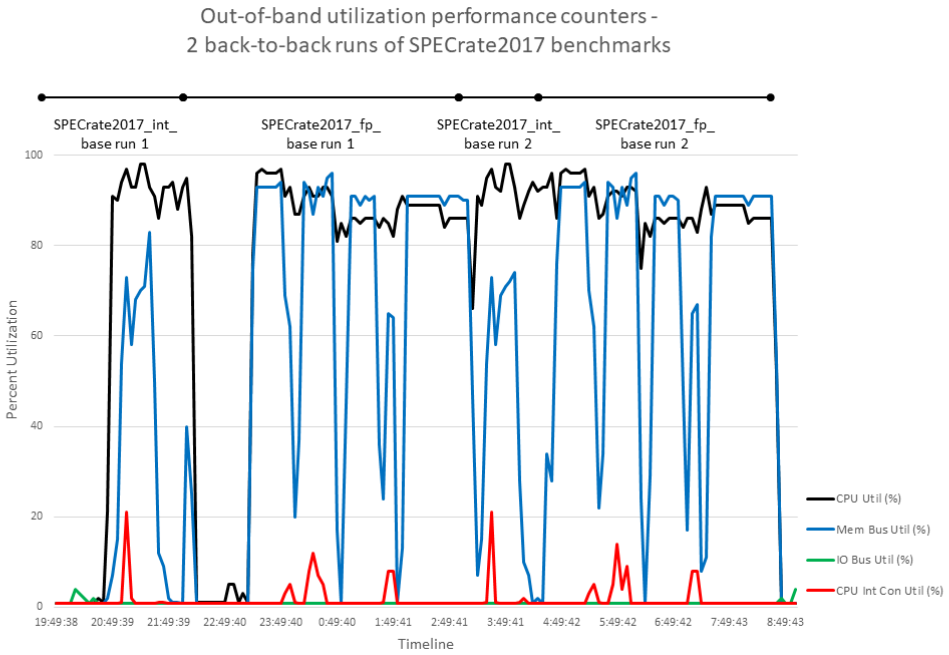

Figure 2. Out of band utilization performance counters

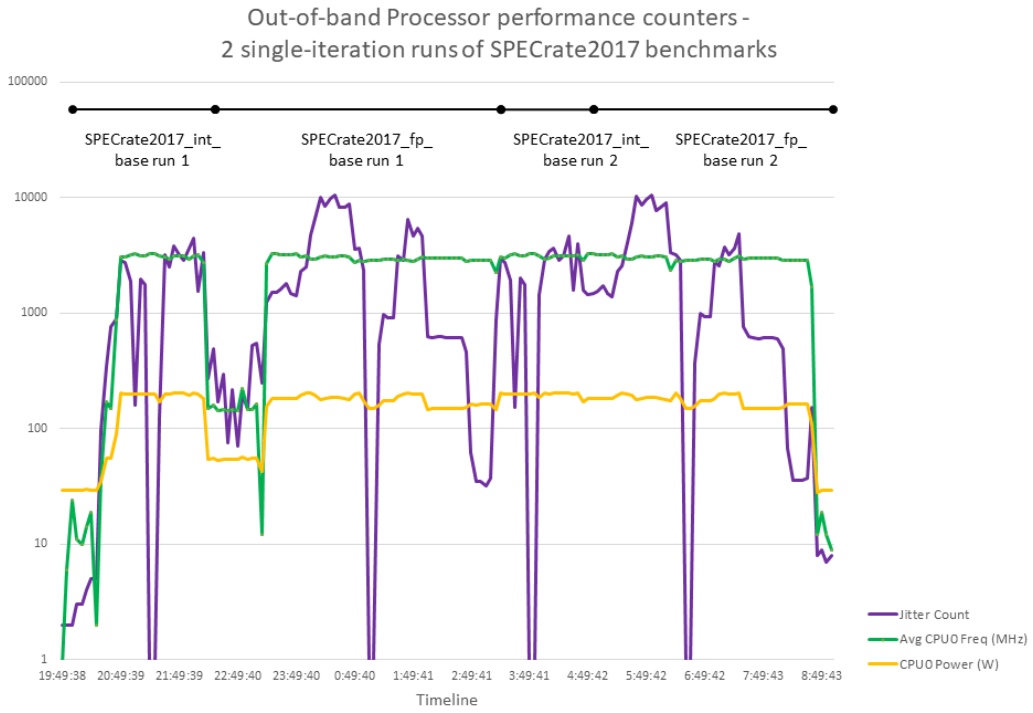

Figure 3. Out of band CPU performance counters

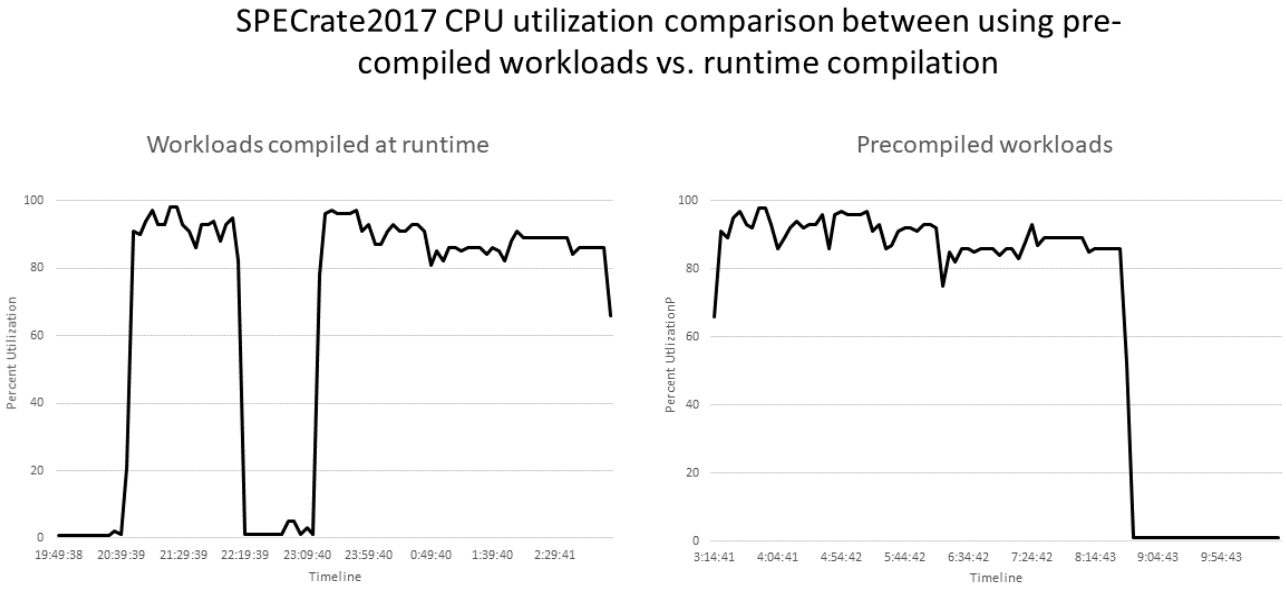

Figure 4. CPU utilization comparison of SPECrate2017 measurements using precompiled workload executables vs. compiling at runtime

4.3 Processor Jitter on SPECjbb2015

For the final test case we utilized the SPECjbb2015 benchmark to demonstrate the effects of setting a BIOS option that eliminates computation jitter due to processor frequency transitions. A BIOS option on newer HPE servers called Processor Jitter Control [4]. The option can be set to find the maximum processor frequency achievable where no frequency variance occurs due to power or thermal constraints. The top frequency for the processor is set to this value. In this test, a SPECjbb2015 Multi-JVM benchmark was run while the Jitter Control was enabled and then a second run was executed with the Jitter Control disabled. The results are shown in Figure [5]. As can be seen, there are no Jitter counts during the first SPECjbb2015 run, where Jitter Control was enabled. The second run show considerable jitter counts throughout the run, with the value increasing as the processor utilization increases.

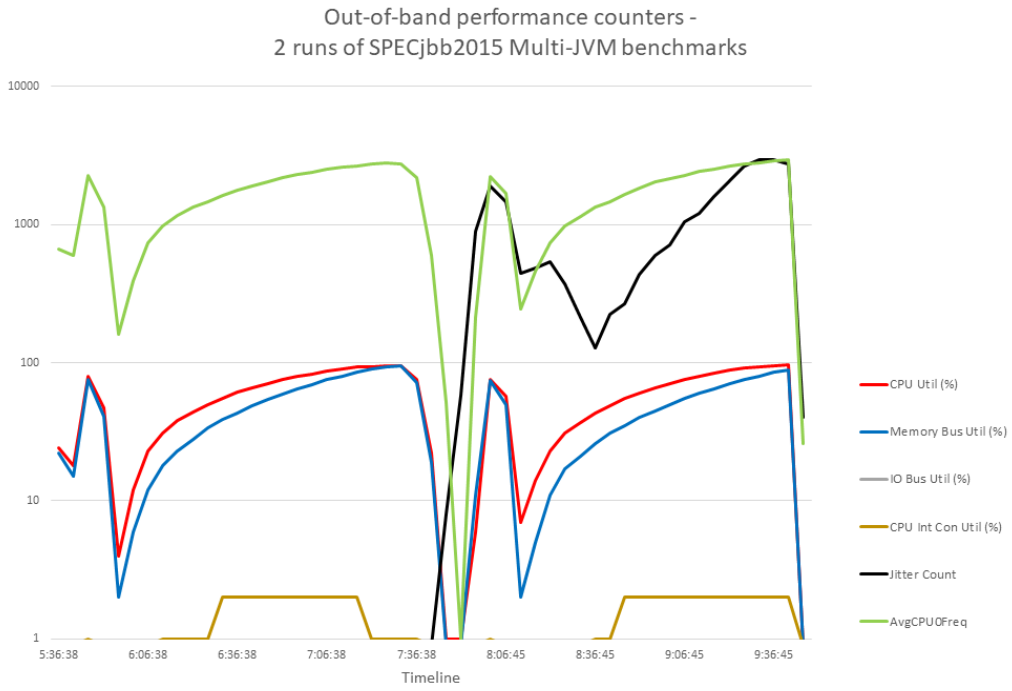

Figure 5. Out-of-band performance counters for back-to-back SPECjbb2015 benchmark runs with different Jitter-smoothing options

5 Conclusions

The goal of our research was to assess the viability of leveraging the out of band performance monitoring capabilities in a production environment. The main conclusions that we wanted to be able to draw were whether or not the access mechanisms would have a negative impact to overall system performance (as they had been in the past) as well as assess the usability of the data that could be mined.

Based on our experience with running benchmarks in a competitive environment, where the results will have a direct impact on overall business success and market growth, we feel that the experimental test environment adequately portrays a "production" environment. Our do no harm testing included a highly competitive compute benchmark where as little as 1% impact to host side performance can mean the difference between 1st and 3rd place in market leadership. The results of the do no harm testing with SPEC CPU 2017 demonstrated that out of band performance monitoring that is available today do not interfere with host side system performance. Since there isn't an impact in performance such as might be seen when running tools that run on the host side, we consider this a major advantage to the out of band approach. A user of this model would essentially get performance data monitoring for "free"; meaning that there isn't a performance overhead expense to obtain it.

When looking at the results of the do no harm testing for SPECpower_ssj2008, we also wanted to make sure we would not take a power efficiency hit. Given that the results show that OOB accesses to the data did not impact the power (or the performance) levels we are concluding that the OOB accesses to performance data as outlined in the paper, do not have a measureable impact to either system power or performance. This differs from traditional in band measurement, where we have observed

periodic sampling of performance data throws a system out of idle, and as a result, increases idle power.

Lastly, we acknowledge that the quantity and quality of data from OOB accessible methods are not as richly featured as many of those available in band; however, we did perform investigations to verify that the data we do have access to can demonstrate value. Our two cited examples (compile times with SPEC CPU 2017 and jitter on SPECjbb2015) are just a few simple scenarios where the data provided can help performance minded engineers correlate performance issues back to system behavior. As an aside, the compile time issue was discovered during our initial analysis of the first run of data in the experimental setup. The long delay was not well understood initially and the ability to visualize the gaps, and correlate the time stamps back to the benchmark logs proved extremely useful into understanding what was being observed. We believe that there is value beyond what we demonstrated here, and additional research will continue.

6 Acknowledgment

The authors wish to acknowledge current and past members of the SPECpower Committee, SPEC CPU Committee, and SPEC Research Power Working Group who have contributed to the design, development, testing, and overall success of SPEC CPU 2017. SPEC, SPECrate, SPECspeed, SPEC CPU, SPECpower, and SPECjbb are registered trademarks of the Standard Performance Evaluation Corporation. All rights reserved; see spec.org as of 12/12/2020.

REFERENCES

[1] Hewlett Packard Enterprise (2019). Restful Interface Tool 2.5.0 User Guide,. https://hewlettpackard.github.io/python-redfish-utility/#overview. .
[2] DMTF's Redfish specification. https://www.dmtf.org/standards/redfish
[3] Performance Monitoring on HPE servers https://community.hpe.com/t5/Servers-The-Right-Compute/Server-performance-monitoring-made-easy-with-HPE-iLO-5/ba-p/7048920#.XZS6ADaWzA0
[4] UEFI System Utilities User Guide for HPE ProLiant Gen10 Servers and HPE Synergy:https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-a00016407ja_jp
[5] SPEC CPU 2017: www.spec.org/cpu2017
[6] SPECvirt_sc2013: www.spec.org/virt_sc2013
[7] SPECpower_ssj2008: www.spec.org/power_ssj2008
[8] SPECjbb2015: www.spec.org/jbb2015
[9] Intelligent Platform Management Interface https://www.intel.com/content/www/us/en/products/docs/servers/ipmi/ipmi-home.html