Measuring and Mitigating Processor Performance Inconsistencies

Kevin Stroup
Cray, Inc.
Los Alamos National Laboratory
Los Alamos, NM, USA
kstroup@{cray.com,lanl.gov}

Paul Peltz Jr.
Oak Ridge National Laboratory
National Center for Computational Sciences
Oak Ridge, TN, USA
peltzpl@ornl.gov

***Abstract*—Application performance inconsistency is a problem that has plagued users and system engineers for a long time. When a user reports that an application took longer than normal to run or was running more slowly than usual, the engineer is faced with a wide range of potential causes. Some of them may be outside the engineer's control, including changes the user has made, interactions with other workload and numerous other factors. One possibility that is detectable and within the engineer's control is that one or more nodes is underperforming, or possibly overperforming. Some sophisticated users may be able to detect this if they have instrumented their application, but quite often the problem report is far from specific or informative. Overperforming nodes can impact application performance in unpredictable ways and may also result in thermal issues that can impact processor lifetime and reliability as well as impacting other components of the system. Los Alamos National Laboratory (LANL) has worked on a number of processes to detect, isolate, and where possible resolve the issue of nodes performing outside expected levels.**

***Keywords*-CPU; Processor; Performance; Variability; Xeon; KNL; ARM**

I. Introduction

During Trinity's acceptance phase there were several incidents in which the acceptance applications and benchmarks did not perform consistently. As a reminder, LANL's Trinity[1] system is a 110 cabinet XC-40 heterogeneous system with approximately 10,000 Intel Xeon Haswell nodes and aproximately 10,000 Intel Xeon Phi Knights Landing nodes. At first the acceptance team believed this was part of the normal hardware shakeout that must be done to weed out various issues, but after a year of the system being in production the engineers were still being asked about inconsistent performance issues with LANL codes running on Trinity. The users were finding that on occasion their jobs, which can take weeks to complete, were not progressing as fast as they normally did. This was leading to hours of wasted compute cycles because one node in their applications was underperforming to such an extent that it was causing nodes to sit idle while waiting for the slowest one to complete its computational cycle.

There were a few cases that the engineers knew of that could cause this that were not specifically a catastrophic hardware failure. Frequently the problem would be that compute nodes would boot and the CPU would underperform and a simple CPU swap or reboot would fix the issue. In the rarer circumstances a node would perform as expected at boot, but then at some point start to underperform in some way. The system engineers at LANL decided to develop a testing strategy that could not only be performed by them when doing maintenance, but that could also optionally be used by users if they needed to be assured the nodes their job was allocated would perform as expected. It was decided that this should not be a universal procedure (like a node health check) since small and short running jobs would see little benefit relative to the overhead.

II. Defining Slow

LANL has found that underperforming nodes seem to come in two distinct types. On occasion a node may perform extremely poorly, often many times slower than expected. In other cases, the variation is a small percentage below, or possibly above, expected performance. In the first case the performance is always lower than expected and generally associated with a hardware issue. A typical example is a node having an extremely high rate of correctable DRAM memory errors (often referred to as a "bursting DIMM"). When the compute node has memory errors, we want the linux kernel to remap the page(s), but if the affected area is in use for Remote Direct Memory Access (RDMA) over the Aries network then we are hit with a double-whammy: The pages are locked and can not be remapped and we have potentially thousands of other nodes requesting those pages with large numbers of RDMA accesses and generating millions or billions of errors that the kernel has to handle and report. The performance impact then becomes huge due to the kernel handling so many errors. No solution to this

This work has been authored by an employee of Triad National Security, LLC, operator of the Los Alamos National Laboratory under Contract No. 89233218CNA000001 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting this work for publication, acknowledges that the United States Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce this work, or allow others to do so for United States Government purposes. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; however, the Laboratory as an institution does not endorse the viewpoint of a publication or guarantee its technical correctness.
This paper is published under LA-UR-19-23142.

page locking has been engineered into the Aries based XC systems, but it is being addressed in the Shasta architecture.

In the second case, where the performance variation is more marginal, the cause is less clearly associated with an obvious hardware issue. The engineers found that many of these were explained by inconsistent processor performance. Differences in manufacturing of the silicon as well as variations in system assembly and integration can impact node performance. During some dedicated testing time, it was found that a small number of nodes associated with the slow jobs were performing below the rest of the nodes in the allocation. The engineers decided that a strategy to do testing was necessary before releasing the system to the users after system maintenance was performed.

III. Testing Methodologies

LANL has pursued two strategies intended to mitigate the impact of performance discrepancies on workloads. One is an administrative task that can be done before releasing the system to users, and another that can be executed at the user's discretion.

A. Sweep the System

The first strategy is performed during system maintenance windows. The entire system is swept to find any nodes that are performing outside of what are determined to be acceptable bounds. These nodes are then isolated from user jobs with a scheduler reservation so that further testing and fault analysis can be done.

To sweep the system for performance discrepancies LANL chose to use the diagnostic tools provided in the Cray diagnostic image. Specifically, xtcpuperf (or xtphiperf for Xeon Phi type processors)[2] with flags selecting only the distributed general matrix multiplication test (DGEMM) to be run was the tool chosen. Multiple passes of DGEMM were run simultaneously (generally 10 to 50 passes) across all nodes in the system. The first pass was eliminated from the analysis to avoid any artifact introduced by temperature variations at start up or artifacts caused by power ramp-up from an idle state. The runs were done as close to simultaneously as possible to ensure that the entire system was at a thermal state typical of running under a full load. The matrix size parameters chosen were typically the defaults provided by the tool. Some limited parameter studies indicated that it was not particularly useful to optimize these parameters, and even testing with sub-optimal parameters was still sufficient to identify performance discrepancies relative to the overall node population.

B. Pre-Job Testing

The second strategy LANL developed was to provide users with an optional utility that the user can choose to run as a preliminary step of a job when the benefits of consistency outweigh the overhead of testing. Again, LANL chose to use the xtcpuperf tool from the Cray diagnostics as it was readily available on all nodes and the runtime parameters were suitable to LANL's needs. This utility will conduct a brief test, lasting less than 10 minutes, of the nodes that the user's job has been allocated. Then subsequent steps of the job can exclude any nodes that fail the performance test. The runtime of the tests were acceptable to the users, given the potential performance impacts that a slow node could cause jobs which typically run for multiple days. The user can over-allocate nodes to their job and thus end up with the desired node count once any of the nodes that do not fall within the acceptable performance range are excluded. To understand how much to over allocate, the user needs to have an estimate of the probability of a given percentage of their allocation being excluded due to not falling within the chosen performance thresholds. This requires the system engineers to have some idea of the overall performance profile of the nodes in their system in order to advise users on the appropriate percentage to over allocate. The screening of nodes during system maintenance windows provides the data to inform this analysis. By using bounds that are approximately three standard deviations from the mean it is possible to project that less than 1% of nodes will be out of range and suggest that jobs that wish to be able to exclude nodes should only need to over allocate by about 1% or 1 node, whichever is greater, to ensure being allocated the desired number of performant nodes. Below is a sample slurm job script.

```
#!/bin/bash
#SBATCH -p knl_any
#SBATCH -N 10
#SBATCH -t 00:10:00

# Check for slow nodes
echo "$(date) - Starting identification
    of slow nodes"
node_checker="/$HOME/node_checker.sh"
if [ -f "${node_checker}" ]; then
     exclude_nodes=$(${node_checker})
else
     echo "${node_checker} does not
         exist. Exiting." 1>&2
     exit 1
fi
echo "$(date) - Done identifying slow
    nodes"

srun -N $(($SLURM_JOB_NUM_NODES -
    $(($(wc -l < ${exclude_nodes}) - 1))
    )) --exclude=${exclude_nodes}
    /$HOME/hello_mpi

if [ -f "${exclude_nodes}" ]; then
```

```
    rm -f "${exclude_nodes}"
fi
```

This is the slow nodes script referenced in the sbatch job above.

```
#!/bin/bash
#
# Determine if there are any slow KNL
   nodes.
# Modify the scratch_home variable to
   write to where you want your
   temporary job files.
# Modify the slow_nodes variable to
   collect the slow_nodes into a file
   for keeping.
# Paul Peltz <peltzpl@ornl.gov> Kevin
   Stroup <kstroup@cray.com>
#
# Setting the environment variable
   GFLOPS to the desired minimum
   acceptable performance
# will over-ride the default of 1700.
#
# Setting the environment variable
   SCRATCH_HOME to a persistent
   location allows for
# long term retention of the
   slow_nodes_list.txt files for
   historical tracking.

if [ -z $SLURM_JOB_ID ] ; then
    echo "Only run this script within a
        job. Exiting" 1>&2
    exit 1
fi
scratch_home=${SCRATCH_HOME:-"/tmp"}
exclude_nodes="${scratch_home}/
   .exclude_nodes.${SLURM_JOB_ID}"
xtphiperf_ini="${scratch_home}/
   xtphiperf.ini.${SLURM_JOB_ID}"
node_results="/tmp/
   node_results.${SLURM_JOB_ID}"
slow_nodes="${scratch_home}/
   slow_nodes_list.txt.${SLURM_JOB_ID}"

loghost=$(grep ^loghost=
   /etc/opt/cray/llm/llm.conf | awk -F
   = '{print $2}')

if [ ! -f "${slow_nodes}" ] ; then
    touch ${slow_nodes}
fi

cat <<EOF > "${xtphiperf_ini}"
#user specified expected performance
[Xeon Phi]
GFLOPS = ${GFLOPS:-1700}
EOF

if [[ $SLURM_JOB_PARTITION == *"knl"*
    ]]; then
    # We will put blahfoo in so it
        doesn't match, but something has
        to be in the file or srun will
        fail
    echo "blahfoo" > "${exclude_nodes}"
    echo "Running slow_node_checker on
        job $SLURM_JOB_ID." |
        /usr/bin/logger -n $loghost -p
        user.error -t "slow_node_checker"
    /opt/slurm/bin/srun -N
        $SLURM_JOB_NUM_NODES
        /opt/cray/diag/bin/knl/xtphiperf
        -v 1 -l 3 -k 35000 -m 35000 -n
        35000 -p "${xtphiperf_ini}" 2>&1
        >/dev/null | grep Fail | sed
        's/,//g' > "${node_results}"
    while read -r line ; do
        echo "$line" | /usr/bin/logger
            -n $loghost -p user.error -t
            "slow_node_checker"
        echo "$line" >> ${slow_nodes}
        echo "$line" | awk '{print $3}'
            >> "${exclude_nodes}"
    done < "${node_results}"
fi

if [ -f ${xtphiperf_ini} ]; then
    rm -f ${xtphiperf_ini}
fi

echo "${exclude_nodes}"
```

IV. Performance Profiles

One advantage of analyzing data from large-scale systems is that the quantity of data allows for a large enough data set to have significant confidence in the expected distribution of node and processor performance. LANL's systems have yielded data on approximately 10,000 Intel Xeon Haswell nodes with 20,000 processors and 10,000 Intel Xeon Phi Knights Landing nodes. With the benefit of multiple passes over time, LANL now has hundreds of thousands of data points from which to determine an expected performance profile for various types of nodes. With the acquisition of two XC-50 systems with ARM nodes, LANL now also has data on approximately 350 nodes and 700 processors to project an expected performance profile for that processor

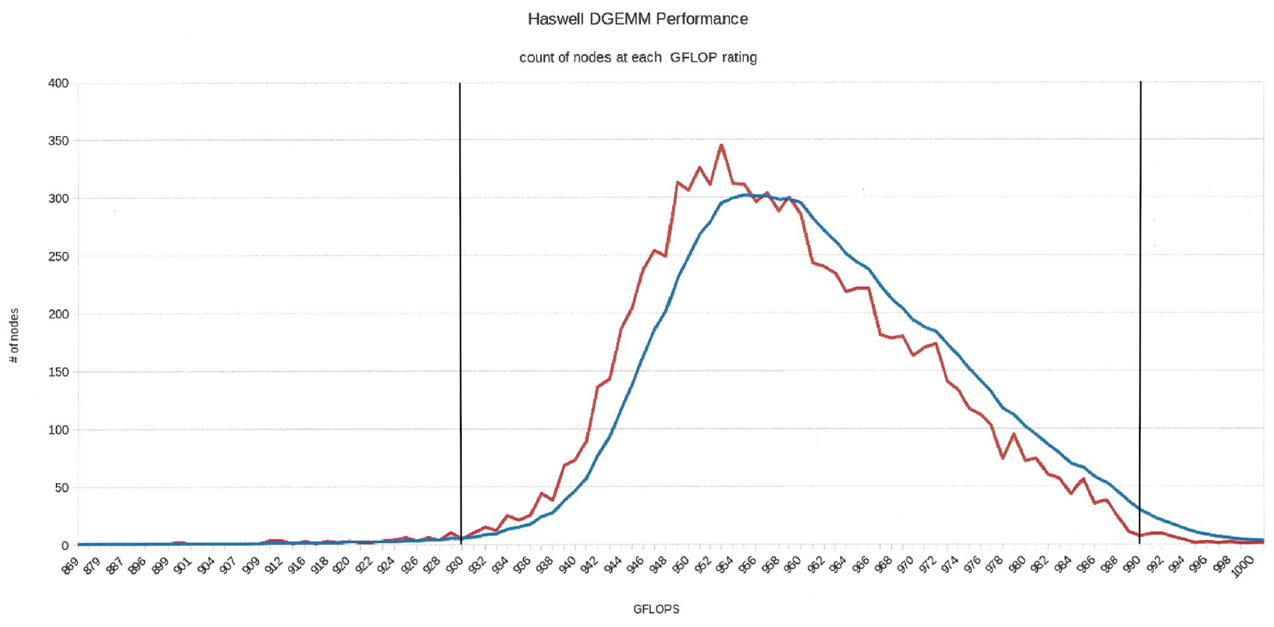

Figure 1. Haswell DGEMM Results. Blue is Smoothed

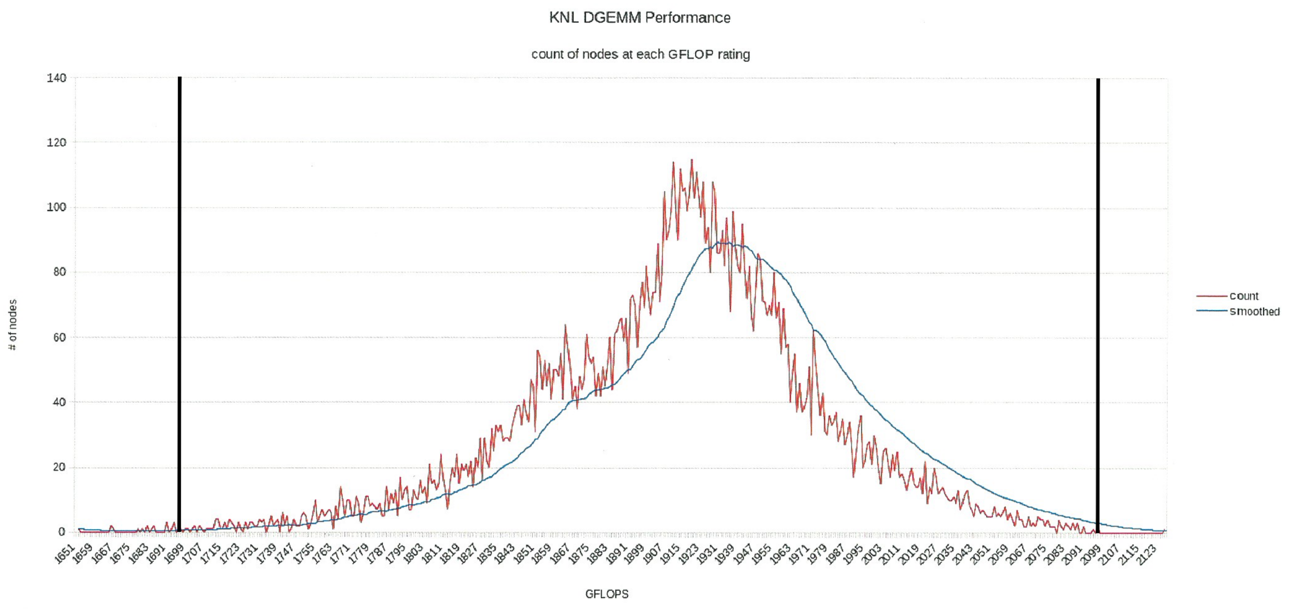

Figure 2. KNL DGEMM Results

type as well. Repeated testing has also given the site an understanding of what to expect in terms of consistency and variability of individual processors over time. This helps inform the decision about how frequently and under what circumstances testing should be repeated.

A. *Intel Xeon Haswell Processors*

Intel Xeon Haswell processors (model E5-2698v3 - 32 core CPU dual socket) perform in a range that shows a good fit to a standard distribution with a node mean of 960 GFLOPs and a standard deviation of 8 GFLOPs. LANL chose to define anything over 3 sigma as out of compliance and address those processors. The procedures that LANL uses are explained in Section V. That gives a definition for the acceptable performance range for those nodes whose performance is between 930 and 990 GFLOPs. See Figure 1 for the results. The vertical bars represent the acceptable performance range.

B. Intel Xeon Phi Knights Landing Processors

The Intel Xeon Phi Knights Landing processors (model KNL 7250 - 68 cores single socket) in LANL's systems perform in a range that shows a fairly good fit to a standard distribution with a node mean of 1900 GFLOPs and a standard deviation of 60 GFLOPs. LANL chose to define anything over 3 sigma as out of compliance and address those processors. The procedures that LANL uses are explained in Section V. That gives a definition for the acceptable performance range for those nodes whose performance is between 1700 and 2100 GFLOPs. See Figure 2 for the results.

C. ARM Processors

Marvell ThunderX2[5] processors (model CN9975 - 28 cores dual socket) in LANL's systems perform in a range that shows a very narrow distribution with a node mean of 590 GFLOPs and a standard deviation under 5 GFLOPs. These are early models that do not support 'turbo' mode. See Figure 3 for the results.

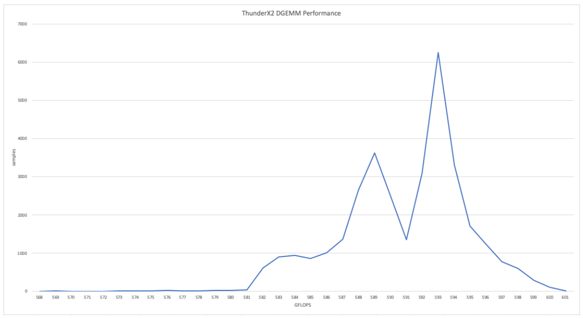

Figure 3. Marvell ThunderX2 DGEMM Results

V. Resolving Issues

A. Obvious Repairs

Resolving performance issues with a node is relatively straightforward where a problematic component can be clearly identified and replaced. An excessive rate of correctable memory errors (often called a bursting DIMM) can have an extremely significant impact on performance, sometimes slowing processing down by an order of magnitude or more. The delineation of what error rate constitutes excessive is somewhat arbitrary, but by looking at the distribution of error counts it becomes fairly obvious. Error rates that are orders of magnitude larger than normal are a clear indicator that a node should be acted upon. Cray does have service guidelines on error rates that warrant hardware action, but a site will need to make its own determination on what error rate will have an impact on their workload throughput.

B. Minimizing Impacts

By implementing procedures to automatically exclude nodes with high rates of non-fatal errors it is possible to avoid additional jobs being impacted. A node health script running regularly on the System Management Workstation (SMW) can detect that a node has an error rate over a given threshold and notify the Slurm scheduler that the node should not be used for future jobs (in Slurm parlance it drains the node). One question that occurred was if it was better to allow the job being impacted by a node with a high error rate to continue running or to terminate the job. LANL engineers chose to generally err on the side of caution and not terminate jobs without the user's permission. In some cases where the job is long-running or has checkpoint and restart capability or the ability to resize the job to use fewer nodes, the users have instructed us to immediately take the node down. Again, the script that detects the condition notified the scheduler, in this case to "down" the node immediately as opposed to draining it and allowing the job to finish with the node still in its allocation. In this case, Slurm notifies the job that the aberrant node has failed and the job, assuming the job is properly configured, goes through a process to reallocate to the remaining number of nodes and restart from the latest good checkpoint.

C. Marginal Cases

If no obvious error conditions exist on a node that is performing anomalously, then identifying the fault may require problem determination procedures such as specific diagnostic tools, swapping or scattering parts among nodes and rebooting or reconfiguring the nodes. The Cray diagnostic image provides several tools to check things like memory and memory bandwidth behavior, etc. Other factors to consider include the power profile (for processors that support this) and temperature behavior of the processor or node, any performance discrepancy between sockets on multi-socket nodes, and the maintenance history of the node.

D. Thermal Impact

Data from the Power Management database (PMDB) was used to check nodes for thermal throttling. KNL processors were seen to reach 88 degrees C and exhibit a sawtooth temperature and power profile. Several aspects of running at LANL made this unsurprising. LANL uses warm-water cooling and is located at an altitude of over 7,000 feet (well over 2 km) and so experiences thinner and drier air than most sites. This makes heat transfer less efficient. See Figure 4 for the graph.

VI. The Unexplained

There were a number of cases where the detected performance inconsistency was never traced back to any root cause. Either the behavior changed, or action was taken

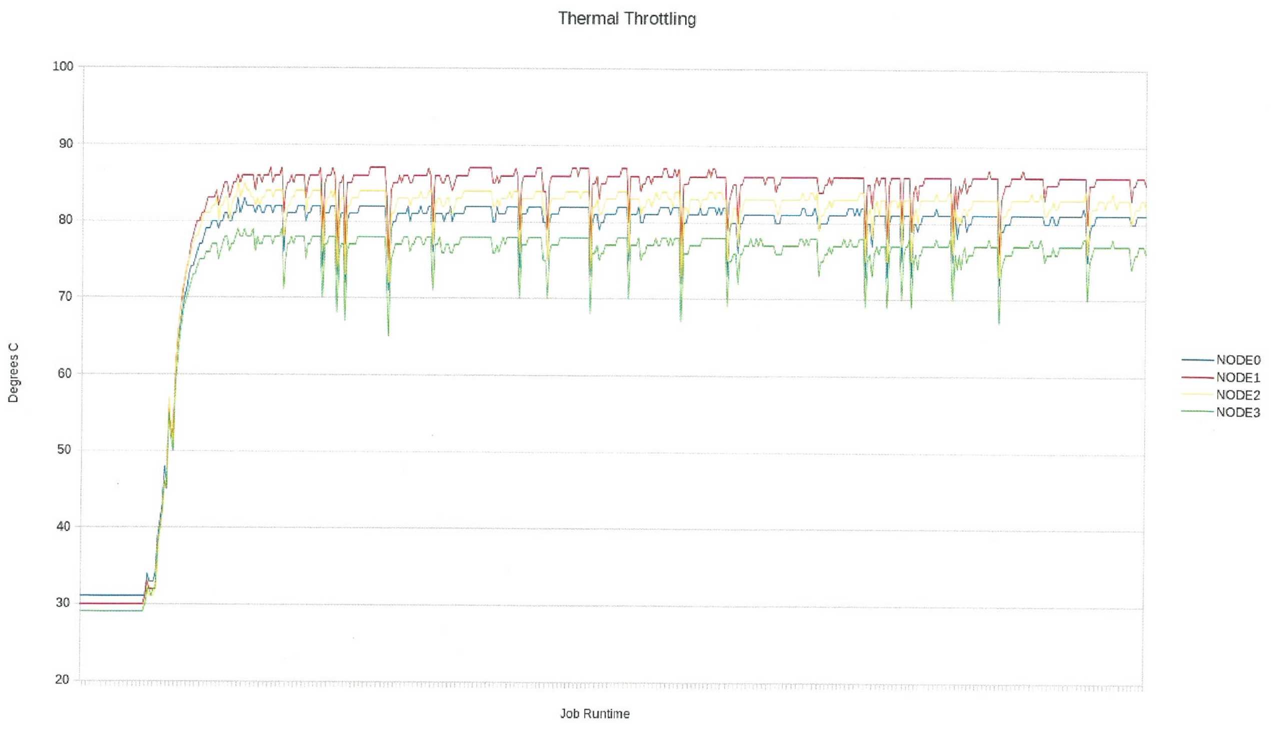

Figure 4. Thermal Throttling Sawtooth Effect

(such as a part replacement) that eliminated the possibility of further diagnosis.

One KNL that was performing at 25% of normal had the components of that node scattered to (swapped with) the other nodes on the same blade as a diagnosis technique. However, rather than the behavior following a part to a new node, the inconsistency simply went away and all nodes were performing normally. In another case a KNL processor which performed 10x slower than normal in a particular NUMA mode was performing normally in all other modes. This CPU was sent back to Intel for further analysis. In another case, a compute blade had two nodes underperforming in the production system but performed normally in the hardware diagnostic system.

In each of these cases a hardware action was performed to fix the problem. It was not always the case that some part was faulty. Sometimes simply reseating a part or moving it to another blade has resolved the issue. While this doesn't get to the root cause of the issue, it is an expedient way to return the system to operation quickly with optimal performance consistency.

VII. Conclusion

The results of these efforts have been worthwhile for LANL. Aside from a better understanding and awareness of the performance characteristics of nodes and their systems overall, the engineers have seen a reduction in reports of job runtime irregularities. The scientists who were most impacted by performance inconsistency tended to be our most sophisticated users running some of the largest and highest profile workloads. They are also the users who have seen the greatest benefits from these efforts. They were saved the effort of having to instrument collective performance checking in their code. By reducing the pain points for LANL's scientists this work has made a significant positive impact on how LANL's systems are perceived by the users, the laboratory's management, and the Department of Energy.

Acknowledgment

The authors would like to thank the entire LANL systems support team for their time and dedication to Trinity. Special thanks to Jim Lujan of LANL for his support of this work. Thanks to Norm Troullier and Larry Kaplan of Cray for their assistance in reviewing and improving this paper. The authors appreciate the guidance they received from the Cray management team.

References

[1] Hemmert, Karl Scott, et al. *Trinity: Architecture and Early Experience*. No. SAND2016-4315C. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2016.

[2] https://cug.org/proceedings/cug2017_proceedings/includes/files/pap140s2-file2.pdf

[3] https://ark.intel.com/content/www/us/en/ark/products/94035/intel-xeon-phi-processor-7250-16gb-1-40-ghz-68-core.html

[4] https://ark.intel.com/content/www/us/en/ark/products/81060/intel-xeon-processor-e5-2698-v3-40m-cache-2-30-ghz.html

[5] https://en.wikichip.org/wiki/cavium/thunderx2/cn9975