# Holistic Measurement-Driven System Assessment

Saurabh Jha*, Jim Brandt†, Ann Gentile†, Zbigniew Kalbarczyk*, Greg Bauer¶, Jeremy Enos¶, Michael Showerman¶, Larry Kaplan§, Brett Bode¶, Annette Greiner‖, Amanda Bonnie‡, Mike Mason‡, Ravishankar K. Iyer*, and William Kramer*¶

*University of Illinois at Urbana-Champaign, Urbana-Champaign, IL 61801

†Sandia National Laboratories (SNL), Albuquerque, NM 87123

¶National Center for Supercomputing Applications (NCSA), Urbana, IL 61801

§Cray, Inc., Seattle, WA 98164

‖National Energy Research Science Computing Center (NERSC), Berkeley, CA 94720

‡Los Alamos National Laboratory (LANL), Los Alamos, NM 87544

***Abstract*—In high-performance computing systems, application performance and throughput are dependent on a complex interplay of hardware and software subsystems and variable workloads with competing resource demands. Data-driven insights into the potentially widespread scope and propagation of impact of events, such as faults and contention for shared resources, can be used to drive more effective use of resources, for improved root cause diagnosis, and for predicting performance impacts. We present work developing integrated capabilities for holistic monitoring and analysis to understand and characterize propagation of performance-degrading events. These characterizations can be used to determine and invoke mitigating responses by system administrators, applications, and system software.**

## I. Introduction

Extreme-scale high-performance computing (HPC) systems require a holistic approach to monitoring and coordination of many disparate subsystems (both hardware and software) to enable continued scaling and efficient execution of applications. HPC systems are typically used for executing tightly coupled simulation applications across hundreds of thousands to millions of processor threads. Mismatches in processor, memory, interconnect, and/or storage performance can significantly influence an application's overall performance; a variation of 50% or more has been observed. A single component failure can cause an entire application to fail at any time. Power and cooling facilities can significantly influence both component performance and failure probability. As a result, effective failure/degradation mitigation response(s) in complex systems require analysis of (i) propagation of faults/errors and (ii) performance issues due to interference among applications or resource exhaustion.

As part of a DOE Office of Science resilience project called Holistic Measurement-Driven Resilience (HMDR), we have been building integrated capabilities for extracting system and application performance- and failure-related data. We have been using these data to build fault-to-failure characterizations, and using these characterizations to determine effective mitigating responses. We are now extending that work to address more generally scenarios that result in performance degradation in which timely and appropriate response can significantly improve both application run times and system throughput. The eventual product of this work will be an interoperable set of capabilities for extreme-scale systems that provides monitoring, analysis, and appropriate response for both resilience and performance issues. These capabilities will support both automated and exploratory analysis, both at run-time and in post-processing. In order to enable analysis of the data sizes to be produced on such systems, we include in our design low overhead collection and extraction of raw and derived information, including system and application-level events and effects, necessary to facilitate the development of machine-learning-based analyses.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award Number 2015-02674.

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

## II. Tracking and Diagnosing System Issues at Runtime

This section illustrates the use of collected data (from NCSA's Blue Waters [1]) and analysis tools for monitoring, tracking, and diagnosis of both failures and performance degradation in systems and applications through use of case studies. Further, these case studies illustrate methods for building features that can help diagnose application resilience and performance problems.

### A. Tracking, predicting, and diagnosing failures

In a large-scale system, faults, errors, and failures are inevitable and can occur in hardware or software from any subsystem. These Events of Interest (EoI) can propagate to manifest as serious system or application stability or correctness issues and may be silently tolerated by the system either through built-in resiliency mechanisms or because they do not coincide with the utilization of resources by an application. In large-scale systems, the volume, frequency, and variety of information output to log files, whether just informational or to document actual errors, can hinder timely problem diagnosis and accurate root cause analysis. Sophisticated and automated diagnostic tools are a requirement for processing relevant information on the time scales required to take effective mitigating action as opposed to post-impact analysis and recovery.

Correlating the observed EoIs to build likely fault-propagation paths, such as the one described in "Case Study 1" below, enables early identification of adverse impact indicators and construction of probabilistic predictors of locality and intensity. The pictorial representation of a fault propagation path shown in Figure 1 was built using our LogDiver [2] tool's semi automatic clustering algorithm [3]. Our current

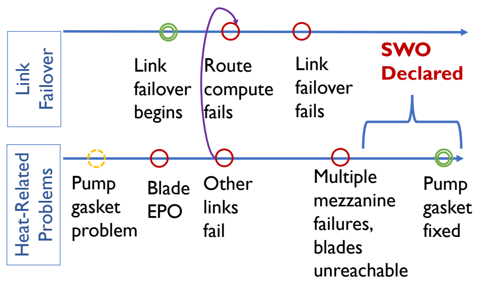

Fig. 1: Network recovery-sequence cluster showing relationship between failure and recovery events

focus is on enhancing and integrating our tools to automate the task of feature discovery and to build fault-propagation paths independent of the particular subsystem, as described in Section III. We will leverage machine learning methods (such as Bayesian networks) to complete this task.

**Case Study 1: A pump gasket failure led to a system-wide outage by way of network component failures and error propagation:** Figure 1 (automatically generated by LogDiver) depicts a network recovery-sequence cluster capturing relationships between network failure and recovery events in the system. The failure of a pump gasket (*pump gasket problem*) caused the temperature to rise in a cabinet. The eventual overheating triggered an emergency power-off of a blade (*blade EPO*), a protection mechanism to guard the blade's components, including Gemini router ASICs, from permanent damage. The associated Gemini router failure triggered link failover (*link failover begins*). As the link failover progressed, additional failures occurred on other blades (*other links fail*) in the same cabinet resulting in route computation failure (*route compute fails*). The failure of the route computation (and its retries) to establish an alternative path for nodes to communicate with other nodes led to the failure of the link failover (*link failover fails*). The overheating effect increased and propagated to nearby cabinets, causing more blades to fail (*multiple mezzanine failures, blades unreachable*). A system-wide outage (SWO) was declared to manually fix the system and restore it to a healthy state.

## B. Tracking, predicting, and diagnosing performance problems

Performance problems can result from failure or non-failure related causes and can also propagate in a system. For example, both link failures and application communication design patterns can result in the occurrence and spread of contention for shared network resources. Performance degradation, rather than application failure, then, may be the outcome. Here we present case studies showcasing the use of monitoring data collected by our LDMS [4] tool to enable analysis of non-failure-related performance issues and propagation paths.

**Case Study 2: Performance degradation due to failure:** Figure 2 shows the impact of network link failure and its corresponding recovery on the Gemini 3D torus [5] interconnection network of Blue Waters. Event logs corresponding to this failure are shown in Figure 2(ii). In this network, traffic from multiple applications may pass through the same Gemini routers along their paths. Sub-figure 2(i) shows the aggregate data passing through all of the Gemini routers. Basic statistics such as this indicate bandwidth utilization of the system. To understand if there is actual performance degradation and, if so, which Gemini router may be the source of the problem requires more in-depth analysis. Converting raw network data into derived metrics that are indicative of performance-degrading conditions and associating these metrics with application performance measures are still areas of active research.

Sub-figure 2(iii) shows the "average packet transmission time", which is calculated by dividing "average packet size" by bandwidth for all the links on a Gemini ASIC router. Continuous derivation of this metric from raw monitored data for each Gemini router can provide a detectable signal for identifying the source of congestion. In this case, failure of one of the links (4 Gemini tiles) on the Gemini network at "9-7-1" x,y,z coordinates in the torus (Gem 9-7-1 in Figure 2) lead to the initiation of network recovery and a network-wide *quiesce*. From the figure it is clear that this particular Gemini router was showing early indication of a problem, as the "average packet transmission time" through this Gemini router was significantly higher than for others. Such signals can be used for training, predicting, and diagnosing problems.

**Case Study 3: Performance degradation due to non-failure related issues:** Figure 3 shows where a 32-node job caused high congestion in the system interconnect, triggering two *congestion protection events*, first at 10:00 am (within 10 seconds of job launch) and then at 15:20 (red lightning bolts in Figure 3(i)). Congestion within the torus can adversely impact the performance of the application responsible for triggering the congestion protection events and other running applications and is a major cause of inconsistent application run times. Blue Waters uses topology aware scheduling (TAS) [6] to maximize communication performance for jobs in its 3D torus by placing a job in a compact rectangular prism-shaped set of physical nodes. Such assignment maximizes locality of inter-node communication patterns within the prism-shaped allocation geometry. I/O calls to the file system, however, are likely (but not guaranteed) to cause communication outside a job's geometry. A linear shape in the "Z" direction is more likely to funnel much of the I/O traffic to shared links, causing high congestion in the system. Detection and diagnosis of the cause of congestion helps optimize scheduling strategies for a job as well as improve application communication performance. Use of derived metrics and anomaly-detection algorithms can enable diagnosis and detection of such issues.

Figure 3(i) shows a plot of maximum *credit* and *inq* stall metrics, indicating when traffic cannot be sent to a port of a different Gemini router (*credit*) or within the same Gemini router (*inq*) because of a lack of receiver buffer space, across links in the job. These unprocessed metrics *alone* do not help in diagnosis of congestion issues in the network or job.

Figure 3(ii) shows a plot of the sum of stall rates on all links for all the Gemini routers local to compute nodes used by the job (normalized to the total stall rate throughout the duration of the job). The two peaks (marked) in this sub-figure correspond to the triggered congestion protection events.

Figure 3(iii) shows a plot of the absolute difference between job input data and output data transferred over the network (normalized by total input or output data throughout the duration of the job). This shows three large peaks that hint at I/O rather than inter-process communication in the application. Only 2 of the 3 peaks in this sub-figure match congestion protection events. Combined knowledge from Figure 3(ii) and Figure 3(iii)

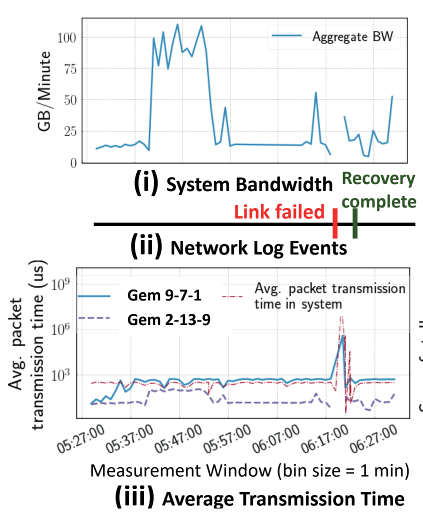

Fig. 2: Case Study 2

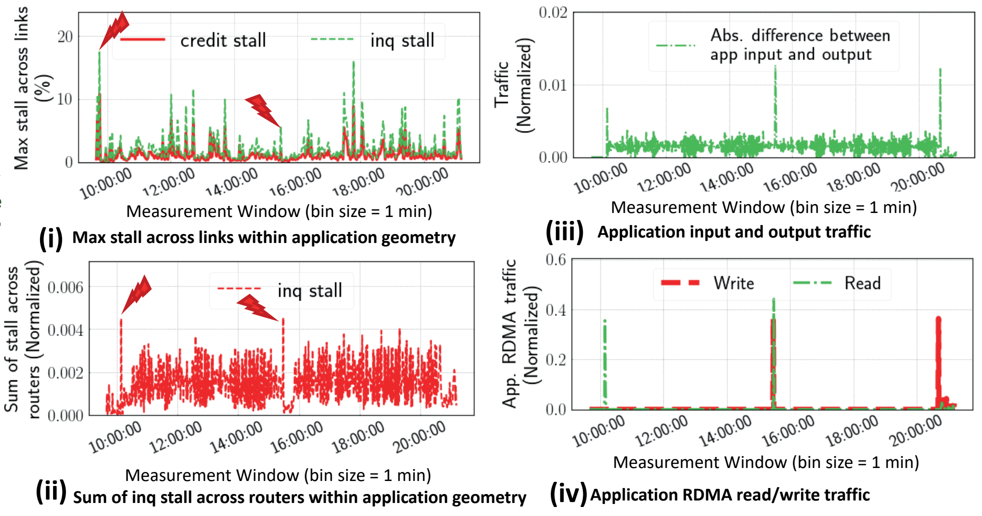

Fig. 3: Case Study 3

enables correlation of the congestion protection events with filesystem I/O.

Figure 3(iv) shows a plot of read and write remote direct memory access (RDMA) values for read and write for this job normalized by total read/write RDMA bytes through the job's duration. It confirms that only large amounts of "read" RDMA trigger *congestion protection events*.

Case studies 1, 2, and 3 illustrate the need to holistically monitor applications and systems to understand application resilience and performance bottlenecks. Moreover, these case studies demonstrate that extraction of features from raw data can help build machine learning models to distinguish and diagnose application performance and resilience issues.

## III. Approach

This section presents our approach to building an integrated tool suite for monitoring, analysis, and response. This suite is comprised of 1) passive tools for gathering and analyzing information produced by typical HPC systems (e.g., system and hardware error logs), 2) active querying tools that collect data not naturally emitted/collected from a system (e.g., performance counters, facilities power and cooling information), and 3) active probing tools that inject stimuli into a system to assess the system response in terms of behavioral characteristics of information gathered by the tools in categories 1 & 2. We are extending our suite of tools with components for post-processing and run-time analytics. The tools are designed to interact seamlessly together and with target system components to provide efficient operation of platforms and applications.

The architecture for our "Holistic Measurement-Driven System Assessment" (HMDSA) infrastructure is shown in Figure 4. We describe the major HMDSA components and their functional interactions here. Our first target deployments are Open Science platforms at NCSA. Currently, on Blue Waters, the Integrated System Console (ISC) [7] provides capabilities for data collection, analysis, visualization, and report generation. HMDSA will provide next-generation advancements to the ISC, with greater capabilities for extreme-scale automated analyses and new capabilities for determining and invoking run-time system-level and application-level responses.

***System and Application Data Collection*** In HMDR, we use LDMS to collect data that are, or can be, exposed on-node,

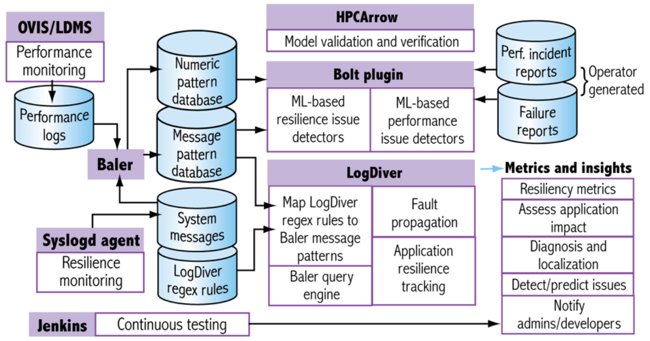

Fig. 4: HMDSA Infrastructure. Data sources (left); analysis results and responses (right). HMDSA will enhance our capabilities for advanced automated analyses of failure and performance degradation EoI and to inform system-level and application-level responses.

for example, CPU, IO, and network utilization data. An LDMS daemon is run on each node to collect data via queries to the daemon. LDMS collects data at user-configurable intervals from milliseconds to minutes. These datasets are the basis of our analyses to determine performance issues and inform propagation scenarios. We are currently extending our collectors to incorporate application-provided information. Timestamped application phase and progress information will enable better understanding of how system events, characteristics, and conditions, including contention for underprovisioned resources, impact an application's performance.

***Propagation Characterizations*** Log data are an important information source for determining an initial problem and the sequence of events in fault handling. LogDiver is our tool for extracting and assessing sequences of events in log data. Currently, message patterns associated with EoIs must be identified in advance and manually categorized as to domain (e.g., network, storage). LogDiver then processes logs, identifies significant messages, and determines spatio-temporal relationships of significant messages. This enables characterization of sequences, such as steps and timings in failure recovery, as well as identification of failures in recovery, which are identified by missing or additional warning lines in sequences. An example is shown in Figure 1.

To facilitate identification of important log messages, particularly in new systems in which the possible and meaningful messages are unknown, we use our Baler [8] tool. The user provides a dictionary of words; Baler turns the log lines into patterns by retaining dictionary terms and treating all other items as variables (indicated by $\star$), e.g., "`handling failed link *`". Baler can also turn numeric value ranges into distinct patterns. Both patterns (textual and numeric) can be treated identically for queries and analyses. This facilitates understanding relationships of numerical characteristics and log events for automated detection of EoIs, abnormal behaviors, and associations.

We are currently integrating LogDiver and Baler (Figure 4) to streamline the analysis process. Patterns determined by Baler will be directly usable by LogDiver, which will process the pattern occurrence data in the Baler database. This will enable more efficient extraction of sequences and timings and will enable LogDiver extensions to include numerical data as well.

***Diagnosis and Prediction*** By determining the text, numerical sequences, and timings of interest, we seek to both diagnose and predict stability and performance-impacting issues. Detection of numeric indicators that can be associated with propagation characterizations could enable use of the sequences to identify both initial causes and impending events in the sequence. The run-time usability will depend on actual time windows between events in a particular sequence.

***Validation Through Fault and Performance Degradation Injection*** In order to expand and refine our characterizations, we augment our production data with data from controlled experiments. We have built a toolkit for fault injection, HPCArrow [9], which includes both injection of faults into system components and launch of applications in order to assess the resulting impact on performance and stability. We capitalize on HPCArrow's flexibility and are extending it to perform injection of performance-impacting events such as memory leaks and contention for shared network and I/O resources.

***Feedback to Applications and System Software*** To mitigate performance-impacting scenarios, we seek to provide data, diagnosis, and prediction information in actionable form to both human and system software consumers. Feedback to system administrators and users can be used for complex diagnosis, improved understanding, and subsequent tuning. However, more direct run-time mitigating responses can be achieved by providing direct feedback to applications and system software where it can be used for load-balancing, task-mapping, or co-scheduling decisions. We have previously shown [10] that in certain congested network scenarios, remapping based on dynamic monitoring of system information can be more effective than that based on static architectural measures alone. Providing information to on-node consumers is facilitated by the LDMS daemons that innately host data. Validated propagation scenarios can be used to prioritize event response, based on potential severity of impact and window of opportunity.

***Future instrumentation*** Historically, systems have provided limited exposure of the information that we require for actionable analysis. Often the reason is a lack of instrumentation. In recent years more information has been exposed; for example, the number and scope of network performance counters has greatly increased in the Cray Aries router ASIC [11] since the previous-generation Cray Gemini router ASIC. However, in some current and upcoming subsystems, while instrumentation has increased, exposure of raw information to the user has been reduced, while the vendor utilizes it for making low level fault and performance decisions. This trend limits the usability of such information by the user and researchers for more global analysis and decision-making processes. As part of our work we seek to demonstrate the utility of processing low-level resource utilization/contention information in a global context to form the basis for discussions with vendors for data exposure. This includes identifying additional potentially actionable information, including location and maximum refresh rates, that would support earlier and more accurate discovery, diagnosis, and prediction of stability/performance issues. Further, we are pursuing advanced analytics, including machine learning and low latency evaluation of raw and derived data against learned behavioral models, to provide the most effective response possible to degradation in stability and/or performance.

## IV. Related Work

There is a substantial body of research, reaching back over a decade [12], in the area of intelligent use of monitoring data. Much of it focuses on characterizing failure rates from text logs in order to set checkpoint intervals for applications. There has been less numerical analysis work due to the complexities of collecting numerical data at sufficient fidelities with low impact and of analyzing large datasets. The work that has been done in that area includes [13], [14]. In contrast to our previous work which focused on development of infrastructures for combined monitoring, analysis, and response [15], [7], here we are developing more sophisticated tools for combined text and numeric analyses to build detailed propagation characterizations reflective of complex interactions among subsystems that will require autonomous resource-utilization optimization in order to deliver on the promise of extreme-scale computing.

## References

[1] "Blue Waters." [Online]. Available: https://bluewaters.ncsa.illinois.edu

[2] C. D. Martino *et al.*, "LogDiver: A tool for measuring resilience of extreme-scale systems and applications," in *Proc. of the 5th Workshop on Fault Tolerance for HPC at eXtreme Scale*. ACM, 2015, pp. 11–18.

[3] S. Jha *et al.*, "Analysis of Gemini Interconnect Recovery Mechanisms: Methods and Observations," in *Cray User Group*, 2016.

[4] A. Agelastos *et al.*, "Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications," in *Proc. Int'l Conf. for High Performance Storage, Networking, and Analysis (SC)*, 2014.

[5] R. Alverson *et al.*, "The Gemini System Interconnect," in *Proc. 2010 IEEE 18th Ann. Symp. on High Perf. Interconnects (HOTI)*, 2010.

[6] J. Enos *et al.*, "Topology-aware job scheduling strategies for torus networks," in *Proc. Cray User Group*, 2014.

[7] J. Fullop *et al.*, "A diagnostic utility for analyzing periods of degraded job performance," in *Proc. Cray User Group*, 2014.

[8] N. Taerat, J. Brandt, A. Gentile, M. Wong, and C. Leangsuksun, "Baler: deterministic, lossless log message clustering tool," *Computer Science - Research and Development*, vol. 26, no. 3-4, pp. 285–295, 2011.

[9] V. Formicola *et al.*, "Understanding Fault Scenarios and Impacts Through Fault Injection Experiments in Cielo," in *Proc. Cray User's Group*, 2017.

[10] J. Brandt *et al.*, "Demonstrating Improved Application Performance Using Dynamic Monitoring and Task Mapping," in *IEEE Int'l Conf. on Cluster Computing*, 2014.

[11] Cray Inc., "Aries Hardware Counters," Cray Doc S-0045-20, 2015.

[12] R. Vilalta *et al.*, "Predictive algorithms in the management of computer systems," *IBM Systems Journal*, vol. 41, no. 3, pp. 161–474, 2002.

[13] T. Evans *et al.*, "Comprehensive Resource Use Monitoring for HPC Systems with TACC Stats," in *Proc. of the First Int'l Wrk. on HPC User Support Tools*, 2014.

[14] S. Gallo *et al.*, "Analysis of XDMoD/SUPReMM Data Using Machine Learning Techniques," in *Proc. IEEE Int'l Conf. on Cluster Comp.*, 2015.

[15] J. Brandt *et al.*, "OVIS-2: A robust distributed architecture for scalable RAS," in *Proc. IEEE Int'l Symp. on Parallel and Dist. Proc.*, 2008.