Runtime HPC System and Application Performance Assessment and Diagnostics

SAND2018-2373 C

J. Brandt[1], A. Gentile[1], Jon Cook[2], B. Allan[1], Jea. Cook[1], O. Aaziz[2], T. Tucker[3], N. Naksinehaboon[3], N. Taerat[3], E. Ates[4], O. Tuncer[4], M. Egele[4], A. Turk[4], and A. Coskun[4]

https://ovis.ca.sandia.gov

ovis-help@sandia.gov

Sandia National Laboratories, Albuquerque NM

New Mexico State University, Las Cruces NM

Open Grid Computing, Austin TX

Boston University, Boston MA

Sandia National Laboratories

NM STATE UNIVERSITY

OGC

BOSTON UNIVERSITY

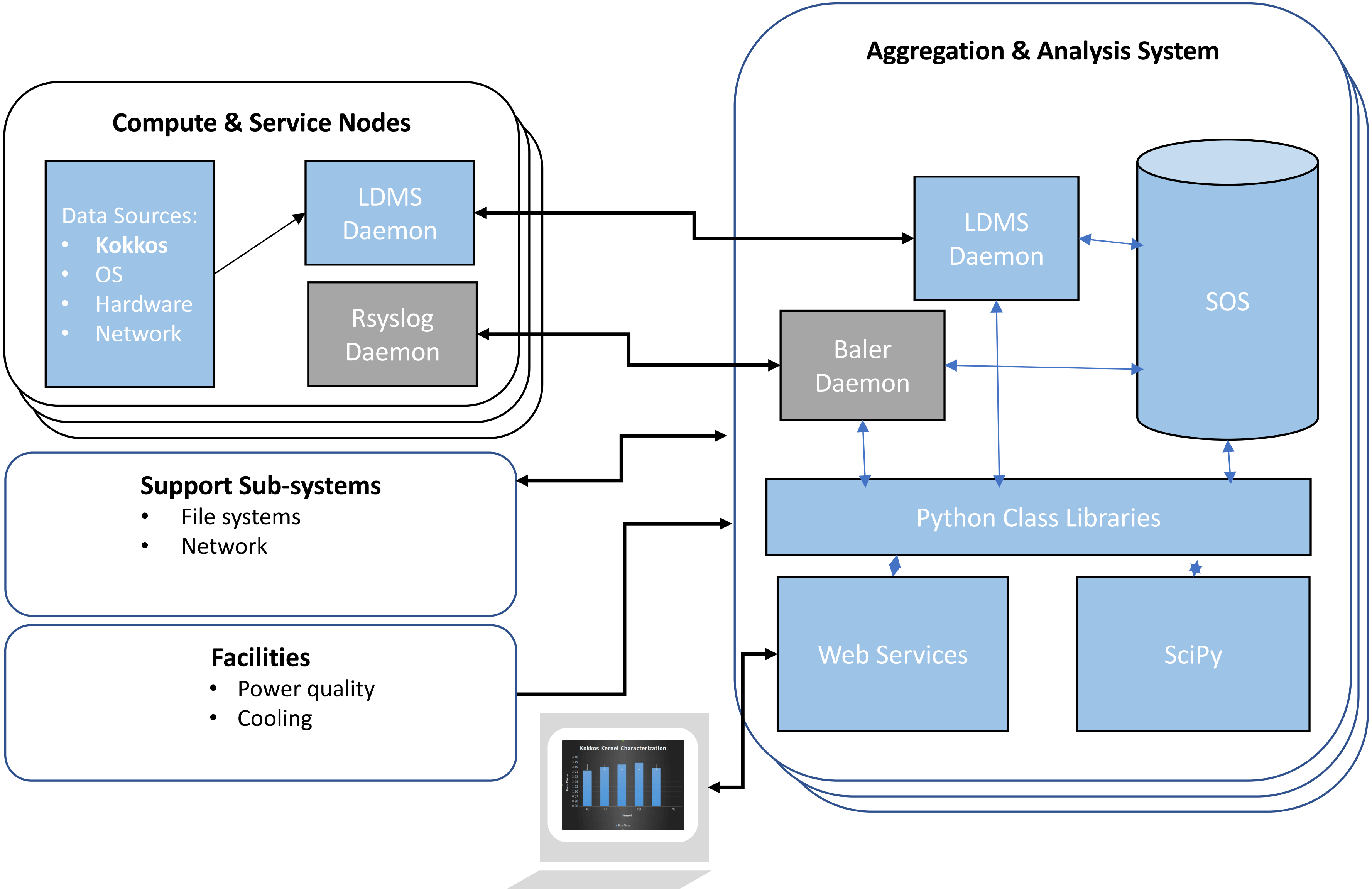
Goal: Understand and Mitigate Performance Variation in Large Scale HPC Systems

Performance variation can come from a variety of sources

- Application code changes
- Compiler changes
- System hardware/software changes/faults
- Resource contention among applications
  - Node, network, storage/file system, power, cooling, etc.

Approach: Use appropriate fidelity collection and analysis of whole system information to reveal reasons for variation and identify solutions to minimize both run times and run time variation

End to End Sensor and Log Collection, Analysis, and Visualization



Whole System Analysis Overview

Scalable end-to-end tool chain for run time collection, transport, and analysis of system wide information:

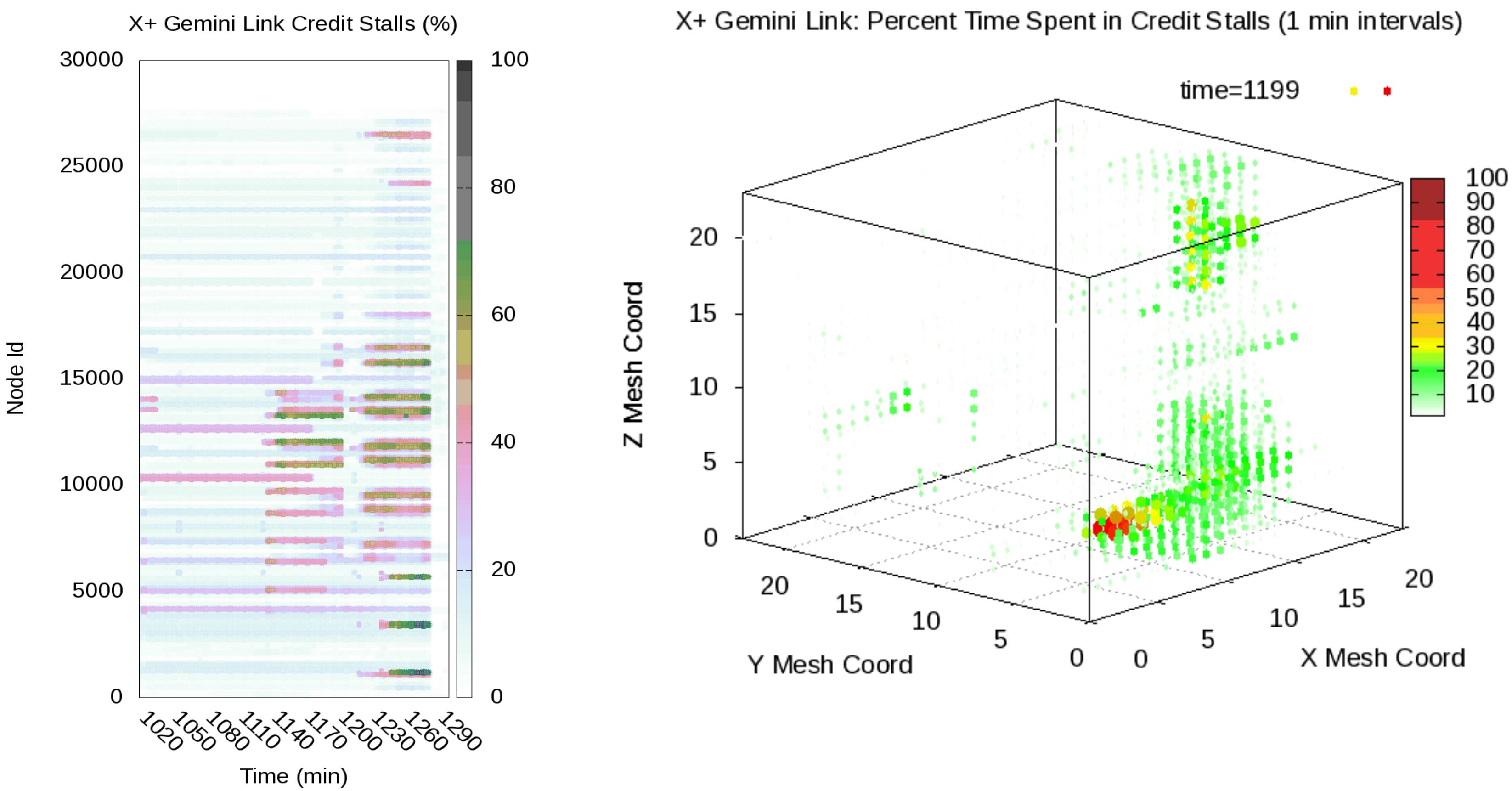
- **Low-overhead**, small footprint data collection and transport (LDMS) - *R&D 100 award winner*
- Integration and joint analysis of numeric and log data (Baler)
- Analysis pipeline (**in situ, in transit, post-processing** with SciPy support)
- Storage (CSV, SOS) and external consumer feeds (named pipe, AMQP)
- Visualization dashboards via Grafana and custom visualization support

System Numeric Data Collection Features

- Synchronized system wide data sampling provides resource utilization “snapshots”
  - Memory
  - Memory Bandwidth
  - Processor
  - Power
  - Network utilization and congestion parameters
  - I/O
- No significant impact on applications at collection rates (1Hz) necessary for resolving resource utilization features
  - Optimized data structures, RDMA
  - Testing at scale on Blue Waters (27648 nodes) and Trinity (20,000 nodes)
- Runtime analysis of large data
  - Custom performant database optimized for inserts and multiple index operations across a variety of ”data types” (e.g., scalars, vectors, log lines, binary blobs)
  - ~ 5TB/day on Trinity

Unprecedented ability to collect system data at resolutions necessary for detecting features and events of interest and to respond on meaningful timescales

Network Congestion Visualizations



NCSA's Blue Waters (27,648 nodes), From: *Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications, SC14*

Minimize application impact by understanding and responding to congestion evolution

Application Performance Insights

LDMS PAPI "Metric Set"

```
test/spapi1505734446162: consistent, last update: Mon Sep 18 08:41:11 2017 [183611us]
M u64        component_id                          162
M u64        job_id                                0
D char[]     Appname                               "lulesh"
D u64        Jobid                                 1186151
D char[]     Username                              "oaaziz"
D u8         NumNodes                              4
D u8         PPN                                   16
D u8         NumThreads                            1
D u64[]      Pid                                   17749,17750,17751,17752,17753,17754,17755,17756,17757,17758,17759,17760,17761,17762,
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
D u64[]      BR_INST_RETIRED:COND:precise=1:u=1    5481289172,5476439288,5479319975,5479187901,5476609604,5484115830,5486034361,5471000
3680,5478787138,5478780103,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
D u64[]      BR_MISP_RETIRED:COND:precise=1:u=1    25062,28508,25330,28528,31715,24893,25337,41362,27098,25894,27211,30637,26173,40281,
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
D u64[]      BR_INST_RETIRED:ALL_BRANCHES:precise=1:u=1 5539782057,5530226180,5535554031,5534985031,5531577864,5544667106,5546656936,5487327
7217,5533621662,5534690788,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
D u64[]      BR_MISP_RETIRED:ALL_BRANCHES:u=1      47739,52183,47813,52661,56491,47695,48067,70879,50684,48385,50565,54872,48977,69469,
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Domain-specific sensor data collection from Trinity testbed

Domain-specific sensor sets (e.g., BRANCH, INSTRUCTION, GENERAL) selected at job launch time for use by application analysts

- Combined analysis with system-level data (e.g., network counters)

Combine application and system data to understand impact on performance of applications, contention, and system state

Application-Driven Information Integration

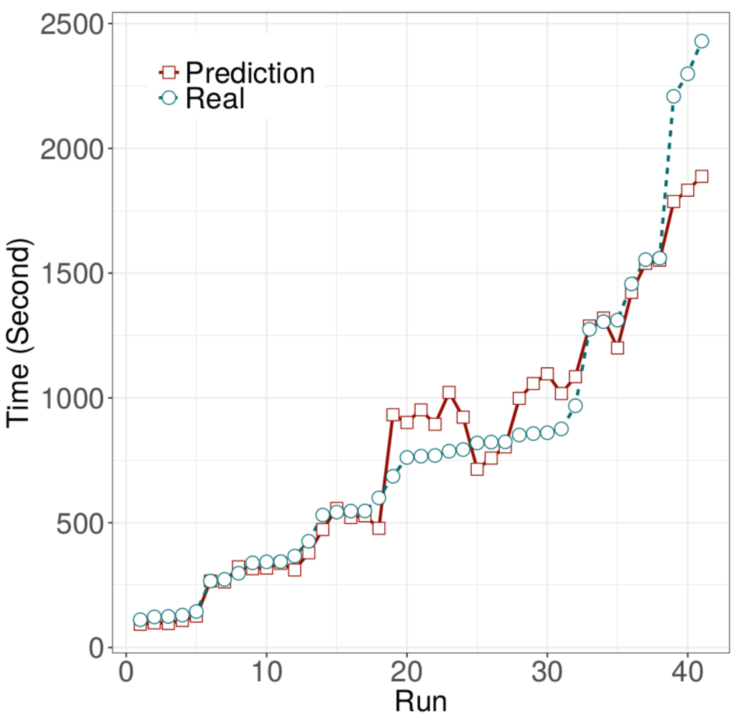

```
{
  "mpi-rank"             :    0,
  "total-app-time"       :    21.935,
  "total-kernel-times"   :    10.032,
  "total-non-kernel-times" :    11.903,
  "percent-in-kernels"   :     45.74,
  "unique-kernel-calls"  :    43,

  "kernel-perf-info"     : [
     {
     "kernel-name"   : "ApplyMaterialPropertiesForElems C",
     "region"        : "",
     "call-count"    : 50,
     "total-time"    : 0.004121,
     "time-per-call" :      0.00008242,
     "kernel-type"   : "PARALLEL-FOR"
   },
   {
     "kernel-name"   : "CalcAccelerationForNodes",
     "region"        : "",
     "call-count"    : 50,
     "total-time"    : 0.040885,
     "time-per-call" :      0.00081771,
     "kernel-type"   : "PARALLEL-FOR"
   },
   {
     "kernel-name"   : "CalcEnergyForElems A",
     "region"        : "",
     "call-count"    : 1750,
     "total-time"    : 0.076308,
     "time-per-call" :      0.00004360,
     "kernel-type"   : "PARALLEL-FOR"
   },...
```

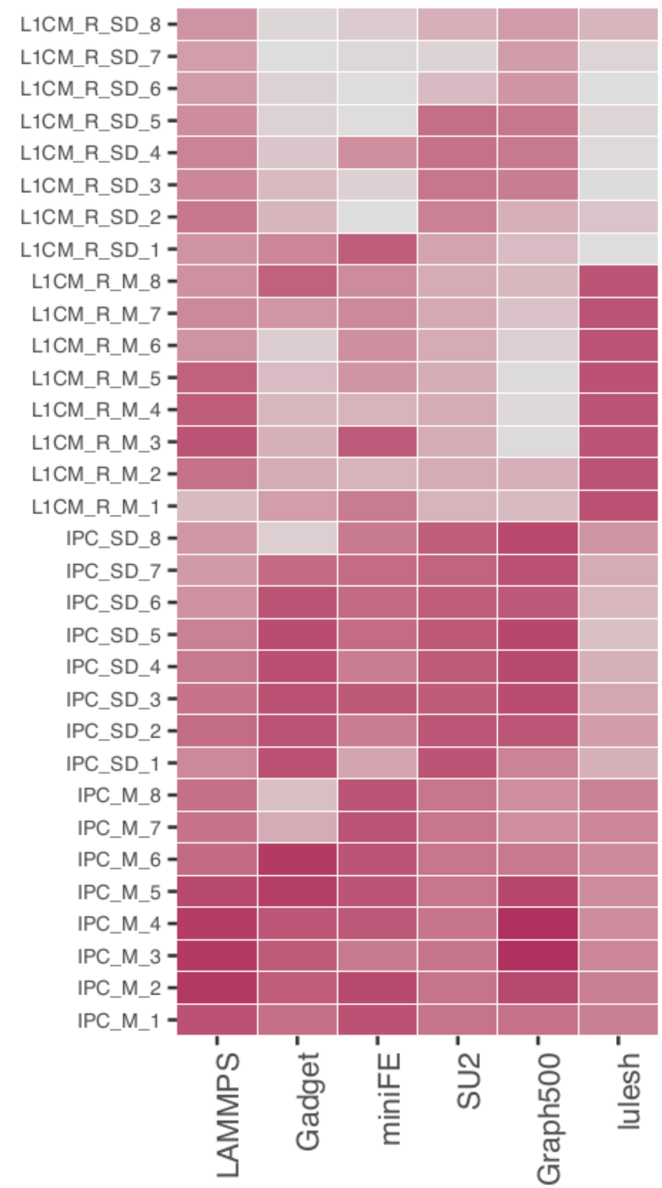

- ***Kokkos*** application kernel information collected and transported as LDMS sets
- Challenges:
  - Variable, run-time data representation
  - Data may be generated asynchronously across all ranks
- Analysis Output:
  - Job-based performance reports
  - Kokkos instrumentation relevant analysis (e.g., stats on kernel behaviors)

Heartbeat Profiling and Performance Prediction

- Assess performance sensitivity based on heartbeat progress in user-determined application regions
- Predict application runtime and detect progress problems

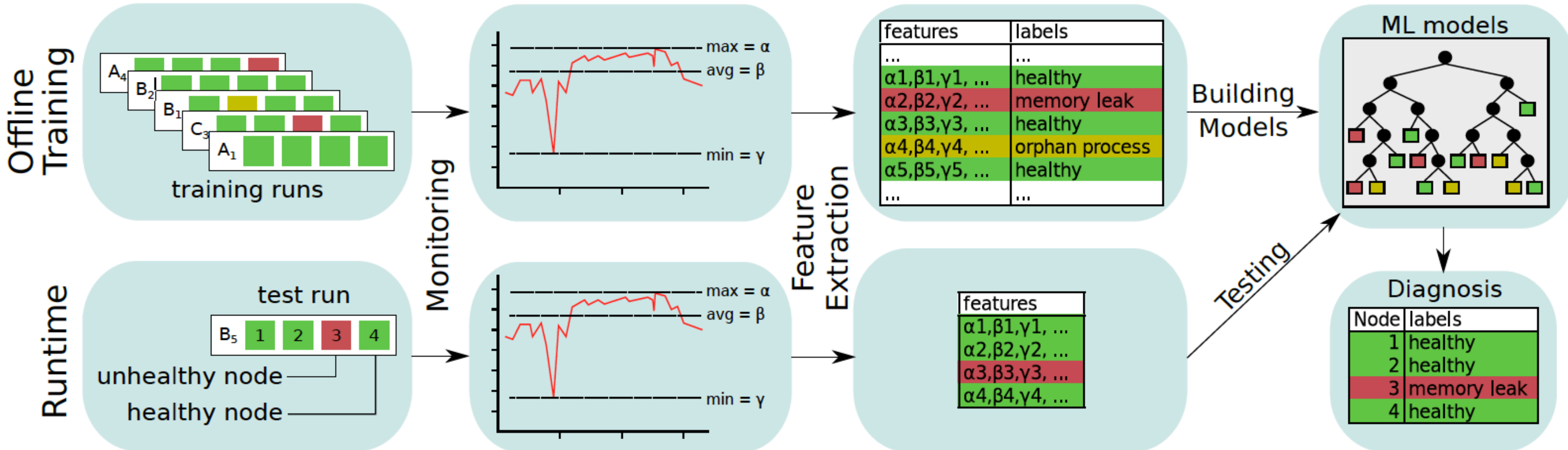


LAMMPS runtime prediction



Interval h/w counter importance heatmap

Anomaly Detection and Problem Diagnosis



Detection and diagnosis of performance problems

- Machine learning models built offline are used for classifying observations at runtime.
- Detect and diagnose behavioral differences due to: memory leaks, errant processes, contention, etc...

From: Diagnosing Performance Variations in HPC Applications Using Machine Learning ISC 2017 -- Gauss Award Winner

Baler Log File Analysis

- Run time processing of message data to discover patterns from messages

Timestamp	Component	Message Text
2016/4/8 06:20	c1-0c2s15n3	| HWERR[c1-0c2s15n3][20531]:0x4d12:SSID RREQ A_STATUS_AT_BOUNDS_ERR Error:Info1=0x82acc05020252:Info2=0x19c0009736000:Info3=0x79091

Count	First Seen	Last Seen	Pattern
594579	2016/4/8 06:20	2016/4/14 07:28	| HWERR[host][dec]:hex:* * A_STATUS_AT_BOUNDS_ERR Error:*=hex:*=hex:*=hex

  - Ease search space and discovery of similar and important events: Trinity Phase 2: Five months 4.5 billion loglines -> 11K patterns
  - Supporting new systems or rare events where the messages are unknown
  - Determine fault propagation via Association Rule Mining

Discover system and application impacts of events via integrated analysis of numeric data and log patterns

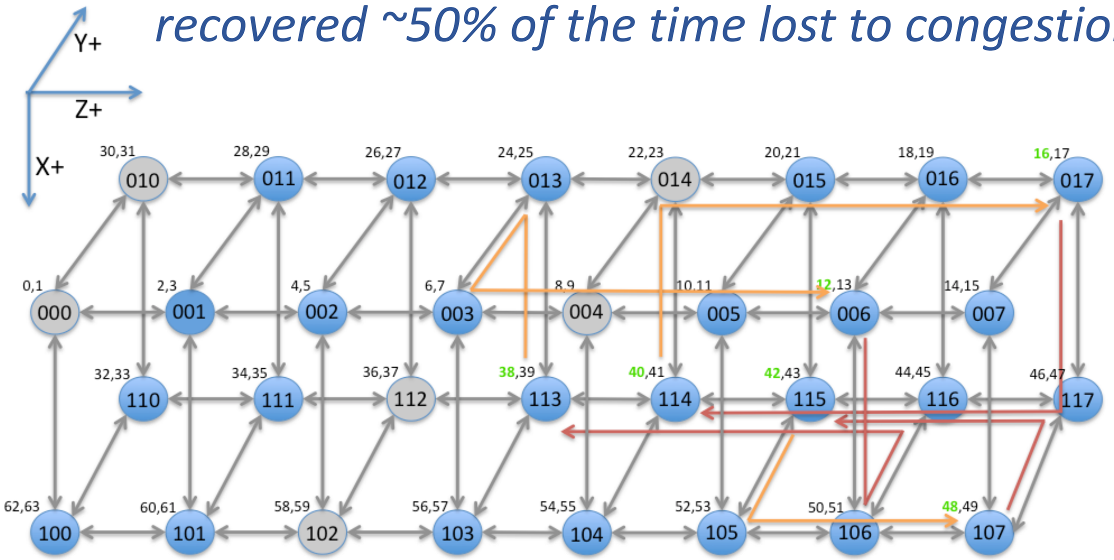
Analysis Framework

- Scalable Object Store (SOS) optimized for scalable storage and analysis of HPC system and application information in flexible formats
- SOS Data Access methods:
  - Command line interfaces for querying data and exporting as Text, CSV, or JSON
  - SQLite command shell
  - Native Application Programming Interfaces through C libraries
  - SciPy & Numpy interfaces to access SOS object data as zero copy ndarray: Arrays of data across components and time
- Supports continuous Analysis loop and/or post-processing
- Grafana visualization support of raw and derived quantities

Continuous analysis and visualization of integrated system and application data, in numeric and log formats. Enables run time understanding and response.

Feedback and Dynamic Response

Task remapping based on dynamic network information in a congested environment recovered ~50% of the time lost to congestion.



From: Demonstrating Improved Application Performance Using Dynamic Monitoring and Task Mapping HPCMASPA 2014

- Communication-heavy application run time affected by network contention
- Map tasks to nodes by minimizing total cost of communication
- Graph analysis: network architecture graph with edges weighted by congestion measures and overlaid with application communication patterns and sizes

Use application+system information for intelligent scheduling and task placement to improve runtime and throughput

Summary

Goal: Understand and mitigate performance variation through collection, analysis, feedback, and response to application needs and system conditions

- Unique ability to collect system data at resolutions necessary:
  - for detecting features and events of interest
  - to respond on meaningful timescales
- Analysis Challenges:
  - Large Data – high dimension, many variables, many components, time dependent
  - Integrated analysis of numeric and log data
  - Complex multi-subsystem interactions (facilities, network, filesystem)
  - Dynamic application demands, system state, and shared resources
  - Quantification of state variables on application performance unknown (e.g., relationship between congestion measures and application performance)
  - Requires run time analysis and decision support