LA-UR-18-28822
Approved for public release; distribution is unlimited.

Title: LANL Site Update

Author(s): Jennings, Michael E.

Intended for: HPCXXL Summer Meeting 2018, 2018-09-16/2018-09-21 (Oak Ridge, Tennessee, United States)

Issued: 2018-09-19 (rev.1)

LA-UR-18-28822

Los Alamos NATIONAL LABORATORY EST. 1943

LANL Site Update

Michael Jennings (*@mej0*) *mej@lanl.gov*

HPCXXL Summer Meeting 2018
Oak Ridge National Laboratory
Oak Ridge, Tennessee

UNCLASSIFIED

Los Alamos National Laboratory

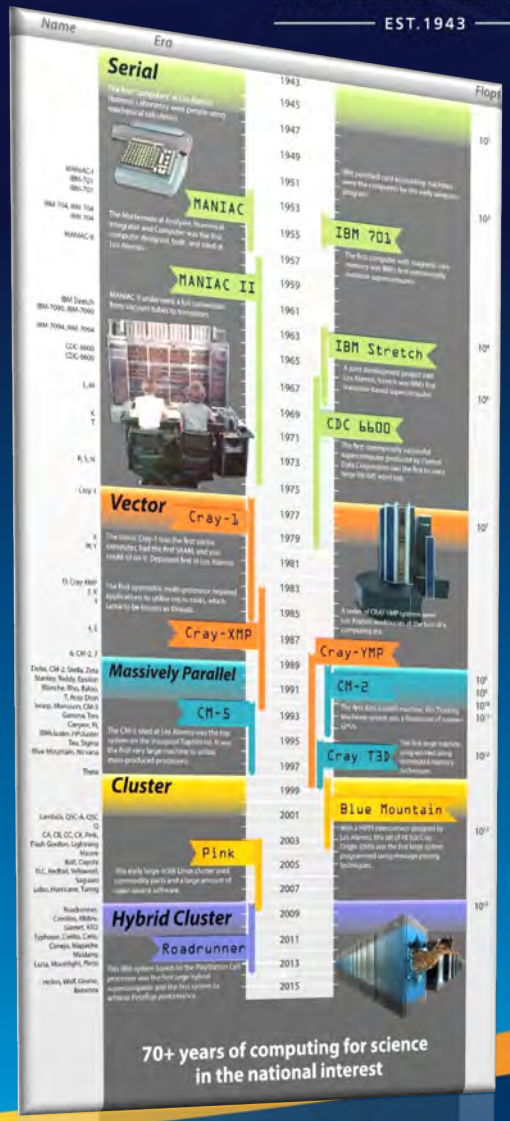
- Established in 1943 as “Site Y” of the Manhattan Project to create atomic bomb
- Mission: To solve National Security challenges through Scientific Excellence
- Part of the NNSA “Tri-Lab” partnership with Lawrence Livermore and Sandia Labs
- We perform a wide variety of classified and open scientific research and development.

- Funded primarily by the Department of Energy, we also do extensive work for/with the Departments of Defense and Homeland Security, the Intelligence Community, et al.
- Our strategy reflects US government priorities including nuclear security, intelligence, defense, emergency response, nonproliferation, counterterrorism, and more.
- We help to ensure the safety, security, and effectiveness of the US nuclear stockpile.
- Since 1992, the United States no longer performs full-scale testing of nuclear weapons. This has necessitated continuous, ongoing leadership in large-scale simulation capabilities realized through investment in high-performance computing.

LANL HPC Division

- LANL's history in high-performance computing is long and storied, dating back to the early '50s.
- Accomplishments include:
  - Helped IBM develop the 1st transistor-based supercomputer, Stretch
  - Our CM-5 was #1 on the inaugural Top500 List
  - The 1st vector computer, Cray-1, deployed here
  - 1st hybrid supercomputer (using IBM POWER and PlayStation Cell processors), Roadrunner, was also 1st to break the PetaFLOP/s barrier
- Led by Gary Grider, creator of Burst Buffer technology

- We support over 2000 unique users across more than 100 different classified/open science projects on 20+ clusters



UNCLASSIFIED

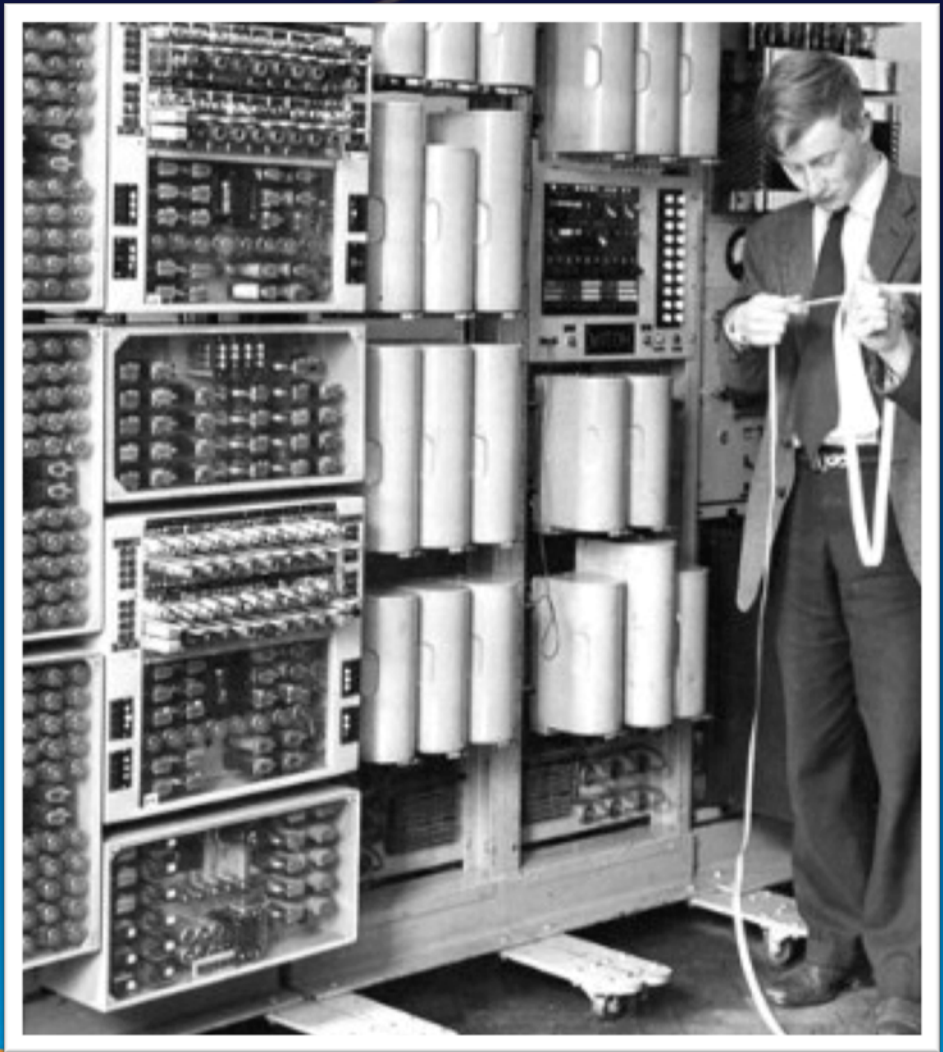
Exascale Compute Project (ECP) Container Working Group

Los Alamos NATIONAL LABORATORY EST. 1943

Two main subgroups have emerged:

- HPC Job Containers
  - Running user jobs in containers
    - Current state-of-the-art in HPC: Charliecloud (user namespaces) & Shifter (setuid-root w/ gateway)
    - Effort underway with Docker, Inc.
- Containers for Microservices
  - User-defined services
    - Web portals, database servers, etc. used by jobs
  - Admin-defined services
    - Standard cluster services, reimagined
- Technology landscape is enormous & changes daily!

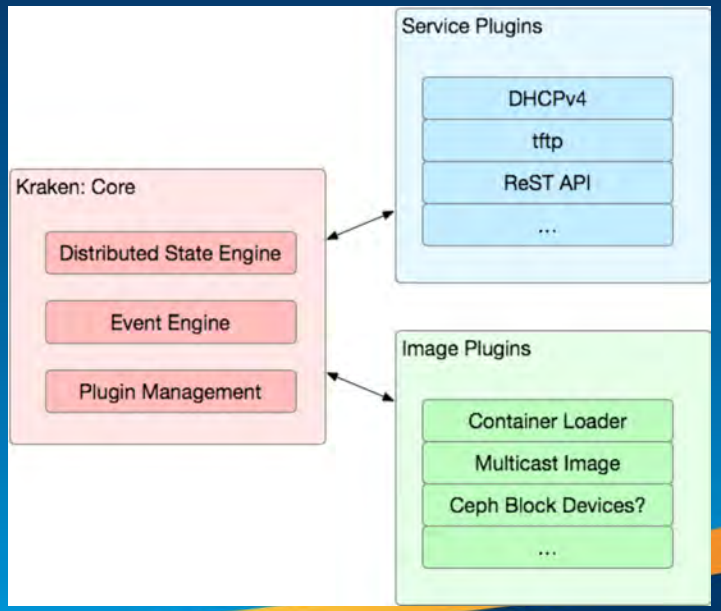
Cluster Provisioning



- Modern HPC cluster provisioning isn't.
- Current FOSS provisioning is outdated and unmaintained (Warewulf) or bulky and complex (xCAT) or massive and cloudy (OpenStack)
- We need something modular, dynamic, scalable, distributed, and **smarter**!
- We want to rethink cluster management and provisioning from the ground up.

Kraken Distributed State Engine

- Replacement for current cluster provisioning systems
- Written in modern Go; will be released under the GPL
- Converges “Config State” with “Discoverable State”
- CAP Theorem-based eventually consistent state
- State changes trigger Events
- Event listeners invoke actions
- Hardware- and OS-agnostic
- Maintain state, manage change
- All of the above are modules



Measuring Resilience: DIMM Fault Injection

Memory Error Injector (MEI)

- Sits between DIMM and DDR SDRAM slot
- Capable of intercepting requests and injecting faults into memory
- Programmable in Python
- Used as part of effort to simulate and measure effects of neutron radiation at high altitudes…like, say, 7300’ ASL.

https://www.wired.com/story/cosmic-ray-showers-crash-supercomputers-heres-what-to-do-about-it/

Measuring Resilience: ECP

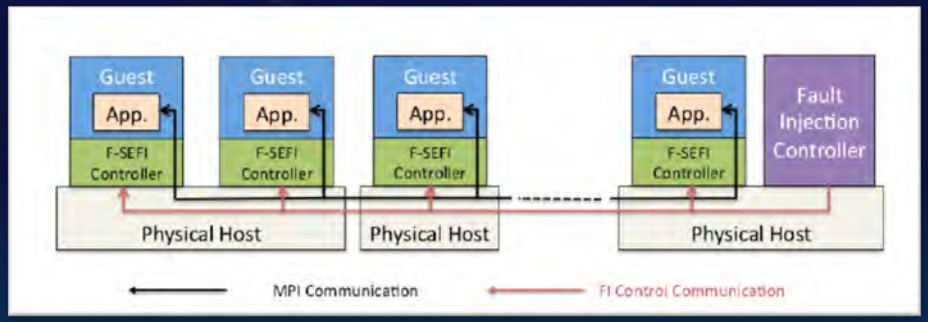
- Evaluate the reliability of DOE supercomputers and the resilience of ECP applications, making recommendations for improvements on both.
- Develop a component-based, hierarchical model used for evaluating hypothetical supercomputer architectures that can be fed field data from real DOE supercomputers.
- Develop a robust software fault injection capability used to quantify the resilience of ECP applications to emulated faults.
- Develop a wide range of fault models for the fault injector to allow it to emulate real world cases and adapt to new fault modes being discovered on the newest supercomputers and newest technology being produced for ECP systems.
- Ensure the fault injector, FSEFI, becomes a self-sustaining open source project
- Develop a robust testing framework for fault injection experiments allowing a user to conduct tens of thousands of trials, assemble the results, and look at vulnerability profiles to determine where to spend resources making their application more resilient.

Instrumenting MPI Applications

Los Alamos NATIONAL LABORATORY EST. 1943

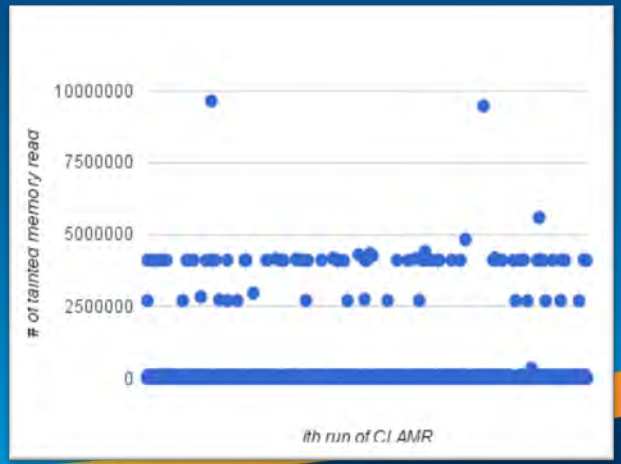
FSEFI's parallel fault injection infrastructure:

- Pluggable fault models
- Next step is demonstrating on appropriate applications



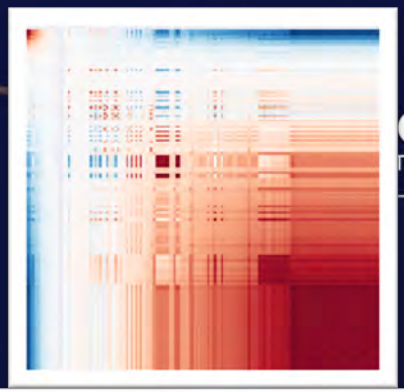
FSEFI v3 adds capabilities for tracing fault corruption as it moves inside a process and across processes in a parallel application

- We will demonstrate the usefulness of this and develop new visualization methods so application designers can understand how corruption spreads and look at techniques for containing the corruption
- Figure depicts the hydrodynamics adaptive mesh refinement (AMR) OpenCL application, CLAMR, and looks at how corrupted elements are used in other calculations and potentially taint their results as well.



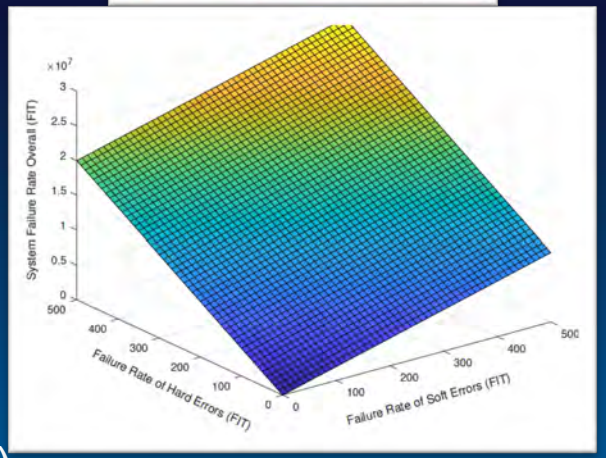
UNCLASSIFIED

Reliability/Resilience at Scale



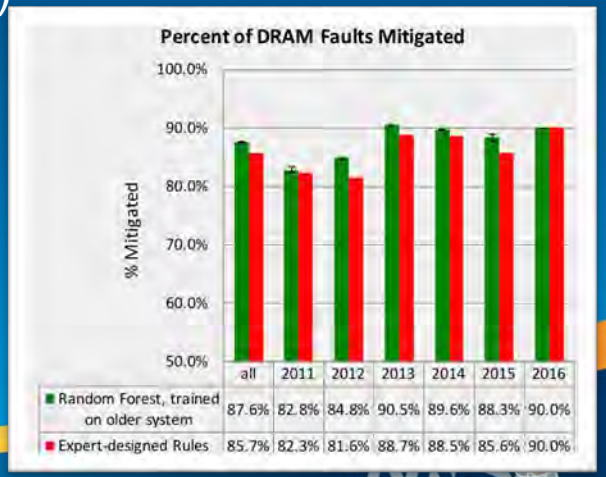
Supercomputer Reliability Model

- Model is developed and undergoing evaluation
- Demonstrate use of model evaluating system resilience given hypothetical vendor data, real data from Trinity, et al. (ORNL Catalog project?)



Machine Learning on System Logs

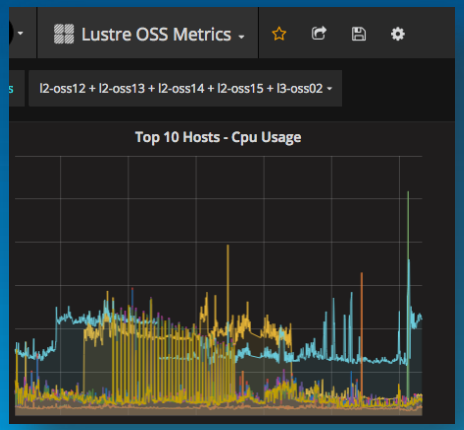
- Develop tool to use temporal characteristics of system logs (including telemetry/environmental data) for anomaly detection
- Develop tool to predict job outcome based on early warnings from system logs
- Test DRAM fault mitigation tool (already developed) on Trinity supercomputer

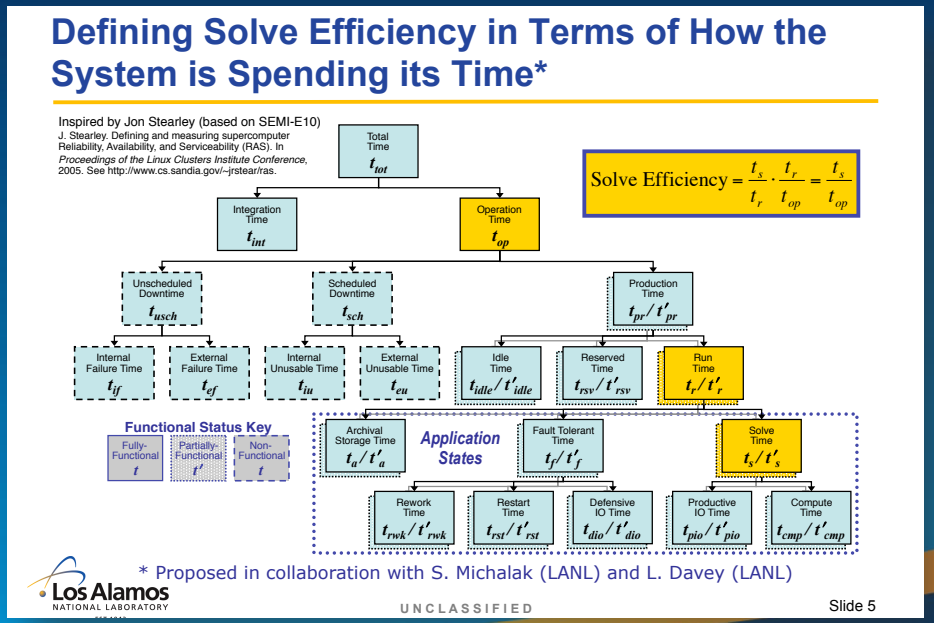


UNCLASSIFIED

What's Ahead in FY19?

- Release analytics framework for studying system reliability
- Demonstrate ElasticSearch and Spark system log analysis framework
- Leverage the cloud community's open source efforts
- Develop data-driven analytics tools for evaluating total cost of ownership for supercomputers such as the one proposed by Stearley (SNL), Michalak (LANL), and Davey (LANL).
- Extend modeling effort to incorporate results



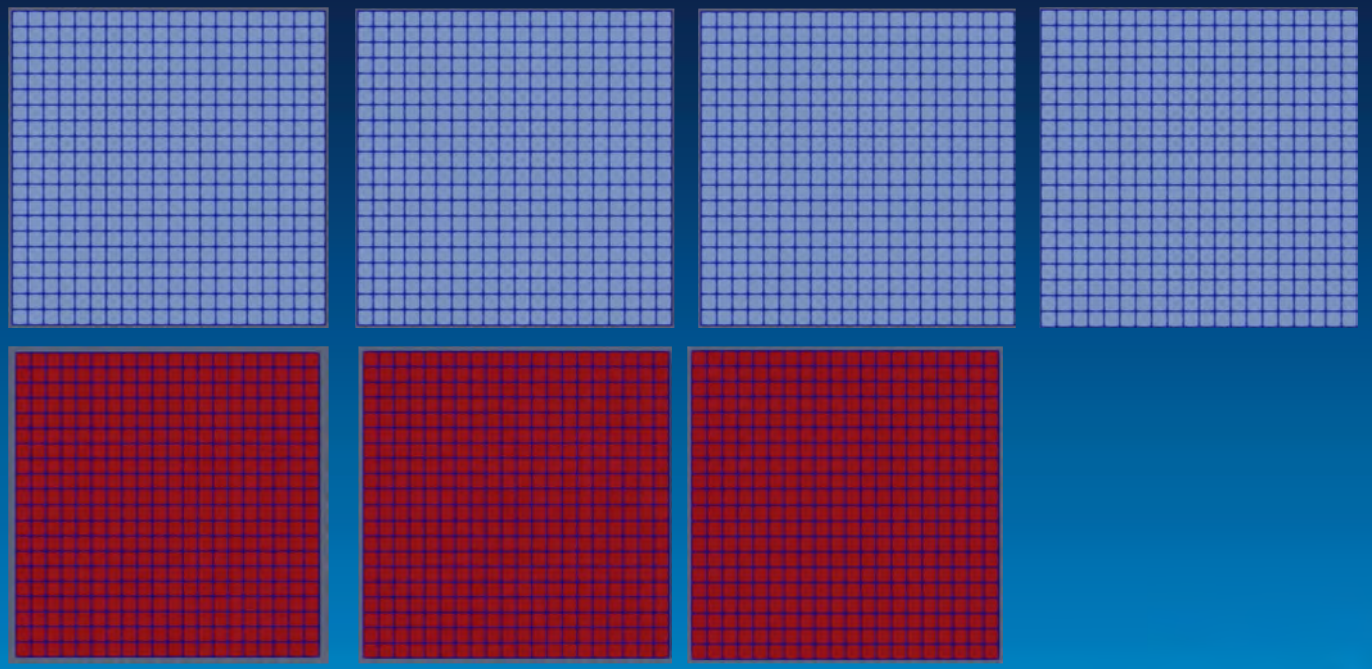


Resilience Project: Impact

- ECP applications need a way to evaluate their resiliency and vulnerability to fault modes of proposed ECP systems
  - This requires a more advanced, honest, fault injector with pluggable and complex fault models rather than simplistic “single bit flips”
  - This requires the ability to inject faults into real, parallel applications – not just toy serial codes
  - This requires a robust meta-framework for conducting experiments, consisting of tens-of-thousands of trials, and providing statistical analysis of the results for the user – fault injection as a service.
  - ECP applications need a way to *quantify* their vulnerability, implement protection mechanisms, and then quantify the resultant level of resiliency as well as the cost – e.g. reduction from 40% SDC rate to 5% SDC rate at the cost of 2.4% performance. Appropriate tradeoff? You decide.
- ECP hardware technology teams / datacenters need a way to evaluate proposed systems from vendors for system reliability
  - Code teams need an estimate of their hard fail / interrupt rate to properly develop mitigation strategies such as checkpoint intervals, quiescence methods, etc.
  - Code teams need an estimate of their soft error / silent data corruption rate to properly develop mitigation strategies such as algorithm-based fault tolerance (ABFT). Note silent corruption tradeoff mentioned above? This depends (1) on the fault models (realistic) and (2) on the fault rate. If it costs 3% performance to reduce vulnerability to a fault that will rarely (if ever) be seen, is it worth it?
  - Estimates from today’s machines are useful as a baseline but probably not relevant to proposed systems due to the extreme scale nature of the systems being proposed
- We provide open source tools for these, like FSEFI (https://github.com/lanl/pfsefi), with install script & user guide
- Model, analytics scripts, and tools to use model will all be open to all

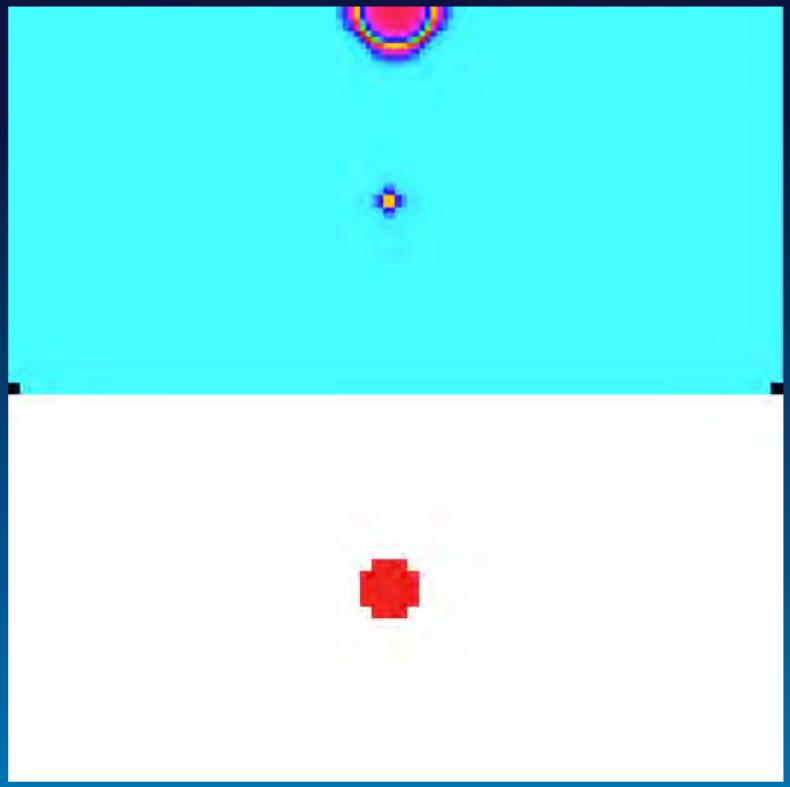
Let's Get Real!

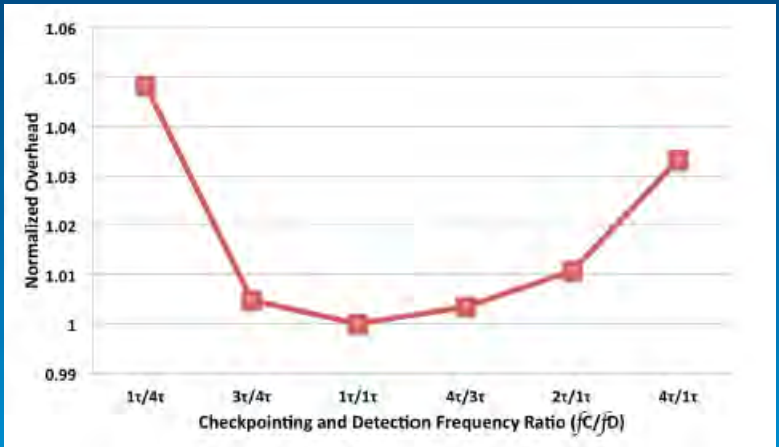
- Collaboration with Ben Bergen's ECP ATDM subproject (*2.2.5.01 ADNN01-ASC ATDM LANL Application*) on FleCSALE
- Studied mini-apps *hydro*, *maire_hydro*, *mish*, and *umma*
- Evaluated vulnerability to targeted fault injection placed deep in the applications' compute intensive kernels
- Corruption rate varied from (as low as) 14% for some mini-apps, in some source locations, to (as high as) 77% in others
- Visual examples of such corruption
- First column is correct simulation
- So what did we do?
- In collaboration with Bergen's team we developed an error detection and correction (EDC) mechanism based on physical properties of the system under simulation
- For a cost of 2.4% in performance we were able to correct 100% of emulated faults with this EDC technique



Another Real-World Win

- In collaboration with CLAMR (hydrodynamics OpenCL adaptive mesh refinement) project we evaluated the resilience
- Developed a threshold-based EDC method which captured 88% of injected faults.
  - Of the remaining 12%, only 4% caused bit-detectable changes
  - Tuning of the threshold could improve this of course
  - For example, this movie showing bit-for-bit differences in the simulation but at least no noticable differences by eye
    - Impactful? User decides
- Example usage, deciding ratio of detection to checkpointing and impacts on performance



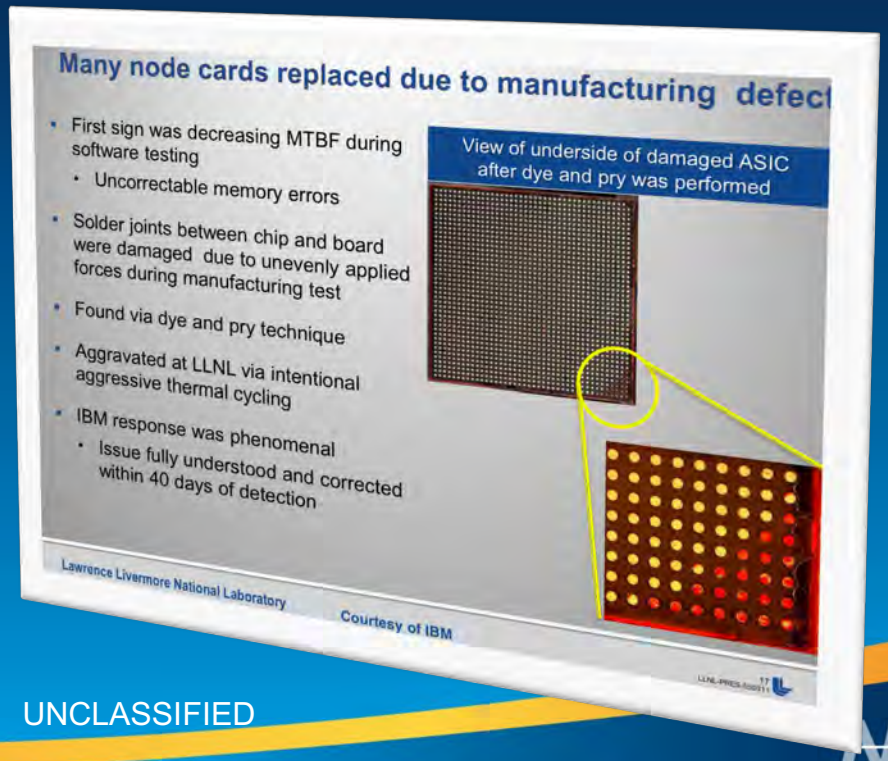


UNCLASSIFIED

Challenge: The Unknown

Hardware vendors can't predict all fault modes, so we can't emulate/model them!

- This is almost a sure thing given every new supercomputer has fault modes which were previously unknown. Hard to explain specifics due to NDA nature, but this is across DOE. This is the nature of pushing the bleeding edge -- we're on the tails of the distribution.
- One public example is from Sequoia at LLNL, from Kim Cupps' presentation at SC a few years ago
- We cannot plan for this, yet it *will* occur!
- Modeling work has elements meant to account for this, but it is unlikely vendors know the failure rate themselves…



UNCLASSIFIED

LBNL Node Health Check

- https://github.com/mej/nhc/ for docs and to participate!
- Yes, that's still its name. Got a better one?
- We *finally* got approval for FOSS contributions!
  - This means a new NHC release can happen soon
- We are working on supplementing Cray NHC with LBNL NHC on Trinity & Trinitite
- Work is underway to replace multiple home-grown tools

- There's an entire Git repo full of code tweaks and new checks just waiting to be pushed….
- Beware issue on large-thread-count platforms (e.g., KNL)

Charliecloud

- LANL's Container Runtime
  https://github.com/hpc/charliecloud
- 2018 R&D 100 Finalist!
  Congratulations to NERSC's Shifter, also a Finalist!
- Recent developments in version 0.9.x (currently 0.9.2):
  - New `vagrantfile` for generating Charliecloud-enabled (and Docker-enabled) VM images based on CentOS 7 Virtualbox image
  - New example containers and tutorials based on MPICH, Spack, and more
  - New `ch-fromhost` utility to seamlessly integrate host-based resources into Charliecloud containers (HSN, GPU, libraries, etc.)
  - Significantly improved documentation (and how it gets generated on RHEL-based platforms)

Questions/Comments?

THANK YOU!

PS: These slides, and all unclassified LANL publications, are available via LA-UR number at:
http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-18-28822

UNCLASSIFIED