



Root Cause Analysis

October 15, 2008

LACSS Resilience Workshop

Presented by Jon Stearley

Sandia National Laboratories

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Agenda

(10 slides, 3 goals)

Get your help on Root Cause

- How to represent interdependencies?

Encourage you towards standardized validation data

- Component Operations Status (COS)

Make you aware of some other work

- Sisyphus (Logs)
- 9Lives (OS)

Context: Resilience Activities at Sandia

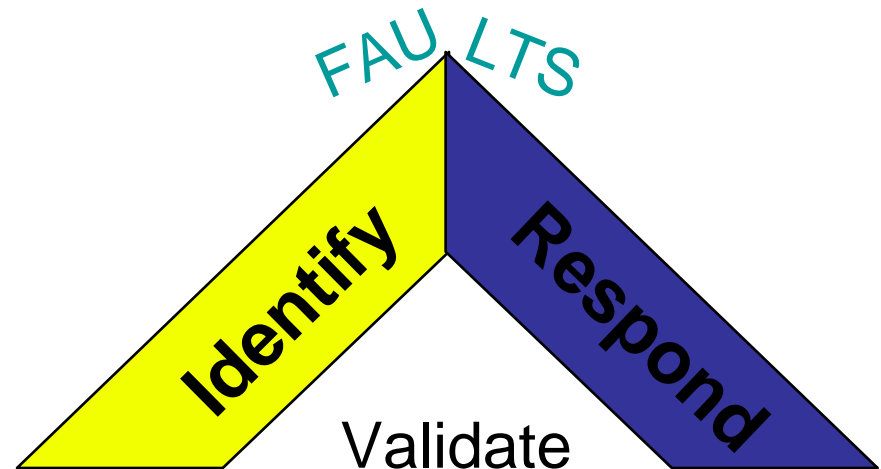
Goal: Automatic *identification* and *response* to faults

Identify Faults (CSSE)

- Failure Prediction
- Root cause analysis
- *Impact: enable timely and focused response*

Respond (LDRD):

- System-Directed Resilience
- *Impact: enable apps to run continuously despite faults*



Root Cause: Big Cheese Model

* Holes: incomplete data, incomplete time, incomplete components, incomplete dependencies

Time

Data

Components & Dependencies!

QuickTime™ and a
IFF (Uncompressed) decompressor
are needed to see this picture.



What Stinks?!#

Given hints (influences search path):

Distinguishing Symptoms

- Text (logs, username, job id, rank id, ...)
- Numbers (temperature, correlation, ...)

Suspect Components

- Hardware (racks, cores, routes, ...)
- Software (daemons, apps, libraries, ...)

Dependencies

- Functional, Physical
- Static, Dynamic

Identify:

Root Cause (likelihood ranked)

Root Cause: Big Cheese Model

* Holes: incomplete data, incomplete time, incomplete components, incomplete dependencies

Given hints (influences search path):

LABELS

Distinguishing Symptoms

- Text (logs, username, job id, rank id, ...)
- Numbers (temperature, correlation, ...)

VERTICES

Suspect Components

- Hardware (racks, cores, routes, ...)
- Software (daemons, apps, libraries, ...)

EDGES

Dependencies

- Functional, Physical
- Static, Dynamic

quickTime™ and a compressed) decomposed to see this picture

$G=(V,E,L)$

Properties?

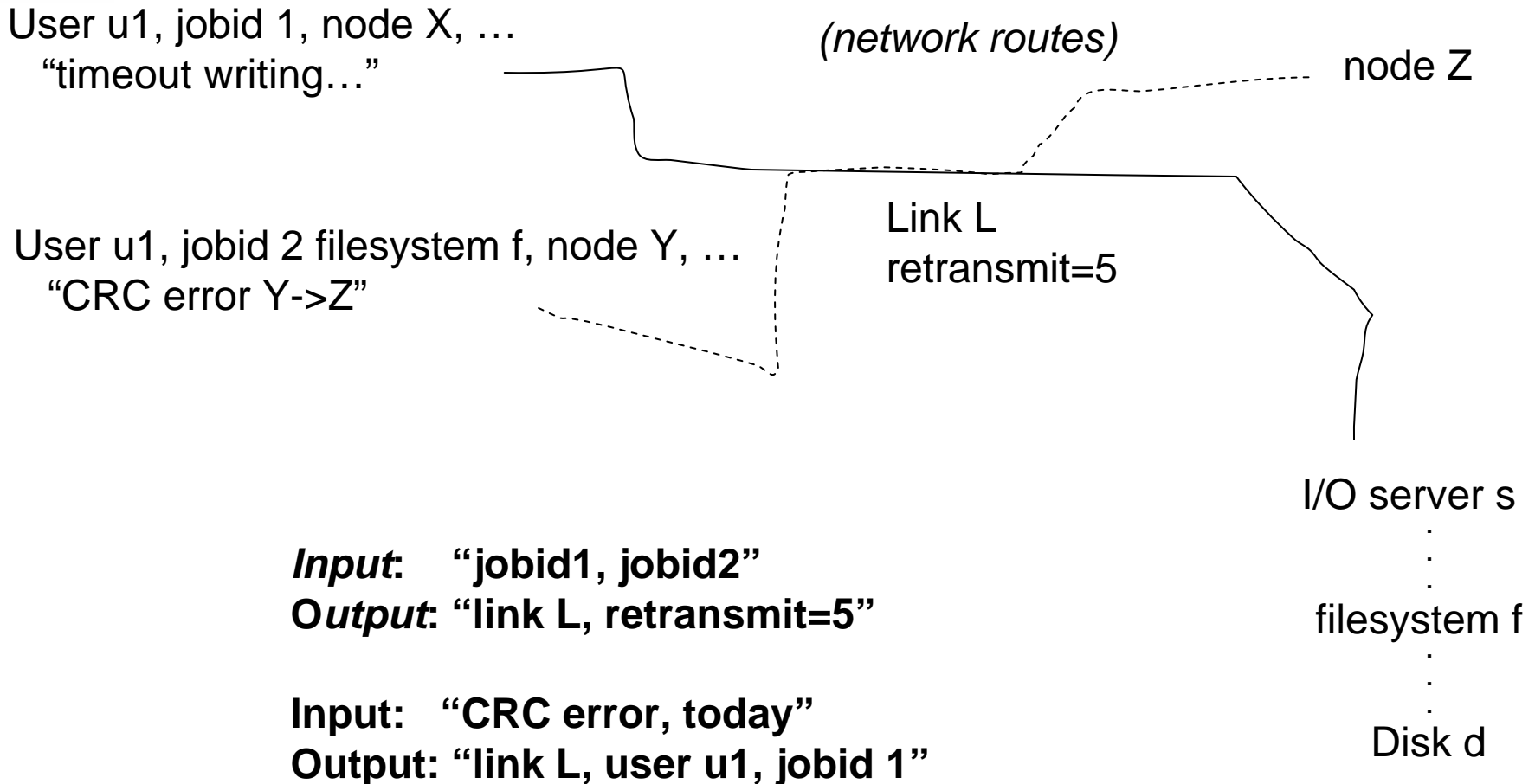
Useful model?

Useful algorithms?

Identify:

Root Cause (likelihood ranked)

Root Cause: Flakey Link (e.g.)



FY'09 Root Cause Deliverables

First Quarter

- Mathematical formulation of at least three important but currently non-computable root cause scenarios based on current systems

Second Quarter

- Select appropriate algorithms for solution.

Third Quarter

- Demonstrate proof-of-concept solutions.

Fourth Quarter

- SAND report

Validation:

“Quantify our ability to predict node failures on the TLCC platform”

Component Operations Status (COS)

Production Uptime (PU) = ready for immediate use by one or more production user

Scheduled Downtime (SD) = not in PU for scheduled reasons

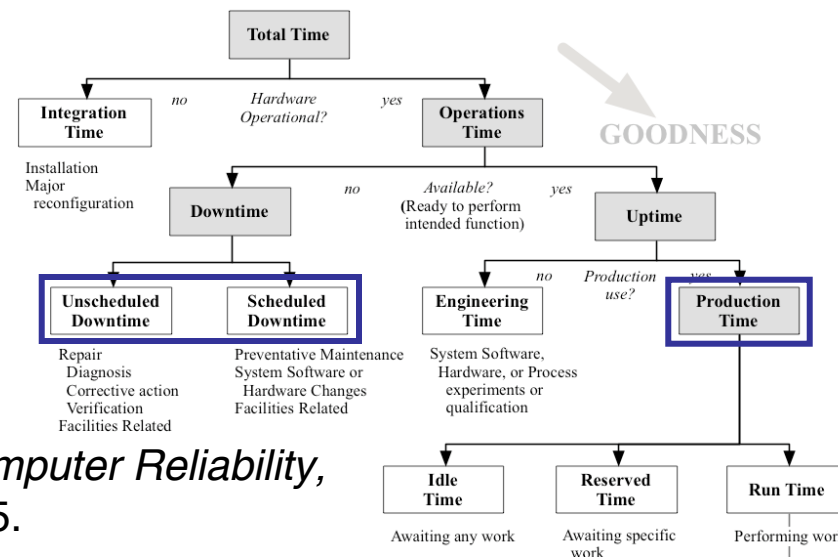
Unscheduled Downtime (UD) = not in PU for unscheduled reasons

FAILURE = the onset of Unscheduled Downtime

FY09:

1. Collect per-node COS on TLCC (admins express “SD” on CLI)
2. Perform prediction experiments on symptomatic data (numeric, text)
3. Use COS to quantify (and validate) prediction results

COS is a proposed standard



Stearley. *Defining and Measuring Supercomputer Reliability, Availability, and Serviceability (RAS)*, LCI05.

Responding to Faults (9Lives)

System-Directed Resilience (LDRD)

Three Primary Research Thrusts

- **Application quiescence**
 - Suspend App, handle in-flight messages and in-progress IO
- **Efficient state management**
 - Identify critical state (app characterization, app guidance, changesets)
 - System guidance for when to extract state (eg MTTI)
 - Explore diskless methods and compression
- **Fault recovery**
 - System software to support dynamic node allocation
 - Explore network virtualization to abstract physical node ID from software.
 - Explore efficient methods for state recovery (roll-back, roll-forward techniques)

Low Overhead (easier to develop and support, faster to run)

- Lightweight Kernel (Kitten)
- Stateless Networking Protocol (Portals)

End

Get your help on Root Cause

- How to represent interdependencies?

Convince you towards standardized validation metrics

- Component Operations Status (COS)

Make you aware of some other work

- Sisyphus (Logs)
- 9Lives (OS)

Extra Slides

Identifying and Predicting Faults

Research to Identify Root Cause

- Develop mathematical methods to describe and track dependencies between system components
- Develop tools to analyze models, identify root cause, and provide feedback to fault-response systems (e.g., the system directed LDRD)

Research on Fault Prediction

- Develop methodologies for anomaly detection and quantification
- Correlate anomalous behaviors with root causes
- Use automated anomaly detection and learned correlations to predict failures with a calculated level of confidence

Impact

- Critical enabler for automated response (also useful to admins)
- Mathematics necessary to design and operate truly resilient systems
- Quantify the system-wide impact of failures.

FY'09 9Lives Deliverables

First Quarter

- Enumerate list of faults, likelihood of fault, and possible response (Stearley)
- Investigate options for quiescence (Brightwell)
- Investigate diskless state management and node recovery (Oldfield)
- Investigate related projects – BLCR (Kordenbrock)

Second Quarter

- Complete design of code to trace application memory usage (Pedretti)
- Complete design of network virtualization layer (Brightwell)
- Complete design of fake RAS system (Riesen)
- Complete design of diskless state management and recovery (Oldfield)

Fourth Quarter

- Memory-use characterizations for selected applications (Pedretti)
- Demonstrate response to link failures – may not need node recovery (Laros)
- Demonstrate network virtualization layer (Brightwell)
- Demonstrate fake RAS – inject faults (Riesen)

FY'09 Prediction Deliverables

- **First Quarter**
 - Selection of algorithms for anomaly detection (multivariate distributions and time series phenomena)
 - Define database schema for collection of system variables and for representation of data to support the different analyses
- **Second Quarter**
 - Quantification methodologies, failure definition and recording mechanisms, data collection (CPM, Log, Workload, COS, etc.)
- **Third Quarter**
 - Implementation and validation of anomaly detection algorithms
 - Selection of algorithms for correlation of failures with anomalies (classification and root cause inference)
 - Attributes selected for predictive analysis
 - Scalable data collection and representation
- **Fourth Quarter**
 - Implementation and validation of failure to anomaly correlation techniques
 - Write SAND report documenting:
 - Quantification of the ability to predict failures on the TLCC platform
 - Additional information/data that should be collected, as well as suggested scalable collection mechanisms, in order to improve failure predictability on future platforms

Proposed Process for Collecting per-node COS on TLCC

Scheduled
Downtime

Production
Uptime

Unscheduled
Downtime

SLURM and MOAB logs and databases are postprocessed into COS records.

All downtimes are counted as Unscheduled, unless distinguishable otherwise:

Only the admins know if a downtime is scheduled or not, so they must be involved in distinguishing them. The easiest way to do this would be to use a flag to `scontrol` and `mrsvctl` like:

```
scontrol -sd [cause...]  
mrsvctl -sd [cause...]
```

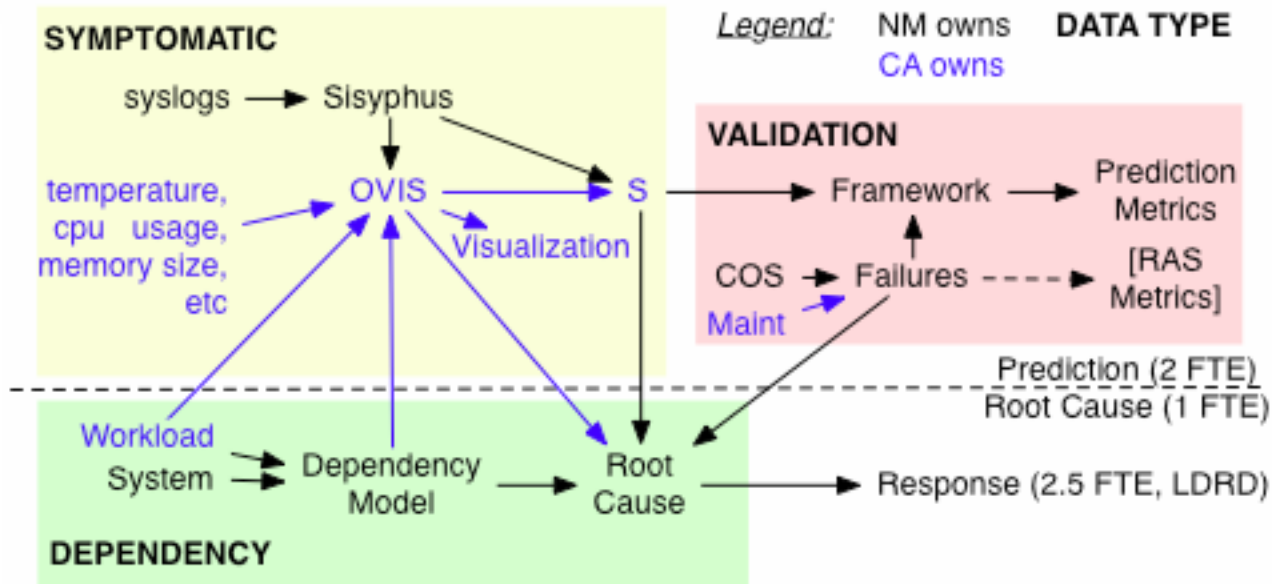
Which would be shorthand for

```
scontrol reason="SD: [cause]"  
mrsvctl -a ACCT=SD -D "SD: [cause]"
```

Where [cause] is an optional argument describing the downtime.

-sd could be added, aliased, or wrapped. More details are on the next slide.

SNL Resilience Interactions



Details:

Prediction Signal S: $s_i=(t,n,c)$ where t is time, n is node, and c is confidence

Failures F: $f_j=(t_i,n,d)$ where t_i is time of failure, n is node, and d is duration

COS is Component Operations Status (uptime, sched and unsched downtime)

Maint is maintenance records (hardware replacements etc)

Framework: how well does S predict F? $P(F | S)$

Prediction Metrics: Precision, Recall, AUC

Workload: jobs (user, nodes, duration,...) and executables (versions, libraries,...)

System: hardware and software configurations (cores, cables, routes, daemons,...)

Dependency Model: $G=(V,E)$ where V are components and E are dependencies

Root Cause: $P(G' | S,F)$ based primarily on dependency model

*Root Cause: $P(n | S)$ based primarily on symptomatic inference

[RAS Metrics]: no FY09 work is planned, but FYI COS is also key for:

MTTI, MTTR, component pareto (including CCF via Dependency Model), etc

Root Cause: Graph Model

* Holes: incomplete data, incomplete time, incomplete components, incomplete dependencies

Given hints (influences search path):

LABELS

Distinguishing Symptoms

- Text (logs, username, job id, rank id, ...)
- Thresholds (temperature limits, ...)
- Waveforms (cpu profile, ...)
- Correlations

VERTICES

Suspect Components

(paths)

- Hardware (racks, cores, routes, ...)
- Software (daemons, apps, libraries, ...)

EDGES

Dependencies

- Functional, Physical
- Static, Dynamic

quickTime™ and a compressed) decomposed to see this picture

$G=(V,E,L)$

Properties?

Useful model?

Useful algorithms?

Identify:

Root Cause (likelihood ranked)