SAND2012-4042P

FINDING FAULTS

Root-Cause Inference in HPC Systems

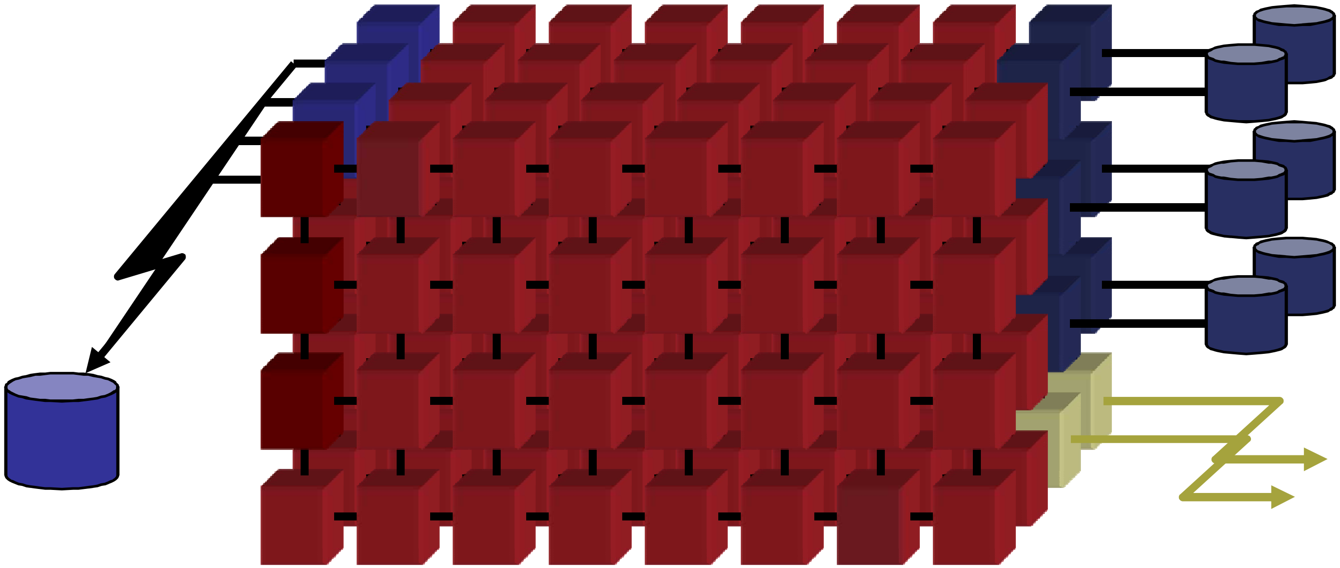
JOWOG-34 Meeting
May 23, 2012

Jon Stearley <jrstear@sandia.gov>

Example: find the silent bad nodes

Goal: Greatest capability job

Fault diagnostic: Job Pass/Fail



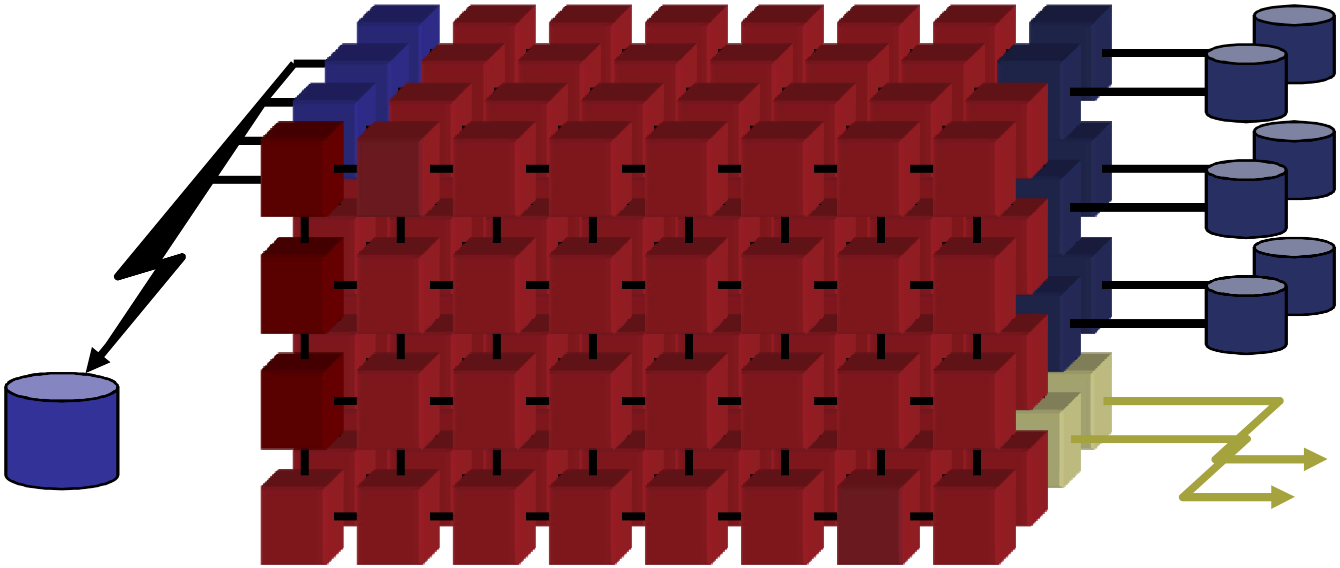
Approach: Recursive bisection

Current Practice: Manual, time-intensive process

Example: find the silent bad <u>cables</u>

Goal: Greatest capability job

Fault diagnostic: Job Pass/Fail



Approach: Recursive bisection (amidst dynamic routing)

Current Practice: <u>Manual, time-intensive process</u>

Overview of Resilience Efforts at SNL

Reducing the <u>effects</u> of faults (undesired events)

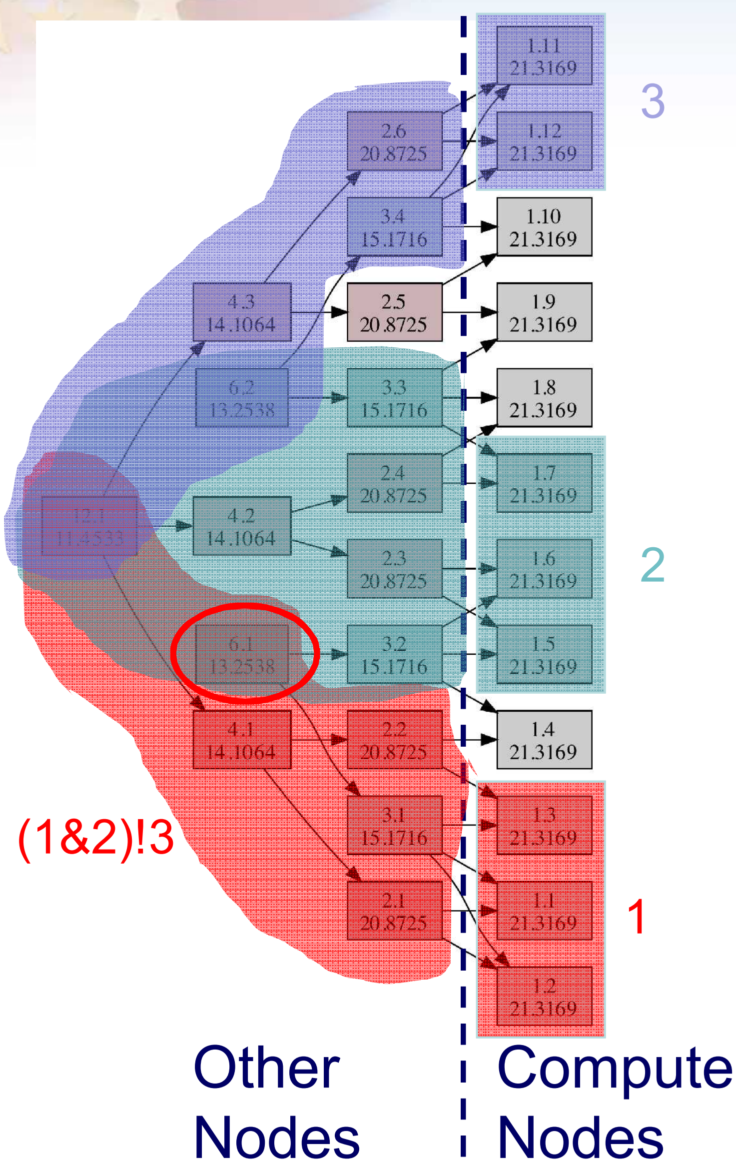
- **Algorithm: Resilience API, GMRES-FT, …**
- **System: Process replication, rMPI, …**

Reducing the <u>occurrence</u> of faults

- **Design: SST, Procurement requirements, …**
- **Operation: Monitoring (e.g. Splunk), Inference, …**

Outline

1. Problem Statement (DONE)
2. Overview of Resilience Efforts (DONE)
3. Approach
4. Results
5. Direction



Given:

System graph

Nodes = components

Edges = dependencies

Job log

JOB	START	STOP	FAIL	NODES		
1	0	321858	1	1.1	1.2	1.3
2	1736	321858	1	1.5	1.6	1.7
3	276498	557283	0	1.11	1.12	

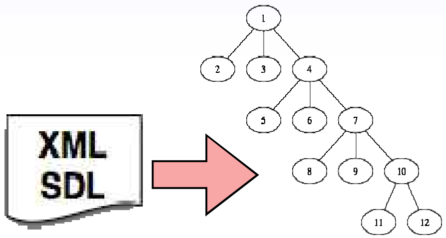
Infer:

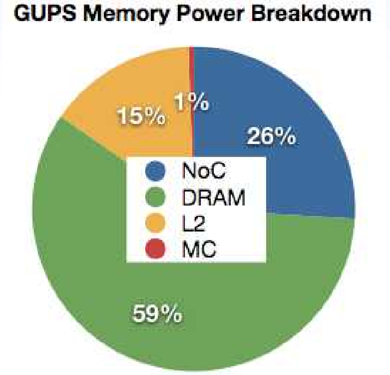
Node failure rates

Structural Simulation Toolkit (SST)

Goals

- Become the standard architectural simulation framework for HPC
- Be able to evaluate DoD/DoE workloads on future system designs
- Use supercomputers to design supercomputers





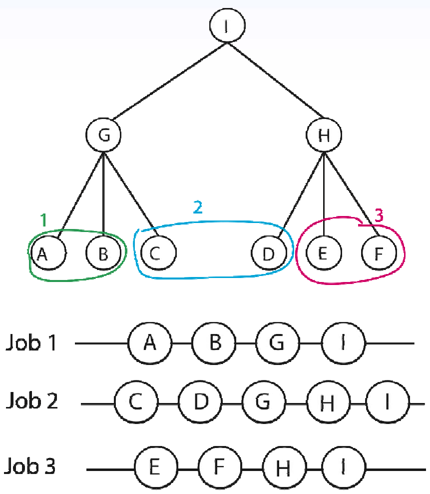
Technical Approach

- Parallel Discrete Event core with conservative optimization over MPI
- End-to-end simulation
  - Integrated Tech. Models for power
  - McPAT, Sim-Panalyzer
- Multiscale
- Open Core, non viral, modular
- Modules include: power, network processor, memory, resilience

Consortium

- Combine Lab, academic, & industry

Conditioned Maximum Likelihood Approach



- **Background**
  - **Set of components in each job is known.**
  - **The true source of failure is *masked*.**
  - **By considering the operational state of other jobs in system, we can find a minimum subset of components M={s_j}, |M|=m_j, that may be responsible for the failure of job *j* .**

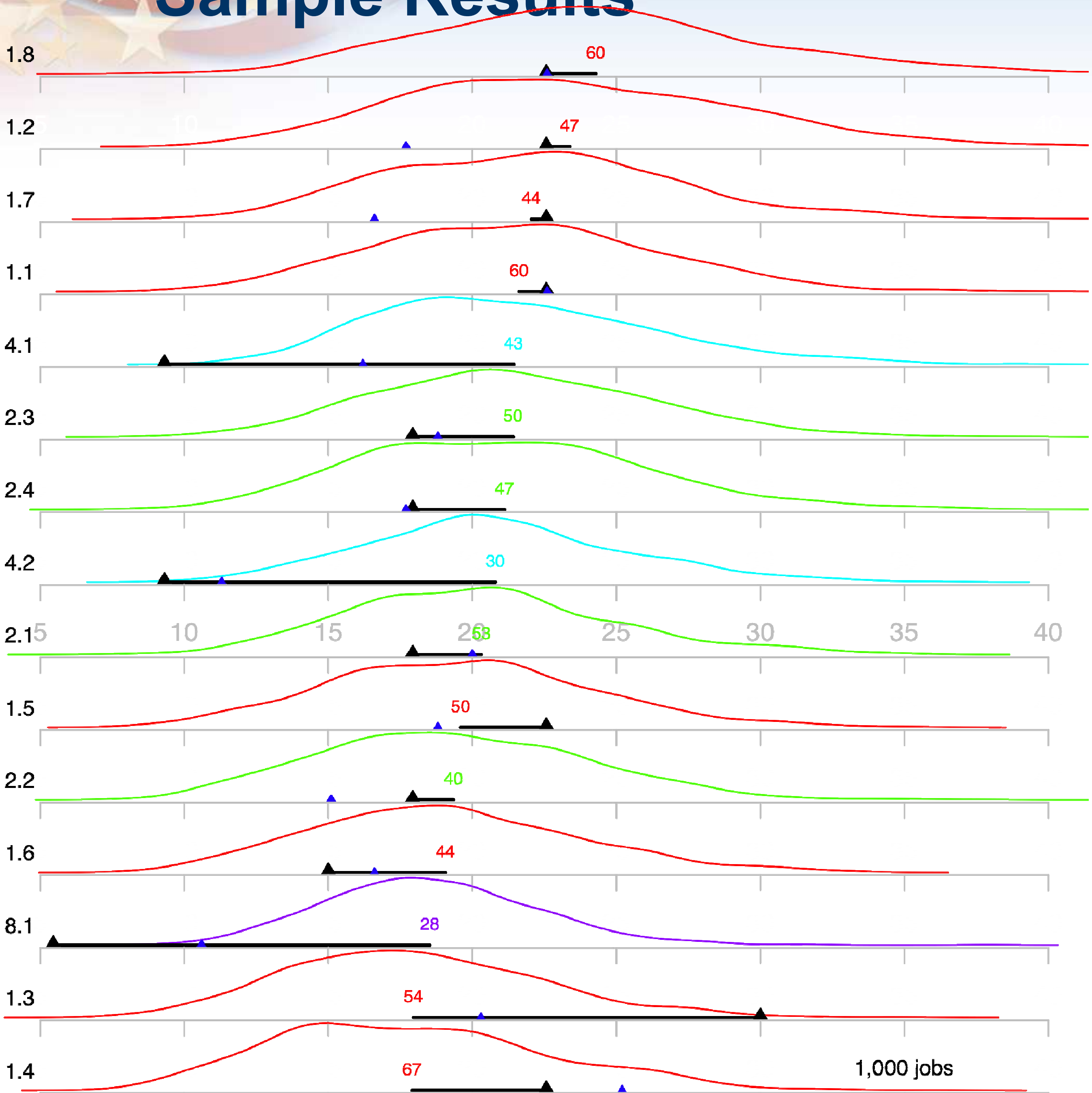
- **Conditional Likelihood Function**
  - **Let $f_i(t|\theta)$ be the PDF of the time to failure for component *i* and**
  - **Let $R_i(t|\theta)$ be the reliability function for component i.**
  - **Θ is vector of unknown distribution parameters with joint PDF conditioned on the job data: g(θ|data)**
  - **Under the environment of masked observations, it can be shown that the likelihood function is given by:**

$$L = \prod_{j=1}^{N}\left[\sum_{i\in s_j}\left(f_i(t_j|\theta_i)g(\theta_i)\prod_{s=1,s\neq i}^{m_j}R_s(t_j|\theta_s)g(\theta_s)\right)^{v_j}\left(\prod_{s=1}^{m_j}R_s(t_j|\theta_s)g(\theta_s)\right)^{1-v_j}\right]$$

  - **As more data is accumulated, the underlying PDF of the distribution parameters, g(θ|data), is updated. v_j is an indicator variable for censoring (=1 if job *j* fails).**

- **Solution Method**
  - **Find the set of parameters θ that maximizes the likelihood function conditioned on the uncertainty in the observations. Very hard…**
  - **Use Markov Chain Monte Carlo methods (e.g. Gibbs sampling) to find the best combination of parameters that explain the data.**
  - **Easy to implement, and parallelize.**

Sample Results



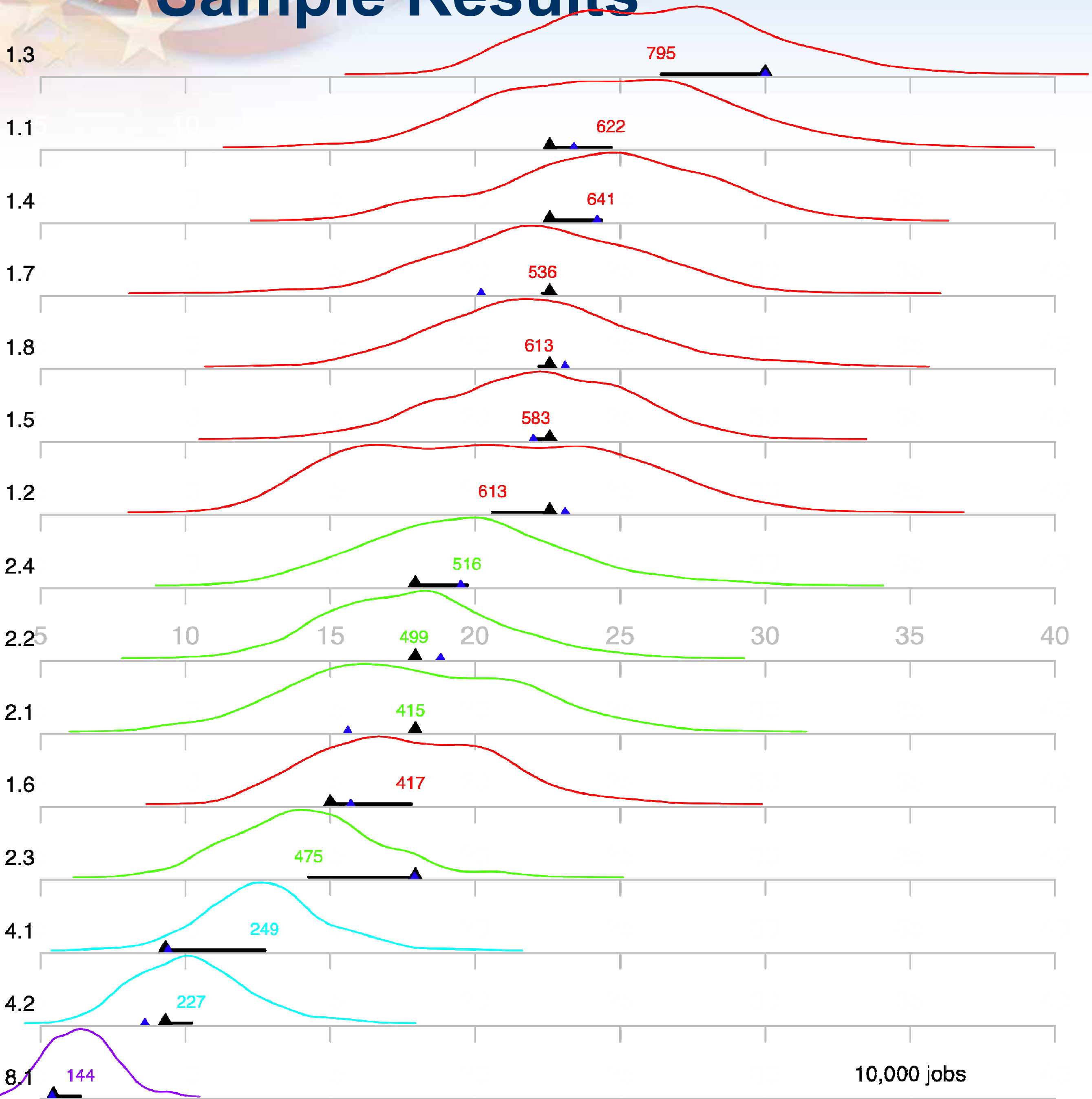
Fault rate PDFs:

- **X-axis is fault rate**
- **Y-axis is likelihood**
- **One row per component**

- **True source of failure is masked.**
- **But underlying failure rates can be inferred.**

- **Components are ranked by decreasing average fault rate**

Sample Results



Fault rate PDFs:

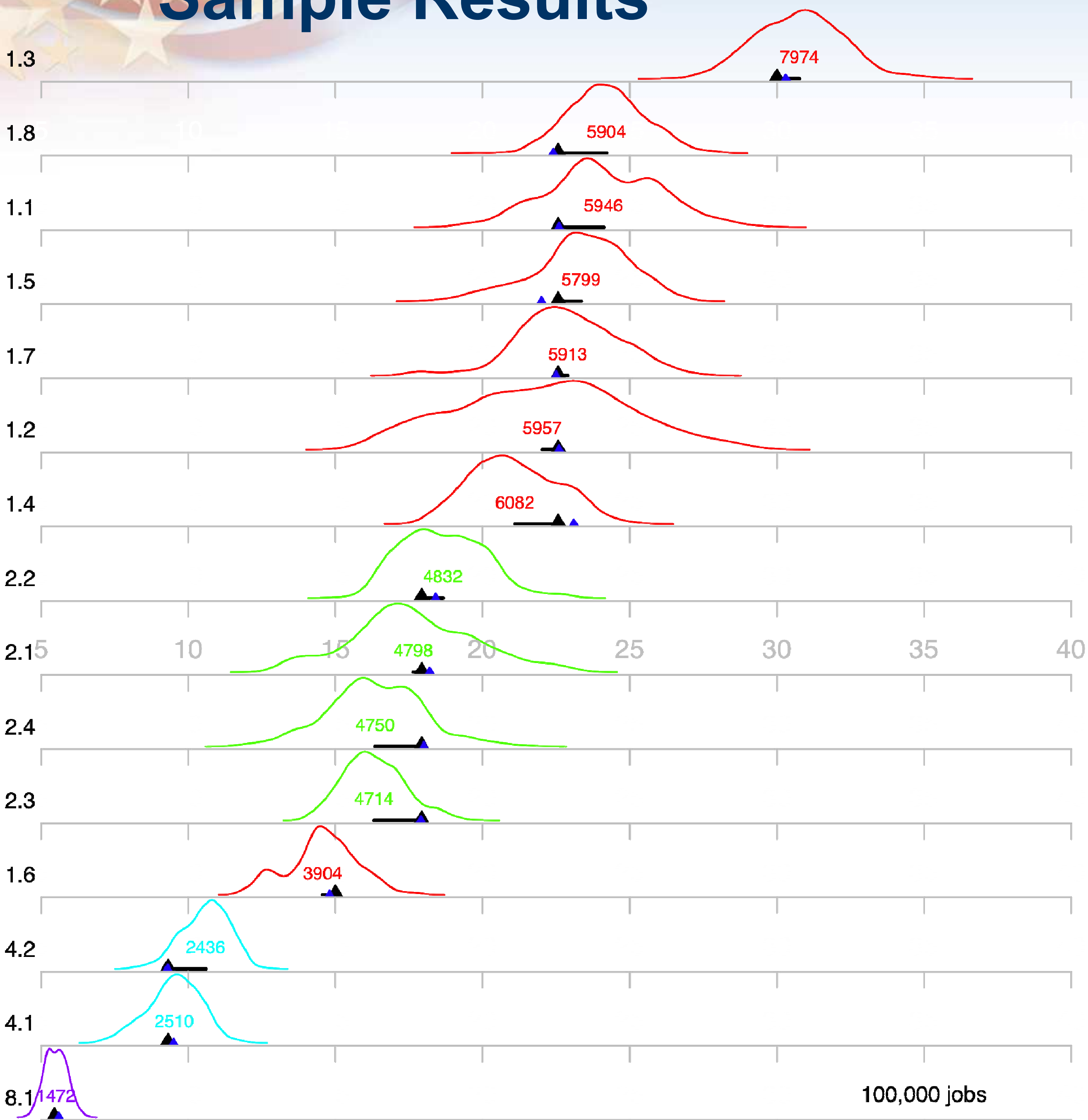
- X-axis is fault rate
- Y-axis is likelihood
- One row per component

- True source of failure is masked.
- But underlying failure rates can be inferred.

- Components are ranked by decreasing average fault rate

- More observations, less uncertainty

Sample Results



Fault rate PDFs:

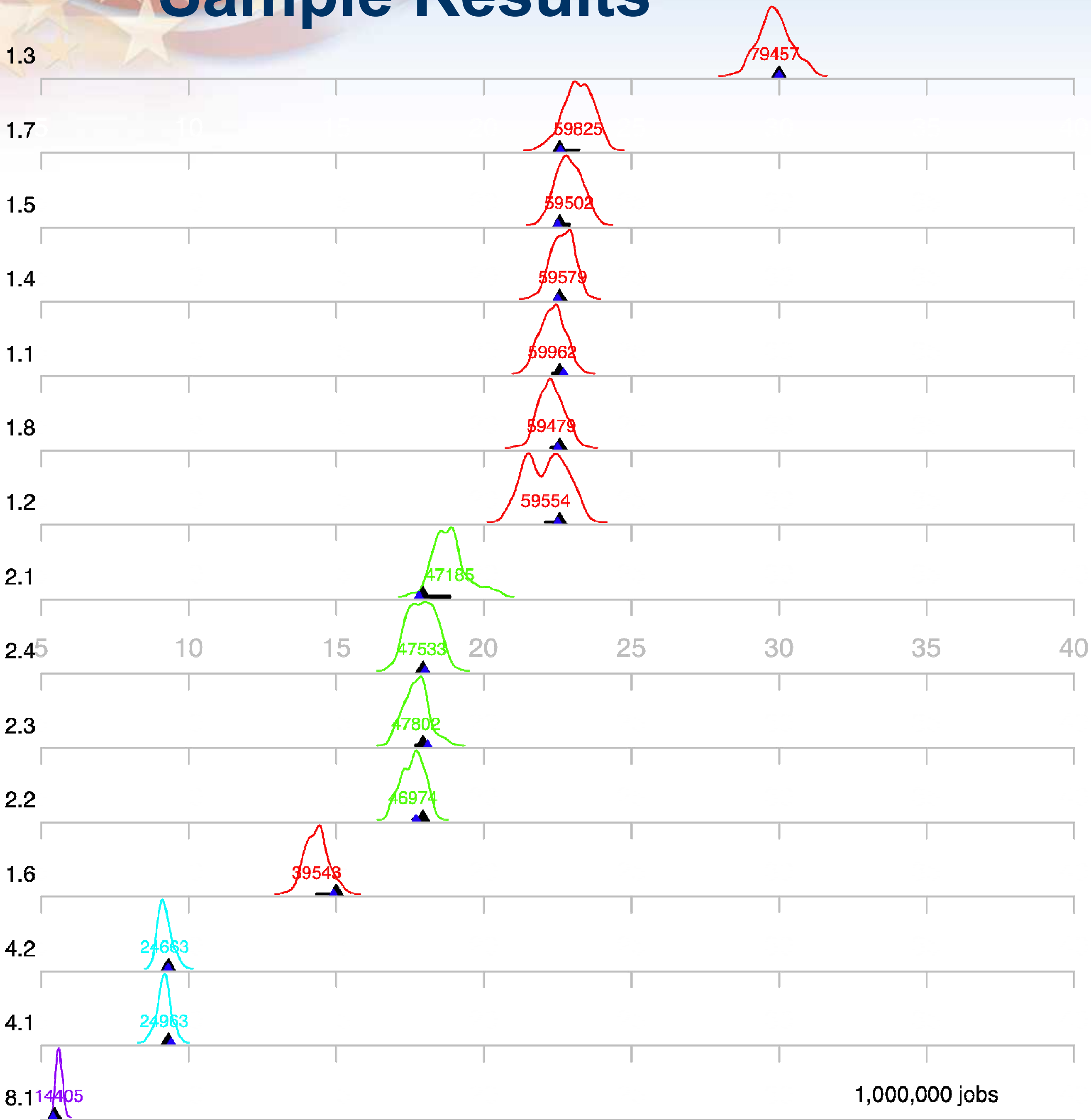
- X-axis is fault rate
- Y-axis is likelihood
- One row per component

- True source of failure is masked.
- But underlying failure rates can be inferred.

- Components are ranked by decreasing average fault rate

- More observations, less uncertainty

Sample Results



Fault rate PDFs:

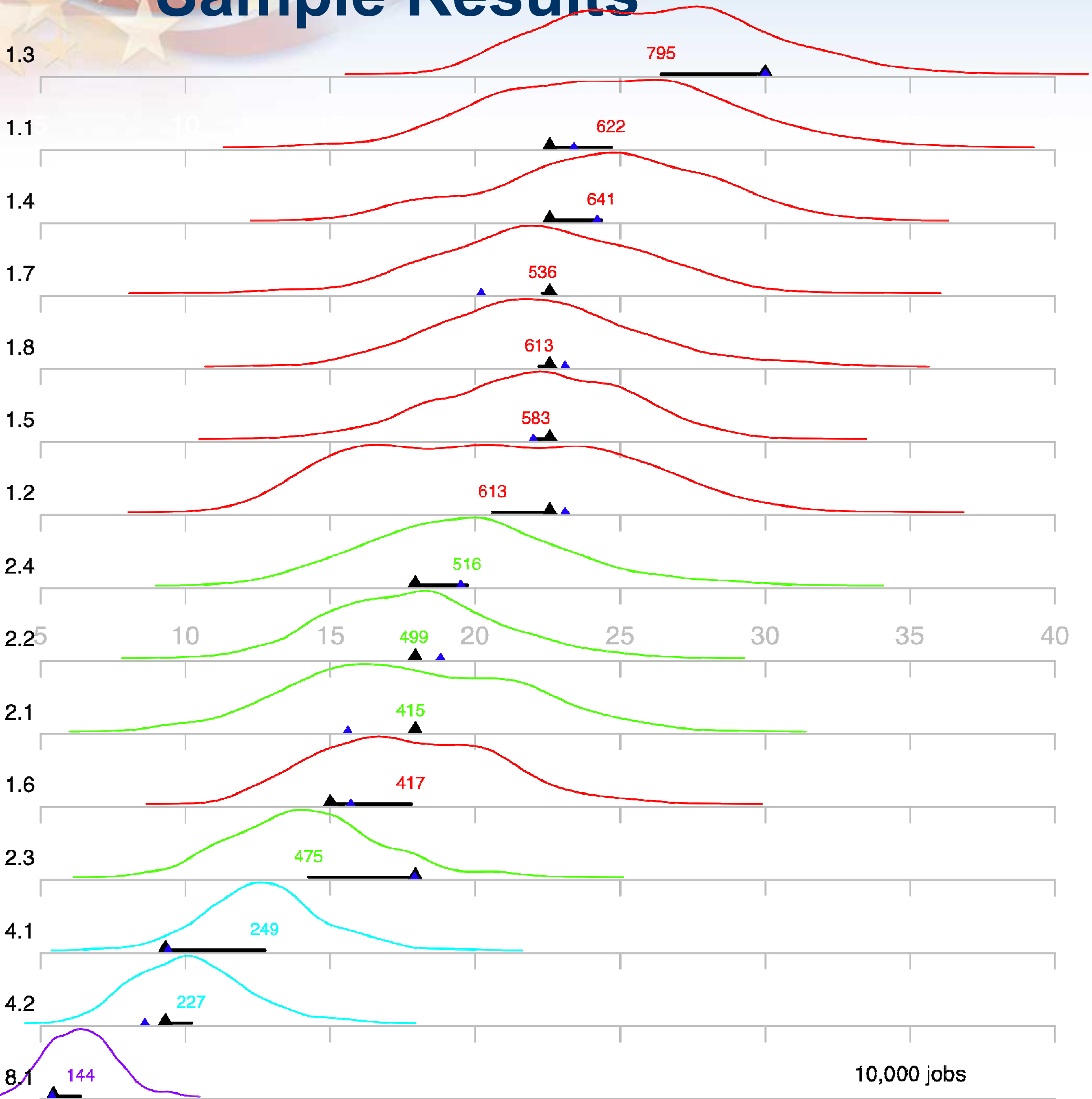
- X-axis is fault rate
- Y-axis is likelihood
- One row per component

- True source of failure is masked.
- But underlying failure rates can be inferred.

- Components are ranked by decreasing average fault rate

- More observations, less uncertainty

Sample Results



Sensitive to:

- Number of observations
- Job sizes and durations
- Job allocation algorithm
- Differences among failure rates
- Graph structure
- Priors used

Comments:

- Requires many failed jobs!
- More information would help!

Questions and Directions

How many jobs must fail before we can confidently intervene?

To what degree can additional information (e.g. system logs) be used to reduce root-cause uncertainty?

What do real system dependency graphs look like? (hardware and software, dynamic routing)

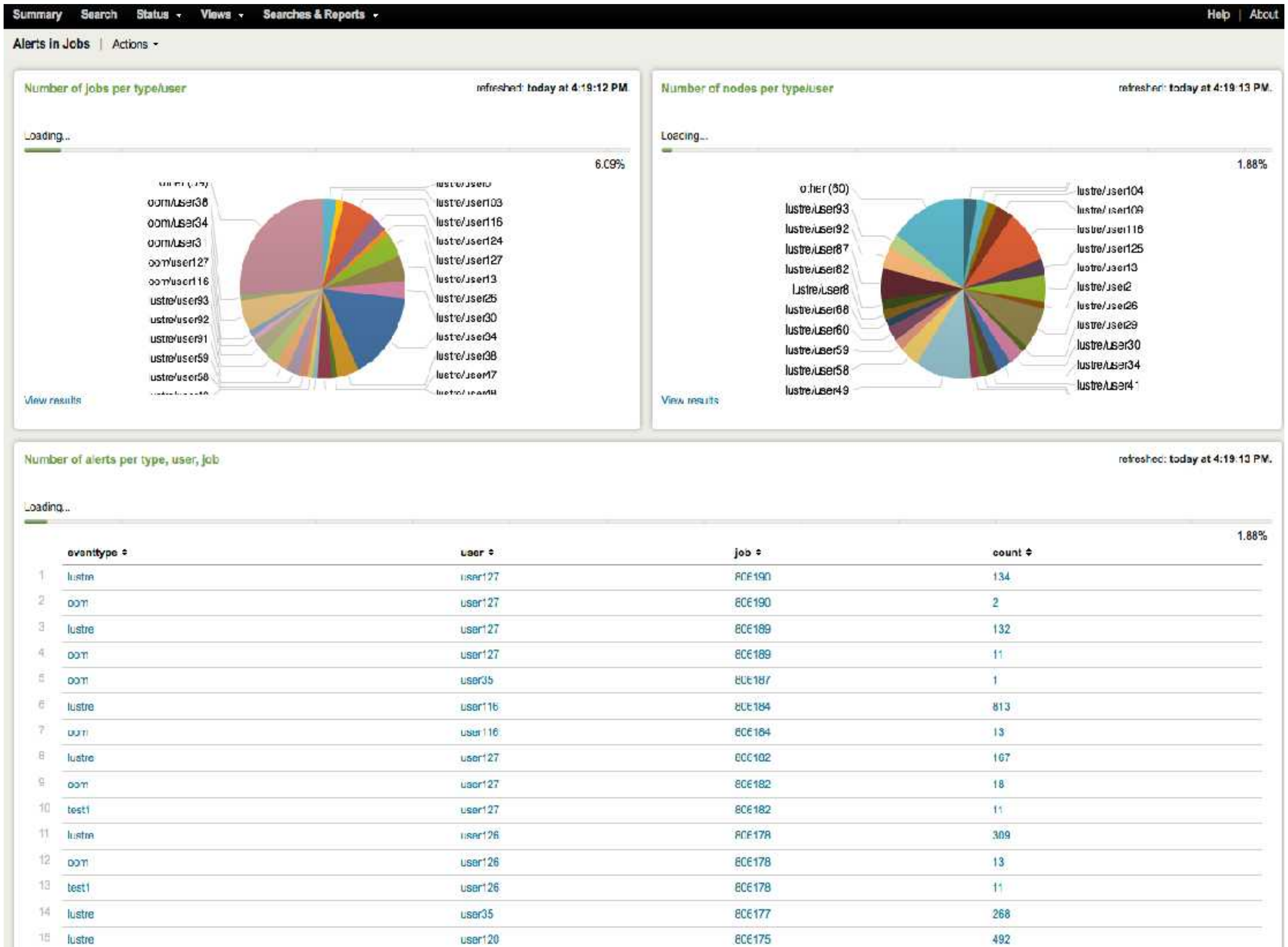
Could this be used during production operation? Influence allocator decisions, to accomplish fault-estimate-driven recursive bisection

Demonstrate it on a real system!

The End

(Extra slides follow)

Splunk Interface?



	eventtype	user	job	count
1	lustre	user127	8CE190	134
2	oom	user127	8CE190	2
3	lustre	user127	8CE189	132
4	oom	user127	8CE189	11
5	oom	user35	8CE187	1
6	lustre	user116	8CE184	813
7	oom	user116	8CE184	13
8	lustre	user127	8CE182	167
9	oom	user127	8CE182	18
10	test1	user127	8CE182	11
11	lustre	user126	8CE178	309
12	oom	user126	8CE178	13
13	test1	user126	8CE178	11
14	lustre	user35	8CE177	268
15	lustre	user120	8CE175	492

System Graph?

(Info to map job->components and component->jobs)

Jobs via scheduler logs (eg SLURM)

Configuration via Genders

And/or Cfengine and promise theory?

Network via routing tables

Configuration changes via git

…and/or others?

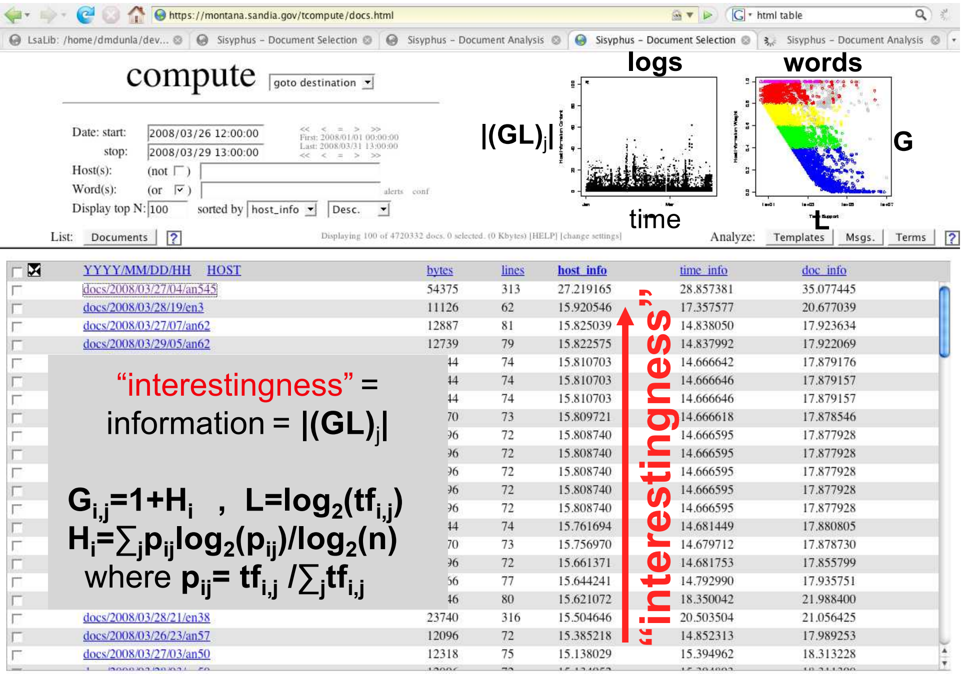
Sisyphus

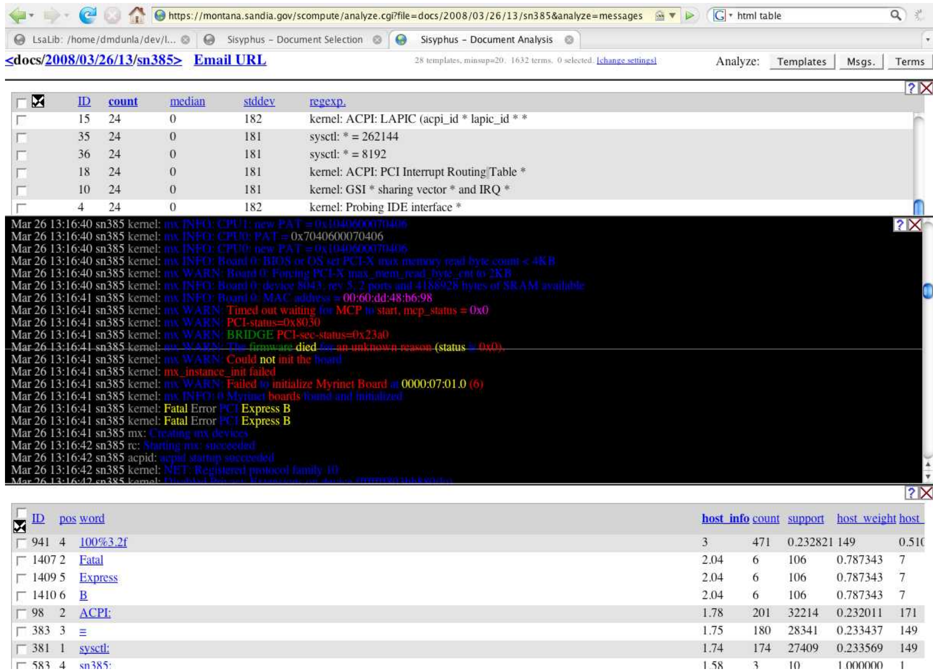
Automatic Fault Detection

Sandia National Laboratories

Jon Stearley

jrstear@sandia.gov



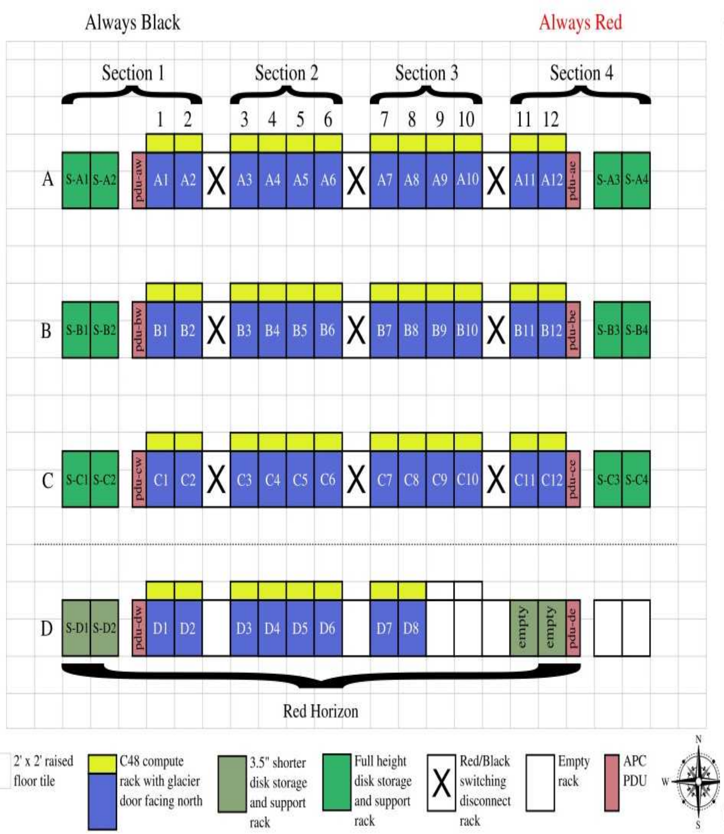


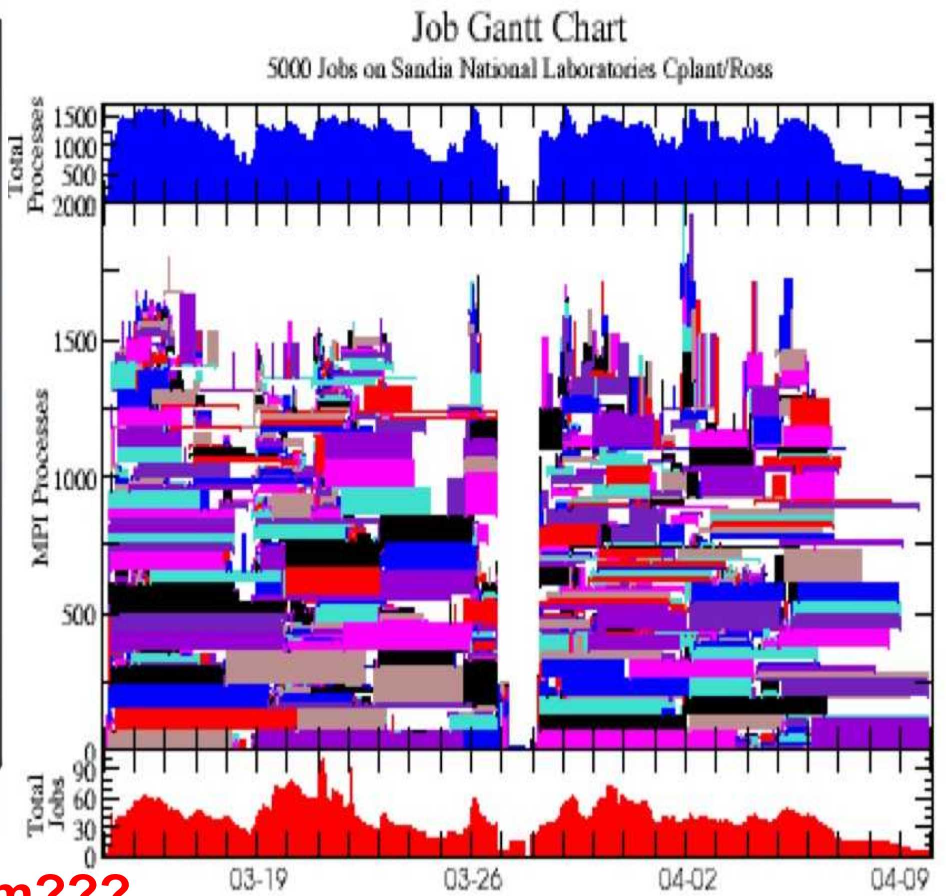
1. Automatically rank logs by information content.

2. Automatically color words by information weight.

3. Automatically deduce word and message patterns.

Similar computers correctly performing similar work should produce similar logs (anomalies warrant investigation).





What is wrong with this $#%&*! system???

```
Oct 17 05:04:06 nid00187 kern crit kernel: LDISKFS-fs error (device sde2) in ldiskfs_setattr: Readonly filesystem
Oct 17 05:04:12 nid00187 kern warning kernel: SCSI error : <1 0 0 0> return code = 0x20000
Oct 17 05:04:12 nid00187 kern warning kernel: end_request: I/O error, dev sde, sector 778694416
Oct 17 05:04:12 nid00187 kern err kernel: Buffer I/O error on device sde2, logical block 7372802
Oct 17 05:04:12 nid00187 kern warning kernel: lost page write due to I/O error on sde2
Oct 17 05:04:12 nid00187 kern warning kernel: SCSI error : <1 0 0 0> return code = 0x20000
Oct 17 05:04:12 nid00187 kern warning kernel: end_request: I/O error, dev sde, sector 779218704
Oct 17 05:04:12 nid00187 kern err kernel: Buffer I/O error on device sde2, logical block 7438338
Oct 17 05:04:12 nid00187 kern warning kernel: lost page write due to I/O error on sde2
Oct 17 05:04:20 nid00187 kern warning kernel: Lustre: 6388:0:(lustre_fsfilt.h:255:fsfilt_commit_wait()) slow journal start 51s
Oct 17 05:04:20 nid00187 kern err kernel: LustreError: 6388:0:(filter_io_26.c:707:filter_commitrw_write()) slow commitrw commit 3511s
Oct 17 05:04:20 nid00187 kern err kernel: LustreError: 6388:0:(filter_io_26.c:707:filter_commitrw_write()) previously skipped 5 similar messages
Oct 17 05:04:20 nid00187 kern err kernel: LustreError: 6388:0:(service.c:583:ptlrpc_server_handle_request()) request 527 opc 4 from U3-1251@ptl processed in 3511s trans 0 rc -5/-5
Oct 17 05:04:20 nid00187 kern err kernel: LustreError: 6388:0:(service.c:583:ptlrpc_server_handle_request()) previously skipped 7 similar messages
Oct 17 05:04:20 nid00187 kern warning kernel: Lustre: 6388:0:(watchdog.c:320:lcw_update_time()) Expired watchdog for pid 6388 disabled after 3511.0309s
Oct 17 05:04:20 nid00187 kern warning kernel: Lustre: 6339:0:(watchdog.c:320:lcw_update_time()) Expired watchdog for pid 6339 disabled after 3511.4820s
Oct 17 05:04:20 nid00187 kern warning kernel: Lustre: 6388:0:(watchdog.c:320:lcw_update_time()) previously skipped 7 similar messages
```

Likelihood Function

Formulate a distribution of machine events based on time and parameterized by failure rates of machine node groups.

$$L(\pi_1, \ldots, \pi_N) = \prod^{E} (\alpha_s \Delta t_s)^{m_s} e^{-\beta_s \Delta t_s}$$

$$\ell(\pi_1, \ldots, \pi_N) = \sum_{s=1}^{E} (m_s \ln(\alpha_s \Delta t_s) - \beta_s \Delta t_s)$$

Find π_i such that

$$\frac{\partial \ell}{\partial \pi_1} = \cdots = \frac{\partial \ell}{\partial \pi_N} = 0$$

Where

$$\alpha_s = \sum_{i \in \mathcal{D}_N} w_{i,s} \pi_i \text{ and } \beta_s = \sum_{i \in \mathcal{D}_N} u_{i,s} \pi_i \text{ where } \mathcal{D}_N = \{\text{Divisors}(N)\}$$

Maximum Likelihood Approach
(with Russell Hooper)

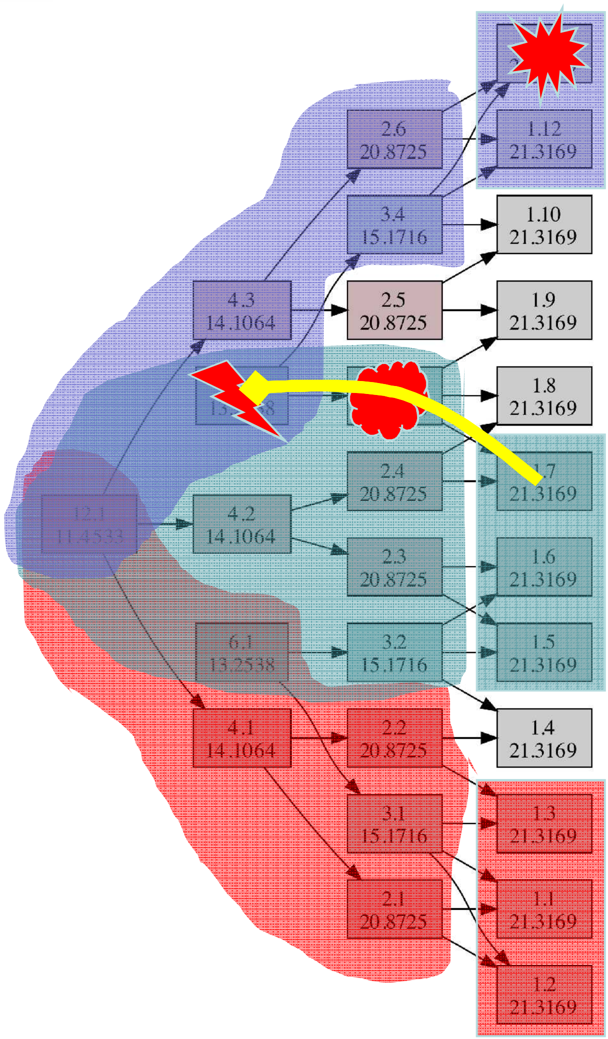
☑ Efficiency ☑ Accuracy ☒ Robustness/Automatic

- Treats optimization by solving a system of nonlinear equations
- Solves equations using Newton method via Trilinos
  - Efficient & accurate with “good” initial guess
  - Can struggle or fail with bad initial guess (failures are readily apparent, eg NaN)
- Strategies exist for obtaining good initial guesses but come at the cost of decreased efficiency
  - “Globalized” Newton – NOX
  - “Homotopy” - LOCA

Another Approach

Maximum Likelihood	**Conditioned Maximum Likelihood**
Exponential distribution (constant failure rate)	Arbitrary distribution
Failure rate is an unknown constant (explore uncertainty indirectly)	Distribution parameters are random variables (examine uncertainty directly)
Per-group failure rate (eg all nodes in a group have the same failure rate)	Per-node distribution parameters (eg each node has own failure rate)
Count of failures based	Time to failure based

Simulator Enhancement Ideas



1. **AND/OR dependency paths in the graph**

   represent redundancy etc
   (eg, both power supplies must fail before this cabinet of nodes are affected)

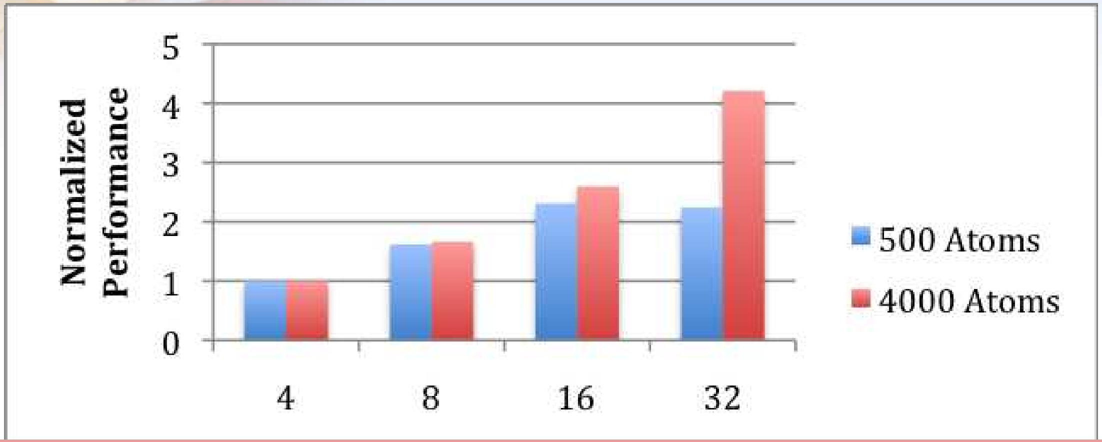
2. **Variety of events, observables, and latency**

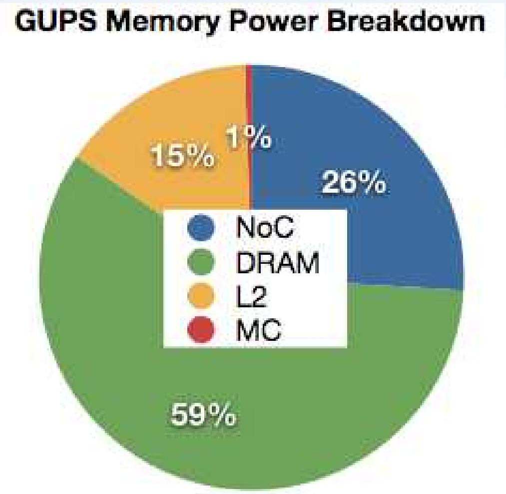
   event type E results in observables O
   (eg, logs or failures on connected nodes)

3. **Job factors affect the likelihood of events**

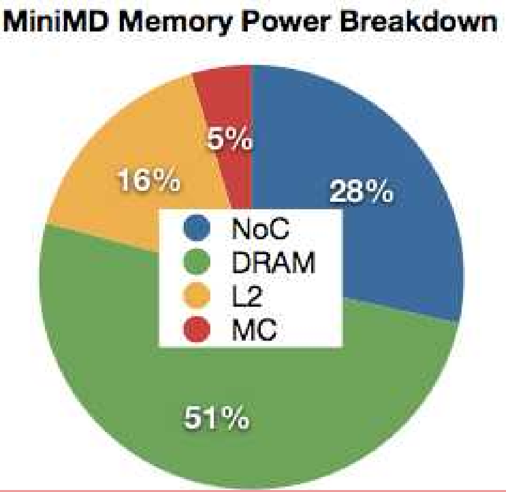
   application A with library L with input deck D causes event type E with observables O

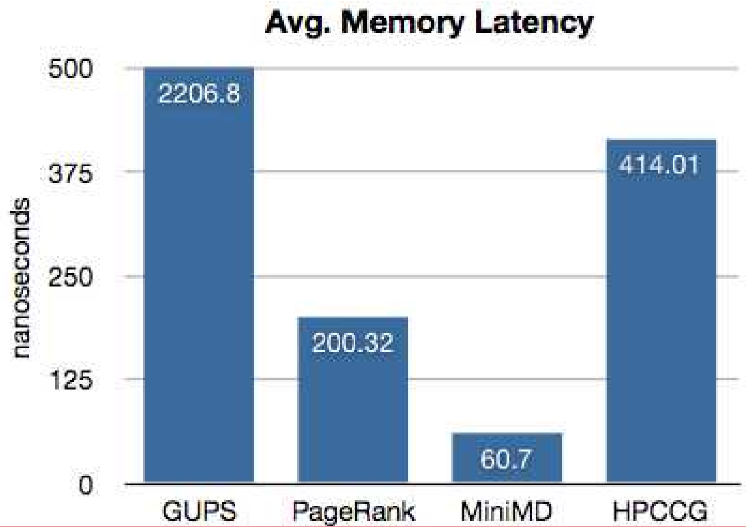
Sample SST Results & Uses





SST Simulation of MD code shows diminishing returns for threading on small data sets



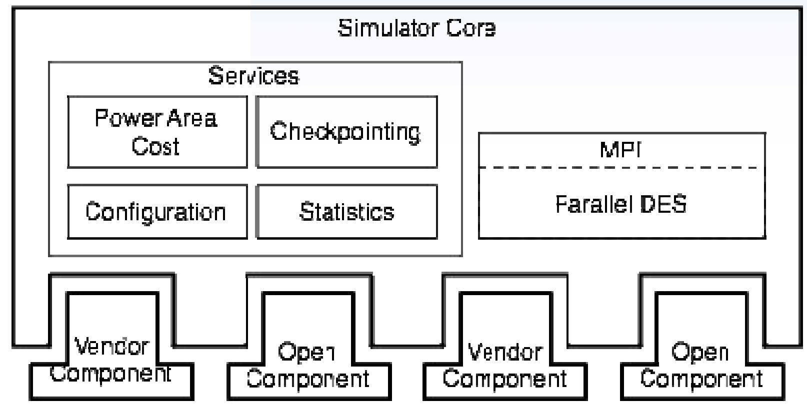


Detailed component simulation highlights bottlenecks

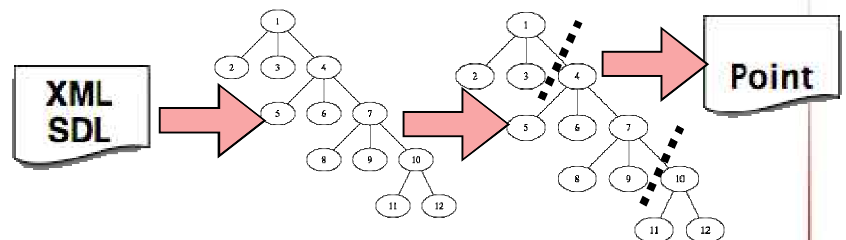
Power analysis help prioritize technology investments

Component Library

- Parallel Core v2
  - Parallel DES layered on MPI
  - Partitioning & Load Balancing
  - Configuration & Checkpointing
  - Power modeling
- Technology Models
  - McPAT, Sim-Panalyzer, IntSim, Orion and custom power/energy models
  - HotSpot Thermal model
  - Supercomputer resilience (YUMYUM)
- Components
  - Processor: Macro Applications, Macro Network, NMSU, genericProc, state-machine, Zesto, GeM5, GPGPU
  - Network: Red Storm, simpleRouter, GeM5
  - Memory: DRAMSim II, Adv. Memory, Flash, SSD, DiskSim



SST Simulator Core



SST Workflow

Sandia National Laboratories