

Introduction

Supercomputers are massive in both component count and complexity, leading to undesirable failure rates and difficulty in isolating root causes. It is not practically feasible to monitor all components, and although extensive system monitoring data exists (i.e. log files and physical sensors), users are frequently the first to be aware of problems based on the way their code is performing.

Current practice: rely on system data to identify problems
New idea: include job data to infer component fault rates

Fault rate estimates can then be used to automatically:

- ▶ maximize the reliability of high-priority jobs,
- ▶ refine confidence intervals regarding suspect components,
- ▶ remove components from service when appropriate,
- ▶ prioritize maintenance activities.

Notation

event - success or failure of job *j*
n - number of jobs in system
T_j - random length of the event associated with job *j*
s_j - set of components in job *j*
T_{ij} - random life of component *i* in job *j*, where *i* = 1, ... *m_j*
F_i, *f_i* - life distribution and density function for component *i*
R_i(t) = 1 - *F_i(t)*
θ_i - parameter vector for component *i*
K_j - true (unknown) cause of job *j* failure
M_j - a minimum random set of components that contains the cause of the failure of job *j*
m_j, *t_j* - realizations from *M_j* and *T_j* respectively for failed jobs

Problem Definition

Challenge: the true cause of job failure is masked.

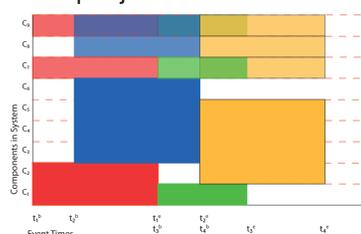
Given: table of job pass/fail observations, graph of connected components

Goal: estimate fault rate distributions of all components

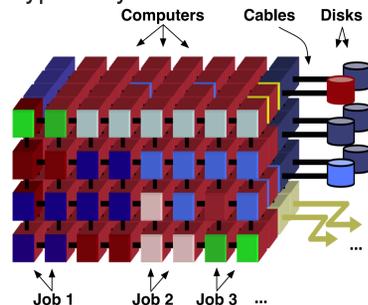
Table: Each row in the table indicates a user job; columns indicate start and stop time, pass or fail outcome, and set of computers used.

Graph: Nodes indicate components, including hardware or software (i.e. computers, cables, network switches, software libraries, etc). Edges indicate functional dependencies, including physical or logical (i.e. connections or configurations). This model is intentionally general to accommodate a wide variety of possible failure causes, and application to other complex systems.

Example jobs:



Typical system:



Likelihood Formulation

Our observations are two dimensional $(t_j, s_j), j = 1, \dots, n$, and the conditional likelihood function for the observed data is given by:

$$L(t|\theta) = \prod_{j=1}^n \left[\sum_{i \in s_j} \left(f_i(t_j) \prod_{j=1, j \neq i}^{|s_j|} R_j(t_j) P[M_j = m_j | T_j = t_j, K_j = i] \right) \right]$$

$P(M_j = m_j | T_j = t_j, K_j = i)$ is the conditional probability that the observed minimum random subset is m_j , given that the job j failed at time t_j and the true cause of failure was component i . $P(M_j = m_j | T_j = t_j, K_j = i) = 0$ if $i \notin s_j$.

The inner term $f_i(t_j) \prod_{j=1, j \neq i}^{|s_j|} R_j(t_j)$ describes the probability that job j fails at time t_j as a result of component i failing.

Intuitively, this is the probability of one component in the minimum random subset failing and the other components surviving through t_j .

Time to Failure Assumptions

Assume that the time to failure for each of the components is characterized by a unique exponential distribution function:

$$f(t|\lambda) = \lambda \exp(-\lambda t) \text{ for } t > 0, \lambda > 0,$$

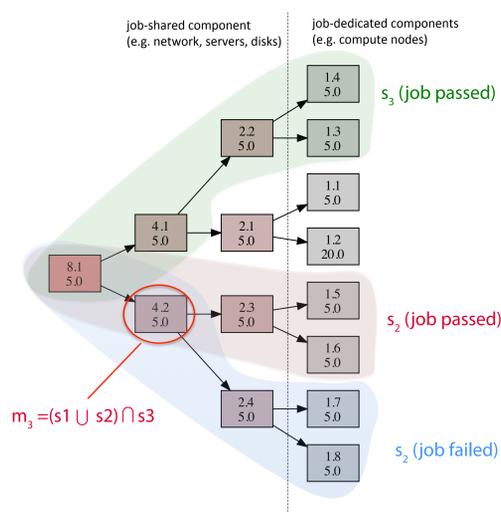
and a prior distribution on λ_i can be approximated by a Gamma distribution:

$$\pi(\lambda_i | a_i, b_i) = \frac{b_i^{a_i}}{\Gamma(a_i)} \lambda_i^{(a_i-1)} \exp[-b_i \lambda_i]$$

Auxiliary Variables

Given the complex conditional likelihood function, it will be necessary to introduce a set of auxiliary variables. For $1 \leq j \leq n$ and $1 \leq i \leq m_j$, let $l_{ij} = I(t_j = T_{ij})$ be an indicator variable, where $I(\cdot) = 1$ when a job j fails as a result of a failure of component i . Define $\mathbf{l}_j = (l_{1j}, \dots, l_{m_j j})$, $\mathbf{l} = (\mathbf{l}_1, \dots, \mathbf{l}_n)$, and denote $\mathbf{l}_{(-j)} = \{\mathbf{l}_k : 1 \leq k \leq n, k \neq j\}$.

Simulation of Operational Data



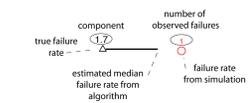
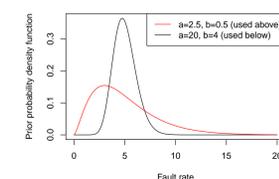
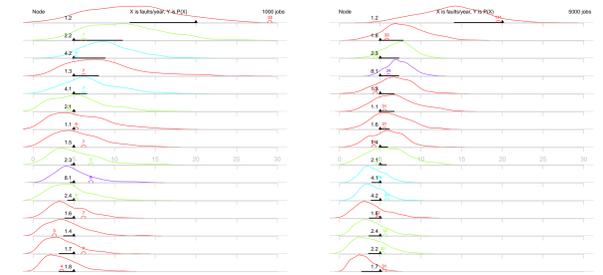
Simulated Data

- ▶ failure density was assumed exponential for all nodes
- ▶ For all nodes $\lambda_{x,x} = 5$ failures per operating year, except: $\lambda_{1,2} = 20$.

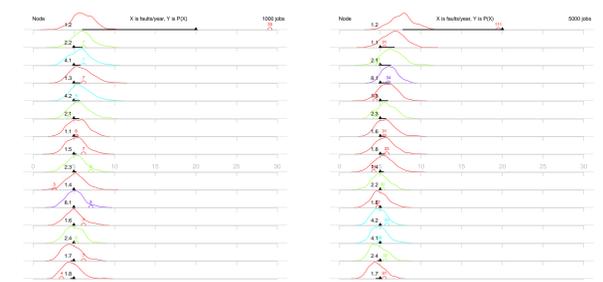
Experimental Results

Parameter Estimation

The complexity of the likelihood function requires the use of a Markov Chain Monte Carlo solution methodology.



Even with unrealistically vague prior distributions, as jobs begin to accumulate, the components with the high failure rates begin to emerge.



As expected, too heavy an emphasis on the prior inhibits detection of underlying failure mechanism.

Conclusion

Algorithm provides a risk-based identification of failed components (or software) within an HPC system where the true source of failure is masked.

MCMC implementation of the algorithm is scalable to large, complex hardware and software architectures.

Number of jobs required to unmask the true culprit is unacceptably high, but this is sensitive to the prior and a history weighted prior will be explored.

The current results are biased toward a high number of jobs as a result of simulation parameters related to:

- ▶ job node assignment in simulation
- ▶ job length distribution

Future Efforts

- ▶ Assess algorithm using HPC operational data
- ▶ Incorporate unstructured data from system log as additional information to decrease uncertainty in fault identification and reduce time required to unmask faulty components.
- ▶ Include data related to the operational environment, e.g. temperature, duty cycles
- ▶ establish capability for adaptive prior