SHERLOCK: TOWARDS HIGHLY RELIABLE ENTERPRISE NETWORK SERVICES VIA INference OF MULTI-LEVEL DEPENDENCIES

Presented by Jim Spadaro and J. Scott Miller
Outline

- Scalability
- False negatives
- False positives
- Lack of prevention
- Unnecessary complexity
Scalability

- Ferret algorithm exponential with number of simultaneous root cause failures
  - Will scale much more poorly with heavily distributed applications
- Evidence seems to scale linearly, but it is not consistent with the theoretical performance
  - $k = 1$?
False Negatives

- Misses elegant software failures
  - Graceful software failures desirable
  - Modifying system to handle these (inspecting each packet’s application-layer data) is expensive

- Dependent/hidden services
  - Clients cannot be instrumented to detect failures when one root-cause server depends on another
  - Requires explicit configuration
False Positives

- Long running processes
  - Maintenance tasks (e.g. backups, large updates)
- Adds noise to data
  - See Figure
  - 1,029 problem notifications in 5 days
  - Less than current configuration, but still unmanageable
Lack of prevention

- Sherlock responds to observed conditions
  - Clients measure response times, Sherlock reasons about causes
  - Point of failure determined after running Ferret
  - Delay of minutes or hours
- Properly tuned existing solutions (look at server attributes) can handle this
Unnecessary Complexity

- Given a lack of failure prediction, overhead and notification delays, why deploy?
- Current monitoring solutions can handle failure states well if properly tuned
- This and existing solutions require network administration to prevent data flood