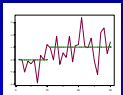


When bad things happen to good systems: detecting and diagnosing problems



Kimberly Keeton
HPL Storage and Content Distribution

Berkeley/Stanford Recovery-oriented Computing Course Lecture
October 18, 2001

2001-10-ROCC-Lecture

Problem definition

- ▼ **Detection:** determining that a problem has (will) occur(red)
- ▼ **Diagnosis:** determining the root cause of the problem
- ▼ **“Problem” can be broadly defined**
 - Performance-related, availability-related, security-related
- ▼ **Fields to draw from:**
 - System administration, operating systems, network management, intrusion detection
- ▼ **Techniques borrowed from:**
 - Statistics, database data mining, AI machine learning

2001-10-ROCC-Lecture, 1
Storage & Content Distribution

Outline

- ▼ Problem definition
- ▼ **Detection techniques**
 - Challenges
 - Change point detection
 - Time series analysis
 - Predictive detection
 - Data mining/machine learning algorithms
- ▼ **Diagnosis techniques**
- ▼ **Additional related work**
- ▼ **Summary**

2001-10-ROCC-Lecture, 2
Storage & Content Distribution

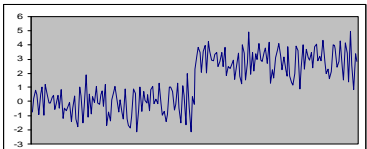
Challenges in detecting problems

- ▼ **Many types of faults**
 - Persistent increase, gradual change, abrupt change, single spike
- ▼ **Time-varying property of observed system behavior**
 - Trends and seasonality (i.e., cyclic behavior)
- ▼ **Distinguishing between the “good,” the “bad” and the “ugly”**
- ▼ **Detecting problems fast enough to minimize service disruption**
- ▼ **Catching false positives vs. neglecting true positives**

2001-10-ROCC-Lecture, 3
Storage & Content Distribution

Change point detection algorithms [Hellerstein98]

- ▼ **Basic idea:**
 - Determine when process parameters have changed
 - Declare change point if I/O response time is “more likely” to have come from a distribution with a different mean



- ▼ **Ex:** maximum likelihood ratio detection rules, such as cumulative sum (CUMSUM)

2001-10-ROCC-Lecture, 4
Storage & Content Distribution

Maximum likelihood ratio

- ▼ Let Y_1, Y_2, \dots, Y_T be i.i.d. random variables
- ▼ Let $f(Y_i, q)$ be the probability distribution function (pdf) of the random variables, where q is the only parameter in the pdf
- ▼ Let $f(q_0)$ and $f(q_1)$ be different distributions
- ▼ **Likelihood ratio:**

$$\frac{\prod_{i=1}^T f(Y_i, q_1)}{\prod_{i=1}^T f(Y_i, q_0)}$$
- ▼ **Large ratio => more likely Y_1, Y_2, \dots, Y_T from $f(q_1)$**

2001-10-ROCC-Lecture, 5
Storage & Content Distribution

Maximum likelihood ratio detection rule

▼ Declare a change has occurred at N if the likelihood ratio after the change exceeds a pre-determined threshold level c

$$N = \inf \left\{ n \geq 1 : \sup_{1 \leq k \leq n} \frac{\prod_{i=k}^n f(Y_i, q_1)}{\prod_{i=k}^n f(Y_i, q_0)} \geq c \right\}$$

▼ Ex: CUMSUM rule for normal random variables

$$N = \inf \left\{ n \geq 1 : \max_{1 \leq k \leq n} \sum_{i=k}^n (Y_i - \bar{Y}) \geq c \right\}$$

2001-10-BOC-Lecture 6
Storage & Content Distribution

CUMSUM example

- Raw data: difficult to detect change
- CUMSUM: easier to detect change
- CUMSUM confidence level

▼ Confidence level compared with bootstrapping (random permutation of data)

- Bootstrap: flat cumulative residuals
- CUMSUM: angle forms at change point

2001-10-BOC-Lecture 7
Storage & Content Distribution

Change point pros/cons

▼ Advantages:

- Well-established statistical technique
- Several variants of on-line and off-line algorithms

▼ Disadvantages:

- Focuses on single type of fault – abrupt changes
- Mostly limited to stationary (non-varying over time) processes
 - Must separately deal with long-term trends and seasonality
- Some dependence on knowledge of and assumptions of data distributions

2001-10-BOC-Lecture 8
Storage & Content Distribution

Outline

- ▼ Problem definition
- ▼ Detection techniques
 - Challenges
 - Change point detection
 - Time series analysis
 - Predictive detection
 - Data mining/machine learning algorithms
- ▼ Diagnosis techniques
- ▼ Additional related work
- ▼ Summary

2001-10-BOC-Lecture 9
Storage & Content Distribution

Time series forecasting algorithms

▼ Basic idea:

- Build model of what you expect next observation to be, and raise alarm if observed and predicted values differ too much

▼ Ex: Holt-Winters forecasting [Hoogenboom93, Brutlag00]

- 3-part model built on exponential smoothing:
- prediction = baseline + linear trend + seasonal effect
 - $y'_{t+1} = a_t + b_t + c_{t-1-m}$
 - baseline: $a_t = \alpha(y_t - c_{t-m}) + (1 - \alpha)(a_{t-1} + b_{t-1})$
 - linear trend: $b_t = \beta(a_t - a_{t-1}) + (1 - \beta)(b_{t-1})$
 - seasonal trend: $c_t = \gamma(y_t - a_t) + (1 - \gamma)(c_{t-m})$
 - where m is period of seasonal cycle

2001-10-BOC-Lecture 10
Storage & Content Distribution

Holt-Winters measure of deviation

▼ Confidence bands to measure deviation in seasonal cycle:

- predicted deviation: $d_t = \gamma|y_t - y'_t| + (1 - \gamma)(d_{t-m})$
- confidence band: $(y'_t - \delta \cdot d_{t-m}, y'_t + \delta \cdot d_{t-m})$

▼ Trigger alarm when number of violations exceeds threshold

- To reduce false alarm rate, measure across moving, fixed-sized window

2001-10-BOC-Lecture 11
Storage & Content Distribution

Holt-Winters example

- ▼ Simplified Holt-Winters: exponential smoothing
- ▼ Generally detects 10-minute changes
 - Violations occur when observation falls outside of lower and upper bounds

2001-10-BOC-Lecture 12
Storage & Content Distribution

Time series forecasting pros/cons

- ▼ Advantages:
 - Well-established statistical technique
 - Considers time-varying properties of data
 - Trends and seasonality (at many levels)
- ▼ Disadvantages:
 - Large number of parameters to tune for algorithm to work correctly
 - Detection of problem after it occurs may imply service disruption

2001-10-BOC-Lecture 13
Storage & Content Distribution

Outline

- ▼ Problem definition
- ▼ Detection techniques
 - Challenges
 - Change point detection
 - Time series analysis
 - Predictive detection
 - Data mining/machine learning algorithms
- ▼ Diagnosis techniques
- ▼ Additional related work
- ▼ Summary

2001-10-BOC-Lecture 14
Storage & Content Distribution

Predictive detection [Hellerstein00]

- ▼ Basic idea:
 - Predict probability of violations of threshold tests in advance, including how long until violation
 - Allows pre-emptive corrective action in advance of service disruption
 - Also allows service providers to give customers advanced notice of potential service degradations

2001-10-BOC-Lecture 15
Storage & Content Distribution

Predictive detection highlights

- ▼ Model both stationary and nonstationary effects
 - Stationary: multi-part model using ANOVA techniques
 - Non-stationary: use auto-correlation and auto-regression to capture short-range dependencies
- ▼ Use observed data and models to predict future transformed values for a prediction horizon
- ▼ Calculate the probability that threshold is violated at each point in the prediction horizon
- ▼ May consider both upper and lower thresholds

2001-10-BOC-Lecture 16
Storage & Content Distribution

Predictive detection example


- ▼ Transform data and thresholds
 - Measured (time-varying) values are transformed into (stationary) values
 - Constant raw threshold also transformed into (time-varying) thresholds
- ▼ Predict future values and probability of threshold violation

2001-10-BOC-Lecture 17
Storage & Content Distribution

Outline

- ▼ Problem definition
- ▼ Detection techniques
 - Challenges
 - Change point detection
 - Time series analysis
 - Predictive detection
 - Data mining/machine learning algorithms
- ▼ Diagnosis techniques
- ▼ Additional related work
- ▼ Summary


2001-10-BOC-Lecture 18
Storage & Content Distribution



Data mining/machine learning algorithms[Lee98]

- ▼ Basic idea:
 - Context: intrusion detection
 - Use data mining techniques to discover patterns describing program and user behavior
 - Compute classifiers (rule sets) that can recognize anomalies
- ▼ Types of algorithms:
 - Classification: map a data item into one of several pre-defined categories
 - Link analysis: determine relations between fields in a database
 - Ex: association rules algorithm
 - Sequence analysis: model sequential (time-based) patterns
 - Ex: frequent episodes algorithm


2001-10-BOC-Lecture 19
Storage & Content Distribution



Classification algorithms

- ▼ Goal: use machine learning to classify “normal” and “abnormal” behavior, and to detect anomalies
 - Ex: system call sequences for sendmail
- ▼ Learning:
 - Training input: pre-labeled “normal” and “abnormal” data
 - Compute “identity” of program by developing rules for normal behavior
 - Output: a set of if-then rules for the “normal” classes, and a default “true” rule for the remaining classes
- ▼ Detection:
 - Raise alarm if percentage of abnormal regions above a threshold
 - May also combine classifiers to do meta-detection
- ▼ Question: how to devise rule sets?
 - Idea: association rules and frequent sets


2001-10-BOC-Lecture 20
Storage & Content Distribution



Association rules algorithm [Srikant95]

- ▼ Used to derive multi-feature (attribute) correlations from database table (or audit trail)
 - Ex: determining what items are often purchased together by customers
- ▼ Motivation:
 - Evidence that program executions and user activities exhibit frequent correlations among system features
 - Consistent behaviors can be captured in association rules
 - New rules can be continuously merged in
- ▼ Format: $X \rightarrow Y$, confidence, support
 - X and Y are subsets of items in a record
 - Support: percentage of records that contain X + Y
 - Confidence: $\text{support}(X+Y)/\text{support}(X)$
- ▼ Command history ex: `trn -> rec.humor; [0.3, 0.1]`


2001-10-BOC-Lecture 21
Storage & Content Distribution



Frequent episodes algorithm [Srikant96]

- ▼ Used to identify a set of events that occur (together) frequently within a time window
 - Serial episode: events must occur in partial order in time
 - Parallel episode: no such ordering constraint
- ▼ Motivation:
 - Evidence that sequence info in program executions and user commands can be used to build profiles for anomaly detection
- ▼ Format: $X \rightarrow Y$, confidence, support, time window
 - X and Y are frequent episodes
 - Support: $\text{frequency}(X + Y)$
 - Confidence: $\text{frequency}(X+Y)/\text{frequency}(X)$
- ▼ Web site ex: home, research -> theory; [0.2, 0.05], [30s]


2001-10-BOC-Lecture 22
Storage & Content Distribution



Overall system design

- ▼ Learning agents to build and maintain the (evolving) rule set:
 - For each new run:
 - Compare rule set from new run against aggregate set
 - If match found, increment match count of matched rule
 - Otherwise, add rule and set match count to 1
 - When rule set stabilizes, prune rule set by eliminating rules with low match count
- ▼ Detection agents
 - Discovered patterns from audit data can be used for anomaly detection


2001-10-BOC-Lecture 23
Storage & Content Distribution



Data mining pros/cons

- ▼ **Advantages:**
 - Training data can be accumulated over time
 - Evidence that normal behavior exhibits correlations
- ▼ **Disadvantages:**
 - Need a large amount of training data to compute rule sets
 - Assumption that training data is nearly "complete" wrt all possible normal behavior, since algorithms only detect patterns present in training data
 - Hard to determine right set of features to include in audit trail
 - May require lots of data pre-processing if incomplete set of features


2001-10-BOC-Lecture 24
Storage & Content Distribution



Outline

- ▼ Problem definition
- ▼ Detection techniques
- ▼ Diagnosis techniques
 - Challenges
 - Dependency models
 - Active dependency discovery
- ▼ Additional related work
- ▼ Summary


2001-10-BOC-Lecture 25
Storage & Content Distribution



Challenges for problem diagnosis

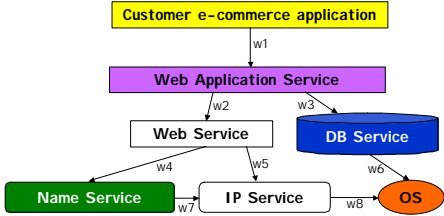
- ▼ Mapping end-user or SLA-level symptoms to deeper root causes
- ▼ Dealing with system complexity to pinpoint problem source
- ▼ Capturing different types of dependencies and their strengths
 - Static
 - Runtime
 - Distributed
- ▼ Capturing dependencies at detailed level of system resources
- ▼ Capturing dependencies that are relevant for a particular workload

2001-10-BOC-Lecture 26
Storage & Content Distribution




Dependency models in a nutshell

- ▼ Use a graph (DAG) structure to capture dependencies between system components
 - if failure of A affects B, then B depends on A
 - edge weights represent dependency strengths



Slide from [Brown01]


2001-10-BOC-Lecture 27
Storage & Content Distribution



Dependency modeling uses

- ▼ Event correlation systems [Yemini96, Choi99, Gruschke98]
 - Incoming events and alarms mapped onto nodes of dependency graph corresponding to origins of event
 - Graph processed to identify nodes on which most alarm/event nodes depend
 - Likely root causes of observed alarm/event
 - Repeat process until likely single root cause selected
- ▼ Model graph as map for systematic examination [Katker97]
 - Incoming problem mapped onto node of dependency graph
 - "Horizontal" search to test each component in path to identify any that are faulty
 - For each faulty node, "vertical" search to recursively apply search to examine nodes on which faulty node depends
 - Repeat process until root cause node (one not dependent on other faulty nodes) is identified


2001-10-BOC-Lecture 28
Storage & Content Distribution



Dependency model pros/cons

- ▼ **Advantages**
 - Do not require a priori existence of detailed knowledge bases
 - Advantages over rule-, code- and case-based root cause analysis
- ▼ **Disadvantages**
 - Most systems don't discuss details of how required dependency models are obtained
 - Static dependency models inadequate
 - Don't capture dynamic dependencies
 - Capture all potential dependencies, resulting in overwhelming state space to search


2001-10-BOC-Lecture 29
Storage & Content Distribution



Outline

- ▼ Problem definition
- ▼ Detection techniques
- ▼ Diagnosis techniques
 - Challenges
 - Dependency models
 - Active dependency discovery
- ▼ Additional related work
- ▼ Summary


2001-10-B0C-Lecture 30
Storage & Content Distribution



Active dependency discovery (ADD) [Brown01]

- ▼ Basic idea:
 - Instrument the system and apply workload
 - Systematically perturb components
 - Measure system change in response
 - Construct dependency model from observed data
- ▼ Dependency model
 - Table with rows defining resources, columns describing requests
 - Value in cell corresponds to dependency strength
 - Strengths computed as slope of linear regression on mean of log of observed data
 - Statistically positive slope gives dependency strength


2001-10-B0C-Lecture 31
Storage & Content Distribution



ADD diagnosis

- ▼ When a problem occurs, diagnose using dependencies:
 - Identify faulty request
 - from problem report, SLA violation, test requests, ...
 - Select the appropriate column in dependency table
 - Select the rows representing dependencies (potential root causes)
 - Investigate potential root causes, starting with those of highest weight


2001-10-B0C-Lecture 32
Storage & Content Distribution



ADD pros/cons

- ▼ Advantages:
 - Can guarantee coverage by explicitly controlling perturbation
 - Causality is easy to establish: perturbation is the cause
 - Simplicity
 - No application modeling or modification necessary
 - Existing endpoint instrumentation may be sufficient
 - No complex data mining required
- ▼ Disadvantages:
 - Invasive nature may make perturbation on production systems difficult
 - Leverage redundancy if available (e.g., cluster system)
 - Run perturbation during non-production periods (initial system setup or during scheduled downtime)
 - Develop low-grade perturbation techniques
 - Extracted models only valid for applied workload


2001-10-B0C-Lecture 33
Storage & Content Distribution



Outline

- ▼ Problem definition
- ▼ Detection techniques
- ▼ Diagnosis techniques
- ▼ Additional related work
 - Presentation of distributed information
 - Introspective systems
 - Self-managing systems
 - Measurement studies of system availability
 - Convergent systems/computer immunology
- ▼ Summary


2001-10-B0C-Lecture 34
Storage & Content Distribution



Presentation of distributed information

- ▼ Sometimes solution to diagnosis is to consult human expert
- ▼ Body of work in how to present information to human expert
- ▼ Ex: CARD system for monitoring large clusters [Anderson97]
 - Hierarchy of databases for collecting monitoring data, using hybrid push-pull model for communication of data
 - Aggregation of information
 - Combine same statistics across different nodes (e.g., avg, stdev of CPU utilization across machines)
 - Aggregate across statistics (e.g., combine CPU, disk and network utilization to get overall machine utilization)
 - Visualization techniques
 - Averages visualized as bar height
 - Standard deviations as bar shade
 - Different colors for different characteristics, and to indicate up/down

2001-10-B0C-Lecture 35
Storage & Content Distribution



Introspective systems

- ▼ **ISTORE [Brown99]**
 - Internal monitoring using sensors
 - Software triggers (predicates over system state) evaluation signals potential problems
 - Adaptation code gets invoked to deal with anomalies
- ▼ **OceanStore [Kubiatowicz00]**
 - Internal event monitoring and analysis of usage patterns, network activity, and resource availability.
 - Info used to:
 - Adapt to regional outages and denial of service attacks
 - Pro-actively migrate data towards areas of use
 - Maintain sufficiently high levels of data redundancy

2001-10-B0C-Lecture_36
Storage & Content Distribution



Self-managing systems

- ▼ **IBM's eLiza autonomous system**
 - Goal: "systems that can configure, optimize, heal and protect themselves, while the user focuses on the more significant things"
 - Includes Oceano
- ▼ **HPL's self-managing storage system [Anderson02]**
 - Automation of initial storage system configuration and workload evolution management

2001-10-B0C-Lecture_37
Storage & Content Distribution



Measurement studies of system availability

- ▼ **[Long95]**
 - Fault-tolerant tool to directly measure TTF, TTR and availability by polling many sites frequently from several locations
- ▼ **[Iyer99]**
 - Availability analysis of a LAN of Unix-based workstations, LAN of Windows NT-based machines and the Internet

2001-10-B0C-Lecture_38
Storage & Content Distribution



Convergent systems/computer immunology

- ▼ **Biological view of autonomous systems [Burgess98]**
- ▼ **Specify "healthy" state of system (e.g., pseudo-invariants)**
- ▼ **When problems arise, don't necessarily try to distinguish between their symptoms and their cause**
 - Shimon Peres analogy: some "illnesses" can't be cured, only their symptoms treated
- ▼ **Use rules to move the systems closer (converging) to healthy**
 - Treat the symptoms
- ▼ **Ex:**
 - Invariant: database transactions shouldn't experience deadlock
 - If deadlock detected, shoot down player(s) and restart
- ▼ **Related to design for restartability**

2001-10-B0C-Lecture_39
Storage & Content Distribution



Summary

- ▼ **Detection techniques**
 - Change point detection
 - Time series analysis
 - Predictive detection
 - Data mining/machine learning algorithms
- ▼ **Diagnosis techniques**
 - Dependency models
 - Active dependency detection
- ▼ **Additional related work**
 - Presentation of distributed information
 - Introspective and self-managing systems
 - Measurement-based studies of system availability
 - Convergent systems/computer immunology

2001-10-B0C-Lecture_40
Storage & Content Distribution



Discussion issues

- ▼ **How to monitor?**
 - Internal vs. external to system
 - Choosing which and how many metrics to monitor
- ▼ **Distinguishing good and bad behavior**
- ▼ **Active vs. passive techniques**
 - What (performance, availability) fault load to use?
- ▼ **Automating root-cause analysis using dependency models**
- ▼ **Methods for tagging requests as they travel throughout the system**
- ▼ **Design for restartability in context of convergent systems**
- ▼ **Methods for automatically detecting human response to failure**

2001-10-B0C-Lecture_41
Storage & Content Distribution



Acknowledgments

- ▼ Aaron Brown, Berkeley
- ▼ Jerry Shan, HPL Decision Technologies Department
- ▼ Angela Hung, Stanford