

Proposal Defense Presentation

Topic: Event Mining for System and Service Management

Presented by
Liang Tang

School of Computing and Information Sciences
Florida International University

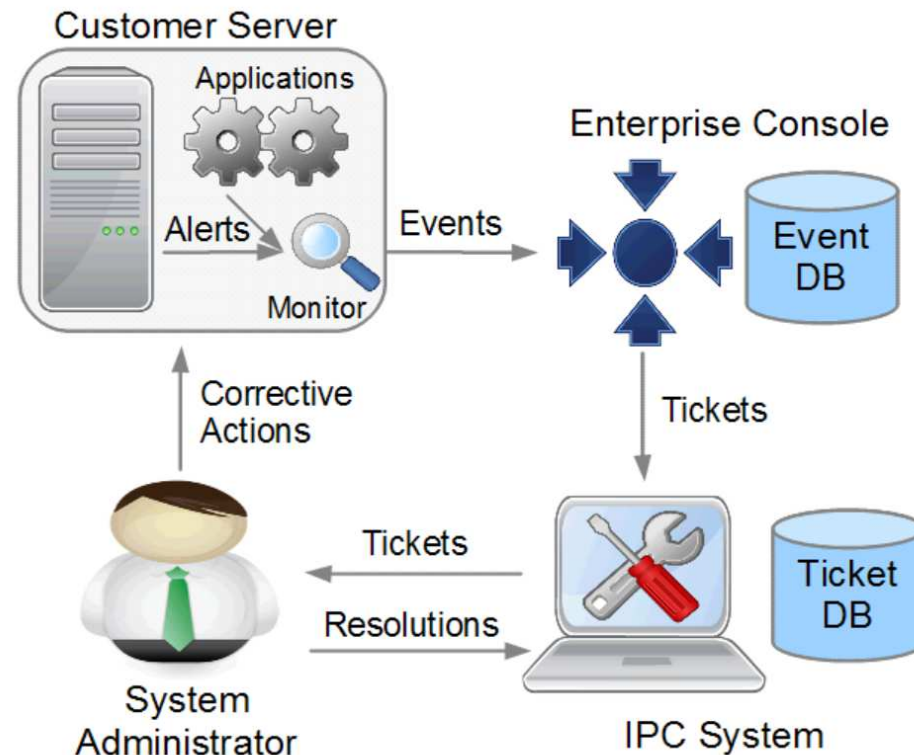
Outline

- Background and Overview
- Research Problems
 - Converting Textual Log to System Events
 - Monitoring Configuration Optimization
 - False Positive
 - False Negative
 - Analysis on Detected System Issues
 - Temporal Dependencies of Events
 - Incident Resolution Recommendation
 - Textual Event Segments Search
- Summary and Timeline

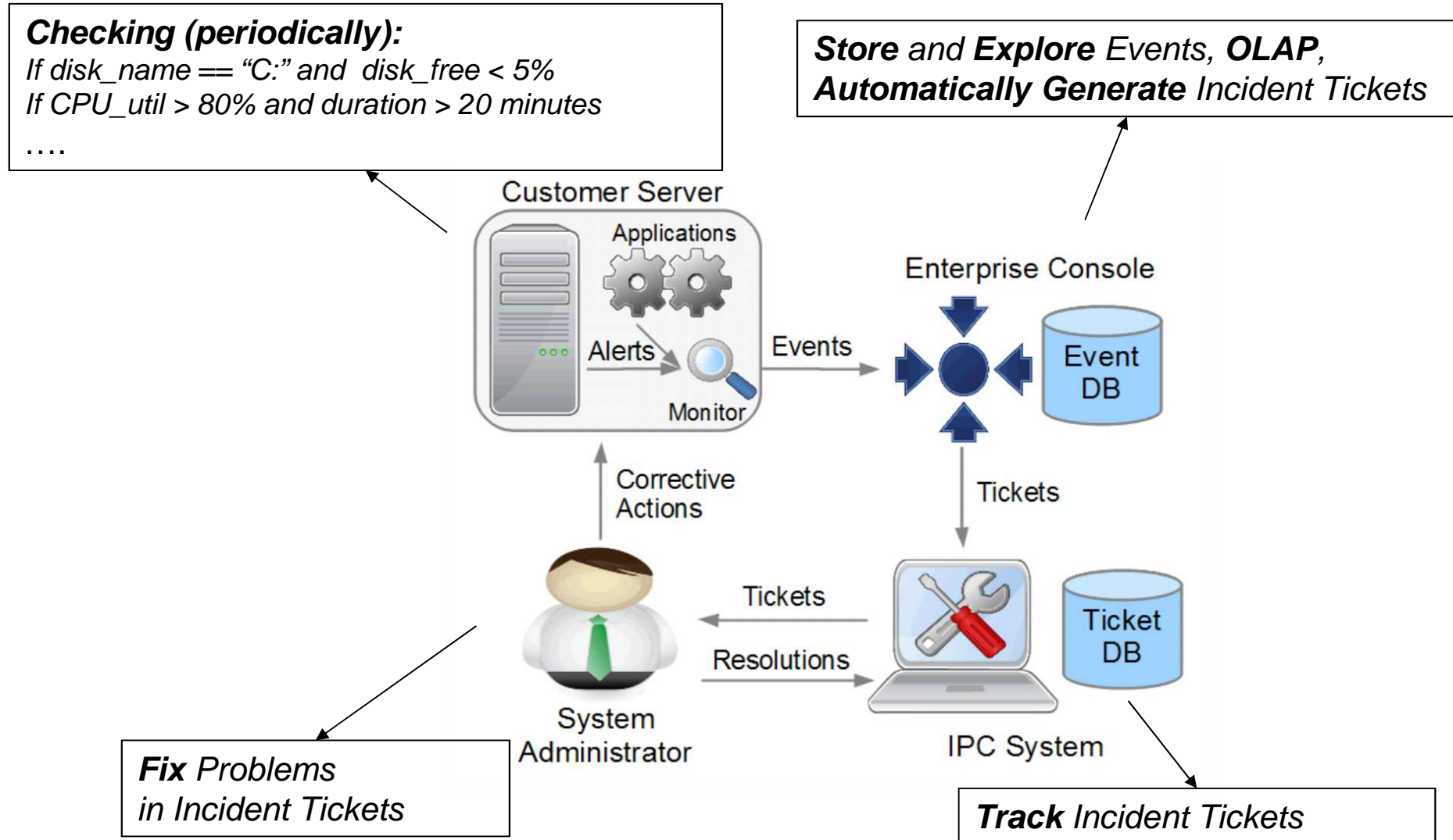
Background

The typical workflow of IT service mainly includes four components (Liang, Tao, Larisa, et al., 2012):

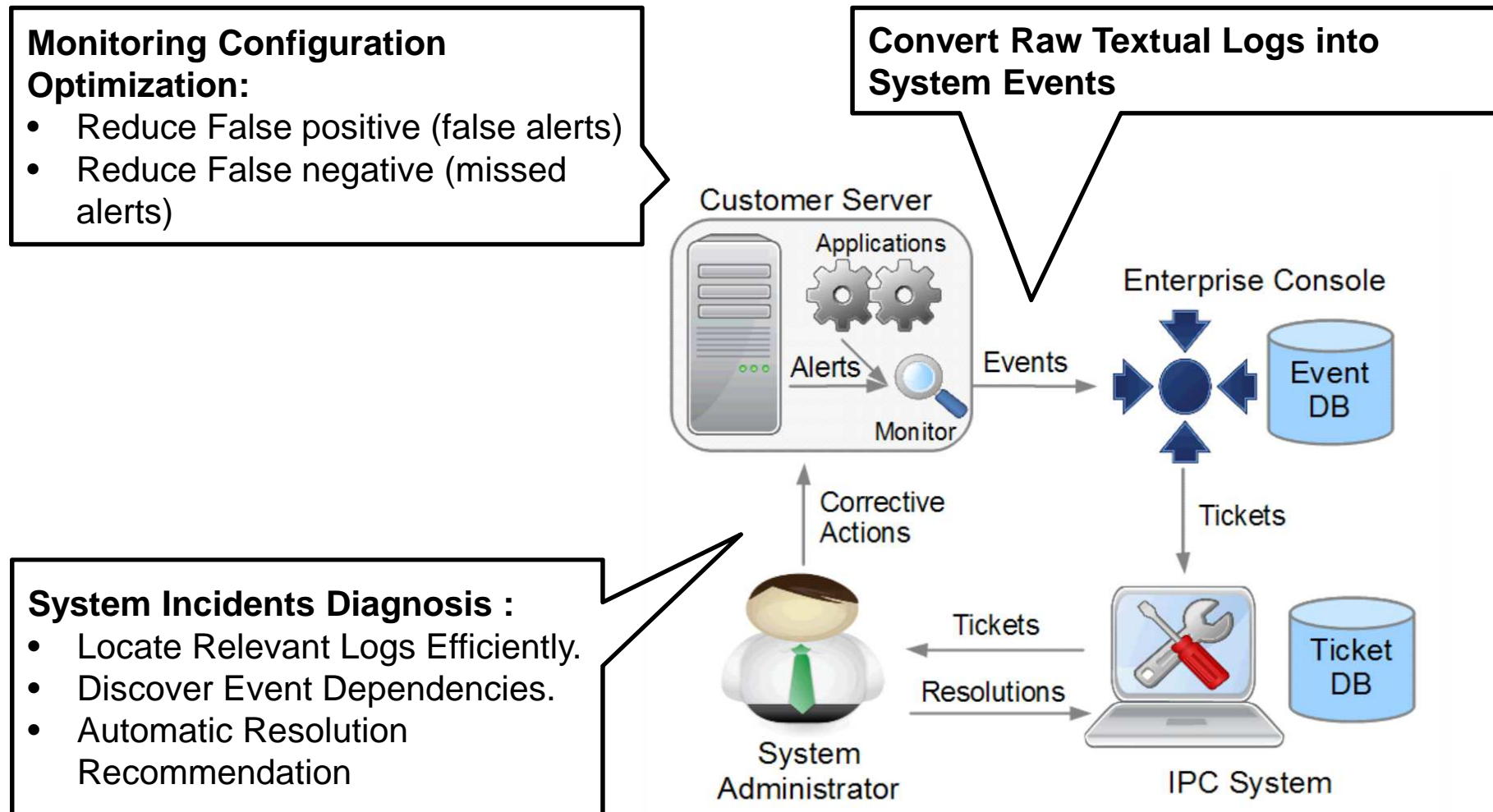
- Customer Servers
- Event DB
- Ticketing System
- System Administrators



Background (Cont.)



Overview of Research Problems



Outline

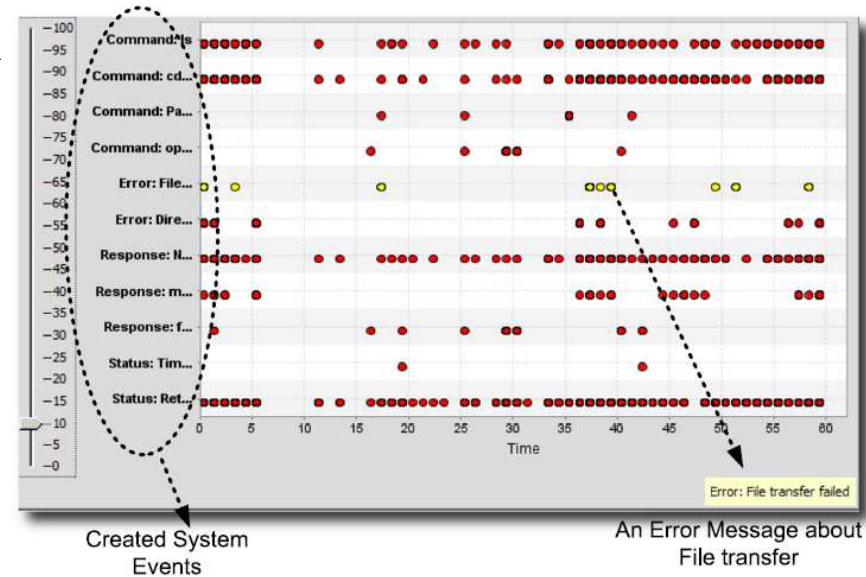
- Background and Overview
- Research Problems
 - Converting Textual Log to System Events
 - Monitoring Configuration Optimization
 - False Positive
 - False Negative
 - Analysis on Detected System Issues
 - Temporal Dependencies of Events
 - Incident Resolution Recommendation
 - Textual Event Segments Search
- Summary and Timeline

Why Convert Textual Logs to System Events?

Table I: An Example of FileZilla's log.

| No. | Message |
|-----------------|--|
| s ₁ | 2010-05-02 00:21:39 Command: put "E:/Tomcat/apps/index.html" "/disk/... |
| s ₂ | 2010-05-02 00:21:40 Status: File transfer successful, transferred 823 bytes... |
| s ₃ | 2010-05-02 00:21:41 Command: cd "/disk/storage006/users/lt... |
| s ₄ | 2010-05-02 00:21:42 Command: cd "/disk/storage006/users/lt... |
| s ₅ | 2010-05-02 00:21:42 Command: cd "/disk/storage006/users/lt... |
| s ₆ | 2010-05-02 00:21:42 Command: put "E:/Tomcat/apps/record1.html" "/disk/... |
| s ₇ | 2010-05-02 00:21:42 Status: Listing directory /disk/storage006/users/lt... |
| s ₈ | 2010-05-02 00:21:42 Status: File transfer successful, transferred 1,232 bytes... |
| s ₉ | 2010-05-02 00:21:42 Command: put "E:/Tomcat/apps/record2.html" "/disk/... |
| s ₁₀ | 2010-05-02 00:21:42 Response: New directory is: "/disk/storage006/users/lt... |
| s ₁₁ | 2010-05-02 00:21:42 Command: mkdir "libraries" |
| s ₁₂ | 2010-05-02 00:21:42 Error: Directory /disk/storage006/users/lt... |
| s ₁₃ | 2010-05-02 00:21:44 Status: Retrieving directory listing... |
| s ₁₄ | 2010-05-02 00:21:44 Command: ls |
| s ₁₅ | 2010-05-02 00:21:45 Command: cd "/disk/storage006/users/lt... |
| ... | ... |

System events are **easier** to analyze other textual logs.



Existing Solutions

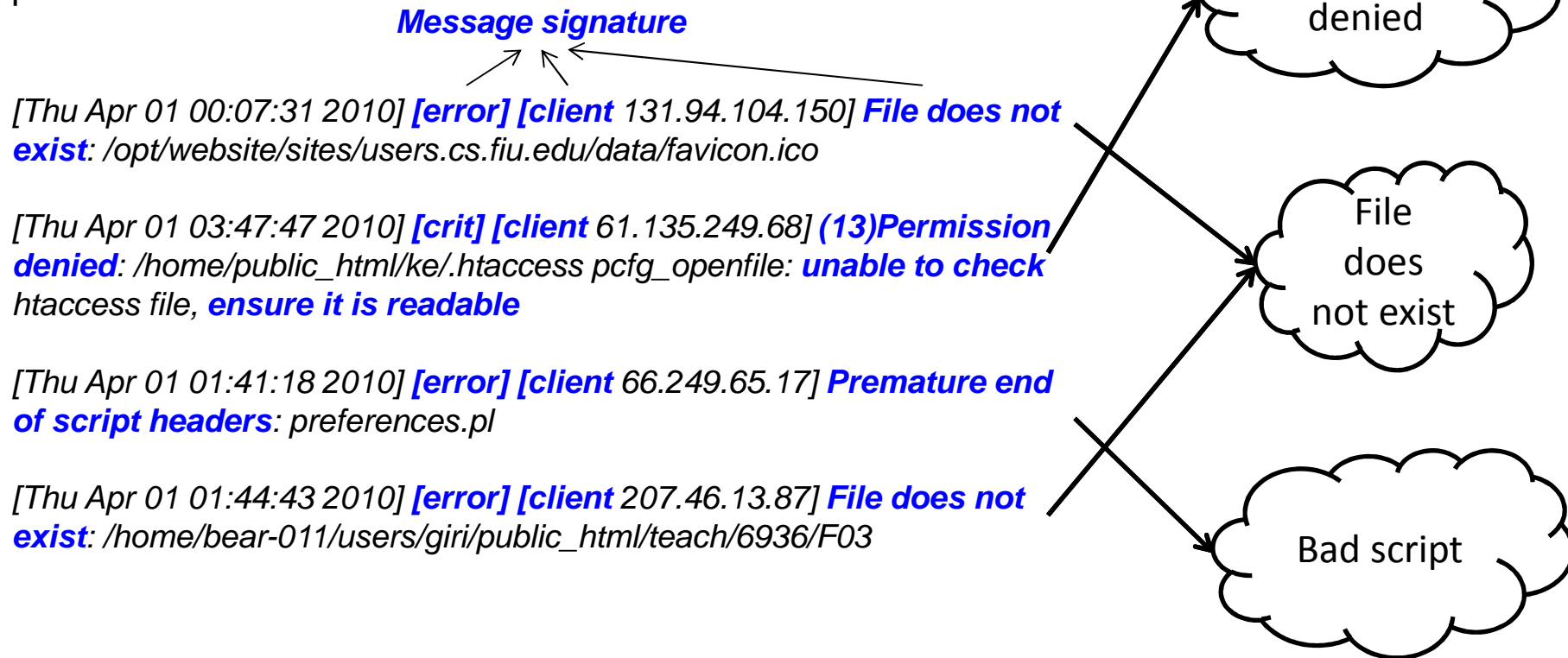
- Write a log parser (W. Xu et al., 2008):
 - Requires to understand all log messages.
 - Document or Source code are not available.
 - Implementation is time consuming.
- Document clustering with Bag-of-Word model:
 - Short lengths of log messages but a Large vocabulary. Due to the **curse of the dimensionality**, the clustering result would be very poor.
- Other clustering methods (M. Aharon et al., 2009; A. Makanju et al, 2009):
 - Extract structure features of log messages, but only work for some types of log messages.

Preliminary Work:

Message Signature Based Clustering

Message signature is the signature of the template.

One type of log messages is generated by **one** template with **different** parameters.



Message Signature based Clustering

- Problem: Find k most **representative** message signatures.
- Question: How to quantify the “representativeness” ?
- **Definition:**
 - Given a message X and a message signature S , the match score is the number of **matched** terms **minus** the number of **unmatched** terms.
 - $match(X,S) = |LCS(X,S)| - (|S| - |LCS(X,S)|) = 2|LCS(X,S)| - |S|$,
LCS=Longest Common Subsequence.
- **Example:**
 - $X = \text{“abcdef”}$, $S = \text{“axcey”}$, $match(X,S) = |ace| - |xy| = 1$

| | | | | | | |
|---|----------|---|----------|---|----------|---|
| X | a | b | c | d | e | f |
| S | <u>a</u> | x | <u>c</u> | | <u>e</u> | y |

Problem Definition

Given a set of log messages \mathcal{D} and an integer k , find k message signature $\mathcal{S} = \{S_1, \dots, S_k\}$ and a k -partition C_1, \dots, C_k of \mathcal{D} to maximize:


$$J(\mathcal{S}, \mathcal{D}) = \sum_{i=1}^k \sum_{X_j \in C_i} \text{match}(X_j, S_i).$$

Problem Analysis:

- Similar to k -means problem, but NOT really.
- Finding the **Optimal** Solution is **NP-Hard**, even if $k=1$.
 - **Multiple Longest Common Subsequence Problem** can be reduced to our problem.

Approximate Problem 1

- Convert each log message into Term Pairs:

2010-05-02 11:34:06 Command: mkdir ".indexes" 

{2010-05-02,11:34:06}, {2010-05-02,Command:},
 {2010-05-02, mkdir}, {2010-05-02, ".indexes" }
 {11:34:06,Command:}, {11:34:06,mkdir},
 {11:34:06,".indexes"}, {Command:,mkdir},
 {Command:,".indexes"}, {mkdir,".indexes"}

- Maximize $F(C, \mathcal{D}) = \sum_{i=1}^k \left| \bigcap_{X_j \in C_i} R(X_j) \right|$. $R(X_j)$: the set of term pairs of log message X_j .
 $C = \{C_1, \dots, C_k\}$ is the partition of log messages

S is the set of signatures

- Lemma: If $F(C, \mathcal{D}) \geq y$, then $J(S, \mathcal{D}) \geq |\mathcal{D}| \cdot \left\lceil \sqrt{\frac{2y}{k}} \right\rceil$

However, F value can be changed by a series of updates of C (not smooth), not good for local search algorithms.

Approximate Problem 2

- Potential for one message group
 - Given a message group C , the potential of C is defined as
$$\phi(C) = \sum_{r \in R(C)} N(r, C) [p(r, C)]^2$$
 - $N(r, C)$ is the number messages in C that contain pair r . $p(r, C) = N(r, C)/|C|$ is the portion of messages in C having r .

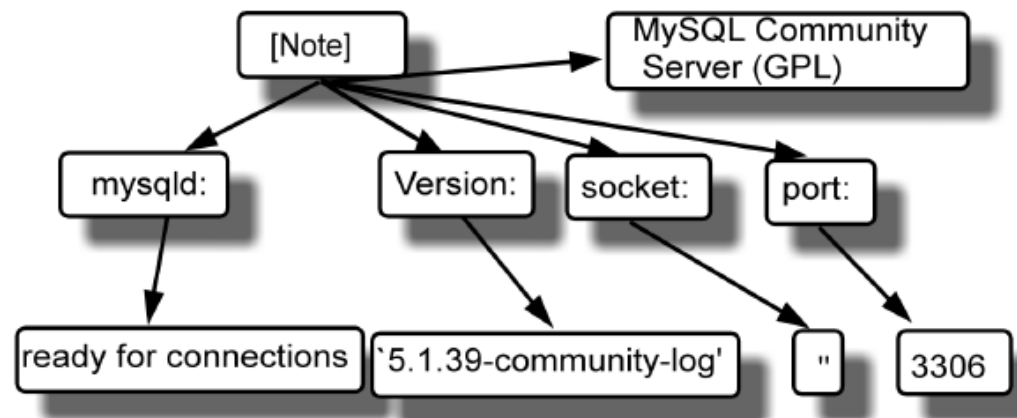
$$\Phi(\mathcal{D}) = \sum_{i=1}^k \phi(C_i)$$

- Overall Potential
 - Sum of all message groups' potentials.
- Lemma: If $F(C, D) \geq y$, then $\Phi(\mathcal{D}) \geq y \cdot |\mathcal{D}|/k$

Tree-Structure based Clustering

- Log messages are too short. Extracting the structural information can help the clustering by a context-free parser. (Most log messages have similar context-free grammars).
- Computing the similarities of trees instead of the raw textual messages.

```
100405 0:00:49 [Note] mysqld: ready for connections.Version:  
`5.1.39-community-log' socket: " port: 3306 MySQL Community  
Server (GPL)
```



Outline

- Background and Overview
- Research Problems
 - Converting Textual Log to System Events
 - Monitoring Configuration Optimization
 - False Positive
 - False Negative
 - Analysis on Detected System Issues
 - Temporal Dependencies of Events
 - Incident Resolution Recommendation
 - Textual Log Segments Search
- Summary and Timeline

What is False Positive (False Alarm)?

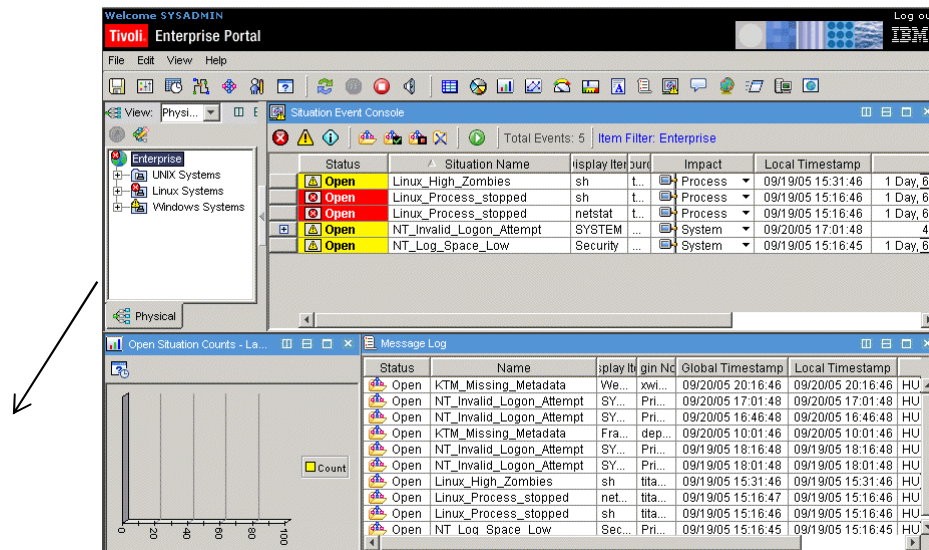
- If *PROCESS_CPU_UTILIZATION* > 50% and *duration* > 10 minutes, then generates a CPU alert
 - “rtvscan.exe” scans the system periodically, it is CPU intensive but it is normal, so it triggers a lot of false positives (false alerts).
- If *PAGING_UTILIZATION_15min* > 400, then generate a paging alert (default situation in IBM Tivoli monitoring)
 - Some customer servers have multiple CPU and huge memories. For those multi-CPU servers, it is normal for page swapping over thousands of times in 15 minutes.

Why We Have False Positive?

- Too Conservative Configurations
 - Missing a real alert would incur system crash, data loss.
- Changes of Monitored Servers
 - New servers and more powerful device are installed.
- Transient Alerts:
 - Temporal CPU, Paging, Disk Spike.
 - Restart of servers, processes, services, routers...

IBM Tivoli Monitoring

Complicated configurations for IBM Tivoli monitoring



Problem Statement

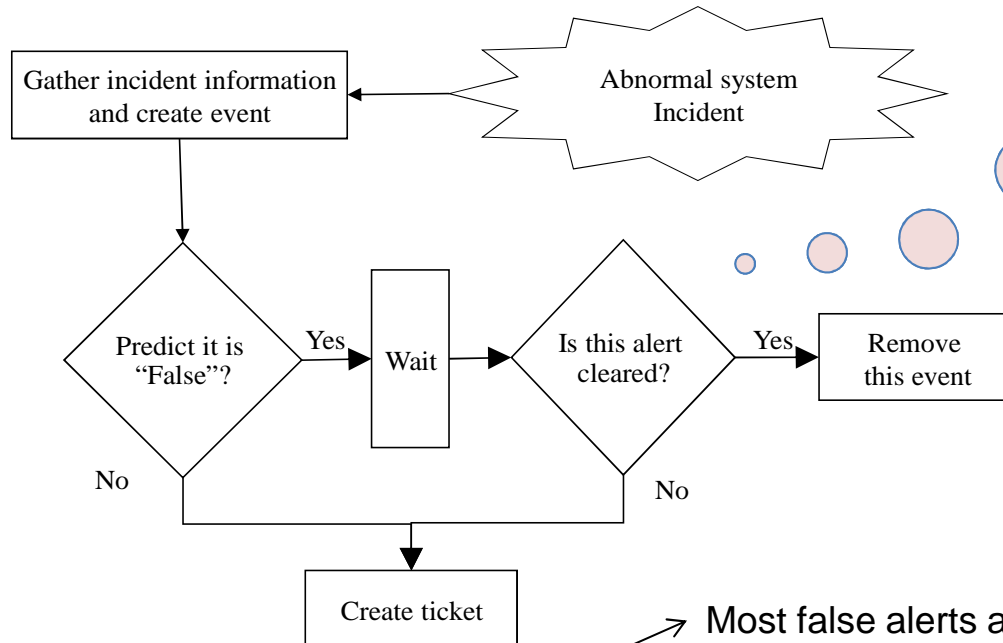
- Eliminate false positives by refining the Monitoring configurations
- Retain all real alerts. No real alert is allowed to miss.

Potential Approaches

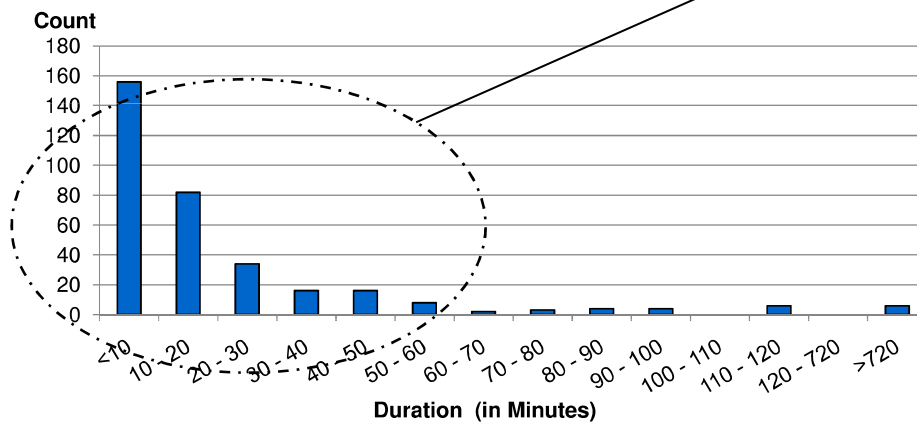
- Supervised Method
 - Build a predictor for predicting false positive.
- Unsupervised Methods
 - Outlier detection (S. Agrawal et al., 2007, K. Xu et al., 2005)
 - Adaptive threshold (S.R. Kashyap et al., 2008)
 - ...

However, they do not guarantee NO real alert is missed.

Preliminary Work



Most false positive alerts are **transient alerts** (automatically disappear in a short time).



Most false alerts are transient alerts.

Some transient alerts may be indications of future real alerts and may be useful. But if those real alerts rise later on, the monitoring system will detect them even if the transient alerts were ignored.

Outline

- Background and Overview
- Research Problems
 - Converting Textual Log to System Events
 - Monitoring Configuration Optimization
 - False Positive
 - False Negative
 - Analysis on Detected System Issues
 - Temporal Dependencies of Events
 - Incident Resolution Recommendation
 - Textual Event Segments Search
- Summary and Timeline

What is False Negative (Missed Alert) ?

- False Negatives are the missed alerts by the monitoring system (not by humans).
- False Negatives are usually captured by human (customers, helpdesk, system administrators).
- False Negatives are NOT recorded in events, but only in manual tickets.

Why We Have False Negatives?

- New devices and software are installed, but not added into the monitoring configurations
 - For example, the customer may install a new database by themselves. But the monitoring team does not know it. When the database has an incident, it cannot be captured by the monitoring system.
- Other changes for existing systems.

Problem Statement

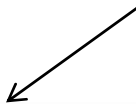
- Eliminate false negatives by refining the monitoring configurations
- It consists of two parts:
 - Scan the historical manual tickets and provide a short list of potential false negatives to the monitoring team (This is our work).
 - Change or add monitoring situations (This is monitoring team's work).

Related Research and Challenges

- **Related research:**
 - Improve the accuracy of the monitoring methodologies. No prior work that mentions how to utilize the ticket data.
- **A straightforward method:** Build a binary classifier to classify the manual tickets
 - Label “1”: a missed alert
 - Label “0”: other issues, such as customer request.
- **Challenges**
 - Highly **Imbalance** manual tickets. Most tickets’ labels are “0”. Only 1% or less are “1”.
 - Labeled data is **rare**.

Proposed Method

- Selective Labeling by *domain words* (labeled features)



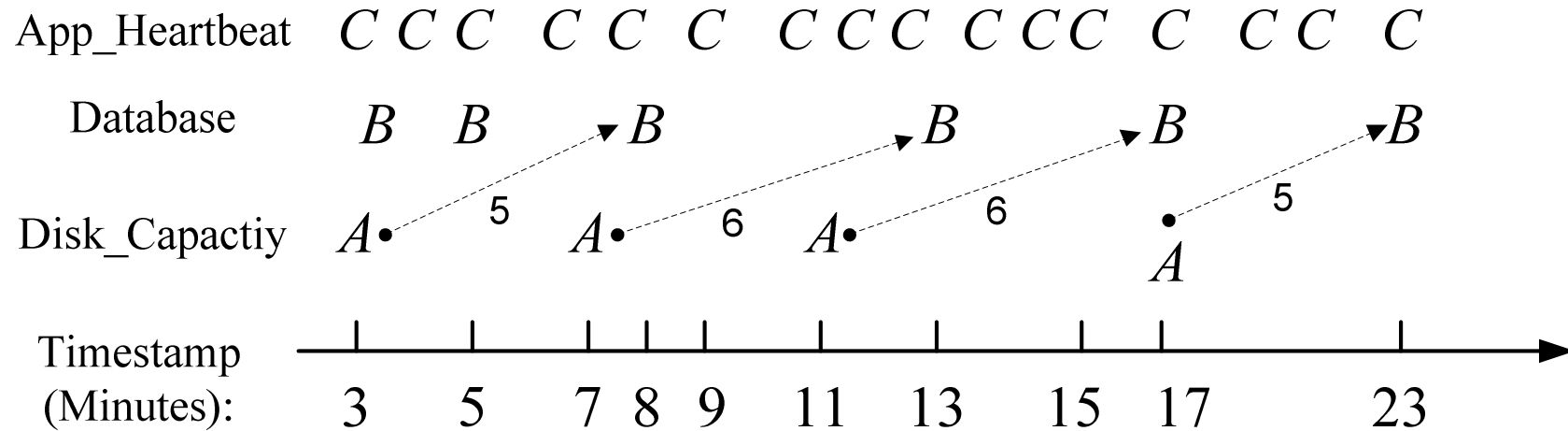
| Situation Issue | Words |
|-------------------------------|-----------------|
| DB2 tablespace Utilization | DB2, tablespace |
| File System Space Utilization | space,file |
| Disk Space Capacity | space,drive |
| Service Not Available | service,down |
| Router/Switch Down | router |

- Use coverage of domain words to rank all tickets.
- Only select top ranked tickets for labeling and training
- Build a binary SVM classifier.

Outline

- Background and Overview
- Research Problems
 - Converting Textual Log to System Events
 - Monitoring Configuration Optimization
 - False Positive
 - False Negative
 - Analysis on Detected System Issues
 - Temporal Dependencies of Events
 - Incident Resolution Recommendation
 - Textual Event Segments Search
- Summary and Timeline

What is Temporal Dependency?



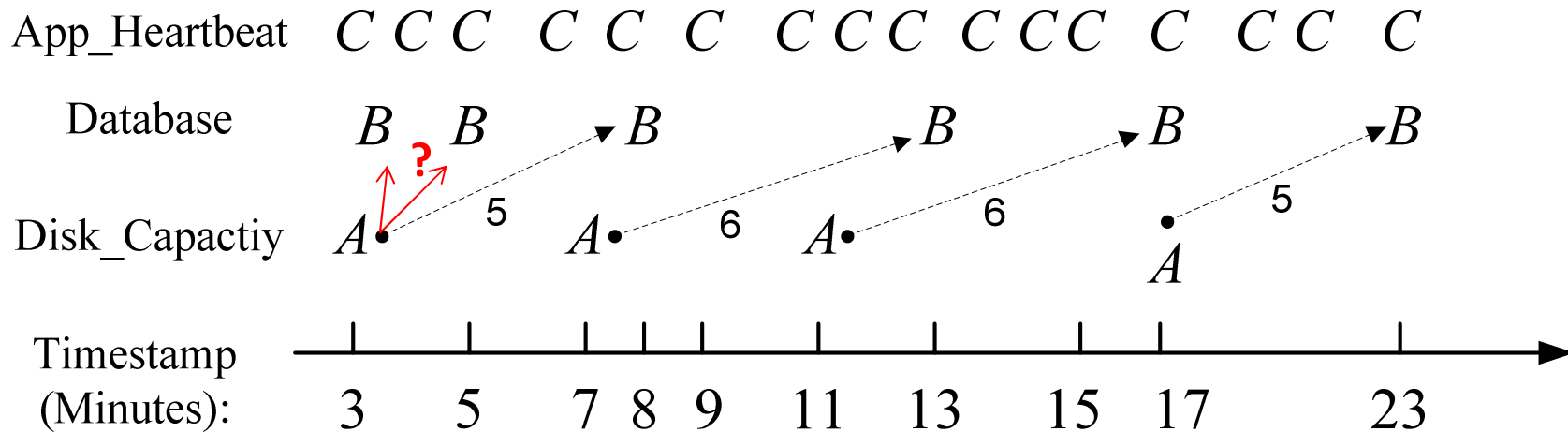
Disk_Capacity $\rightarrow_{[5\text{min}, 6\text{min}]}$ Database, [5min, 6min] is the lag interval.

Why We Want to Find the Temporal Dependencies?

- Finding dependent system components
 - Identify the root cause of system problems
 - ...
- Finding correlated monitoring situations
 - Remove redundant situations
 - Event correlation
 - ...

Existing Approaches

- Predefine the lag interval (H. Mannila et al., 1997)
- No interleaved dependency (T. Li et al., 2005, K. Bouandas et al., 2007)



Disk_Capacity \rightarrow $[5\text{min}, 6\text{min}]$ Database, $[5\text{min}, 6\text{min}]$ is the **lag interval**.

Relation with Other Temporal Patterns

Those temporal patterns can be seen as the temporal dependency with **particular** constraints on the time lag.

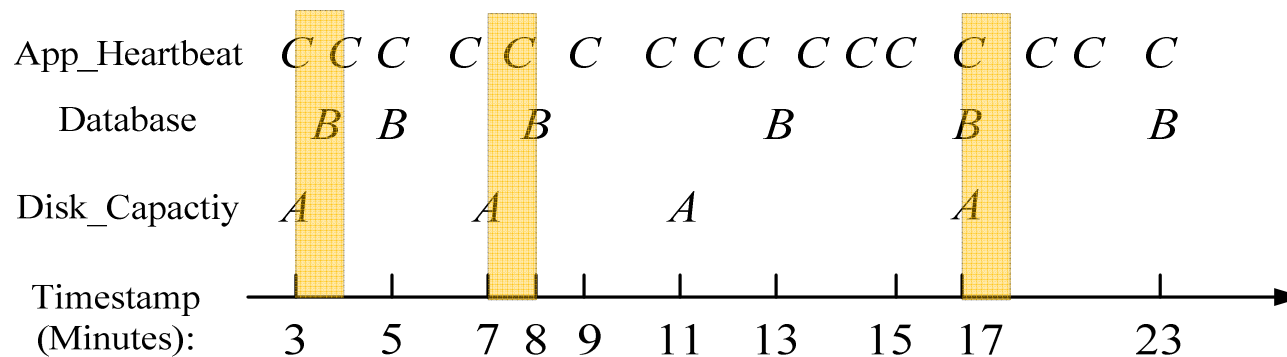
| | | |
|--------------------|---|--|
| Mutually Dependent | $\{A, B\}$ | $A \rightarrow_{[0,t]} B, B \rightarrow_{[0,t]} A$ |
| Partial Periodic | A with periodic p and time tolerance δ | $A \rightarrow_{[p-\delta, p+\delta]} A$ |
| Frequent Episode | $A \rightarrow B \rightarrow C$ | $A \rightarrow_{[0,t]} B, B \rightarrow_{[0,t]} C$ |
| Loose Temporal | B follows A before t | $A \rightarrow_{[0,t]} B$ |
| Stringent Temporal | B follows A about t | $A \rightarrow_{[t-\delta, t+\delta]} B$ |

Challenges for Finding Time Lag

- Given a temporal dependency, $A \rightarrow_{[t1, t2]} B$, what kind of lag interval $[t1, t2]$ we want to find?
 - If the lag interval is too **large**, every A and every B would be “dependent”.
 - If the lag interval is too **small**, real dependent A and B might not be captured.
- Time complexity is too high.
 - $A \rightarrow_{[t1, t2]} B$, $t1$ and $t2$ can be any distance of any two time stamps. There are $O(n^4)$ possible lag intervals.

What is a Qualified Lag Interval

- If $[t_1, t_2]$ is qualified, we should observe **many** occurrences for $A \rightarrow_{[t_1, t_2]} B$.



| Lag Interval | Number of Occurrences |
|--------------|-----------------------|
| [0,1] | 3 |
| [5,6] | 4 |
| [0,6] | 4 |
| [0,+∞] | 4 |

Length of the lag interval is larger, the number of occurrences also becomes larger.

What is a Qualified Lag Interval

- Intuition (Statistical Testing):

Expected value

- If A and B are **randomly** and **independently** distributed, how many occurrences observed in a time interval $[t_1, t_2]$?
- What is the minimum number of occurrences (threshold)?
 - Consider the number of occurrences in a lag interval to be a variable, n_r . Then, use the *chi-square* test to judge whether it is caused by **randomness** or not?

$$\chi_r^2 = \frac{(n_r - n_A P_r)^2}{n_A P_r (1 - P_r)}$$

The number of As

$$P_r = |r| \frac{n_B}{T}$$

Total time length of the event sequence

Straightforward Algorithms for Finding Qualified Lag Intervals

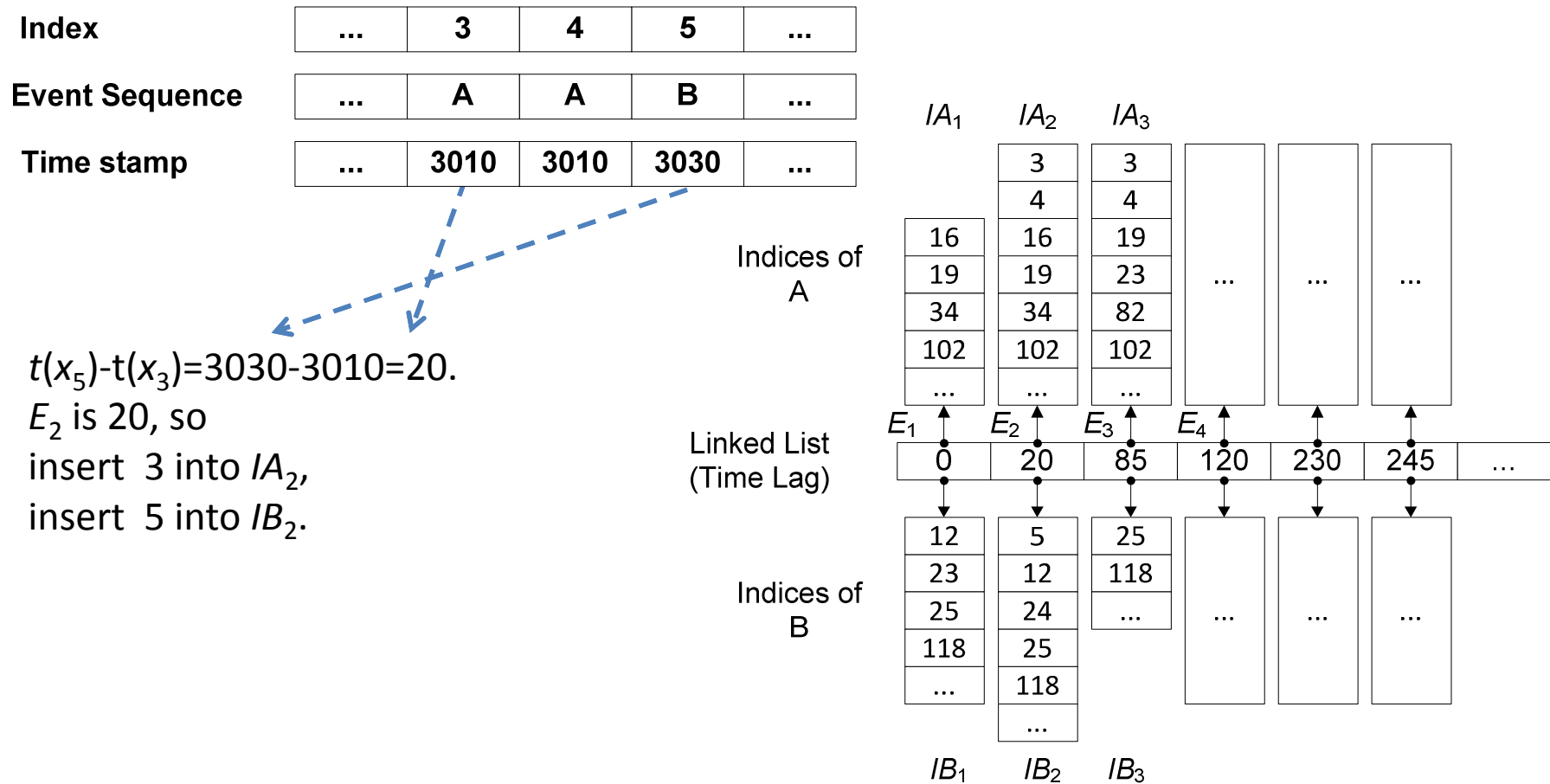
- **(Brute-Force) Algorithm:** For $A \rightarrow_{[t1, t2]} B$, for every possible $t1$ and $t2$, scan the event sequence and count the number of occurrences.
- Time Complexity
 - The number of distinct time stamps is $O(n)$.
 - The number of possible $t1$ and $t2$ is $O(n^2)$.
 - The number of possible $[t1, t2]$ is $O(n^4)$.
 - Each scanning is $O(n)$. The total cost is $O(n^5)$.
- Cannot handle large event sequences.

Time Complexity Lower Bound

- The problem of finding all qualified time intervals is 3SUM-Hard, so there is no $o(n^2)$ algorithm in the worst case (A. Gajentaan et al., 1995).
- **3SUM problem:** Given a set of n integers, is there three integers a, b, c in the set such that $a+b=c$?
- No $o(n^2)$ algorithm can solve this problem in the worst case.

Preliminary Work: STScan Algorithm

- Idea:
 - Avoid **redundant** scanning, store all time lags into a sorted table.



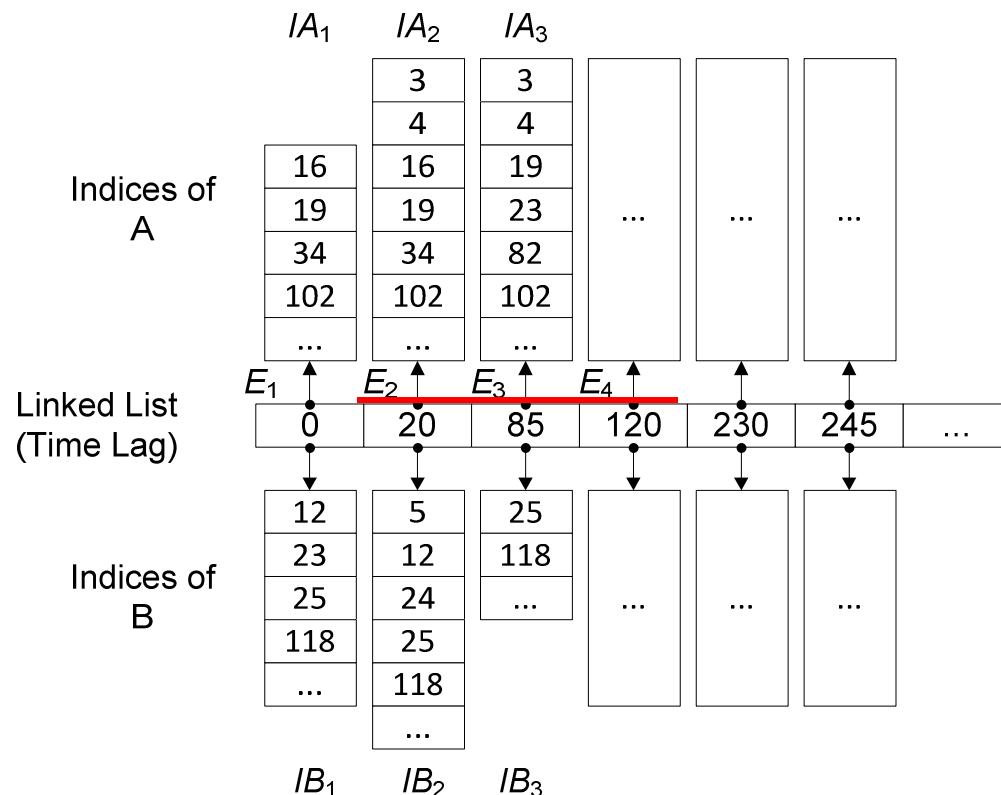
STScan Algorithm

- Every lag interval is represented as a **sub-segment** of the linked list.
- For example: $[20, 120]$ is $E_2 E_3 E_4$, the number of occurrences is $|IA_2 \cup IA_3 \cup IA_4|$

Time cost for creating this table is $O(n^2)$.

The number of elements is $O(3n^2) = O(n^2)$.

Time cost for scanning is $O(n^2)$.



Outline

- Background and Overview
- Research Problems
 - Converting Textual Log to System Events
 - Monitoring Configuration Optimization
 - False Positive
 - False Negative
 - Analysis on Detected System Issues
 - Temporal Dependencies of Events
 - Incident Resolution Recommendation
 - Textual Event Segments Search
- Summary and Timeline

Incident Ticket with Resolution

| Ticket ID | Resolver Name | Open Time | Problem Description | Solution Description | ... |
|-------------|---------------|----------------------------------|--|--|------|
| IBM-C102203 | John | May 7, 2012 6:51:49 AM EDT | Summary: MS SQL issue: database AEFA_DB very little log free space in ... | *** Clearing Event Received. Event Follows *** | |
| IBM-C422013 | XC | May 6, 2012 9:06:40 AM EDT | MS SQL issue: bad server status: on server TMPD0371.... | Server reboots resolved_full | ... |
| ... | ... | ... | ... | ... | ... |

Observation: If the **problem descriptions** are similar, the **solution descriptions** are likely to be similar or identical.

Problem Statement

- Build a recommendation model for recommending the relevant resolutions for new tickets.
 - When a new ticket arrives, recommends top K relevant resolutions

Why We Study this Problem?

- Help administrators diagnose new tickets.
 - Many system problems are **not isolated**.
 - Share the knowledge between different administrators.
 - ...

Related Work

- User-based Recommendation Algorithms
- Item-based Recommendation Algorithms
- Constraint-based Recommender Systems
- Multiple Objective Optimization...

Preliminary Work

- User-based Top-K Recommendation
 - Find top K similar tickets based on problem descriptions
- Incorporating the **penalty** of “wrong” recommendation
 - Recommending a false ticket’s resolution to a real ticket would mislead the administrators (may incur serious consequences).
 - However, we do not know a new ticket is real or false.
 - Combine the ticket prediction confidence with the ticket relevance.

Proposed Method

- Measuring the **quality** of the resolutions
 - No explicit ratings of resolutions.
 - Many resolutions are not **informative**, like “solved”, “*cleared*”, “*is fixed*”.

Outline

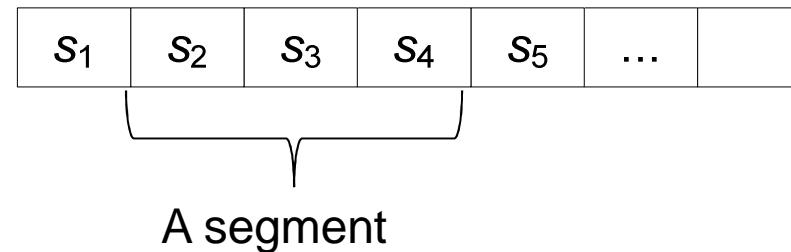
- Background and Overview
- Research Problems
 - Converting Textual Log to System Events
 - Monitoring Configuration Optimization
 - False Positive
 - False Negative
 - Analysis on Detected System Issues
 - Temporal Dependencies of Events
 - Incident Resolution Recommendation
 - Textual Event Segments Search
- Summary and Timeline

What is Textual Event Segment?

A textual event sequence is a sequence of events, where each event is a text (e.g., textual log sequence)

Table I: An Example of FileZilla's log.

| No. | Message |
|-----------------|--|
| s ₁ | 2010-05-02 00:21:39 Command: put "E:/Tomcat/apps/index.html" "/disk/... |
| s ₂ | 2010-05-02 00:21:40 Status: File transfer successful, transferred 823 bytes... |
| s ₃ | 2010-05-02 00:21:41 Command: cd "/disk/storage006/users/lt... |
| s ₄ | 2010-05-02 00:21:42 Command: cd "/disk/storage006/users/lt... |
| s ₅ | 2010-05-02 00:21:42 Command: cd "/disk/storage006/users/lt... |
| s ₆ | 2010-05-02 00:21:42 Command: put "E:/Tomcat/apps/record1.html" "/disk/... |
| s ₇ | 2010-05-02 00:21:42 Status: Listing directory /disk/storage006/users/lt... |
| s ₈ | 2010-05-02 00:21:42 Status: File transfer successful, transferred 1,232 bytes... |
| s ₉ | 2010-05-02 00:21:42 Command: put "E:/Tomcat/apps/record2.html" "/disk/... |
| s ₁₀ | 2010-05-02 00:21:42 Response: New directory is: "/disk/storage006/users/lt... |
| s ₁₁ | 2010-05-02 00:21:42 Command: mkdir "libraries" |
| s ₁₂ | 2010-05-02 00:21:42 Error: Directory /disk/storage006/users/lt... |
| s ₁₃ | 2010-05-02 00:21:44 Status: Retrieving directory listing... |
| s ₁₄ | 2010-05-02 00:21:44 Command: ls |
| s ₁₅ | 2010-05-02 00:21:45 Command: cd "/disk/storage006/users/lt... |
| ... | ... |



A textual event segment is a segment of the textual event sequence. For instance, $s_2s_3s_4$, but $s_2s_3s_5$ is not.

Problem Statement

- Given an event sequence S and a query sequence Q , find all segments with length $|Q|$ in S that are **similar** to Q .
- What is the dissimilarity:

$$N_{dissim}(L_1, L_2, \delta) = \sum_{i=1}^l z_i \leq k,$$

where

$$z_i = \begin{cases} 1, & \text{sim}(e_{1i}, e_{2i}) < \delta \\ 0, & \text{otherwise} \end{cases},$$

and δ is a user-defined threshold for the event similarity.

In other words, similar segments have **at most** k dissimilar events.

Why We Study This Problem?

- Investigating historical logs is a common way to diagnose the system incident.
 - Similar event segments reveal similar system behaviors.
 - Comparing similar event segments help to find out the reason for a system problem.

We allow similar segments have some dissimilar events, because they may be the abnormal behavior.

Potential Solutions

- Unordered Text Similarity Search
 - LSH(Locality Sensitive Hashing) (A. Gionis et al., 1999) + Min Hash Function(A. Z. Broder et al., 1998).
 - ...
- Code Sequence Match or Alignment
 - Suffix Tree, Suffix Array (U. Manber, 1993)
 - ...

Preliminary Work

- LSH + Suffix Array = Suffix Matrix

EXAMPLE 1. Let S be a sequence of events, $S = e_1e_2e_3e_4$. H is a set of independent hash functions for events, $H = \{h_1, h_2, h_3\}$. For each event and hash function, the computed the hash value is shown in Table 1.

Table 1: An Example of Hash Value Table

| Event | e_1 | e_2 | e_3 | e_4 |
|-------|-------|-------|-------|-------|
| h_1 | 0 | 2 | 1 | 0 |
| h_2 | 3 | 0 | 3 | 1 |
| h_3 | 1 | 2 | 2 | 0 |

Let $h_i(S)$ denote the i -th row of Table 1. By sorting the suffixes in each row of Table 1, we could get the suffix matrix $\mathbf{M}_{S,m}$ below.

$$\mathbf{M}_{S,m} = \begin{bmatrix} 3 & 0 & 2 & 1 \\ 1 & 3 & 0 & 2 \\ 3 & 0 & 2 & 1 \end{bmatrix}.$$

For instance, the first row of $\mathbf{M}_{S,m}$: 3021, is the suffix array of $h_1(S) = 0210$.

Offline Indexing:

Step 1. Construct m random hash functions

Step 2. For each hash function, compute the hash value of each event.

Step 3. For each hash value sequence, build the suffix array as a row of the suffix matrix.

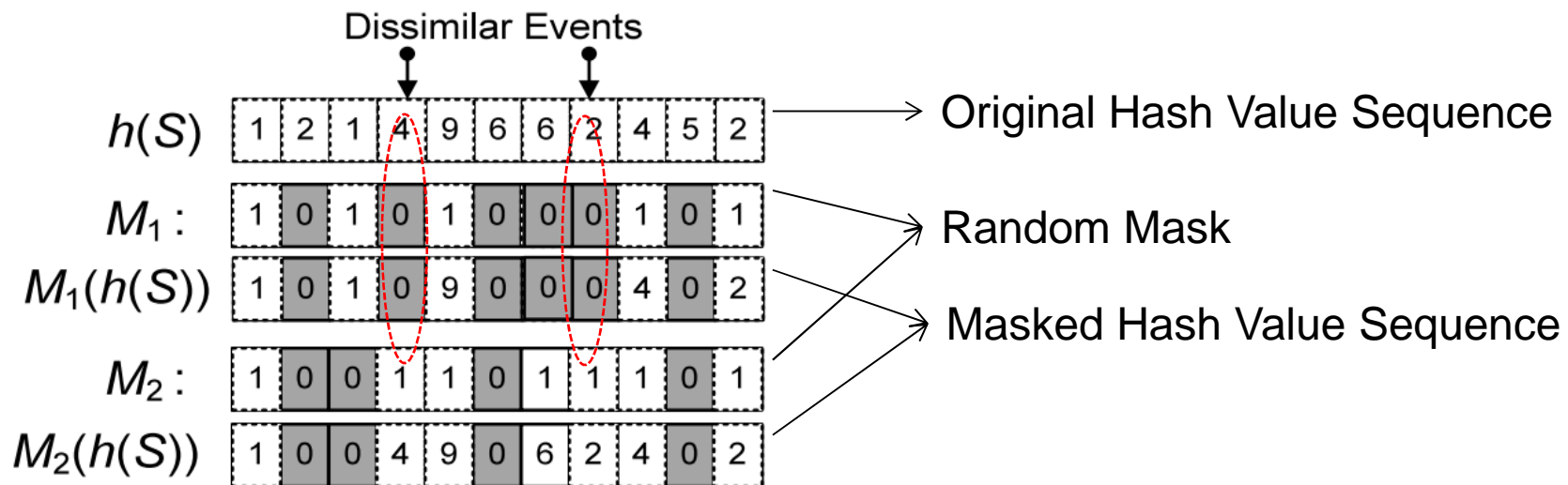
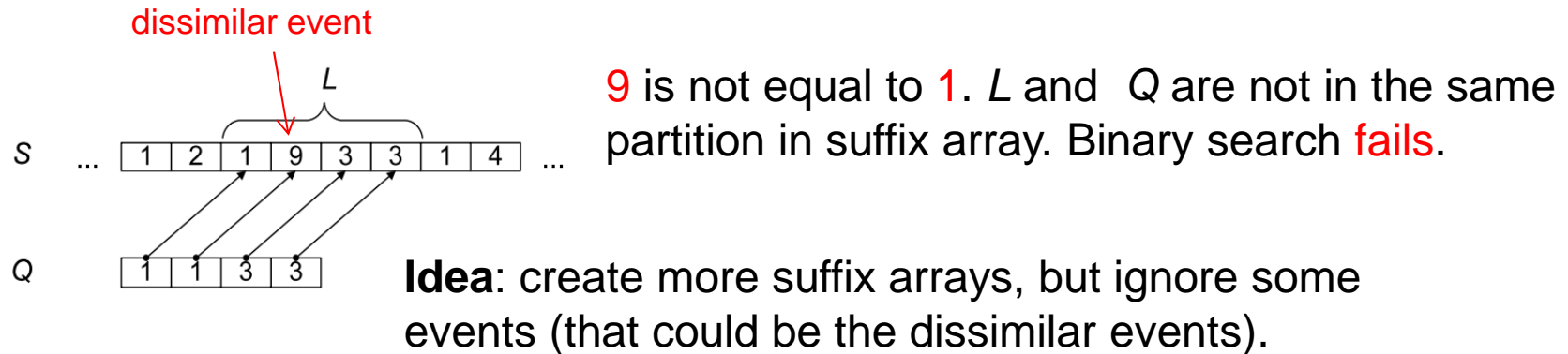
Online Search:

Step 1. Use the m hash functions to hash query Q and get m hash value query sequences.

Step 2. Use every hashed query sequence to do *binary search* over suffix arrays and get candidate segment positions.

Step 3. If one segment appears in many candidate sets, pick it as the final candidate.

Handling Dissimilar Events



Dissimilar events are masked in $M_1(h(S))$ and do NOT hurt the binary searches.

Outline

- Background and Overview
- Research Problems
 - Converting Textual Log to System Events
 - Monitoring Configuration Optimization
 - False Positive
 - False Negative
 - Analysis on Detected System Issues
 - Temporal Dependencies of Events
 - Incident Resolution Recommendation
 - Textual Event Segments Search
- Summary and Timeline

Summary

Monitoring Configuration Optimization:

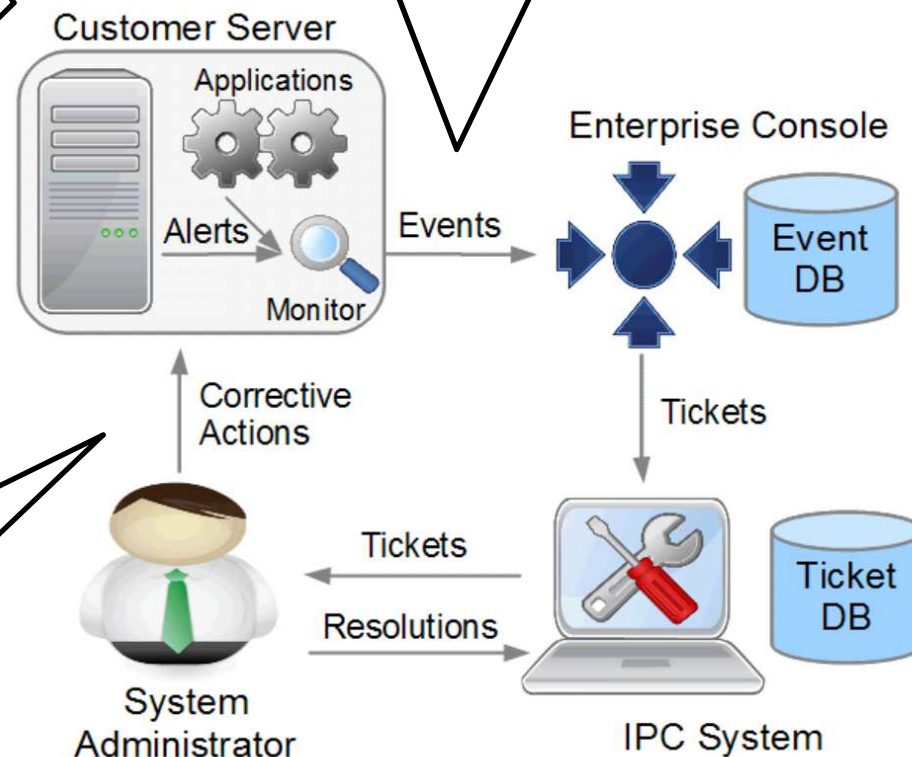
- Reduce False positive (false alerts)
- Reduce False negative (missed alerts)

Green one is already done.
Red one is ongoing.

System Incidents Diagnosis :

- Locate Relevant Logs Efficiently.
- Discover Event Dependencies.
- Automatic Resolution Recommendation

Convert Raw Logs into System Events



Timeline

- End of 2013 Fall
 - Finish “False Negative Discovery”.
 - Finish “Ticket Resolution Recommendation”.
- 2014 Spring
 - Defense and Finish Dissertation.
- 2014 Summer
 - Apply Graduation.

References

- A. Gajentaan and M. H. Overmars, “On a class of $O(n^2)$ problems in computational geometry”, in Computational Geometry, 5:165-185, 1995.
- A. Gionis, P. Indyk, and R. Motwani, “Similarity search in high dimensions via hashing”, in VLDB 1999.
- A. Makanju, A. N. Zirc-Heywood, and E. E. Miliotis, “Clustering event logs using iterative partitioning”, in KDD 2009.
- A. Z. Broder, N. Charikar, A. M. Frieze, and M. Mitzenmacher, “Min-wise independent permutations” in STOC 1998.
- H. Mannila, H. Töivonen, and A. I. Verkamo, “Discovery of frequent episodes in event sequences”, in DMKD, 1(3):259-289, 1997.
- K. Bouandas and A. Osmani, “Mining association rules in temporal sequences”, in CIDM 2007.
- M. Aharon, G. Barash, I. Cohen, and E. Mordechai, “One graph is worth a thousand logs: Uncovering hidden structures in massive system event logs”, in ECML/PKDD, 2009.
- K. Xu, Z.-L. Zhang, and S. Bhattacharyya, “Profiling internet backbone traffic: behavior models and applications”, in SIGCOMM 2005.
- L. Tang, T. Li, F. Pinel, L. Schwartz, and G. Grabarnik, “Optimizing system monitoring configurations for non-actionable alerts”, in IEEE/IFIP NOMS, 2012.
- S. Agrawal, S. Deb, K.V.M. Naidu, and R. Rastogi, “Efficient detection of distributed constraint violations”, in ICDE 2007.
- S. R. Kashyap, J. Ramamirtham, R. Rastogi, and P. Shukla, “Efficient constraint monitoring using adaptive thresholds”, in ICDE 2008.
- T. Li and S. Ma, “Mining temporal patterns without predefined time windows”, in ICDM 2004.
- U. Manber and E. W. Myers, “Suffix arrays: A new method for on-line string searches”, in SIAM J. Comput., 22(5):935-948, 1993.
- W. Xu, L. Huang, A. Fox, D. A. Patterson, and M. I. Jordan, “Mining console logs for large-scale system problem detection”, in SysML, 2008.

Recent Publications (11 full conference papers, 1 short paper, 2 journal papers)

2013

- **Liang Tang**, Romer Rosales, Ajit Singh, Deepak Agarwal. "[Automatic Ad Format Selection via Contextual Bandits](#)", in *Proceedings of the 22th ACM Conference on Information and Knowledge Management (CIKM 2013)*, San Francisco, USA, Dec. 2013. (full paper, invited paper).
- **Liang Tang**, Tao Li, Shu-Ching Chen, Shuzhi Zhu. "[Searching Similar Segments over Textual Event Sequences](#)", in *Proceedings of the 22th ACM Conference on Information and Knowledge Management (CIKM 2013)*, San Francisco, USA, Dec. 2013. (full paper, acceptance rate: 143/848=16.8%).
- Li Zheng, Chao Shen, **Liang Tang**, Chunqiu Zeng, Tao Li, Steve Luis, Shu-Ching Chen. "[Data Mining Meets the Needs of Disaster Information Management](#)", *IEEE Transactions on Human-Machine Systems*, 2013, to appear.
- **Liang Tang**, Tao Li, Larisa Schwartz, Genady Ya. Grabarnik. "[Identifying Missed Monitoring Alerts based on Unstructured Incident Tickets](#)", in *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, Zurich, Switzerland, Oct. 2013. (short paper), to appear.
- **Liang Tang**, Tao Li, Larisa Schwartz, Florian Pinel, Genady Ya. Grabarnik. "[An Integrated Framework for Optimizing Automatic Monitoring Systems in Large IT Infrastructures](#)", in *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'13)*, Chicago, USA, Aug. 2013. (Industrial/Government Track, full presentation).
- **Liang Tang**, Tao Li, Yexi Jiang, Zhiyuan Chen. "[Dynamic Query Forms for Database Queries](#)", *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2013.
- **Liang Tang**, Tao Li, Larisa Schwartz, Genady Grabarnik. "[Recommending Resolutions for Problems Identified by Monitoring](#)", in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM'2013)*, 2013 (regular technical track, acceptance rate: 27.0%) .

2012

- Li Zheng, Chao Shen, **Liang Tang**, Chunqiu Zeng, Tao Li, Steve Luis, Shu-Ching Chen and Jainendra K. Navlakha. "[Disaster SitRep - A Vertical Search Engine and Information Analysis Tool in Disaster Management Domain](#)", in *Proceedings of the 13th IEEE International Conference on Information Integration and Reuse (IRI'12)*.
- **Liang Tang**, Tao Li, Larisa Schwartz. "[Discovering Lag Intervals for Temporal Dependencies](#)", in *Proceedings of the 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'12)*, Beijing, China, Aug. 2012. (research track, full presentation, acceptance rate: 133/755=17.6%)
- **Liang Tang**, Tao Li, Florian Pinel, Larisa Schwartz, Genady Grabarnik. "[Optimizing System Monitoring Configurations for Non-Actionable Alerts](#)", in *Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS'2012)*, 2012 (main technical paper, acceptance rate: 26.2%)

2011

- **Liang Tang**, Tao Li, Chang-Shing Perng. "[LogSig: Generating System Events from Raw Textual Logs](#)", in *Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM'11)*, 2011 (Full paper, acceptance rate: 15%)
- Li Zheng, Chao Shen, **Liang Tang**, Tao Li, Steve Luis, Shu-Ching Chen. "[Applying Data Mining Techniques to Address Disaster Information Management Challenges on Mobile Devices](#)", in *Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'11)*, 2011 (Industrial/Government Track, full presentation, acceptance rate: 8%)

2010

- **Liang Tang**, Tao Li. "[LogTree: A Framework for Generating System Events from Raw Textual Logs](#)", in *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM'10)*, 2010 (Full paper, acceptance rate: 9%)
- Li Zheng, Chao Shen, **Liang Tang**, Tao Li, Steve Luis, Shu-Ching Chen, Vagelis Hristidis. "[Using Data Mining Techniques to Address Critical Information Exchange Needs in Disaster Affected Public-Private Networks](#)", in *Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'10)*, 2010 (Industrial/Government Track, Full presentation, acceptance rate: 11%)