Using Syslog Message Sequences for Predicting Disk Failures

Wes Featherstun and Errin Fulp

Wake Forest University

Department of Computer Science

Winston-Salem, NC USA

November 11, 2010

HPC Trends and System Events

- Computing improvements achieved by adding more processors
  - IBM Blue Gene at LLNL has 212,992 processors
  - System failures will become more problematic
- As systems become larger, frequency of critical events will increase
  - Hardware failure, software failure, and user error
  - Lower overall system utilization
- Cannot easily improve failure rates; can we manage failures?
  - Minimize the impact of failures
  - Smarter scheduling of applications and services
- Accurate event predictions are key for event management
  - Are accurate predictions possible?
  - Need system status information to make predictions

System Status Information

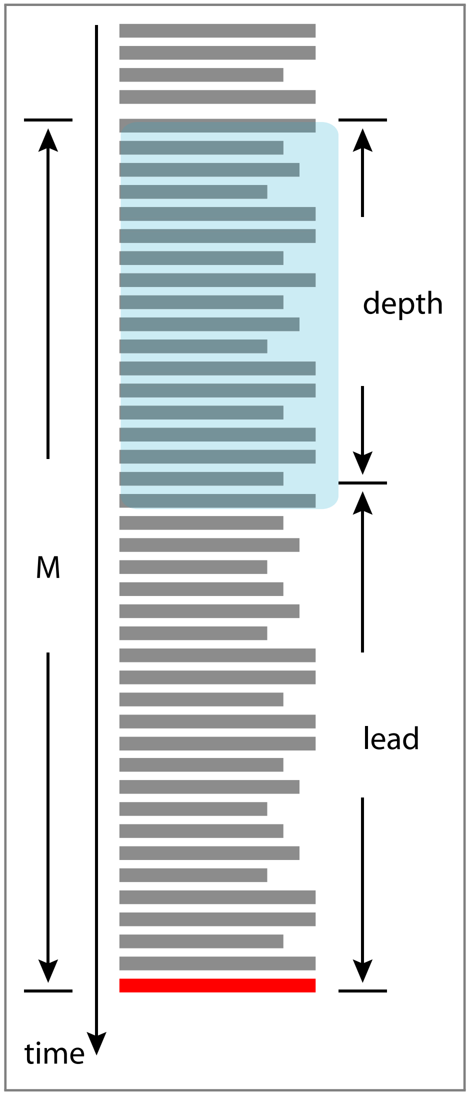
- *Almost* every computer maintains a system log file
  - Provide information about system events
- An event represents a change in *system state*
  - Include hardware failures, software failures, and security

Host	Facility	Level	Tag	Time	Message
198.129.8.6	kern	alert	1	1171062692	kernel raid5: Disk failure on sde1, disabling device

- Entries contain information such as: time, message, and tag
  - Time identifies when the message was recorded
  - Message describes the event, typically natural language
  - Tag represents criticality, low values are more important
- Trying to <u>predict</u> future events

Example System Event to Predict

- An interesting event is *disk failure*
  - By 2018 [large systems] could have 300 concurrent reconstructions at any time [SG07]
  - Predicting disk failure is important
  - *Easy to identify event in the log...*
- Predict failure as **early as possible**
  - Min depth d and max lead l

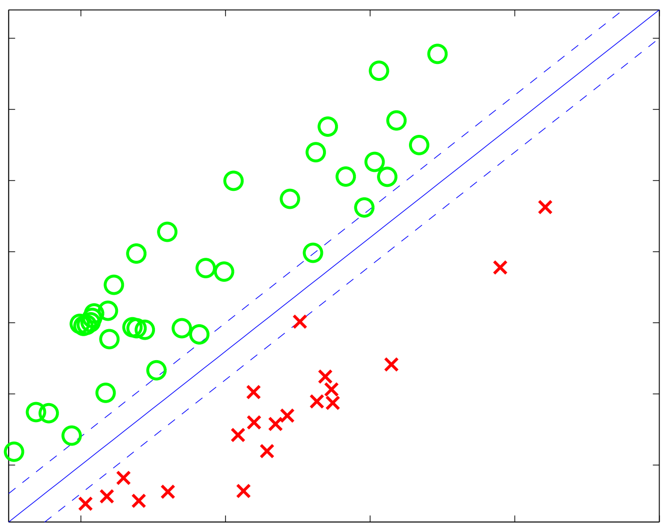


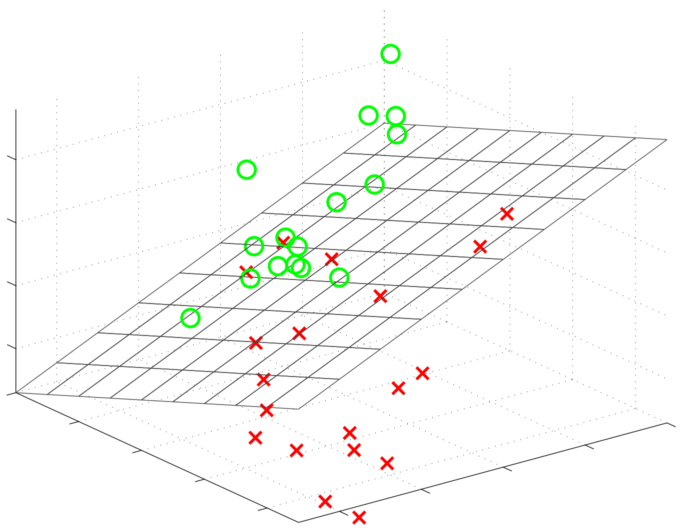
Previous Work

- Hammerly *et al.* used a naive Bayesian classifier to predict with 52% accuracy within 48 hours [HE01]
- IBM achieved over 80% accuracy, but with a specialized logging system [LZXS07]
- Broadwell achieved 100% accuracy in predicting SCSI cable failures, but the approach is not easily scalable [Bro02]
- Turnbull *et al.* used an approach similar to that in this presentation to predict system board failures [TA03]

Support Vector Machines

- Support Vector Machine (SVM) is a classification algorithm
  - Consider a set of samples from two different classes
  - Each vector consists of features describing the sample
  - SVM finds a hyperplane separating the classes in hyperspace
  - The vectors closest to the plane are the *support vectors*





Spectrum Kernel

- Assume two symbols $\{A, B\}$ and sequence length $k = 2$
  - There are 2^k possible sequences (features) (AA, AB, BA, BB)
  - Value of a feature is the number of occurrences

$M = \{A, A, B, A, A, B, B, A\}$
AA: 2
AB: 2
BA: 2
BB: 1

  - The spectrum kernel uses a *sliding window* to create sequences
  - There are b^k possible sequences, where b is number of symbols
- Used to provide *context* to each item
- *How does this work for syslog messages?*

`tag` Sequences

- Each message has a `tag` that indicates criticality
  - Sequence of messages represented by sequence of `tag` values
  - Need to reduce number of symbols, assume three levels
  - high ($\texttt{tag} < 10$), medium ($10 < \texttt{tag} < 140$), low ($\texttt{tag} > 140$)
- Given a series of messages M, process using a *sliding window*
  - Count the number of occurrences of k-length sequences

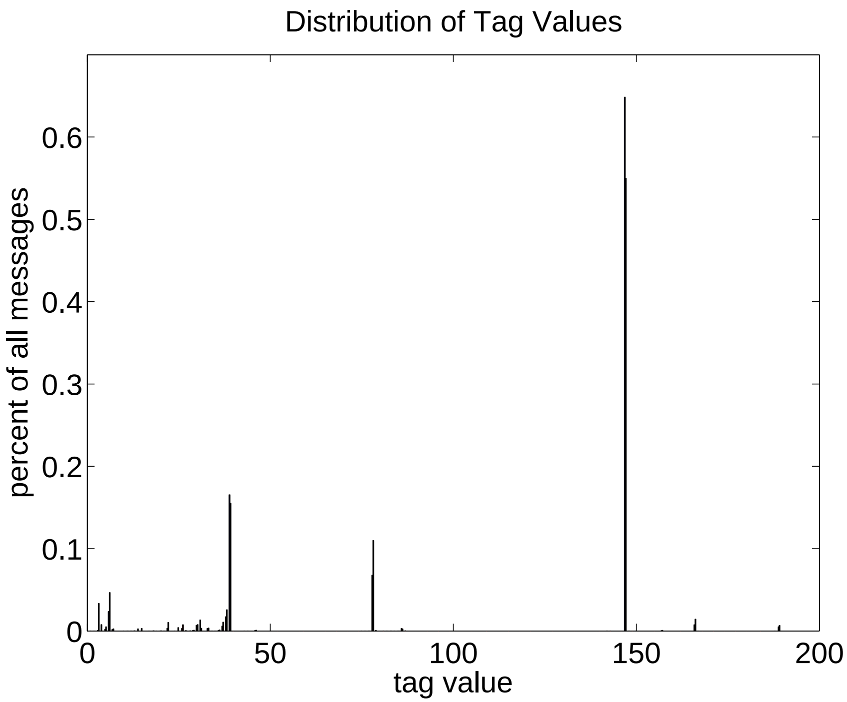
Example of Tag Sequences

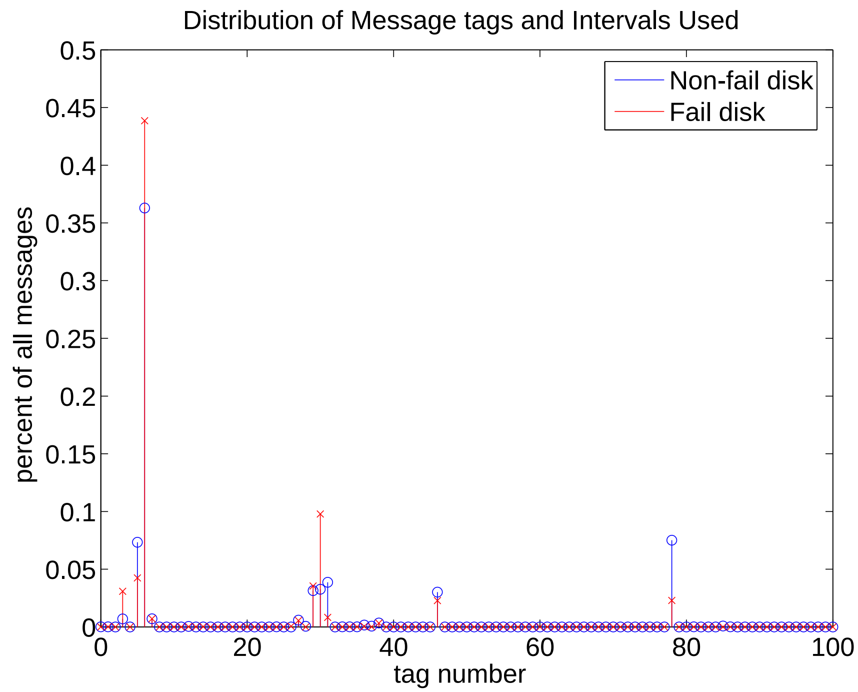
- Let M $= \{$148, 148, 158, 40, 5$\}$
- Assume $b = 3$ and $k = 3$, then $3^3 = 27$ possible features
  - Feature number is $f_{t+1} = \mod (b \cdot f_t, b^k) + e$
  - Vector for *M* would be (5:1, 141:1, 148:2, 158:1)

`tag`	**Encoding(e)**	**Sequence**	f**(base10)**
148	2	2	
148	2	22	
158	2	222	26
141	2	222	26
5	0	220	7

System Data Used for Experiments

- About 24 months of `syslog` file from 1024 node Linux cluster
  - Averaged 3.24 message an hour (78 a day) per machine
  - Observed 125 disk failure events
- Tag values ranged from 0 to 189
  - 61 unique `tag` values were observed during this time



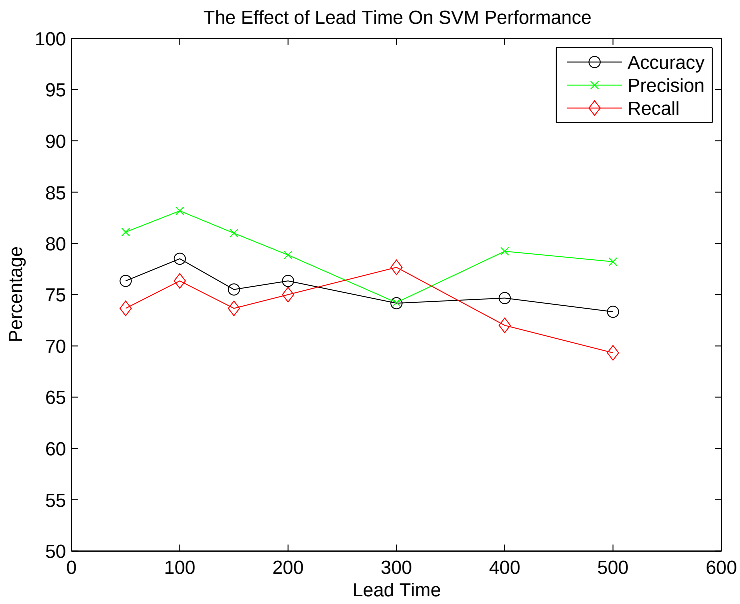


Experimental Setup Tags

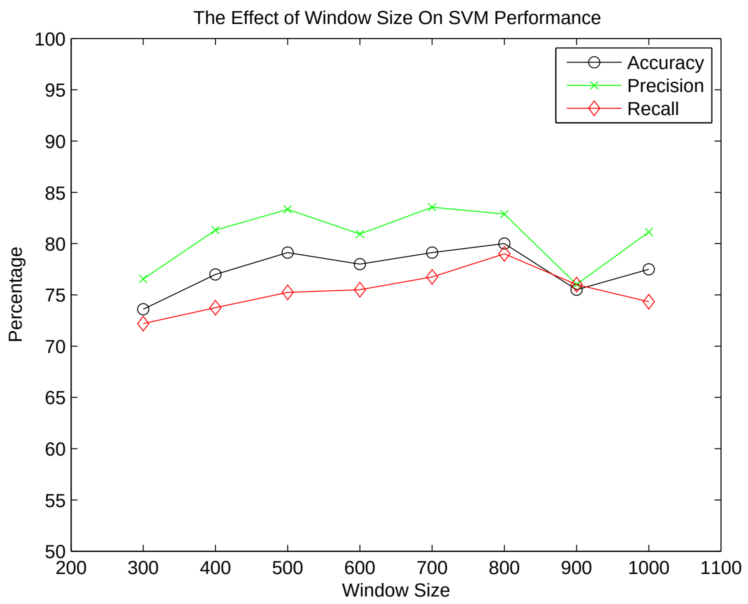
- Use an SVM based on tag count and sequence count
- Determine optimal lead time
- Determine optimal window size
- Use the window size and lead time results to find the best sequence length
- Analyze results using accuracy, precision, and recall
  - Accuracy: the number of correct classifications
  - Precision: how many predicted failures actually occurred?
  - Recall: of all failures, how many were predicted?

Tag Results

- Lead time best at about 100 messages
- Optimal window size is about 800 messages



(a) The accuracy, precision and recall of tag-based methods as lead time changes



(b) The accuracy, precision, and recall of tag-based methods as the window size changes

Changing Sequence Length

- Increasing the sequence length adds more contextual information
  - Do longer sequences improve effectiveness?
  - Longer sequences exponentially increase feature space...
- Experiments performed with window size of 800 messages and 100 messages of lead time

Sequence Length	Accuracy	Precision	Recall
3	73.166	74.9003	75.0011
4	75.6666	80.8341	72.6681
5	79.9993	82.8838	79.0012
6	79.4994	80.5503	80.6674
7	80.999	85.4837	78.668
8	78.4992	85.7335	73.3339

Table: A comparison of sequence lengths

- $k = 5$ provides best balance of speed and effectiveness

Timing Information

- Perhaps a change in message frequency is an indicator of imminent failure
- Keep track of the time (in seconds) during which each k-length sequence arrives

Feature Space	Accuracy	Precision	Recall
Sequences Using Tags	79.9993	82.8838	79.0012
Sequences Using Tags and Time	77.8329	82.2338	71.667

Table: Comparing performance between features using only tags and features including time information using sequences of length 5

Feature Space	Accuracy	Precision	Recall
Sequences Using Tags	80.999	85.4837	78.668
Sequences Using Tags and Time	81.1661	86.9337	76.005

Table: Comparing performance between features using only tags and features including time information using sequences of length 7

Keyword Motivation

- Tag numbers aren't always available
  - Instead, try to classify on *message* field
- Create a dictionary of "words" (any space delineated string)
  - Alphabet is far too large
- Created dictionaries of "keywords" (originally 52, reduced to 24)
- Convert each keyword to a number
- Use sliding window of 2+ words to create word sequences

`keyword`	**Encoding(e)**	**Sequence**	f
`disk`	0	0	
`hpc`	1	01	
`error`	2	012	5
`lustre`	3	0123	18

Keywords Results

- The 54 keyword dictionary contains specific node names
  - Does the SVM just train on nodes which tend to fail?
- Create a reduced 24-keyword dictionary
  - All identifiers of a certain type are assigned to the same number
  - Training on general categories instead of specific nodes, IPs, etc

Dictionary	Accuracy	Precision	Recall
54	77.6661	81.8171	76.0008
24	77.6659	79.1004	78.6676

Table: A comparison of keystring dictionaries

Sequence Length	Accuracy	Precision	Recall
3	77.6659	79.1004	78.6676
4	79.4996	82.9838	80.6676
5	82.1428	85.0008	80.9543

Table: Performance as sequence length increases

Keywords with Timing Information

- Timing information hurt performance for tag based methods
  - Since keywords ignore message bounds, might timing info add useful context?

Experiment	Accuracy	Precision	Recall
Without Time Info	79.4996	82.9838	80.6676
With Time Info	80.1657	85.567	78.6679

Table: A comparison of the 24-keystring dictionary with and without the addition of time information

- $k = 4$, despite $k = 5$ performing better without timing information
  - Training time for $k = 5$ with timing information can be massive...

Combination Features

- Both the tag and keyword approaches work well in isolation
  - Can the effectiveness of predictions be increased by combining methods?

Approach	Accuracy	Precision	Recall
Tags Without Time	80.999	85.4837	78.668
Keystrings With Time	80.1657	85.567	78.6679
Combination Without Time	77.9995	82.317	74.334
Combination With Time	80.6664	88.567	74.6673

Table: A comparison of tag based, keystring based, and combination methods

- Combination approach has fewer false positives, but also fewer true positives

Conclusions and Future Work

- Best method depends on priorities
  - High recall: tags without time or 24-keywords with time
  - High precision: combining tags, keywords, and timing information
- Several areas for improvement
  - Try different classification algorithms
  - Different data sets
  - Can this approach be used for other failure events?
- Questions? Interested? Email:
  - Dr. Errin Fulp: fulp@wfu.edu
  - Wes Feathesrtun: wes.featherstun@gmail.com

Further Reading

Peter Broadwell.
Component failure prediction using supervised naive bayesian classification, December 2002.
Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi= 10.1.1.3.4641. Accessed on April 19, 2010.

Greg Hamerly and Charles Elkan.
Bayesian approaches to failure prediction for disk drives.
In Proceedings of the Eighteenth International Conference on Machine Learning (ICLM), June 2001.

Yinglung Liang, Yanyong Zhang, Hui Xiong, and Ramendra Sahoo.
Failure prediction in ibm bluegene/l event logs.
In Proceedings of the Sevent IEEE International Conference on Data Mining, 2007.

Bianca Schroeder and Garth A Gibson.
Understanding failures in petascale computers.
Journal of Physics: Conference Series, (28), 2007.

Doug Turnbull and Neil Alldrin.
Failure prediction in hardware systems, 2003.
Available at: http://www.cs.ucsd.edu/ dturnbul/Papers/ServerPrediction.pdf. Accessed on: April 19, 2010.