

# System Behavior Analysis by Machine Learning

## CSC456 OS Survey

Yuncheng Li  
raingomm@gmail.com

December 6, 2012

# Table of contents

- 1 Motivation and Background
  - Motivation
  - Background
- 2 Analysis Examples
  - Windows Error Reporting System[XHF<sup>+</sup>09]
  - Web server performance metric[BSP<sup>+</sup>08]
  - Console logs with source code[DB08]
- 3 Conclusion
- 4 References

# Table of Contents

- 1 Motivation and Background
  - Motivation
  - Background
- 2 Analysis Examples
  - Windows Error Reporting System[XHF<sup>+</sup>09]
  - Web server performance metric[BSP<sup>+</sup>08]
  - Console logs with source code[DB08]
- 3 Conclusion
- 4 References

# Scenarios

- Debugging and improving Operating Systems
- Dynamic resource allocation
- Data mining the console logs

# Challenges

- Badly organized data
- Big Data
- Noisy Data

# Methodology

- Data driven, rather than problem specific
- Pattern Recognition and Data Mining

# Machine Learning Components

- **Raw Data**, which are usually not well structured, such as console logs from arbitrary program
- **Feature Engineering**, refining raw data by removing unnecessary information and transforming data representation into a form that could be used by general data model
- **General Data Model**, such as Tree Based Model (CART), Principal Components Analysis (PCA), Logistic Model . . .

# Table of Contents

- 1 Motivation and Background
  - Motivation
  - Background
- 2 Analysis Examples
  - Windows Error Reporting System[XHF<sup>+</sup>09]
  - Web server performance metric[BSP<sup>+</sup>08]
  - Console logs with source code[DB08]
- 3 Conclusion
- 4 References



## Data Source

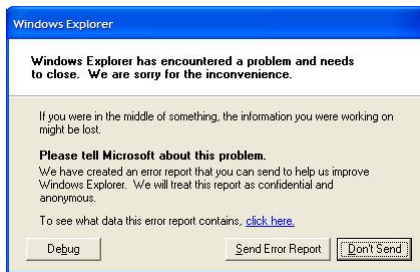


Figure: Windows Error Reporting Interface

## Raw Data

- Failure Type, such as segmentation fault and deadlocks
- Process Name
- Exception code, such as 0x c0 00 00 00 05
- Offending callstack, such as the output of *backtrace* in *gdb*

## Callstack Example

<b>Module</b>	<b>Function</b>	<b>Offset</b>
kernel32	ByteCallback	0x3
kernel32	WideCharExpand	0x2
kernel32	MultiByteToWideChar	0x9
A3DHF8Q	---	0x3820523
A3DHF8Q	---	0x3723952
A3DHF8Q	---	0x3945323
kernel32	ProcUserText	0x4
user32	TextDecode	0x4096
user32	ReadDialog	0x4096
user32	OpenDialog	0x4096
ntdll	RtlThreadStart	0x0
ntdll	RtlInitThreadThunk	0x0

**Figure:** An example of Callstack in Windows Operating System

## Features

- Indicator Function for *Event Type, Process Name and Exception Code*
- Extended Minimum Editing Distance for pairwise Callstack comparison ( $O(\min(m, n))$  space complexity)
  - Editing: insertions, deletions or substitution
  - Editing Target: letters, frames and callstack groups

Each **pair** of raw data (*failure type, process name, exception code ...*) will generate a **similarity vector**.

# Model

Logistic Model  $h(x) = \frac{1}{1+e^{-\beta \cdot x}}$  to predict whether two failures are equal based on the similarity vector.

- Full Model: No constraints on the  $\beta$
- Reduced Model: Enforce some constraints on the penalty coefficients  $\beta$

## Results – Parameters Learning

Param.	Model Estimates				
	M1	M2	M3	M4	
$\alpha$	3.45	5.04	2.99	2.89	
$\beta_{ET}$	1.36	1.46	2.43	2.23	
$\beta_{PN}$	1.18	1.19	3.56	3.30	
$\beta_{EC}$	2.14	1.82	2.19	1.71	
$\beta_{CS}$	-6.99	-6.57	-7.21	-6.74	
$\gamma_{InsSame}$	0.72	1.22	0.94	1.00	
$\gamma_{InsNew}$	1.48				
$\gamma_{DelSame}$	0.56	1.13	0.94		
$\gamma_{DelLast}$	1.54				
$\gamma_{SubMod}$	2.44	1.17			
$\gamma_{SubFunc}$	0.25				
$\gamma_{SubOffset}$	0.00				
<b>LL</b>	<b>-149.6</b>	<b>-155.6</b>	<b>-191.0</b>		<b>-197.5</b>

Figure: Learned Parameters

## Results – Similarity Prediction Performance

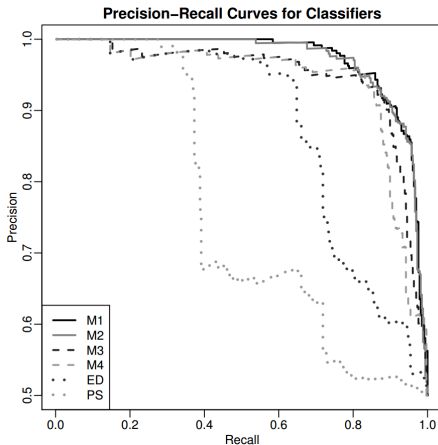


Figure: Precision and Recall Performance

# Data Source

Monitor Edit View Help

System Processes Resources File Systems

Load averages for the last 1, 5, 15 minutes: 0.00, 0.00, 0.00

Process Name	Status	% CPU	Nice	ID	Memory	Waiting Channel	Session
abrt-applet	Sleeping	0	0	2455	1.2 MiB	poll_schedule_time	
abrt-d	Sleeping	0	0	1984	2.1 MiB	poll_schedule_time	
acpid	Sleeping	0	0	1721	116.0 KiB	poll_schedule_time	
aio/0	Sleeping	0	0	53	N/A	worker_thread	
aio/1	Sleeping	0	0	54	N/A	worker_thread	
aio/2	Sleeping	0	0	55	N/A	worker_thread	
aio/3	Sleeping	0	0	56	N/A	worker_thread	
async/mgr	Sleeping	0	0	22	N/A	async_manager_thr	
ata/0	Sleeping	0	0	37	N/A	worker_thread	
ata/1	Sleeping	0	0	38	N/A	worker_thread	
ata/2	Sleeping	0	0	39	N/A	worker_thread	

End Process

Figure: Gnome System Performance Monitor



# Features

Time series of the following metrics

- CPU utilization
- number of disk IO
- network traffic
- ...

## Analysis Techniques

- *Clustering* for identifying distinct system states. (Kmeans or Tree Based Model)
- *Correlation Analysis* to find which set of attributes always comes with failures, which is an indication of failure causes
- *Baseline analysis* and detect anomaly that far from baseline

## Data Source

```
295
audit(1216470015.968:3): policy loaded auid=4294967295 ses=4294967295
INIT: version 2.86 booting
      Welcome to Red Hat Enterprise Linux Server
      Press 'I' to enter interactive startup.
Setting clock (utc): Sat Jul 19 05:20:22 MST 2008      [ OK ]
Starting udev:                                       [ OK ]
Loading default keymap (us):                        [ OK ]
Setting hostname rhce-prep.example.com:             [ OK ]
No devices found
Setting up Logical Volume Management:
  No volume groups found
                                                    [ OK ]

Checking filesystems
/: clean, 4871/263232 files, 72321/263056 blocks
/home: clean, 117/130560 files, 27384/522000 blocks
/var: clean, 1165/130560 files, 65117/522000 blocks
/dev/md0: clean, 12/883872 files, 45604/883456 blocks
/usr: clean, 81733/524288 files, 427747/524120 blocks
/boot: clean, 33/66264 files, 24068/265040 blocks
                                                    [ OK ]

Remounting root filesystem in read-write mode:     [ OK ]
Mounting local filesystems:                         [ OK ]
Enabling local filesystem quotas:                   [ OK ]
```

Figure: Linux Booting Logs

## Features Extraction

- Message types and message variable, such as *sent\_packet 196.168.3.1*
- Identify *significant* message types (Have a lot of *variations*)
- Grouping message values, by clustering in the feature space
- Generate *bag of words* feature, which is a histogram of message groups

# PCA-Based Anomaly Detection

- PCA is short for Principal Components Analysis
- Certain feature instance is identified as *anomaly* if the *reconstruction error* by principal components is higher than a threshold

# Results

Table 1: Detection Precision

Anomalous Event	Events	Detected
Empty packet	1	1
Failed at the beginning, no block written	20	20
WriteBlock received java.io.IOException	39	38
After delete, namenode not updated	3	3
Written block belongs to no file	3	3
PendingReplicationMonitor timed out	15	1
Redundant addStoredBlock request	12	1
Replicate then immediately delete	6	2

Table 2: False Positives

False Positive Type	False Alarm
Over replicating (actually due to client request)	11
Super hot block	1
Unknown reasons	2

Figure: Manual Labeled and Detection Results comparison

# Table of Contents

- 1 Motivation and Background
  - Motivation
  - Background
- 2 Analysis Examples
  - Windows Error Reporting System[XHF<sup>+</sup>09]
  - Web server performance metric[BSP<sup>+</sup>08]
  - Console logs with source code[DB08]
- 3 Conclusion
- 4 References

# Conclusion

- Unstructured data calls for good feature engineering
- Feature engineering is really the key
- Data analysis and visualization techniques are just routines



# Table of Contents

- 1 Motivation and Background
  - Motivation
  - Background
- 2 Analysis Examples
  - Windows Error Reporting System[XHF<sup>+</sup>09]
  - Web server performance metric[BSP<sup>+</sup>08]
  - Console logs with source code[DB08]
- 3 Conclusion
- 4 References

# Bibliography I



Kevin Bartz, Jack W. Stokes, John C. Platt, Ryan Kivett, David Grant, Silviu Calinoiu, and Gretchen Loihle.

Finding similar failures using callstack similarity.

*In Proceedings of the Third conference on Tackling computer systems problems with machine learning techniques, SysML'08, pages 1–1, Berkeley, CA, USA, 2008. USENIX Association.*



Songyun Duan and Shivnath Babu.

Empirical comparison of techniques for automated failure diagnosis.

*In Proceedings of the Third conference on Tackling computer systems problems with machine learning techniques, SysML'08, pages 2–2, Berkeley, CA, USA, 2008. USENIX Association.*

## Bibliography II



Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I. Jordan.

Detecting large-scale system problems by mining console logs.  
In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, SOSP '09, pages 117–132, New York, NY, USA, 2009. ACM.