System Behavior Analysis by Machine Learning

CSC456 OS Survey

Yuncheng Li
raingomm@gmail.com

December 6, 2012

Table of contents

Table of Contents

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

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

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

- Full Model: No constraints on the β
- Reduced Model: Enforce some constraints on the penalty coefficients β

Results – Parameters Learning

Param.	Model Estimates M1	M2	M3	M4
α	3.45	5.04	2.99	2.89
β_{ET}	1.36	1.46	2.43	2.23
β_{PN}	1.18	1.19	3.56	3.30
β_{EC}	2.14	1.82	2.19	1.71
β_{CS}	-6.99	-6.57	-7.21	-6.74
$\gamma_{\mathrm{InsSame}}$	0.72	1.22	0.94	1.00
γ_{InsNew}	1.48			
$\gamma_{\mathrm{DelSame}}$	0.56	1.13	0.94	
$\gamma_{\mathrm{DelLast}}$	1.54			
γ_{SubMod}	2.44	1.89	1.17	
$\gamma_{\mathrm{SubFunc}}$	0.25	0.40		
$\gamma_{\mathrm{SubOffset}}$	0.00	0.00		
LL	**-149.6**	**-155.6**	**-191.0**	**-197.5**

Figure: Learned Parameters

Results – Similarity Prediction Performance

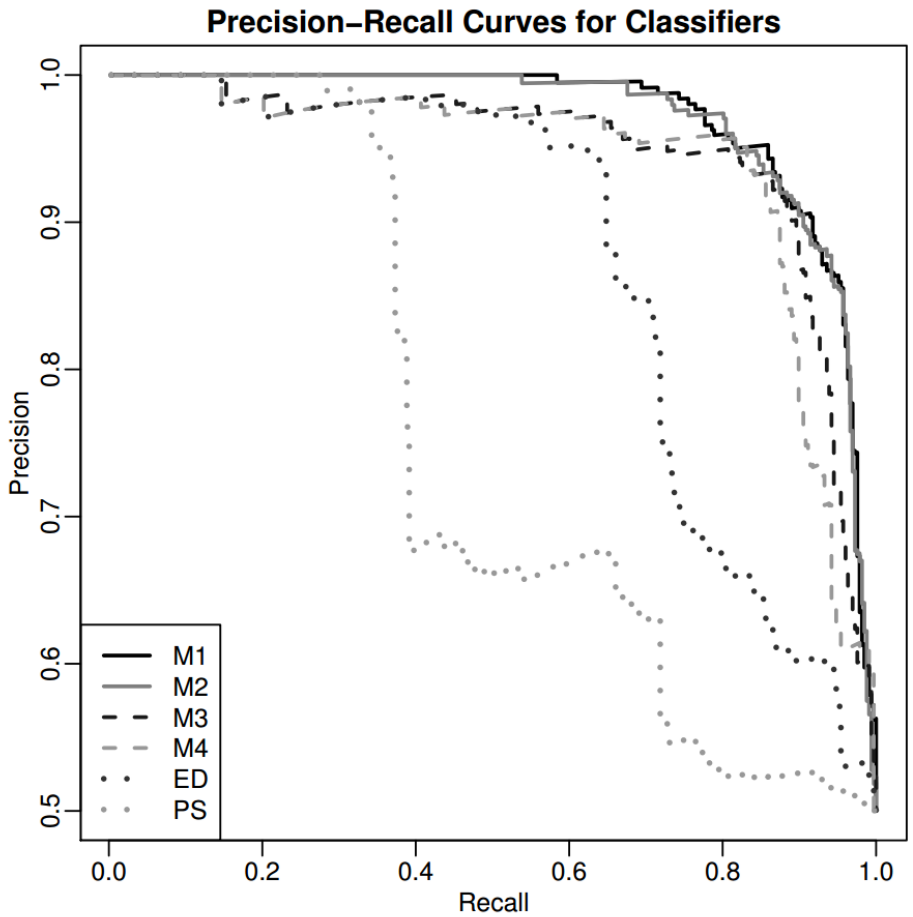

Figure: Precision and Recall Performance

Data Source

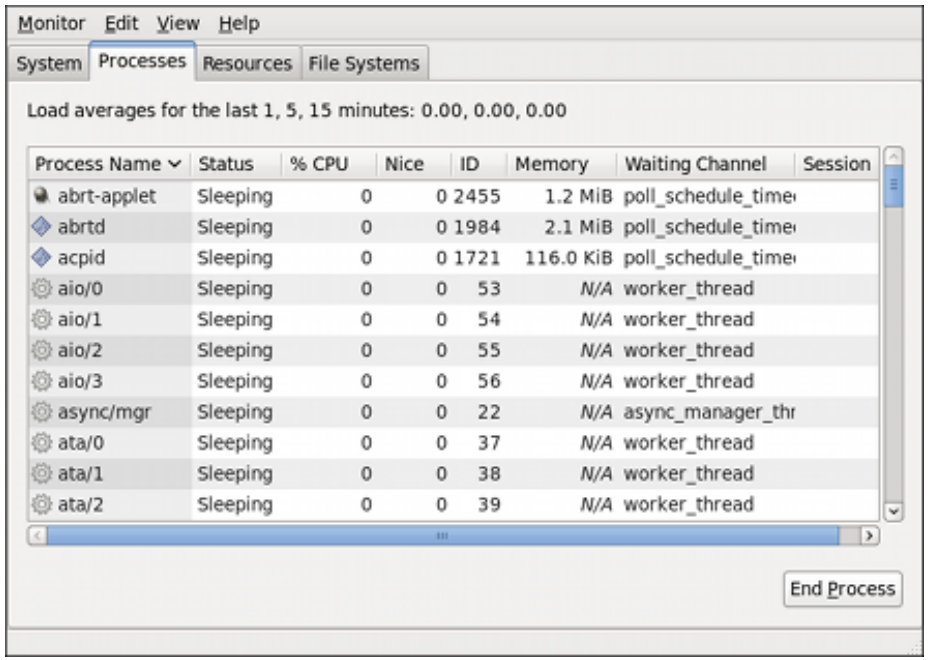

Figure: Gnome System Performance Monitor

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

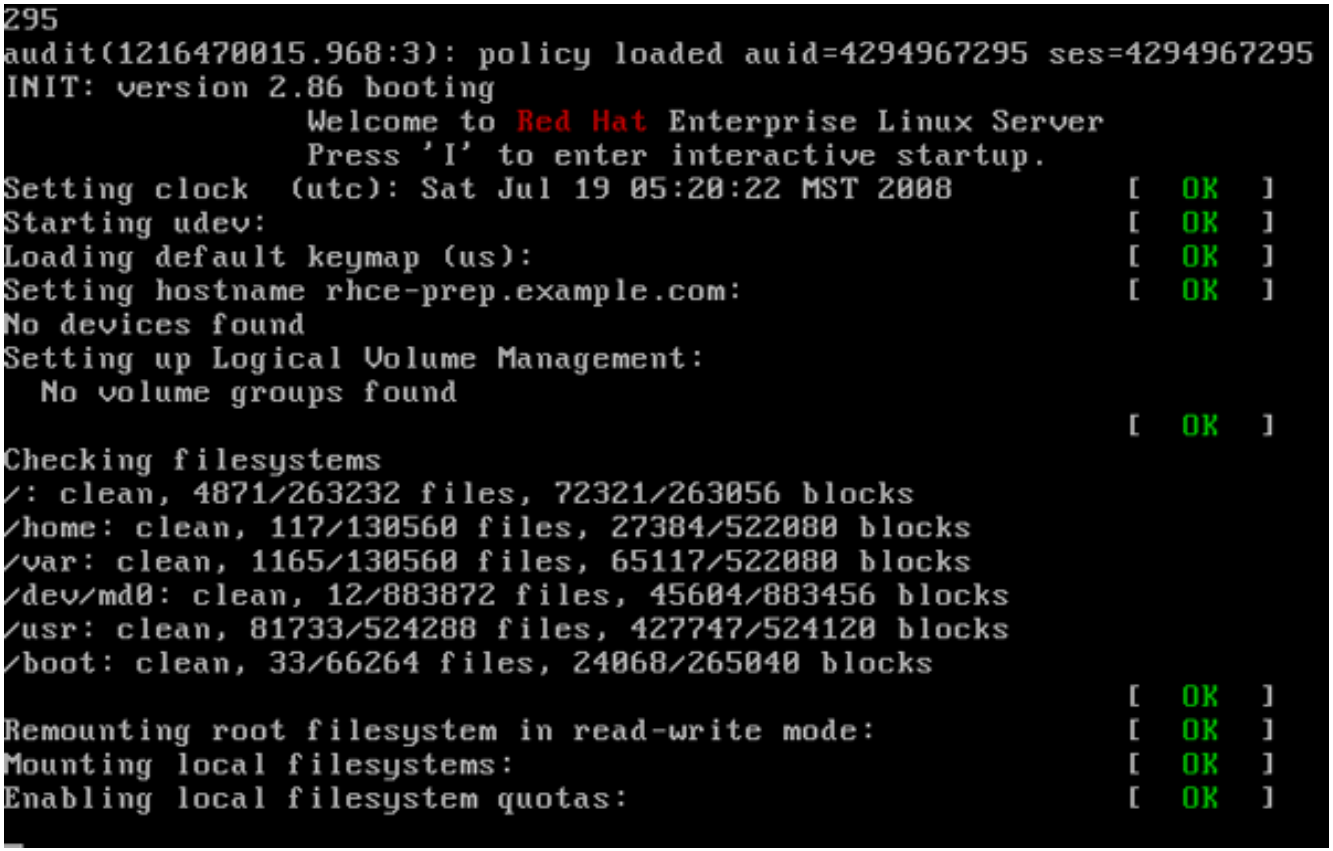

Figure: Linux Booting Logs

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

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

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.