Data-Driven Device Failure Prediction

Thesis Defense

Paul L. Jordan

Thesis Advisor
G.L. Peterson

Committee Members
M.J. Mendenhall, A.C. Lin, and A.J. Sellers

18 August 2016

Overview

Problem

- Computer systems fail, not often, but could have devastating consequences
- Redundancy can help, but it is expensive, complex, and only masks the root cause
- Survey paper of techniques for predicting failure using machine learning [1]
- Difficulty is in finding or obtaining training data

Solution?

- Adaptive Failure Prediction (AFP) framework [2]
- AFP wasn't capable of running on modern operating system
- AFP didn't exhaustively emulate all possible/realistic faults [3]

Solution

- This work presents an extended AFP
- New realistic fault loads:
  - Memory Corruption
  - CPU Limitation
  - Memory Limitation (due to leak)
- Modernized fault injection tool: Windows Software Fault Injection Tool (W-SWFIT)
- Workload Generator: Distributed PowerShell Load Generator (D-PLG) [4]

Related Work

Proactive Fault Management (PFM)

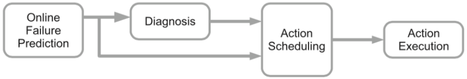


Figure: The stages of proactive fault management [1].

- **Online Failure Prediction (OFP)**
- Diagnosis
- Action Scheduling
- Action Execution

Online Failure Prediction (OFP)

- Act of evaluating a running system in real time to make a prediction about whether a failure in a future state is imminent [1]

Online Failure Prediction (OFP)

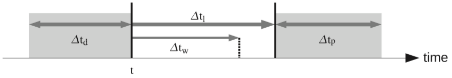


Figure: The timeline for OFP [1].

- Present Time: t
- Data Window: Δt_d, represents the time window of data used for a predictor to make its assessment.
- Lead Time: Δt_l, represents the time between when failure is predicted and when that failure will occur.
- Minimal Warning Time: Δt_w, is the amount of time required to avoid a failure if one is predicted.
- Prediction Period: Δt_p, is the time for which a prediction is valid.

Faults, Errors, and Failures

- Failure: Delivered service deviates from correct service
- Error: The point when things go wrong (Detected vs. Undetected)
- Fault: Hypothesized root cause of an error

Faults, Errors, and Failures

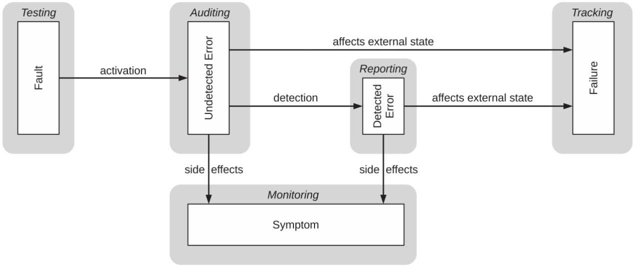


Figure: How faults and errors evolve into failure with the associated methods for detection represented by enclosing gray boxes [1].

Base Adaptive Failure Prediction (AFP)

- Virtually clones target system
- Generates realistic workload for target to accomplish
- Synthesizes realistic faults that lead to failure
- Capture data from system to train failure prediction model

Base Adaptive Failure Prediction (AFP)

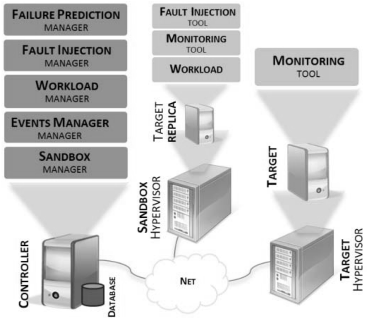


Figure: The AFP framework [2].

Methodology

Adaptive Failure Prediction (AFP) Implementation

Table: Hypervisor 1 configuration (sandbox/target).

Qty.	Role	Operating System	CPU / Mem.
1	DC	Win. Server 2008 R2	2 / 2 GB
1	Web	Win. Server 2008 R2	2 / 2 GB
5	Client	Win. 7	1 / 512 MB

Table: Hypervisor 2 configuration (controller).

Qty.	Role	Operating System	CPU / Mem.
1	RDP	Win. Server 2008 R2	1 / 4 GB
1	Log	Ubuntu 14.04 LTS	1 / 1 GB

Adaptive Failure Prediction (AFP) Implementation

- Workload Generator: D-PLG with five client machines
- Fault-Load Generator:
  - W-SWFIT
  - Memory Corruption
  - CPU Limitation
  - Memory Limitation (due to leak)
- Events Manager: rsyslog server with SolarWinds syslog forwarder
- Prediction Model: Support Vector Machine (SVM) and boosted decision trees in *R*

Experimental Results

Results

- Four fault loads tested on a Windows Domain Controller (DC) and an Apache web server:
  - W-SWFIT
  - Memory Corruption
  - CPU Limitation
  - Memory Limitation (due to leak)

Fault Injection

- Target process crashes immediately
- DC: restarts the computer
- Apache: starts a new child server process or parent process halts
- No indicators to use to train machine learning algorithm

Under-Resource CPU

- Third party application consumed all CPU time
- Virtual Machine (VM) resources were reduced
- Results
  - Both cases resulted in same behavior
  - Slower response times for both the DC and Apache web server
  - Target process would not fail

Memory Corruption

- Different from fault-injection in that it corrupts heap-space instead of program memory
- DC: corrupted the user database
- Apache: corrupted web content
- Results
  - Same as fault injection: either would not fail, or would crash immediately with no warning signs

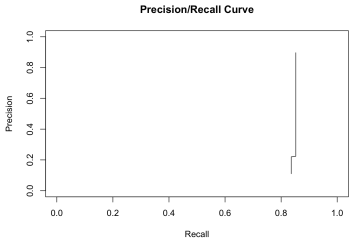
Memory Leak

- Third party application consumed all available memory
- Only fault load that caused failure with indicators present in log messages prior to failure
- Trained two statistical models (SVM, and Boosted Decision Trees)
- As expected, both predictors performed adequately before software update, then poorly after
- After re-training with newly generated data performance once again was adequate

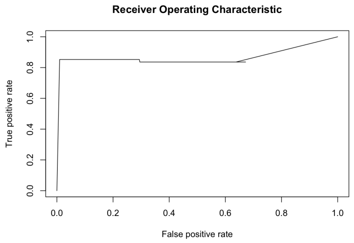
Memory Leak

- What is adequate?
  - Naïve predictor predicts non-failure prone at all times
  - Currently no form of prediction is taking place in operational environment
  - Machine learning classification algorithms evaluated using confusion matrix at best F-Measure, Receiver Operating Characteristic (ROC), Area Under the Curve (AUC), and Precision/Recall Curves [1, 5]

Support Vector Machine (SVM) Performance



(a) Precision/Recall Curve.



(b) ROC Curve (AUC = 0.8664).

Figure: Test data performance of the SVM prediction method on failure data obtained by consuming all available memory until target application fails.

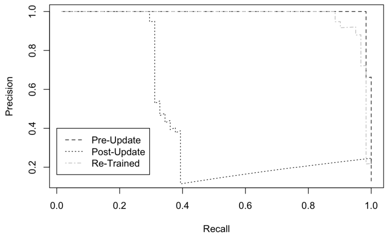
- Before software update, AUC: 0.8664
- After re-training, AUC: N/A (highest F-Measure: 0.4380)

Support Vector Machine (SVM) Performance

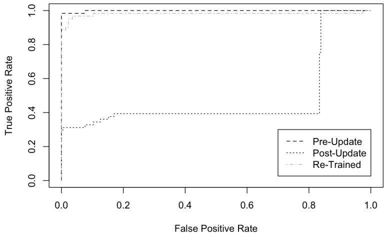
Table: Confusion matrix on test data created before software updates on threshold with highest F-Measure (0.8739) using SVM.

		Actual	
		Fail	**No-Fail**
Predicted	**Fail**	52	6
	No-Fail	9	607

Boosted Decision Tree Performance



(a) Precision/Recall Curves.



(b) ROC Curves.

Figure: Performance of the boosting prediction method on data generated by consuming all available memory until target application fails.

- Before software update, AUC: 0.9984
- After software update, AUC: 0.4854
- After re-training, AUC: 0.9801

Boosted Decision Tree Performance I

Table: Confusion matrix on test data created before software updates on threshold with highest F-Measure (0.9917) using boosting.

		Actual	
		Fail	No-Fail
Predicted	Fail	60	0
	No-Fail	1	412

Boosted Decision Tree Performance II

Table: Post-update failure data confusion matrix on threshold with highest F-Measure (0.4691) using model trained on failure data generated before software update.

		Actual	
		Fail	**No-Fail**
Predicted	**Fail**	19	1
	No-Fail	42	222

Boosted Decision Tree Performance III

Table: Post-update failure data confusion matrix on threshold with highest F-Measure (0.9355) using model trained on failure data generated after software update.

		Actual	
		Fail	**No-Fail**
Predicted	**Fail**	58	5
	No-Fail	3	218

Conclusions & Future Work

Conclusions

- Extended AFP presented can predict realistic failure in production systems
- Capable of adapting to underlying system changes
- Vulnerability to certain types of failure can come and go, consequently, all fault loads should be used

Future Work

- Further explore how best to implement fault injection and its true impact
- Integrate real failure data
- Further validation and automation
- Implement and make operational

Summary

Questions?

References I

[1] F. Salfner, M. Lenk, and M. Malek, “A survey of online failure prediction methods,” *ACM Computing Surveys (CSUR)*, vol. 42, no. 3, 2010.

[2] I. Irrera, M. Vieira, and J. Duraes, “Adaptive failure prediction for computer systems: A framework and a case study,” in *Proceedings of the 2015 IEEE 16th International Symposium on High Assurance Systems Engineering (HASE 2015)*, pp. 142–149, 2015.

[3] N. Kikuchi, T. Yoshimura, R. Sakuma, and K. Kono, “Do injected faults cause real failures? a case study of linux,” in *Proceedings of the 25th IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW 2014)*, pp. 174–179, 2014.

References II

[4] P. Jordan, C. Van Patten, G. Peterson, and A. Sellers, "Distributed powershell load generator (D-PLG): A new tool for dynamically generating network traffic," in *Proceedings of the 6th International Conference on Simulation and Modeling Methodologies, Technologies, and Applications (SIMULTECH 2016)*, pp. 195–202, July 2016.

[5] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.