Università della Svizzera Italiana

Master Thesis

Accelerator for Event-based Failure Prediction

Problem Statement

Author:
Simon Maurer

Supervisor:
Prof. Miroslaw Malek

March 10, 2014

Contents

1 Introduction

This document provides an outline of the planned work for the master thesis "Accelerator for Event-based Failure Prediction". The document addresses the problem considered by the thesis, a state of the art outline as well as important questions and hypothesis that arise or are needed for the thesis. Further the document provides a list of necessary theory bases and introduces the design and experimenting methods applied in the thesis. Finally, the document gives an overview of necessary material in terms of literature and equipment, shows the preliminary work progress and provides a broad time schedule.

2 Problem and Research Area

In today's live it becomes increasingly important, that computer systems are dependable. The reason being, that computer systems are used more and more in areas where the failure of such a system can lead to catastrophic events. Banking, public transportation and medical engineering are only a few examples of areas employing large and extremely complex systems. The increasing complexity of computer systems has a direct impact on their maintainability and testability. It is simply impossible to guarantee that a piece of software comes without any faults. On top of that, the same problematic arises with the hardware components which also may contain faulty parts but also get increasingly prone to failures due to decay of material.

In the event of a system failure it is of course desirable to fix the system as soon as possible in order to minimize the downtime of the system (maximize the availability). This can be accomplished by using different types of recovery techniques, e.g. Check-pointing (create checkpoints to roll back/forward), system replication (switch to a redundant system), fail over (reboot). All these techniques require a certain amount of time to complete the recovery process, time that is very expensive. In order to minimize this time, techniques have been developed to anticipate upcoming failures. Such a technique is described in [12].

The work presents a new algorithm to predict failures and compares the results with other techniques. The accuracy of the presented algorithm to predict failures proves to be better compared to the other techniques, has however the drawback of increased complexity and hence increased computation time. It is very important to keep the computation overhead very low in order to maximize the time between the prediction of a failure and the actual event of the failure. One way to decrease the computation time is to design a hardware accelerator for the prediction algorithm. The design of such an accelerator is outlined in this document.

The presented algorithm uses system information, e.g. log entries, to predict failures during runtime (online). To be able to interpret the log files, it is important that they are structured in a standardized format. Further it is necessary input real log files to the algorithm in order to verify its performance (in terms of precision and speed). Unfortunately it is not possible to use the same data already used in the reference work, because of confidentiality issues. For this reason, new test data have to be generated by respecting the data properties described in the reference work as well as respecting a log representation standard.

3 State of the Art Outline

This section provides a brief outline of the state of the art in the different fields of research that are relevant for the thesis. This includes a small overview of failure prediction methods, existing solutions to accelerate failure prediction algorithms and concepts of log standardization.

3.1 Failure Prediction

A very detailed overview of failure prediction methods is given in [13]. The techniques used as comparison in the main reference [12] as well as the described technique are listed below.

- Event-based Failure Prediction [12]
- Dispersion Frame Technique (DFT) [7, 6]
- Eventset [15]
- SVD-SVM [4]

Maybe some words about the counter ideas (SEER: a lightweight online failure prediction approach)

Discuss the arguments of Felix

Too many parameters to be identified, estimated and set
Considering an embedded system, this is usually not a problem because the parameters are defined during the design phase and will never be changed.

Limited performance scalability
...to be analyzed ...

Industry trends towards cloud, not single node prediction
In embedded systems it will still be beneficial to predict failures of single nodes.

3.2 Accelerator

The following list presents designs to accelerate algorithms related to machine learning and the computation of hidden Markov models:

- [2] presents an architecture for a lightweight Viterbi accelerator designed for a embedded processor datapath.
- [3] describes a FPGA based accelerator for the SVM-SMO (support vector machine - sequential minimal optimization) algorithm used in the domain of machine learning.
- [5, 8, 10] describes a FPGA based accelerator for protein sequence HHM search. The Viterbi algorithm is used.
- [16] describes i.a. an approach to accelerate the Viterbi algorithm from the HMMER library using GPUs.

3.3 Log Standardization

[14] describes an approach on how to standardize logs. Still missing are sources for:

- data generation in order to verify an algorithm
- metrics to measure log quality
- more on log standardization

4 Questions and Hypothesis

The following list is intended to give an overview of open questions:

- how to parallelize the algorithm
- what kind of accelerator (FPGA, GPU, just use multiple cores)
- how to generate test data to verify the accelerator
- ...

Specific questions concerning the offline algorithm:

- what type of kernels (for transition durations $D(t)$) and how many
- gradient of kernel partially derived by parameters: also sum over k
- how to compute the step size η (line search: $\eta_k = h(\eta)$, choice of $h(\eta)$)
- choice of bounds to stop iteration (step size, sequence likelihood)
- ...

Ideas on how to accelerate the online part of the algorith

- use high speed multiplier-accumulator (MAC) devices on a FPGA
- use MACs only on integer numbers and compute FP later manually
- minimize division (compute scaling factor once and then multiply)
- if precision allows use pipelining to precompute the factor $b(j, o[k]) * v(i, j, k)$
- ...

Possible optimizations of the algorithm:

- use a regularization term in the cost function to prevent overfitting
- incorporate the offline part of the algorithm into the online part in order to deal with model aging
- ...

The accelerator will be designed by taking the following hypotheses into account:

- Algorithm of Felix has been verified in his work
- the system using the accelerator provides standardized log events
- only the online part of the algorithm needs to be accelerated
- ...

5 Theory Base

5.1 Taxonomy of Dependable Computing

- Fault-Error-Failure chain [1]
- Precision and Recall [11]
- F-Measure [9]

5.2 Algorithm Description

This section provides an brief overview of the computational steps done by the proposed algorithm. The following description should provide a complete blueprint of the algorithm, that allows to implement it, but without any explications or proofs related to the formulation. To be able to understand the formal expression of the algorithm, first a definition of the used parameters is provided.

- N: number of states

- M: number of observation symbols
- L: observation sequence length
- R: number of cumulative probability distributions (kernels)

The delay of the event at time t_k with respect to the event at time t_{k-1} is described as

$$d_k = t_k - t_{k-1} \tag{1}$$

One part of the algorithm is the model training. The features to be trained are the following:

- π_i, forming the initial state probability vector $\boldsymbol{\pi}$ of size N
- $b_i(o_j)$, forming the emission probability matrix B of size $N \times M$
- p_{ij}, forming the matrix of limiting transmission probabilities P of size $N \times N$
- $\omega_{ij,r}$, the weights of the kernel r
- $\theta_{ij,r}$, the parameters of the kernel r

For simplification reasons, in a first step only one kernel is used. Due to this, the kernel weights can be ignored. Choosing the gaussian distribution results in the kernel parameters μ_{ij} forming Θ_μ and σ_{ij} forming Θ_σ.

$$\kappa_{ij,gauss}(d_k|\mu_{ij}, \sigma_{ij}) = \frac{1}{\sigma_{ij}\sqrt{2\pi}} \exp(-0.5\frac{(d_k - \mu_{ij})^2}{\sigma_{ij}^2}) \tag{2}$$

The training of the model is accomplished by the following steps.

1. initialize $\boldsymbol{\pi}$, B, P, μ and σ

$$d_{ij}(d_k) = \sum_{r=1}^{R} \omega_{ij,r}\kappa_{ij,r}(d_k|\theta_{ij,r}) \tag{3}$$

$$v_{ij}(d_k) = \begin{cases} p_{ij}d_{ij}(d_k) & \text{if } j \neq i \\ 1 - \sum\limits_{\substack{h=1 \\ h\neq i}}^{N} p_{ih}d_{ih}(d_k) & \text{if } j = i \end{cases} \tag{4}$$

$$\alpha_0(i) = \pi_i b_{s_i}(O_0) \tag{5}$$

$$\alpha_k(j) = \sum_{i=1}^{N} \alpha_{k-1}(i) v_{ij}(d_k) b_{s_j}(O_k); \quad 1 \leq k \leq L \tag{6}$$

Description of D(t), P, G, pi, kernel, weights, distribution function,

Data Processing
: Tupling and grouping of events...

Model Training
: Baum-Welch algorithm...

Sequence Processing
: Forward algorithm. Sequence likelihood.

Classification
: Bayes Decision.

6 Method, Design and Experiments

In a first step, the proposed algorithm will be implemented in Octave[1]. The main reason for choosing Octave is on one hand the simplicity to implement complex algorithms with only a few lines of code and on the other hand the immediate revelation of possible parallelization possibilities. This implementation is also meant to help to fully understand the algorithm and possibly provide some ideas about optimizations related to the design of an accelerator.

The next step will consist of generating synthetic data to verify the correctness of the implementation. This data will also be used to verify further implementations and to compute a speedup by comparing the accelerated and the non-accelerated implementation. Knowing this, it is important to generate data that can easily be imported into Octave but also been used as input stream for a C/C++ implementation.

Further, the online part (sequence processing, classification) of the algorithm will be implemented in C++. To be able to run and verify the algorithm, the training data computed with Octave will be used. It is possible that due to optimization decisions it will also be necessary to implement the off-line part of the

[1]https://www.gnu.org/software/octave

algorithm (training) in C++. In this case, the training data will be computed anew. To efficiently implement the algorithm in C++ a linear algebra library will be used (e.g. Armadillo[2], Eigen[3], LAPACK[4], etc.). It will be analyzed which library is best fitting for the purposes of this work. This sequential implementation will represent the reference to which the accelerated implementation will be compared to.

Finally, the accelerator will be designed. The design will include discussion concerning the choice of accelerator (e.g. FPGA, GPU, multiple cores). As a minimum approach, only the online part of the algorithm will be accelerated. However, due to optimization reasons it may be desirable to also accelerate the off-line part and alter the whole algorithm in such away as to compute the learning process online and hence being able to react to model aging problems. Again the generated synthetic data will be used to verify the algorithm as well as to compute a speedup compared to the non-accelerated implementation.

In addition to the steps listed above it would be very desirable to verify the accelerated algorithm with real data. To accomplish this, one idea is to set up the following experiment:

Run high-load procedures (e.g. prime number computation) on a undervolted CPU-core and observe the internal hardware counters of the CPU-core. Whenever a counter overflows, the counter id and the time of the overflow is recorded. Additionally also the time and the type of failures of the CPU are recorded. This data can then be fed to the algorithm:

- each recorded counter id corresponds to an event
- the difference between two consecutive events corresponds to the delay $d_k = t_{k+1} - t_k$
- the failure events are used as oracle to compute F-measure, Precision and Recall

Should the experiment result in a reasonable F-measure, it would not only have been showed that the accelerated algorithm works on real data, but also that it is

[2]http://arma.sourceforge.net
[3]http://eigen.tuxfamily.org
[4]http://www.netlib.org/lapack

possible to predict HW failures of a CPU-core by using hardware counters. This experiment must first be performed with a non-optimized version of the algorithm (i.e. the exact same implementation as described in [12]) before it is repeated with an optimized version (if any optimization has been implemented).

7 Preliminary Work Progress

todo…

8 Time Schedule

todo…

References

[1] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, *Basic concepts and taxonomy of dependable and secure computing*, IEEE Transactions on Dependable and Secure Computing, 1 (2004), pp. 11–33.

[2] M. Azhar, M. Sjalander, H. Ali, A. Vijayashekar, T. Hoang, K. Ansari, and P. Larsson-Edefors, *Viterbi accelerator for embedded processor datapaths*, in IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP), July 2012, pp. 133–140.

[3] S. Cadambi, I. Durdanovic, V. Jakkula, M. Sankaradass, E. Cosatto, S. Chakradhar, and H. Graf, *A massively parallel FPGA-based coprocessor for support vector machines*, in IEEE Symposium on Field Programmable Custom Computing Machines (FCCM), April 2009, pp. 115–122.

[4] C. Domeniconi, C.-s. Perng, R. Vilalta, and S. Ma, *A classification approach for prediction of target events in temporal sequences*, in Principles of Data Mining and Knowledge Discovery, T. Elomaa, H. Mannila, and H. Toivonen, eds., vol. 2431 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, pp. 125–137.

[5] A. Jacob, J. Lancaster, J. Buhler, and R. Chamberlain, *Preliminary results in accelerating profile hmm search on fpgas*, in IEEE International Parallel and Distributed Processing Symposium (IPDPS), March 2007, pp. 1–8.

[6] T.-T. Lin and D. Siewiorek, *Error log analysis: statistical modeling and heuristic trend analysis*, IEEE Transactions on Reliability, 39 (1990), pp. 419–432.

[7] T.-T. Y. Lin, *Design and evaluation of an on-line predictive diagnostic system*, PhD thesis, Carnegie-Mellon University, Pittsburgh, PA, 1988.

[8] R. P. Maddimsetty, J. Buhler, R. D. Chamberlain, M. A. Franklin, and B. Harris, *Accelerator design for protein sequence hmm search*, in International Conference on Supercomputing, ICS '06, New York, NY, USA, 2006, ACM, pp. 288–296.

[9] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel, *Performance measures for information extraction*, in In Proceedings of DARPA Broadcast News Workshop, 1999, pp. 249–252.

[10] T. Oliver, L. Yeow, and B. Schmidt, *High performance database searching with hmmer on fpgas*, in IEEE International Parallel and Distributed Processing Symposium (IPDPS), March 2007, pp. 1–7.

[11] C. J. V. Rijsbergen, *Information Retrieval*, Butterworth-Heinemann, Newton, MA, USA, 2nd ed., 1979.

[12] F. Salfner, *Event-based Failure Prediction*, PhD thesis, Humboldt-University of Berlin, February 2008.

[13] F. Salfner, M. Lenk, and M. Malek, *A survey of online failure prediction methods*, ACM Comput. Surv., 42 (2010), pp. 10:1–10:42.

[14] F. Salfner, S. Tschirpke, and M. Malek, *Comprehensive logfiles for autonomic systems*, in IEEE International Parallel and Distributed Processing Symposium (IPDPS), April 2004, pp. 211–218.

[15] R. Vilalta and S. Ma, *Predicting rare events in temporal domains*, in IEEE International Conference on Data Mining (ICDM), December 2002, pp. 474–481.

[16] J. Walters, V. Balu, S. Kompalli, and V. Chaudhary, *Evaluating the use of gpus in liver image segmentation and hmmer database searches*, in IEEE International Parallel Distributed Processing Symposium (IPDPS), May 2009, pp. 1–12.