A bayesian model for system level reliability estimation

A. Vallero*, A. Savino*, S. Tselonis†, N. Foutris†, M. Kaliorakis†, G. Politano*, D. Gizopoulos†, S. Di Carlo*
*Control and Computer Engineering Department, Politecnico di Torino, 10129 Torino, Italy
†Department of Informatics & Telecommunications, University of Athens, 15784, Athens Greece

***Abstract*—[1] Nowadays, the scientific community is looking for ways to understand the effect of software execution on the reliability of a complex system when the hardware layer is unreliable. This paper proposes a statistical reliability analysis model able to estimate system reliability considering both the hardware and the software layer of a system. Bayesian Networks are employed to model hardware resources of the processor and instructions of program traces. They are exploited to investigate the probability of input errors to alter both the correct behavior and the output of the program. Experimental results show that Bayesian networks prove to be a promising model, allowing to get accurate and fast reliability estimations w.r.t. fault injection/simulation approaches.**

I. Introduction

Reliability is an important design aspect for computer systems due to the aggressive technology miniaturization. Unreliable hardware components affect computing systems at several levels. After a raw error manifests in a hardware block (e.g., microprocessor, memory), it can be propagated to other layers composing the full system and eventually reach the software stack of the system and corrupt either data or instructions of software applications. Nevertheless, the software stack itself can play an important role in masking errors generated in the underlying layers [1]. Therefore, the role of the software stack coupled with the target hardware architecture must be carefully considered when system reliability is analyzed.

This paper proposes a new statistical approach for the estimation of the reliability of a microprocessor-based system considering both the hardware and the software layer. The software execution on the microprocessor is modeled in the form of a Bayesian network that describes relations among resources (e.g., registers, memory elements, functional units) involved in the execution of the instructions composing a program. The Bayesian model is then exploited to compute a probability of correct (error-free) execution of the software in the presence of a hardware fault, used to estimate the overall reliability of the system.

Transient, intermittent and permanent faults can be considered in this phase without affecting the way the high-level model is constructed. Due to limitation in space, without loosing generality, in this paper we will focus on transient faults as a special case. Experimental results performed on a program executed on top of a x86-64 microprocessor show that reliability estimations are accurate when compared to those obtained by time-consuming fault-injection experiments performed using a micro-architectural fault injector. At the same time, the proposed approach enables a significant reduction of the time required to perform the reliability analysis, enabling fast and accurate reliability evaluations.

[1]This paper has been fully supported by the 7th Framework Program of the European Union through the CLERECO Project, under Grant Agreement 611404.

II. Reliability analysis

The proposed statistical reliability analysis methodology estimates the probability of failure of a computer system running a program.

Bayesian networks are employed to model program traces, i.e., sequences of instructions issued by the microprocessor when the program is executed with a specific workload. Several traces can be obtained from a single program by profiling its execution with different workloads using a dedicated profiler. Fig.1 shows an example of how a program trace can be modeled resorting to a Bayesian network. Nodes of the network represent program's resources and input errors while edges model possible error propagation/masking paths among resources.

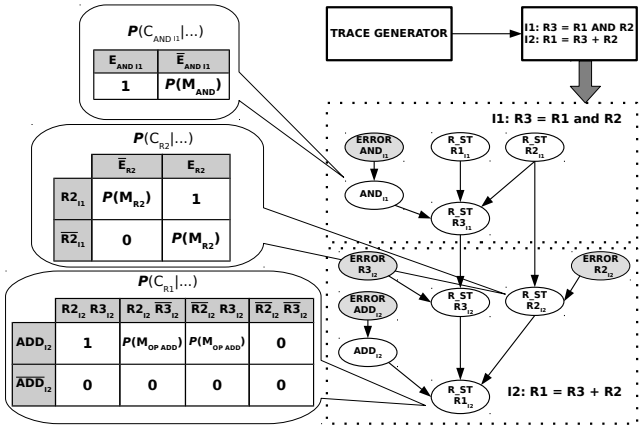

Fig. 1. Example of a bayesian network model for a simple sequence of two instructions.

Once conditional probabilities are set up properly for every node, the BN network can be solved and the probability of correct execution of a trace can be estimated. However, only a subset of the instructions of a trace directly affect resources that identify the final outcome of the computation. We denote this subset of instructions as active state instructions (A_I).

To compute the probability that a program trace is correct, error probability is taken into account only for output resources

978-1-4799-7603-4/15/$31.00 ©2015 IEEE

of the active state instructions. We define such resources as active resources (A^{res}) and we denote with A_i^{res} the subset of active resources modified by instruction I_i . Given these definitions, the probability for the active instruction I_i to be correct ($P(I_i)$) can be computed as the probability that all its active resources are correct:

$$P(I_i) = \prod_{R \in A_i^{res}} P(R) \tag{1}$$

since the probabilities of correctness of output resources of an instruction are statistically independent.

A program trace T is correct (error-free) if all active state instructions are correct. The probability of correctness of a trace ($P(T)$) can therefore be computed as:

$$P(T) = P(\bigcap_{I \in A_I} I) = \prod_{I \in A_I} P(I|J, \forall J < I) \tag{2}$$

Once the probability of correctness of a trace is computed, the error rate of the system while executing the trace can be computed as:

$$\lambda_{estimated}^{T} = -\frac{ln(P(T))}{t_T} \tag{3}$$

where t_T is the execution time of the analyzed program trace. To obtain the system Soft Error Rate (SER), λ_{BN}, a simple or a weighted average can be applied to all $\lambda_{estimated}$ of the analyzed traces:

$$\lambda_{BN} = \sum_{T \in analyzed\, traces} \lambda_{estimated}^{T} \times w_i \tag{4}$$

where $\sum_i w_i = 1$.

III. Experimental results

The proposed reliability estimation methodology has been validated on the *QSORT* MiBench benchmarks [2] running on the x86-64 architecture. Traces have been extracted resorting to the MARSSx86 simulator [3]. The same benchmark with the related workloads is analyzed resorting to the proposed Bayesian model and resorting to an extensive architectural fault injection campaign. Computed results are then compared to evaluate the accuracy and the performance of the proposed model. In this preliminary study, only microprocessor physical registers belonging to the Integer Register File have been considered as sources of hardware faults.

Fault injection experiments have been performed using the MARSSx86-FI [4] fault injector. We performed a statistical fault injection campaign, on MARSSx86-FI, adopting the statistical sampling of [5]. We compute a fault population for 99% confidence and 3% error margin. The calculation leads to a total of 1843 different single-bit transient faults, which are generated for the injections on the integer physical register file.

Results of fault injection based reliability estimations and bayesian based estimations are first compared in terms of accuracy. Figure 2 compares for the case study the SER computed through fault injection with the one computed resorting to the Bayesian model assuming a raw error rate for the register file $\lambda_{RF} = 1.9 \cdot 10^{-8}$.

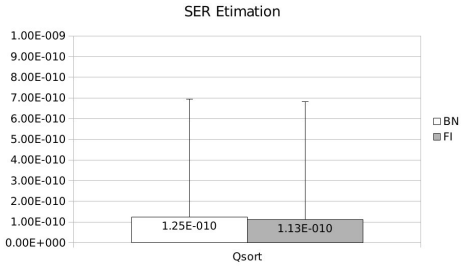

Fig. 2. Estimation of the SER of both Bayesian network and Fault Injection approaches. We also report error bars for both approaches.

We can state results show that prediction error with BN have almost the same order of magnitude of FI.

Finally, Figure 3 compares simulation time between fault injection based and bayesian based reliability estimations. The figure clearly shows that, resorting to the statistical model, estimation time is reduced by two orders of magnitude thus enabling very fast estimations.

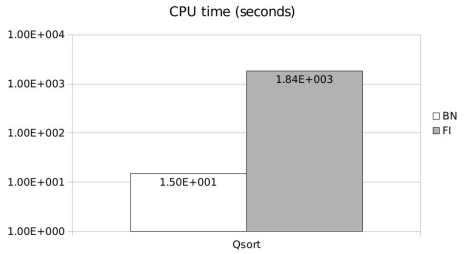

Fig. 3. Performance comparison of simulation time of a single trace for both the Bayesian Network and the Fault Injection approaches

IV. Conclusion

This paper proposes a new statistical approach for the estimation of the reliability of a microprocessor-based system, taking into account both the hardware and the software layer. Preliminary experimental results performed on the MiBench benchmark clearly show that the proposed approach is able to provide an accurate and fast estimations when compared to a similar analysis, performed using micro-architectural level fault injection.

References

[1] A. Savino, S. Carlo, G. Politano, A. Benso, A. Bosio, and G. Di Natale, "Statistical reliability estimation of microprocessor-based systems," *Computers, IEEE Transactions on*, vol. 61, no. 11, pp. 1521–1534, Nov 2012.

[2] University of Michigan at Ann Arbor. Mibench version 1.0. [Online]. Available: http://www.eecs.umich.edu/mibench/

[3] A. Patel, F. Afram, S. Chen, and K. Ghose, "Marss: a full system simulator for multicore x86 cpus," in *Proceedings of the 48th Design Automation Conference*. ACM, 2011, pp. 1050–1055.

[4] N. Foutris, M. Kaliorakis, S. Tselonis, and D. Gizopoulos, "Versatile architecture-level fault injection framework for reliability evaluation: A first report," in *On-Line Testing Symposium (IOLTS), 2014 IEEE 20th International*. IEEE, 2014, pp. 140–145.

[5] R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence," in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, April 2009, pp. 502–506.