A Survey of Research Platforms for Fault Injection-Based Reliability Evaluation (Post-print)

Authors: Hu Yu, Li Xiaowei

Date: 2017-03-09T00:00:00+00:00

Abstract

The primary objective of reliability assessment research platforms is to rapidly and accurately evaluate the impact of various faults on computing system reliability, thereby providing system designers with quantitative analysis data. This paper provides an overview of reliability evaluation methods for computing systems, focusing on representative research efforts in fault injection-based reliability assessment platforms both domestically and internationally, while also presenting the progress made by our research group in related studies.

Full Text

Preamble

Information Technology Letters, Vol. 8, No. 2
A Survey of Fault Injection-Based Reliability Evaluation Research Platforms
Yu Hu, Xiaowei Li

Abstract

The primary purpose of reliability evaluation research platforms is to rapidly and accurately assess the impact of various faults on computing system reliability, thereby providing quantitative analysis data for system designers. This paper provides an overview of reliability evaluation methods for computing systems, focusing on representative work conducted domestically and internationally on fault injection-based reliability evaluation platforms, while also presenting the progress of our research group in related research.

Keywords: High-End Computing Systems, Reliability Evaluation, Fault Injection

1 Introduction

As high-end computing systems employ tens of thousands of high-performance processors to achieve peak performance of hundreds of teraflops to petaflops per second through massive parallelism, the Mean Time Between Failures (MTBF) continues to decline with the expanding scale of system hardware. Figure 1 [Figure 1: see original paper] shows how system MTBF decreases as the number of nodes increases, given individual component failure probabilities of 0.0001, 0.00001, and 0.000001 per hour (i.e., MTBF of 10^4, 10^5, and 10^6 hours) [1]. As evident from the figure, a Petaflops-scale system with 200,000 nodes has an MTBF of only a few hours. Additionally, as integrated circuit feature sizes shrink, supply voltages decrease, and frequencies increase, integrated circuit chips become more sensitive to various noise interferences such as voltage fluctuations, electromagnetic interference, and radiation. The combined effect of these two factors has made reliability one of the six major challenges facing high-end computing systems [2]. The primary goal of building reliability evaluation research platforms is to rapidly and accurately assess the impact of various faults on high-end computing system reliability, providing quantitative analysis data for system designers.

Figure 1. MTBF of High-End Computing Systems [1]

Reliability evaluation methods for computing systems include measurement-based approaches, analytical model-based approaches, and fault injection-based approaches. Measurement-based methods involve observing the behavior of actual systems when natural errors occur during real workload execution, thereby obtaining authentic data. However, due to the low frequency of errors during actual operation, these methods often require long periods to collect sufficient data for statistical analysis. Analytical model-based methods involve constructing mathematical models of computing systems, such as Markov chain models and Petri net models, to calculate relevant metrics through mathematical analysis. Nevertheless, inaccuracies in the models themselves or their input parameters may lead to significant deviations in the analysis results. Fault injection refers to the experimental process of artificially introducing faults into a target system running a specific workload according to a pre-selected fault model, using certain strategies, and then observing and analyzing the system' s behavior after fault introduction to obtain qualitative or quantitative results. As a primary method for evaluating computing system reliability, fault injection technology was first proposed in an internal IBM technical report in the early 1970s [3], subsequently adopted by industry for reliability evaluation of fault-tolerant computing systems, attracted attention from universities and research institutions in the mid-1980s, and is now widely used in reliability evaluation of various computing systems. Compared with measurement and analytical model methods, fault injection is more economical and flexible, occupying an increasingly important position in reliability evaluation. Therefore, this paper focuses on introducing representative work on fault injection-based reliability evaluation platforms conducted domestically and internationally.

2 Overview of Fault Injection-Based Reliability Evaluation Platforms

A fault injection-based reliability evaluation platform primarily includes the target system and modules such as fault injection, fault library, workload generator, workload library, controller, monitor, data collector, and data analyzer [4], as shown in Figure 2 [Figure 2: see original paper].

Figure 2. Framework of Fault Injection-Based Reliability Evaluation Platform

The fault injection module is responsible for injecting faults into the target system; the monitor tracks the execution commands of the fault injection module and initiates the data collector when necessary. The data collector gathers data online, while the data analyzer processes and analyzes data offline. The controller manages the entire experimental process. Fault types can include Stuck-At Faults, Bit-flip Faults, Bridging Faults, Spurious Current, and Power Surge, among others.

Based on the implementation approach for fault injection, methods are primarily categorized into Hardware-Implemented Fault Injection (HWIFI), Software-Implemented Fault Injection (SWIFI), and Simulation-Based Fault Injection.

2.1 Hardware-Implemented Fault Injection (HWIFI)

Hardware-implemented fault injection utilizes additional hardware to introduce faults into the target system' s hardware. According to whether the fault injector directly contacts the target system, HWIFI can be further divided into contact and non-contact types. As the name implies, contact HWIFI uses pin-level probes [5][6], fixtures [29], or sockets [7] to directly generate voltage or current changes on the target chip. Non-contact HWIFI employs heavy-ion radiation [8][9][10], electromagnetic effects [11], lasers [12], or scan chains [13] to generate spurious current on the target chip. The advantage of HWIFI is its ability to produce faults at any location within the target system, while its disadvantages include difficulty in precisely controlling the timing and location of fault injection and the potential risk of damaging the target system.

2.2 Software-Implemented Fault Injection (SWIFI)

Software-implemented fault injection modifies programs at compile-time or runtime [14][15][16][17][18][19], causing the target system' s normal state to change during program execution. SWIFI can inject faults not only into applications but also into operating systems. Its advantages include ease of implementation and low cost, while its disadvantages include: inability to inject faults into locations inaccessible to software, thus only partially simulating actual fault conditions; relatively low timing precision, making it unsuitable for simulating faults with long latency periods such as bus and processor faults; and the need to modify applications, which may alter the workload.

2.3 Simulation-Based Fault Injection

Simulation-based fault injection involves injecting faults into systems designed with VHDL[1] or Verilog hardware description languages during the system design phase. It can be performed at various abstraction levels, including switch-level [20], gate-level and register-transfer level [21][22][30], and behavioral level [23][24][25]. Simulation-based fault injection offers good controllability and observability, enabling reliability evaluation during the early stages of system design and helping designers adopt appropriate fault-tolerant measures to improve system reliability early on. The disadvantage is that fault injection simulation experiments at lower abstraction levels, while providing higher simulation accuracy, are slower and require longer time to obtain meaningful statistical data. In recent years, researchers have utilized fault pruning to accelerate simulation [26] and employed FPGA[2] emulation technology for reliability evaluation [27][28].

In the following section, we select several representative research efforts to provide detailed introductions to fault injection-based reliability evaluation platforms.

3 Representative Research Work

3.1 AFIT (Advanced Fault Injection Tool) [6]

AFIT is a hardware-implemented fault injection platform developed by the University of València, Spain, which uses fixtures for pin-level fault injection on target chips. Figure 3 [Figure 3: see original paper] shows the overall framework of AFIT, including a personal computer serving as the controller, synchronization and trigger module, timing module, target system activation module, event reading module, high-speed fault injection module, and target system prototype.

Figure 3. Overall Block Diagram of AFIT

The synchronization and trigger module controls the start time of fault injection experiments; the timing module provides a 40MHz clock and fault injection enable signal AI for the high-speed fault injection module, with the AI signal waveform varying according to the number and type of injected faults; the target system activation module initializes the target system prototype; the event reading module uses counters and trace memory to determine when to read system responses. The structure of the high-speed fault injection module is shown in Figure 4 [Figure 4: see original paper], comprising injection activation logic and an effective error detector. Upon receiving the AI signal from the timing module, the injection activation logic connects transistors. When the transistor connected to the power supply is activated, it injects faults with a logic value of 1 into the prototype system through the IOUT signal; when the transistor connected to ground is activated, it injects faults with a logic value

[1]VHSIC Hardware Description Language, Very High-Speed Integrated Circuit Hardware Description Language

[2]Field Programmable Gate Array

of 0 through IOUT. The effective error detector sets the Memory of Effective Error (MEE) signal connected to the personal computer—MEE is 1 when IOUT actually changes the logic value of the prototype system' s pins, indicating successful fault injection, and 0 otherwise, indicating unsuccessful injection. AFIT can inject faults into the target system at frequencies up to 40MHz and can selectively inject transient, intermittent, or permanent faults, with controllable fault location, duration, and frequency. Specifically, transient fault duration is controllable from 100ns to 4 s, intermittent fault duration from 100ns to 2 s with intervals from 1 to 65ms, and permanent fault duration is 1.2s.

Figure 4. Structure of High-Speed Injection Module

3.2 Ftape (Fault Tolerance and Performance Evaluator) [17]

Ftape is a software-implemented fault injection platform developed by the University of Illinois at Urbana-Champaign. Figure 5 [Figure 5: see original paper] shows the overall framework of Ftape. Fault injection locations include software-accessible CPU registers, memory, and disk subsystems, with fault types being single or multiple bit-flips, sets, or resets. The timing and location of fault injection can be randomly generated according to probability distribution functions (such as exponential or normal distributions) or generated based on workload characteristics (e.g., injecting faults into frequently used components to accelerate fault effects). Disk system fault injection is implemented by running code in the driver, such as bus faults or timer faults, thus incurring no additional hardware overhead. Ftape has been used to measure the reliability of two Tandem fault-tolerant computer prototypes and system performance degradation during failures.

Figure 5. Overall Framework of Ftape

3.3 DEPEND [24]

DEPEND is another reliability evaluation platform developed by the University of Illinois at Urbana-Champaign, employing a simulation-based fault injection approach. It describes computing system behavior through a collection of communicating processes, with functional fault models. The overall block diagram of DEPEND is shown in Figure 6 [Figure 6: see original paper]. To use DEPEND, developers first write C++ control programs using objects from the DEPEND library, then compile and link them into a runtime environment for fault injection. Subsequently, repair processes are initiated, and statistical reports are generated. Objects in the DEPEND library include active components (simulating basic servers with first-come-first-served and round-robin service policies, providing manual fault injection and repair), fault injectors (injecting faults according to set probability distributions or workload characteristics), checksum calculators (computing checksums), fault reporters (collecting fault statistics, displaying MTBF, MTBR[3], availability, and coverage, providing specific information for

[3]Mean Time Between Repairs

each fault), voters (simulating a basic voter with timeout functionality, allowing user-defined voting strategies), servers (having active component attributes but simulating servers with redundant components, providing automatic fault injection, automatic repair, and reconfiguration capabilities), links (simulating communication channels, supporting link faults, packet errors, packet loss, and user-defined faults, with automatic retry support), multi-mode redundancy modules (simulating dual-module testing, triple-module redundancy, and N-module redundancy), and fault managers (logging faults and shutting down components exceeding fault thresholds).

Figure 6. Overall Framework of DEPEND

3.4 FuSE (Fault injection using SEmulation) [28]

FuSE is an FPGA-based emulation reliability evaluation platform developed by the Vienna University of Technology, designed to accelerate fault injection simulation experiments. Its overall block diagram is shown in Figure 7 [Figure 7: see original paper]. The SEmulator engine supports three modes: simulation mode—where both the testbench and Device Under Test (DUT) run on the host, with simulation speeds of approximately thousands of clock cycles per second; co-simulation mode—where the entire or partial DUT is downloaded into an HMX2-AS2 FPGA development board, so testbench output signals are fed into the emulated circuit through the FPGA' s PCI Express interface under the control of an I/O Manager, while observed signals are sent back to the host, achieving emulation speeds up to millions of clock cycles per second; and emulation mode —where both testbench and DUT run on the FPGA board, reaching speeds up to tens of millions of clock cycles per second. The FuSE configuration data logging module records configuration information and experimental results from fault injection experiments.

Figure 7. Overall Framework of FuSE

3.5 SRP Processor Application Example

Figure 8 [Figure 8: see original paper] illustrates the application of the SRP (Self-Repairing Processor) processor in wireless sensor network nodes. The workload is a wireless sensor network application where light intensity data collected by SRP sensor nodes and ordinary sensor nodes is transmitted through a data acquisition node to host control software for analysis and display. Figure 8(b) shows the process where, after injecting 8 faults into the SRP, the SRP restores normal operation through self-test, self-diagnosis, and self-repair.

Figure 8. Application of SRP Processor in Wireless Sensor Network Nodes

5 Conclusion

This paper surveys research on fault injection-based reliability evaluation platforms domestically and internationally, introducing four representative works

based on hardware-implemented fault injection, software-implemented fault injection, simulation-based fault injection, and FPGA emulation, respectively. It also presents our own reliability evaluation work using simulation-based fault injection in a processor design with self-sensing, self-diagnosis, and self-repair (3S) capabilities. Future work will focus on developing corresponding reliability evaluation research platforms for multi-core processors and high-end computing systems.

References

[1] Daniel A. Reed, Charng-da Lu, Celso L. Mendes, Big Systems and Big Reliability Challenges, Proceedings of Parallel Computing: Software Technology, Algorithms, Architectures and Applications (PARCO), 2003, pp. 729-736.

[2] Carl G. Tengwall, IBM®Blue Gene®/P - An Overview of a Petaflop Capable System, http://www.nsc.liu.se/lcsc2007/presentations/LCSC_{2007}-tengwall.pdf.

[3] Harlan D. Mills, On the Statistical Validation of Computer Programs, Technical Report, FSC-72-6015, IBM Federal Systems Division, 1972.

[4] Mei-Chen Hsueh, Timothy K. Tsai, Ravishankar K. Iyer, Fault Injection Techniques and Tools, IEEE Computer, 1997, 30(4): 75-82.

[5] Henrique Madeira, Mário Zenha Rela, Francisco Moreira, João Gabriel Silva, RIFLE: A General Purpose Pin-level Fault Injector, Proceedings of European Dependable Computing Conference on Dependable Computing (EDCC), 1994, pp. 199-216.

[6] J. R. Martínez, P.J. Gil, G. Martín, C. Pérez, J. J. Serrano, Experimental Validation of High-Speed Fault-Tolerant Systems Using Physical Fault Injection" , Proceedings of International Working Conference on Dependable Computing for Critical Applications (DCCA), 1999, pp. 233-249.

[7] J. Arlat, Y. Crouzet, and J.C. Laprie, Fault Injection for Dependability Validation of Fault-Tolerant Computer Systems, Proceedings of International Symposium on Fault-Tolerant Computing (FTCS), 1989, pp. 348-355.

[8] U. Gunneflo, J. Karlsson, J. Torin, Evaluation of error detection schemes using fault injection by heavy-ion radiation, Proceedings of International Symposium on Fault-Tolerant Computing (FTCS), 1989, pp. 340-347.

[9] Cristian Constantinescu, Neutron SER Characterization of Microprocessors, Proceedings of International Conference on Dependable Systems and Networks (DSN), 2005, pp. 754-759.

[10] Jeffrey Kellington, Ryan McBeth, Pia Sanda and Ronald Kalla, IBM® POWER6™ Processor Soft Error Tolerance Analysis Using Proton Irradiation, Proceedings of Workshop on Silicon Errors in Logic-Systems Effects (SELSE), 2007.

[11] J. Karlsson, J. Arlat, and G. Leber, Application of Three Physical Fault Injection Techniques to the Experimental Assessment of the MARS Architecture, Proceedings of International Working Conference on Dependable Computing for Critical Applications (DCCA), 1995, pp. 150-161.

[Sampson1998] J. R. Sampson, W. Moreno, F. Falquez, A technique for automatic validation of fault tolerant designs using laser fault injection, Proceedings of International Symposium on Fault-Tolerant Computing (FTCS), 1998, pp. 162-167.

[13] P. Folkesson, S. Svensson, J. Karlsson, A Comparison of Simulation Based and Scan Chain Implemented Fault Injection, Proceedings of International Symposium on Fault-Tolerant Computing (FTCS), 1998, pp. 284-293.

[14] J.H. Barton, E.W. Czeck, Z.Z. Segall, and D.P. Siewiorek, Fault Injection Experiments Using FIAT, IEEE Transactions on Computers, 1990, 39(4): 575-582.

[Kanawati1995] G.A. Kanawati, N.A. Kanawati, and J.A. Abraham, FERRARI: A Flexible Software-Based Fault and Error Injection System, IEEE Transactions on Computers, 1995, 44(2):

[16] S. Han, K.G. Shin, and H.A. Rosenberg, Doctor: An Integrated Software Fault-Injection Environment for Distributed Real-Time Systems, Proceedings of International Computer Performance and Dependability Symposium (IPDS), 1995, pp. 204-213.

[17] T.K. Tsai and R.K. Iyer, An Approach to Benchmarking of Fault-Tolerant Commercial Systems, Proceedings of International Symposium on Fault-Tolerant Computing (FTCS), 1996, pp. 314-323.

[Carreira1998] J. Carreira, H. Madeira, J.G. Silva, Xception: A technique for the experimental evaluation of dependability in modern computers, IEEE Trans. on Software Engineering, 1998, 24(2):

[19] A. Baldini, A. Benso, S. Chiusano, P. Prinetto, "BOND" : An interposition agents based fault injector for Windows NT, Proceedings of International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT), 2000, pp. 387-395.

[20] H.R. Zarandi, G. Miremadi, and A. Ejlali, Fault Injection into Verilog Models for Dependability Evaluation of Digital Systems, Proceedings of International Symposium on Parallel and Distributed Computing (ISPDC), 2003, pp. 281-287.

[21] V. Sieh, O. Tschäche, and F. Balbach, VERIFY: Evaluation of reliability using VHDL-models with embedded fault descriptions, Proceedings of International Symposium on Fault-Tolerant Computing (FTCS), 1997, pp. 32-36.

[22] D. Gil, J. Gracia, J. C. Baraza, and P. J. Gil, A study of the effects of transient fault injection into the VHDL model of a fault-tolerant microcomputer sys-

tem, Proceedings of International On-Line Testing Workshop (IOLTW), 2000, pp. 73-79.

[23] E. Jenn, J. Arlat, M. Rimén, J. Ohlsson, and J. Karlsson, "Fault injection into VHDL models: The MEFISTO tool, Proceedings of International Symposium on Fault-Tolerant Computing (FTCS), 1994, pp. 356-363.

[Goswami1997] Kumar K. Goswami, Ravishankar K. Iyer, and Luke Young, DEPEND: A Simulation-Based Environment for System Level Dependability Analysis, IEEE Transactions on Computers, 1997, 46(1): 60-74.

[25] Juan-Carlos Baraza, Joaquín Gracia, Sara Blanc, Daniel Gil, and Pedro-J. Gil, Enhancement of Fault Injection Techniques Based on the Modification of VHDL Code, IEEE Transactions. on VLSI Systems, 2008, 16(6): 693-706.

[26] L. Berrojo, F. Corno, L. Entrena, I. González, C. López, M. Sonza, G. Squillero, An Industrial Environment for High-Level Fault-Tolerant Structures Insertion and Validation, Proceedings of VLSI Test Symposium (VTS), 2002, pp. 229-236.

[27] P. L. Civera, L. Macchiarulo, M. Rebaudengo, M. Sonza Reorda, M. Violante, Exploiting FPGA for accelerating Fault Injection Experiments, Proceedings of International On-Line Testing Workshop (IOLTW), 2001. pp. 9-13.

[28] Marcus Jeitler, Martin Delvai, Stefan Reichör, FuSE - A Hardware Accelerated HDL Fault Injection Tool, Proceedings of Southern Conference on Programmable Logic (SPL), 2009, pp. 89-94.

[29] 王建莹，孙峻朝，容错计算机系统可靠性评估工具：HFI-2 故障注入器，电子学报，1999，27(11):

[30] 黄海林，唐志敏，许彤，龙芯 1 号处理器的故障注入方法与软错误敏感性分析，计算机研究与发展，2006，43(10): 1820-1827.

Author Biographies

Yu Hu is an Associate Professor and Master' s Supervisor at the Institute of Computing Technology, Chinese Academy of Sciences (huyu@ict.ac.cn).

Xiaowei Li is a Professor and Doctoral Supervisor at the Institute of Computing Technology, Chinese Academy of Sciences, Deputy Director of the Key Laboratory of Computer System and Architecture, Chinese Academy of Sciences, Council Member of the China Computer Federation, Director of the Fault-Tolerant Computing Professional Committee, and a member of the JCST editorial board.

Note: Figure translations are in progress. See original paper for figures.