2022 IEEE Symposium on Industrial Electronics & Applications (ISIEA) | 978-1-6654-8012-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISIEA54517.2022.9873679

A Survey on Software/Hardware Fault Injection Tools and Techniques

Nadir K.Salih[1] D Satyanarayana[2] , Abdullah Said Alkalbani[3], R. Gopal[4]
College of Engineering, University of Buraimi, Oman [1,2,3,4]
nadir@uob.edu.om [1] degala.s@uob.edu.om[2] abdullah.s@uob.edu.om[3] gopal.r@uob.edu.om[4]

Abstract
Computational systems need to be dependable, but they are vulnerable to defects, errors, failure and faults, and they affect their behaviour. Fault injection is one of the useful techniques that have been used to evaluate the behaviour of the system by intentionally introduced fault, so as to determine its effect on the system behaviour. There are many tools and techniques used to inject faults, categorized under software and hardware techniques. To keep pace with progress and development over time, developers need to be aware off tools and techniques according to their needs, when using either software or hardware. In this paper we make a survey for several tools and techniques which are mostly used these days. As we compare between them based on their basic functions, source need, how to inject faults and feedback about each one.

Keywords: Fault injection, Tools, Techniques, Dependability.

I. Introduction

In the last decades, the use of computational systems has been increased in a huge way. Each of these systems has been employed for specific tasks and required to perform them even if there are faults. For example, ATM does not allow someone withdraw cash with limitless, even if there is a fault in the system. It has to give an acceptable answer to the fault. As a response, the ATM should display that there is a problem in its system, which means that it is unavailable [14]. To ease the understanding, a few basic definitions are provided. **Fault** is a physical deficiency, scarcity/shortage or shortcoming that occurs within some hardware or software component; hence it may cause a huge damage in the system depending on its effect. **Error** is a deviation from accuracy or correctness. In addition, it demonstrates the fault. So detecting errors implies faults presence. **Failure** is the non-execution of some procedures that is due or expected [15][16]. Computer systems can be vulnerable to different problems such as environmental conditions, faults, errors or failures that may affect them and change the expected system behaviour. Hence, they need to be monitored in order to detect them and maintain the system behaviour. The fault injection is a technique used to evaluate the effects in computer systems, based on the observation of the system behaviour in the presence of intentionally induced faults [17]. It also can be defined as an assessment of fault tolerant systems and can be performed at different levels of the system [18] [19]. Fault injection can be considered as a method to assess fault tolerant systems. Different fault injection methods can be classified into:

Hardware fault injection, which it the external physical sources used to affect the actual hardware system.

Simulation-based fault injection, which it a fault simulator used to simulate and model the target system and the faults.

Emulation-based fault injection, where the target system is emulated (usually with Field Programmable Gate Arrays (FPGAs)) and faults are injected in the emulator [18]. To inject faults, different tools are needed. They require many procedures that should be taken into account. In the beginning, the system under test should be activated by the used tool. Then the faults may be injected as the user determined. After the injection, the target system should be monitored to ensure that the expected system behaviour is verified. The fault injection, monitoring and activation activities should be controlled and coordinated. In order to specify the faults that are intended to inject, the user needs a user interface. Also he or she needs it to specify the conditions of the experiment, and also where the results of the experiment can be got [14]. According to software/hardware changes, there are two types of fault injection techniques: Intrusive techniques: To enable fault injection in this technique, the original model of software or hardware need to be modified and Non-intrusive techniques, in which the original model code or hardware remains unchanged during all processes [20].Fault injection techniques also can be grouped into invasive and non-invasive techniques. Invasive techniques are those which leave behind a footprint of the testing mechanism, during testing which is impossible to remove. Non-invasive techniques are able to hide their presence so as to have no effect on the system unlike the faults they inject [1]. This paper is organized as follows: Section I is the introduction of the research. Section II presents the related Research. Section III describes the motivations of fault injection. Section IV demonstrates the fault injection environment. Section V presents the fault injection techniques. Section VI demonstrates the fault injection tools. Section VII introduces the advantages and disadvantages of fault injection. Finally, section III concludes the paper. To gain insight about the fault injection tools and techniques and after well studying and extracting the most relevant articles, we expose the research in a form of sex questions related to the topic as in table1

NO	Research questions
RQ#1	What are the main motivations toward Fault injection tools and techniques?
RQ#2	What are the assumptions for software fault injection and hardware fault injection as well?
RQ#3	What are the main types of contribution?
RQ#4	What are the used evaluation methods?
RQ#5	Which environment used for fault injection?
RQ#6	What are the open issues and future directions addressed by the studies?

Table 1 Research Questions

II. Related Researches

As computer systems are vulnerable to damage as a result to different conditions, developers should provide them with tolerant system so as to guarantee that they give the expected answers even in the presence of faults. Fault injection has been widely used as a dependability evaluation technique. This section summarizes the most relevant researches in this field. There are two main categories of techniques used to inject faults, which are software and hardware. [22],[18],[16],[23], [24] and [31] present the three main techniques to implement the fault injection in the hardware of a system, which are; the physical fault injection, which is complemented at the physical level, embarrassing the hardware with environmental parameters such as heavy ions radiation or modifying the value of the integrated circuits pins; Software Implemented Fault injection (SWIFI),which aims to reproduce the errors that would have been produced based on faults taking place in the hardware. As it is encouraged to avoid the difficulties and the inveterate cost to the implementation of physical fault injection approaches or heavy ion; and simulated fault injection, in which the system under test is simulated in another computer system. In [1] and [16], the hardware faults are classified according to their duration into:

1) Permanent faults: They are usually caused by irreversible component damage, improper manufacture, or misuse/overwork.
2) Transient faults: They are also called soft errors. They are triggered by environmental troubles, such as electromagnetic interference or radiation. Despite of their ability to induce an improper state in the system, they can return to a normal operating state in a short duration without causing permanent damage. According to several studies, transient faults occur much more and harder to detect than permanent ones.
3) Intermittent faults: In-case of the presence of this type of faults, the system becomes unstable and oscillates between the active and the sleep state. They are usually caused by unstable hardware. They can be repaired by redesign or replacement.

There is a great need of different tools to inject faults in hardware, using the previous techniques. One of these tools is named FAUMachine, that its design goal is the simulation of the hardware to be as close as possible to the corresponding physical hardware. Moreover, it can simulate even the environment like classes and the user interaction [25]. Another hardware-based injection tool is called RIFLE. The injection is in the pin-level of any module. In addition of its ability to inject faults, it can also analyze their impacts on the systems. Moreover, it has an innovative feature which is its capability of defining different certain types of faults via the combination of fault trigger capabilities with multilateral definition software [18].Software fault injection also needs different tools. Library Fault Injector (LFI) is one of them. It automates the preparation of fault scenarios and their injection between shared libraries and applications as well. It works for several platforms such as Windows and Linux. LFI has been evaluated taking into account its ease of use, efficiency, accuracy and performance overhead and the conclusion is that it is useful even if it is running without human intervention. As it makes the fault injection based testing more adequate and accessible for the testers and developers as well. The performance overhead undertaken during fault injection is negligible. Jaca is different from other software fault injection tools, because of its ability to inject faults in both low-level and high-level fault injection applications affecting Assembly language element (buses, CPU registers, etc) in the former one and the attributes and methods of objects in a Java program in the later one. It monitors the system under test, observing its behavior if it is as expected or not in the presence of faults, using a logfile [36]. A new SWIFI and monitoring environment, targeted to the modern and complex processors has been presented in [22][35], which is called Xception.It differs from other SWI FI tools [18], because of the minimum interference of the injected faults with the target application. Also they can affect any running process in the target application, because of the Xception operation which is close to the hardware. Moreover this tool can inject faults even if there is no source code. For realistic faults injection and monitoring the faults activation and their impact on the target system behavior in detail, Xception uses the advance debugging and performance monitoring features exist in most of the modern processors. One more important feature of this tool is that there are no inserted software traps and the target application is not modified. An experimental study revealed that 44 percent of the software faults cannot be emulated [26]. Another SWIFI tool described in [22] is called DOCTOR has the ability to inject memory, processor and communication faults on distributed real time system which is named HARTS. It has the ability to inject faults in the communication subsystems of their target systems. They have been used for different purposes, such as the evaluation of the distributed diagnosis algorithms and the communication faults effect in parallel applications. One more mostly used software fault injection tools is LIFTING, which is an open source simulator has the ability to simulate both logic and fault simulations for both single and multiple stuck at faults and Single Event Upset (SEU). It can be modified so as to meet the user's requirement [18]. In [27] there is a description for FERRARI tool, which has the ability to inject both transient and permanent faults. As it can inject faults in object code. Moreover, in [34] there is experimental Analysis of Binary-Level Software Fault Injection in Complex Software using one of the desired Software Fault Injection technique (G_SWIFT (Generic Software Fault Injection Technique)), which differs from other techniques because of its ability to inject faults in case of the unavailability of the source code [34].

III. Motivations of Fault Injection

Computer systems required to be highly dependable. Fault injection has been defined by Arlat as the dependability validation technique of fault tolerant systems that consists in the controlled experiments completion and observing the system behaviour in the presence of faults that induced intentionally "by the writing introduction (injection) of faults in the system".[1]. In order to provide systems with high dependability, the fault life cycle areas classified into: fault prevention, fault removal, fault tolerance and fault forecasting. Fault prevention and removal are primarily handled in the phases those are before releasing the application and mostly done by reviewing and testing processes. Fault tolerance is about providing the system with the ability to resist failures. Fault forecasting helps in understanding the remaining faults in the system and the probability of their occurrence in the future. As it calls to use of the models of software reliability which supposes that sufficient database of observed failures has been collected during the actual process for preference [21][28].Fault Injection attempts to determine if the outcome of the system matches the specifications or not, in the presence of faults that are injected in specific chosen points and states.

IV. Fault Injection Environment

A computer system that is under fault injection testing should have the following components:

Fault Injector: It is responsible for the fault injection into the target system and the commands execution, which are received from the workload generator.

Fault Library: It stores different fault types, fault locations, fault times, and appropriate hardware semantics or software structures.

Workload Generator: It is responsible for the workload generation for the target system as input.

Workload Library: It stores sample workloads/test-benches to motivate the target system.

Controller: It controls the experiment or it realizes the stimulation.

Monitor: It tracks the commands execution and initiates data collection when it is necessary. In other words it realizes the observations.

Data Collector: It is responsible for collecting the online data.

Data Analyzer: It processes data and analyzes it [16,20].

V. Types of Faults

Fault can be considered as a deviation of the required task in one of the components. It may arise during the computer system design processes (specification, design, development, manufacturing, assembly and installation). [1] The Fault type defines exactly what is damaged and how is that damage carried out.

There are many types of faults can be injected using the above-mentioned tools and techniques, listed in table II.

It is worth mentioning that there are many faults can be created during the software development. They are catalogued to: Function faults: It happens when there is missing implementation or incorrect one, which requires a design change correction. They are less frequent faults. Algorithm faults: It happens when there is missing implementation or incorrect one that can be fixed with no need to the design change. They are dominant faults. Timing/serialization faults: Missing or incorrect serialization of shared resources.

Checking fault: it Missing or incorrect validation of data, or incorrect loop, or incorrect conditional statement [7] [1]. The following table shows different types of faults:

Fault	**Type Description**
MVIV	Missing variable initialization using a value
MVAE	Missing variable assignment with an expression
MIFS	Missing IF construct + statements

MLAC	Missing AND in expression used as branch condition
MLPA	Missing small and localized part of the algorithm
WPFV	Wrong variable used in parameter of function call
MFC	Missing function call
MVAV	Missing variable assignment using a value
MIA	Missing IF construct around statements
MIEB	Missing IF construct + statements + ELSE construct
MLOC	Missing OR in expression used as branch condition
WVAV	Wrong value assigned to variable
WAEP	Wrong arithmetic expression in function call parameter

Table II: Types of faults

VI. Fault Injection Techniques

Fault injection techniques validate the dependability of fault-tolerant systems [1]. Testing of the dependability and reliability of systems is critical as systems' failures may have devastating effects especially in life-critical applications. The failure and the explosion of Ariane 501 were traced back to chain of events which includes software faults occurred on the On-Board Computer software [2]. The failure of Tritan rocket launch which costs over 3 billion dollar was due to software failure as admitted by the US Air Force [3]. Faults can also occur in medicine especially with the recent medical technologies that rely on fine engineering devices with complex software. For an analysis of software faults of medical devices; see [4]. The consequences of systems failures caused by faults make it imperative to verify the reliability and dependability of fault-tolerant systems prior to their deployment. This paper presents a concise survey of fault injection techniques (which are exposed in this section) and tools (which are exposed in the later one) that can be used as a gentle introduction to the area. Fault injection techniques can be grouped into: software-based fault injection, hardware-based fault injection, simulation-based fault injection, emulation-based fault injection and hybrid-based fault injection. In this section, there is a brief explain about the last three mentioned techniques, while in the next two sections there will be more details about the first two techniques.[1,33]

Simulation based fault injection injects faults "in high-level models (most often VHDL models)". It allows the system dependability evaluation only in case of a model of the system availability. One of its advantages is that there is no risk to damage the used system. As they need less time and effort that the hardware techniques. Moreover, they have higher observation and controlling of the system in the existence of faults.[30]

Emulation based fault injection enables the designer studying the actual behavior of the circuit in the application environment, taking the real time interactions into account. It has been presented as an alternative solution to reduce the time spent during simulation-based fault injection operations. It can be performed in both software and hardware. In hardware, there must be a reconfiguration, i.e adding hardware structures to the model because of its necessity to perform the experiments [1][30].

Hybrid fault injection This technique includes both software fault injection versatility and the hardware monitoring accuracy [1].

Hardware based fault injection

In which the actual hardware system is affected by external physical sources [17].

Hardware fault injection techniques can be grouped into two categories based on the way used to inject faults.

1) Hardware Fault Injection with Contact

The fault is injected by making direct, physical contact with the hardware by attaching/adding new components in order to introduce changes in the current or voltage of the target chip. This category contains two main methods:

•Active probes. This technique attaches probes to the pins which results in the current changes in the target chip. This technique requires special care as the current may results in damaging the hardware if not controlled carefully. The types of faults injected by this technique include stuck-at faults and bridging faults.

•Socket insertion. This technique inserts a socket between the target hardware and its circuit board which insert logic faults by changing the pin signals. The types of faults injected by this technique include stuck-at faults, open faults and logic faults.

These techniques have high level of controllability as the exact time and location of the faults can be controlled. However, there is an intrinsic risk of causing hardware damage when employing such techniques.

2) Hardware Fault Injection without Contact

The faults are injected by an external source which has no direct, physical contact with the hardware. The external source produces radiation (heavy ion) or electromagnetic fields which causes changes in the current inside the target chip. This category contains main methods

•Heavy ion radiation:

Which has a major feature which is its ability to inject faults into VLSI (Very-large-scale integration) circuits at locations which are impossible to be reached by other techniques such as software-implemented fault injection and pin-level.[32]

•Electromagnetic field: It is used by MARS (Maintainable Real-Time System) to conduct the contact less faults.[11]

These techniques imitate some natural phenomena and thus provide, to some extent, a realistic environment for testing dependability of prototypes. The downside of employing those methods is their low-level of controllability as the exact time and location of the faults cannot be controlled.In general hardware fault injection techniques have high-time resolution and low perturbation which makes them useful for verifying the dependability of system/characteristics which require high-time resolution such as CPU or characteristics that require accessing exact locations (memory). Hardware fault injection works under the assumptions that the injected faults into the system are representative to the actual faults that occur in the system. In addition to an additional physical hardware is needed to inject faults. As it is necessary mentioning that, the additional hardware does not affect the basic functional system behavior. The potential of software fault injection in validating systems dependability has been well recognized [5,6]. Software fault injection techniques are not as expensive to implement as hardware fault techniques as they do not require manufacturing hardware. Software fault injection works under the assumption that the injected faults produce a behavior similar to the behavior of the systems in the presence of real faults [1], Although it is still unknown whether this assumption. Fault injection has been used for long decades as a useful technique, in order to evaluate both software and hardware dependability, but life contains both positives and negatives and computer systems as well. [13] and [18] holds [7], it has been a general framework for testing the reliability of software [8,9]. Further, the experimental studies confirm the validity of the aforementioned assumption to a great extent. In [10], it is shown that 85 percent of the times the mutation faults generate the same error produced by real faults. However, when presenting with new software, it is hard or impossible to detect the most likely hidden errors/faults in the software and their likelihood of occurring when the software is in use; hence the fault injection technique should not be used as absolute measure of the dependability, exception handling mechanisms and risks [7][37]. However, it is well-established fault injection provides better understanding of the behavior of software. Another approach to the emulation of software faults is model-based whereby reliability models are proposed to estimate the reliability of software and to predict software faults [11][12]. Software fault injection methods can be categorized on the basis of when the faults are injected into:

1- During compile-time, where the program instruction must be modified before the program image loading and execution.

2- During run-time, where fault injection is triggered using a mechanism. There are many triggering mechanisms, but the commonly used include:

(a) Time-out:

In this type, a timer expires at a predetermined time, which triggers injection. It injects faults based on time instead of specific events or system state.

(b) Exception/trap:

In this case, the fault injector is the controller instead of the hardware exception or a software trap. Unlike time-out in which fault can be injected by the exception/trap, whenever specific events or circumstances occur.

(c) Code insertion:

There are many instructions added to the target program, in which fault injection can occur before certain instructions, much like the code-modification method. A relatively recently technique emulate software faults injects mutation faults into the executable code of the target software (instead of injecting the mutation into the source code) using mutation operators that are classified into groups based on the observations [7].

VII. Conclusion

As time progresses and life requirements increase in terms of technology, computer systems need to evolve to coincide with this progress and development. Computer systems need to be protected and provided with a high level of security so they need to be capable of carrying out the faults/errors that are exposed to them. In order to do this, they need high-effective techniques and tools. Fault injection is one of these techniques that have been widely used by developers and testers to assess the dependability of systems. It has certain techniques and tools depending on whether it will be in software or hardware. In this paper we expose many of them and compare between different tools in terms of their uses, need of source code, how they inject faults and their main function. As we compare between the fault injection techniques, which are grouped into: hardware based fault injection, software based fault injection, simulation based fault injection, emulation based fault injection and hybrid based fault injection, by exposing their advantages and disadvantages.

ACKNOWLEDGEMENT

This work has been supported by the project Deign of Intelligent Robot, sponsored by Internal Research Grant, University of Buraimi, Oman.

References:

[1] P Kumari, P Kaur,. " A survey of fault tolerance in cloud computing." Int. Arab J. Inf. Technol. 1.2 (2021):

[2] Lions,. "Ariane 5 flight 501 failure report by the inquiry board." (1996).

[3] Sobek, S. "Human error called culprit in 3 rocket launch failures." Florida Today Space Online, (1999).
[4] Wallace, "Failure modes in medical device software: an analysis of 15 years of recall data." Journal of Reliability, Quality and Safety Engineering 8.04 (2001):
[5] Christmansson, Jörgen,. "Error injection aimed at fault removal in fault tolerance mechanisms-criteria for error selection using field data on software faults." Software Reliability Engineering, 1996. Proceedings., Seventh International Symposium on. IEEE, 1996.
[6] A Uchôa, C Barbosa. " Predicting design impactful changes in modern code review: A large-scale empirical study." IEEE Software 14.4 (2021): 73-83.
[7] K Yu, Q Fu, H Ma, TR Lin, X Li. " Simulation data driven weakly supervised adversarial domain adaptation approach for intelligent cross-machine fault diagnosis" IEEE transactions on software engineering 32.11 (2021): 849-867
[8] F Oliveira, J Araujo, R Matos. Software Aging in Container-based Virtualization: An Experimental Analysis on Docker Platform.No. UILU-ENG-94-2231. ILLINOIS UNIV AT URBANA CENTER FOR RELIABLE AND HIGH-PERFORMANCE COMPUTING, 2021.
[9] H Xiong, C Jin, M Alazab, KH Yeh, et al. " On the design of blockchain-based ECDSA with fault-tolerant batch verication protocol for blockchain-enabled IoMT." IEEE Transactions on Reliability 42.2 (2021): 190-204.
[10] Daran, Murial, and Pascale Thévenod-Fosse. "Software error analysis: A real case study involving real faults and mutations." ACM SIGSOFT Software Engineering Notes. Vol. 21. No. 3. ACM, 1996.
[11] Lyu, Michael R. " Prediction of software reliability." (2021): P-3-25.
[12] Khoshgoftaar, Taghi M., et al. "Process measures for predicting software quality." High-Assurance Systems Engineering Workshop, 1997, Proceedings. IEEE, 1997.
[13] JAMES H. BARTON, EDWARD W. CZECK, ZARY Z. SEGALL, AND DANIEL P. SIEWIOREK , Fault Injection Experiments Using FIAT
[14] Eliane Martins, Cecilia M.F. Rubira, Nelson G.M.Leme, Institute of Computing – State University of Campinas (UNICAMP) Jaca: A reflective fault injection tool based on patterns.
[15] Laura L. Pullum, Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud. Artech House, Boston London.2019
[16] C Bozzato, R Focardi, F Palmarini: optimizing voltage fault injection attacks,2019.
[17] A Mahmoud, N Aggarwal, A Nobbe, Pytorchfi: A runtime perturbation tool for dnns.2020
[18] GJ Uriagereka, Design-time safety assessment of robotic systems using fault injection simulation in a model-driven approach. 2019
[19] MM Abaei, R Abbassi, Reliability assessment of marine floating structures using Bayesian network.2018.
[20] E Kaja, NO Leon, Extending Verilator to Enable Fault Simulation , Wolfgang Ecker, and Thomas KrusInfineon Technologies AG - 85579 Neubiberg, Germany Technische Universit¨at M¨unchen,2021.
[21] A. Aviiienis, H. Kopetz, J. C. Laprie, Dependable Computing and Fault-Tolerant Systems, volume 4, page(24-30),2019.
[22] João Carreira, Henrique Madeira, and João Gabriel Silva Dep. Engenharia Informática Universidade de Coimbra, Portugal, Xception: A Technique for the Experimental Evaluation of Dependability in Modern Computers.2017.
[23] RK Lenka, S Padhi,, Fault Injection Techniques-A Brief Review.IEEE,2018
[24] D. Gil, R. Martínez, J. C. Baraza, J. V. Busquets, P. J. Gil, Fault Injection into VHDL Models: Experimental Validation of a Fault Tolerant Microcomputer System .2010.
[25] Nicolas Guelfi, Henry Muccini, Patrizio Pelliccione, Alexander Romanovsky, 4 September Dubrovnik, CROATIA Workshop Proceedings.
[26] Henrique Madeira .On the Emulation of Software Faults by Software Fault Injection, Marco Vieira.2011.
[27] FERRARI: A Flexible Software-Based Fault and Error Injection System, Ghani A. Kanawati, Nasser A. Kanawati, and Jacob A. Abraham, Fellow, IEEE.
[28] K. Umadevi, S. Brintha Rajakumari,A Review on Software Fault Injection Methods and Tools, International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization) Vol. 3, Issue 3, March 2015.
[29] Paul D. Marinescu and George Candea, LFI: A Practical and General Library-Level Fault Injector, Proceedings of the Intl. Conference on Dependable Systems and Networks (DSN), June 2009.
[30] Roberta Piscitelli, Shivam Bhasin and Francesco Regazzoni, Fault Attacks, Injection Techniques and Tools for Simulation, Chapter2, page(17)
[31] M. Rimén, J. Ohlsson, J. Karlsson, E. Jenn, J. Arlat, Design Guidelines of a VHDL-based Simulation Tool for the Validation of Fault Tolerance.
[32] Johan Karlsson, Peter Folkesson, Jean Arlat, Yves Crouzet, Günther Leber, Johannes Reisinger, Application of Three Physical Fault InjectionTechniques to the Experimental Assessment of the MARS Architecture.
[33] VHDL SIMULATION-BASED FAULT INJECTION TECHNIQUES, D.Gil, J.C Baraza, P.J.Gil, page 2.2015.
[34] Nadir K Salih, Tianyi Zang. Variable service process for SaaS Application. Research Journal of Applied Sciences, Engineering and Technology. vol. 4, Issue 22, 2012, pp 4787-4790.
[35] Nadir K Salih, Tianyi Zang, Mingrui Sun. Multi-database in healthcare network. IJCS, vol. 8, Issue6, No3, 2011, pp 210-214.
[36] Regina Lúcia de Oliveira Moraes Eliane Martins, Jaca - A Software Fault Injection Tool, Proceedings of the

2003 International Conference on Dependable Systems and Networks (DSN’03) 0-7695-1959-8/03.IEEE.2003
[37] Nadir K Salih, Tianyi Zang. Survey and comparison for Open and closed sources in cloud Computing. International Journal of Computer Science Issues, vol. 9, Issue3,No1,2012,pp118-123.

https://youtu.be/bpho5XPY604