RESEARCH ARTICLE OPEN ACCESS

Designing a Dump-Collection-and-Validation Automation Software

Swathi N R*, Mamatha T**

*(Computer Science and Engineering, R.V College of Engineering, India
Email: swathinr.cs17@rvce.edu.in)

**(Computer Science and Engineering, R.V College of Engineering, India
Email: mamathat@rvce.edu.in)

--************************----------------------------------

Abstract:
Dumps help in understanding the state of a computer server (system) during a crash. Accurate dump data can help debug the faulty system and pinpoint the reason for the failure. Processing dump contents to extract useful data can be an extremely complicated and time-consuming task. This task can be automated using software tools to reduce time and effort for developers and testers. In this paper, we discuss the design of such software. We propose a technique that involves forcibly making a system crash to generate dump data. The overarching idea is to process this dump data by parsing and matching them with the list of fields from a directory.

***Keywords* — system dump, parser, error injection, active cores, stress application, logical partitions.**

--************************----------------------------------

I. INTRODUCTION

A system dump is a state of a system at a given instance. A platform system dump captures information about the system hardware and optionally, portions of the hypervisor memory in case of system crash or failure. The information collected in system dump is very critical and is used for debugging the system. This time-consuming process needs to be automated to save effort and time spent by testers. With the new software, we can capture the system dump and verify its contents automatically. The system is setup, and the required error is injected into the system. Once the dump is generated, it is stored in a disk storage. The contents of the dump are validated using parsing tools. The data is cross verified with a directory and the system is reset to the original state.

II. SYSTEM ARCHITECTURE

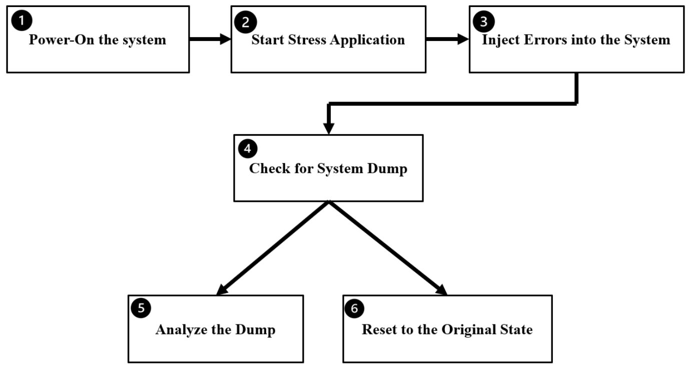
1 Power-On the system
2 Start Stress Application
3 Inject Errors into the System
4 Check for System Dump
5 Analyze the Dump
6 Reset to the Original State

Fig.1Flow of events between modules in the Dump-Collection-and-Validation system

The system architecture includes 6 phases. Dump collection from the server is tested by following the steps as shown. The arrows indicate the flow of events between phases.

A. *System Setup Module*

This module performs the following functionalities:

- Boots the system.
- Verifies if the hypervisor is up and running.
- Creates logical system partitions and install OS on the logical partition.

It takes as input the name of the machine and the logical partition in the machine that needs to be turned on. Once the system starts to power on, it goes into booting mode. After the completion of the system boot, the system goes into the operational state where it is ready to be used.

B. *Stress Application Module*

This module performs the following functionalities:

- Starts the stress application to exercise the hardware units.
- Lists the set of active cores with their respective node numbers.
- Picks an active core from this list and sends it to the error injection module.

The system is stress tested by exercising all or selected hardware components of the system to uncover any potential flaws or issues. The application takes as input the details of the system. The same application is used to find the active cores of the processor by scanning through the hardware of the system.

C. *Error Injection Module*

This module performs the following functionalities:

- Injects various kinds of error in different locations of the system.
- Restarts the System after every error.

The Error Injection module helps to inject error into the system using commands for specific dump initiations. The type of the dump and the site of injection as given by the stress application is used to inject the appropriate error into the system. This triggers the dump in the system which generates the dump file which is collected in the storage.

D. *Dump Collection Module*

This module performs the following functionalities:

- Collects and stores the dumps on a disk.
- Sends the dump to be analysed.

The Dump Collection module regulates and manages the collection of dump data in the storage. The error which is injected causes the dump to be generated and stored. The module also makes sure that the system is on and does not allow restart of the system which might cause incomplete collection of dump data.

E. *Dump Analyzer Module*

This module performs the following functionalities:

- Verifies if all the data has been saved in the dump file.
- Validates the contents of the dump by matching it with the directory using the parsing tools.

The Dump Analyzer Module uses parsing tools to parse the data stored in the dump file. After parsing the file, tokens are matched with the data in the directory. The directory has the list of all allowed field names. The value of the fields in the dump need to match that of the directory.

F. *System Reset Module*

This module performs the following functionalities:

- Cleans up the logs.
- Restarts off the system.

The system needs to be reset after the above steps. The state of the system needs to be changed from error state to operational state. Once this is done, on cycle of testing is complete.

III. DATA FLOW DESIGN

Data Flow Diagrams (DFDs) are used to understand the flow of data in the system. It shows how the data is getting transferred between different modules. It consists of 4 elements namely data flow, process, external entity, and storage. It gives the end users an abstract understanding of the data provided to the system as input. The level-0, level-1, and level-2 DFDs are as shown below.

A. Level-0 DFD

The level-0 DFD describes general operation of the system. It represents the system and user and the inputs and outputs between the user and the system.

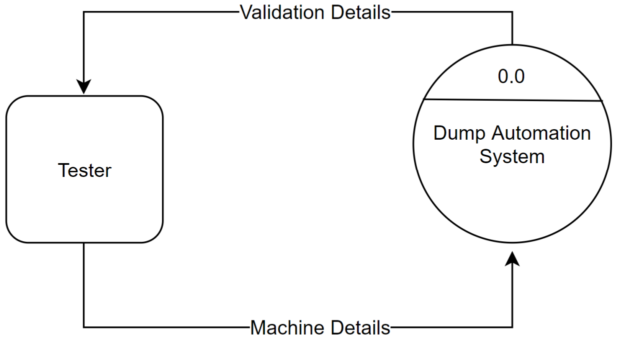

Fig. 2 Level-0 Data Flow Diagram

The abstract function of the Dump Automation System shown in Fig 2, is to provide the developers and testers a centralized platform to check the validity of the dump data generated by the system. The detail about the machine is sent and in return the validation details are received by the software.

B. Level-1 DFD

The level-1 DFD describes the system more in detail than the level-0 DFD. It specifies the main modules involved in the system. This is as shown in the Fig 3. The level-1 DFD illustrates the data flow through various stages in the system.

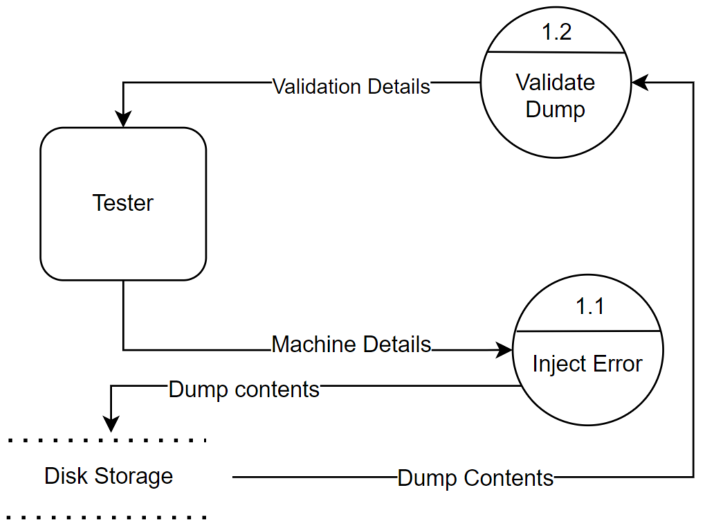

Fig. 3 Level-1 Data Flow Diagram

Module 1.1: The Inject Error Module takes as input the details of the machine that needs be injected with the error from the tester. After the error is injected, the system goes in error state and starts collecting system dump in the Linux storage.

Module 1.2: The Validate Dump Module takes as input the contents of the dump generated by the system and validates its contents It sends the status back to the user who initiated the software.

C. Level-2 DFD

A level-2 DFD offers a more detailed look at the processes that make up an information system than a level-1 DFD does. It can be used to plan or record the specific makeup of a system. This is shown in Fig4.

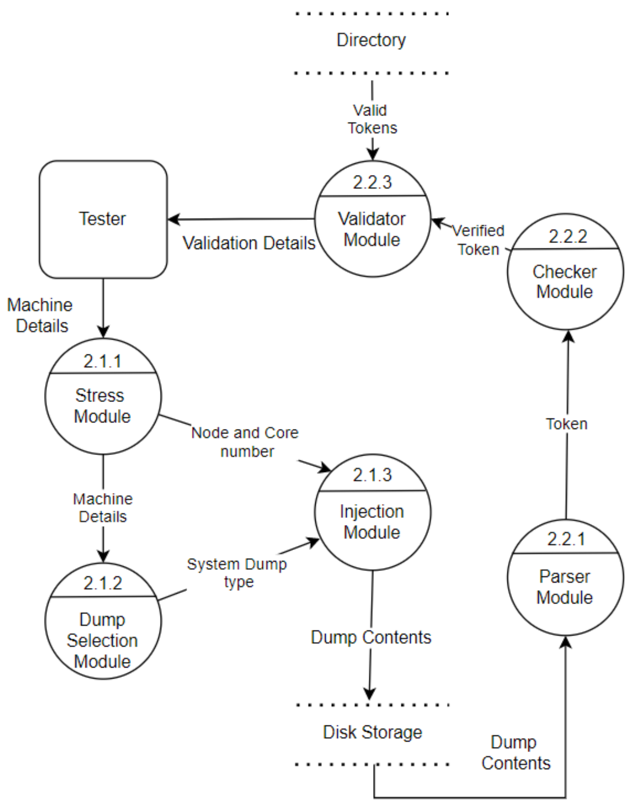

Fig. 4 Level-2 Data Flow Diagram

Module 2.1.1: The Stress Module starts the stress application which identifies the active cores in the system. One of the cores are picked randomly and sent to the next module.

Module 2.1.2: The Dump Selection Module selects the dump that needs to be generated next. It sends the details of the type of dump to the Injection Module.

Module 2.1.3: The Injection Module injects the error into the system by taking the type and the core details from the previous two modules.

Module 2.2.1: Parser Module uses the parsing tool to parse through the generated dump. It sends tokens to the Checker Module.

Module 2.2.1: The Checker Module takes token and checks if the details are complete with all fields filled.

Module 2.2.3: The Validator Module validates filled fields by checking with the allowed list of values from the directory. It sends the validated details.

IV. STRUCTURE DESIGN

The design of the system is described using a structure chart. It describes the structure of the system along with data and control flow. Data flow is shown using empty arrows and control flow is depicted using filled arrows. The structure chart is as shown in Fig 5.

Modules to inject error, collect dump and validate the dump are called by the main interface. Dump type as well as node and core numbers are picked by stress application. The dump validate module calls submodules to parse, check and verify the fields of the dump file.

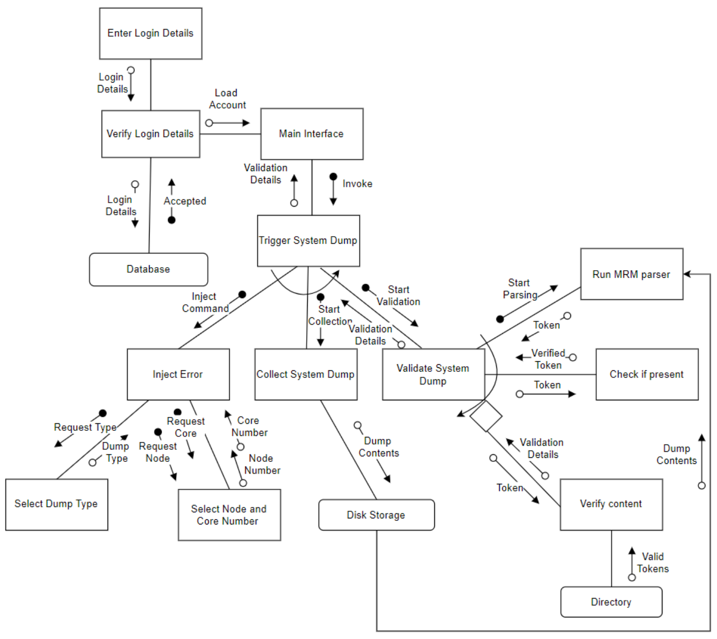

Fig. 5 Structure chart for the system

Disk storage helps in saving the contents of the dump file while the directory helps store valid fields for dump data. Validated details from the system is received back to the user after going through one complete dump file.

V. CONCLUSION AND SUMMARY

Evaluation metrics are the criteria for testing different algorithms. The evaluation metric accuracy can be used here. It is defined as the ratio of number of correct outcomes to total number of outcomes. The contents of the dump need to be completely accurate to be able to recover the system that has failed.

The input to the software consists of the name of the system along with the name of the logic partition whose dump that needs to be validated. Information like node number and core number are picked randomly from the list of active cores in the processor. The crash can happen due to various reasons in a system. Hence all kinds of errors need to be simulated to verify the system dumps under all kinds of circumstances.

The process of automation will stop the need for manual verification. The software will cater to the need of the user for easy dump collection and validation technique to save time and effort spent by its employees. The software can also test the quality of the dump generation software. With such technologies, the complete dump automation testing can be completed in a single click, reducing manpower.

REFERENCES

[1] J. K. Ousterhout, "Scripting: higher level programming for the 21st Century," in Computer, vol. 31, no. 3, pp. 23-30, March 1998J. Breckling, Ed., *The Analysis of Directional Time Series: Applications to Wind Speed and Direction*, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.

[2] Aryanti Aryanti and Ade Silvia Handayani and Ibnu ziad and Ikhthison Mekongga and Farid Jatri Abiyyu , "Compatibility of Linux Architecture for Diskless Technology System", Proceedings of the 4th Forum in Research, Science, and Technology (FIRST-T1-T2- 2020), Jan 2021.

[3] Lucas Correa, Fabian Vargas, Letícia Poehls, "Hardware-based approach to guarantee trusted system boot in embedded systems", Microelectronics Reliability, Volumes 100–101, 2019

[4] Jun Xu, Dongliang Mu, Ping Chen, Xinyu Xing, Pei Wang, and Peng Liu. "CREDAL: Towards Locating a Memory Corruption Vulnerability with Your Core Dump" in the Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). Association for Computing Machinery, New York, NY, USA,2016, pp. 529–540

[5] S. Giraddi, S. Yaligar and H. S. Kavitha, "Problem and Project-Based Learning in Scripting Lab," 2016 IEEE 4th International Conference on MOOCs, Innovation and Technology in Education (MITE), Madurai, 2016, pp. 152-156

[6] Q. Meng, S. Wu, and Y. Shang, "Simulation and Optimization of Large Aircraft Landing Gear System based on AMESim and Python Script," 2019 IEEE 8th International Conference on Fluid Power and Mechatronics (FPM), Wuhan, China, 2019, pp. 1328-1333

[7] J. K. Ousterhout, "Scripting: higher level programming for the 21st Century," in Computer, vol. 31, no. 3, pp. 23-30, March 1998

[8] Aryanti Aryanti and Ade Silvia Handayani and Ibnu ziad and Ikhthison Mekongga and Farid Jatri Abiyyu , "Compatibility of Linux Architecture for Diskless Technology System", Proceedings of the 4th Forum in Research, Science, and Technology (FIRST-T1-T2-2020), Jan 2021

[9] Lucas Correa, Fabian Vargas, Letícia Poehls, "Hardware-based approach to guarantee trusted system boot in embedded systems", Microelectronics Reliability, Volumes 100–101, 2019.

[10] Jun Xu, Dongliang Mu, Ping Chen, Xinyu Xing, Pei Wang, and Peng Liu. "CREDAL: Towards Locating a Memory Corruption Vulnerability with Your Core Dump" in the Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). Association for Computing Machinery, New York, NY, USA,2016, pp. 529–540

[11] S. Giraddi, S. Yaligar and H. S. Kavitha, "Problem and Project-Based Learning in Scripting Lab," 2016 IEEE 4th International Conference on MOOCs, Innovation and Technology in Education (MITE), Madurai, 2016, pp. 152-156

[12] Q. Meng, S. Wu, and Y. Shang, "Simulation and Optimization of Large Aircraft Landing Gear System based on AMESim and Python Script," 2019 IEEE 8th International Conference on Fluid Power and Mechatronics (FPM), Wuhan, China, 2019, pp. 1328-1333

[13] Izraelevitz J., Mendes H., Scott M.L. , "Linearizability of Persistent Memory Objects Under a Full-System-Crash Failure Model" in: Gavoille C., Ilcinkas D. (eds) Distributed Computing. DISC 2016. Lecture Notes in Computer Science, vol 9888. Springer, Berlin, Heidelberg, 2016

[14] Yijian Huang, Haibo Chen, and Binyu Zang. "Optimizing crash dump in virtualized environments" in Proceedings of the 6th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments (VEE '10), 2010. Association for Computing Machinery, New York, NY, USA, 25–36.

[15] G. A. Kanawati, N. A. Kanawati and J. A. Abraham, "FERRARI: a flexible software-based fault and error injection system," in IEEE Transactions on Computers, vol. 44, no. 2, pp. 248-260, Feb. 1995, doi: 10.1109/12.364536.

[16] T. Mori et al., "A Power, Performance Scalable Eight-Cores Media Processor for Mobile Multimedia Applications," in IEEE Journal of Solid-State Circuits, vol. 44, no. 11, pp. 2957-2965, Nov. 2009, doi: 10.1109/JSSC.2009.2028936.

[17] T. Wissink and C. Amaro, "Successful Test Automation for Software Maintenance," 2006 22nd IEEE International Conference on Software Maintenance, 2006, pp. 265-266, doi: 10.1109/ICSM.2006.63.

[18] V. Garousi and F. Elberzhager, "Test Automation: Not Just for Test Execution," in IEEE Software, vol. 34, no. 2, pp. 90-96, Mar.-Apr. 2017, doi: 10.1109/MS.2017.34.