A Survey of Approaches to Healing in Computing Systems

Abhishek Bhavsar, Ameya More, Chinmay Kulkarni, Dheeraj Oswal, Jagannath Aghav
Department of Computer Engineering and Information Technology
College of Engineering, Pune

***Abstract*—Computing systems are prone to errors and faults and a large amount of time is wasted in maintaining the system and bringing it back to a stable state after a fault. This maintenance currently is handled by human resources in the cloud computing architecture. As a result of this a lot of resources in the form of money and manpower and efforts in the form of man months are wasted. We review the basic technologies available in the healing of computing systems in this document. This will give us an insight of features each of the approaches provide. We compare each of the available approaches based on identified criteria. These criteria help us to weigh the various approaches and identify pros and cons, hence giving us a clear understanding of the healing methodologies.**

***Keywords*—Autonomic Computing, Self-Healing Systems, Fault Tolerance**

I. Introduction

The advent of computers kicked off a race for development of a unified computing platform that could provide for a centralised computing facility, large and reliable storage and increased accessibility.[1] This race brought about revolutionary technologies like Grid Computing, Clustered Systems, Distributed System and many others. Cloud Computing is the most recent development in this field and shows a lot of promise.

II. Motive and Objective

Cloud computing is the next clustered computing architecture that supports distributed computing. It allows for dynamic creation and tear-down of computing infrastructure. It provides immense computing power without the specific need for maintaining and designing the infrastructure. Cloud Computing though promising is relatively nascent. The distributed architecture that forms the backbone of a Cloud Computing system is prone to faults. Thus, to improve the reliability of such systems, it is essential to detect and avoid faults in such systems. There is, therefore, a need to develop methods that will efficiently monitor cloud computing systems for faults and also correct them if possible. The motive of the paper is to shed light on the existing healing and autonomic computing methods for fault detection and healing. The paper also proposes a holistic approach to tackle faults and implement healing technologies. The new architecture proposed, makes use of process specific characteristics to reduce statistical data gathering while still achieving fault healing.

III. Literature Review

A. Autonomic computing system with model transfer

A single normalized view of information is synthesized by an autonomic manager. This autonomic manager receives different data and commands from different devices.[2] The desired state of the managed resource is compared with the normalized view of this disparate data, which is also used to identify the actual state. Configuration is adjusted to reach the desired state in case that there is a difference. In the case that there is a match between the two states, i.e. the actual state and the desired state, a definite action is taken in the form of maintenance procedures associated with the resource that is being managed. In case of a mismatch dependency processing, machine learning and action determination logic is used to decide the corrective action. This patent also claims a policy rule which is used to identify and choose between the different data and commands that are received from the managed resource.

B. Autonomic Architecture for a Self-Healing Computer System

In this patent, a self-healing processor and an error mitigation system is used to constitute a self-healing system. This self-healing processor also consists of a dynamic signature analysis circuit and a code block which is associated with the operation of a section of digital logic. This portion of digital logic is executed by the self-healing processor.[3] A dynamic signature is generated by the dynamic signature analysis circuit. This dynamic signature represents the operation of the section of digital logic which is associated with the code block. This dynamic signature is then passed on to the error mitigation system by the dynamic signature analysis system, after which the error mitigation system compares this dynamic signature with a static signature in order to determine whether the signatures match. The results that are obtained after this comparison is performed by the error mitigation system are essential in confirming the detection of error. This comparison can either conclude that that the signatures do not match, in which case, the digital logic associated with the code block encompasses an error. In this case, the error mitigation system tries to execute the code block once again. These events are stored by the error mitigation system in the form of log information. On the other hand, if the error mitigation system concludes that the signatures do match, then there is no error in the code block. Thus, the error mitigation system is the most important part of the entire architecture, forming its core. The error mitigation system contains a large number of components that are required for finding out the errors and correcting

them. This entire process involves the use of multiple learning techniques as well as methods that rely on the use of lookup tables. There is a continuous interaction between the error mitigation system and the self-healing processor in the form of message passing. This message passing is essential for the error mitigation system to effectively detect errors, though such interactions make this architecture resource-intensive. The intensive interactions between the self-healing processor and the error mitigation system also considerably affects the efficiency of the overall self-healing system.

C. System and Method for Achieving Autonomic Computing Self Healing Utilising Meta Level Reflection and Reasoning

In this patent, self-healing is achieved through autonomic computation by utilizing meta-level reflection procedures. This architecture comprises of two main levels. The Meta level receives reification messages from a monitor present in the Base level. This monitor detects an error in a production environment after which it is in a position to generate these reification messages.[4] The Meta level consists of a reasoning system which receives the reification messages and then analyses the data using the knowledge of computational components present in the Base level. This analysis is required for identifying a self-healing action for the error that has been detected. The reasoning system then uses the results of this analysis to return a reversion message to the Base level. The reversion message consists of a signal which is used to carry out the self-healing action. Once the Base level receives the signals of the reversion message from the reasoning system of the Meta level, it then executes the corrective action, thus implementing the self-healing action designed by the reasoning system. Thus the Meta level is the decision making level which decides the appropriate action to be taken in order to carry out the self-healing process. The Base level simply implements these actions as received from the reasoning system of the Meta level. The use of the Meta Level allows a level of abstraction and makes the Base level less complicated thus making it modular which makes the architecture flexible. Meta Level also enables a state of transparency in the architecture by transparently suggesting methods to avoid a fault. Due to the separation between the Base level and the Meta level, the Meta level implementations can be independent of the Base level, thus allowing changes in the meta level without affecting the Base level. This modularity of the Meta Level and Base Level is very advantageous as changes in either level can be made without affecting the working of the other level. Another advantage of this architecture is that message passing between the Meta level and the Base level is minimal. The reason for this is that the Meta level is idle most of the time, only getting triggered when an anomaly is detected in the Base Level.

D. Checkpointing

To be done[5]text [6] text[7]

IV. Conclusion

....To be done...

TABLE I. Comparison of the various approaches

Features	**Autonomic Computing System with Model Transfer**	**Architecture for a Self Healing Computing System**	**System and Method for Achieving Autonomic Computing Self Healing Utilising Meta Level Reflection and Reasoning**
Comparison Parameters	Sensor State Comparison	Dynamic Signature Comparison	Reification Message
Separate Level used	No	No	Yes-Meta Level
Action Recommendation Logic	Learning based on creating models and mapping them to ontology includes semantic processing, dependency processing and action determination.	Static rules based. Uses Lookup table to map fault to strategy. Also has alternate configuration in case the basic strategy fails and a retry module. Strategy Optimisation is proposed not implemented.	Uses a reasoning system. Job is divided into two parts - detection (Base level) and action recommendation (Meta level). Transparency provided because of Meta level.
Architecture specific requirements	Sensors	Error Mitigation System	Reasoning Expert System
Transparency	No	No	Yes- Multilevel
Feasibility	Excessive use of sensors. Too much of information collected, requires comparable hardware to process so much information.	Uses a specific processor that executes a code block and needs to interact with error mitigation system. Too much message passing	Meta level is idle most of the time. Fewer messages passed. Base level only sends messages when error is detected. Strategy can be easily decided and changed without modifications to the Base level.
Dynamic Learning	Yes	No	Possible

References

[1] A. Oliner, "What supercomputers say: A study of five system logs," in *Proc. of DSN 2007*.

[2] J. C. STRASSNER and B. J. MENICH, "Autonomic computing method and apparatus," Patent, 06 2009, uS 7542956. [Online]. Available: http://www.patentlens.net/patentlens/patent/US_7542956/en/

[3] R. D. MELEN, N. W. MOUSSA, M. HONDA, H. IKAI, and K. KATO, "Architecture for a self-healing computer system," Patent Application, 11 2010, uS 2010/0281134 A1. [Online]. Available: http://www.patentlens.net/patentlens/patent/US_2010_0281134_A1/en/

[4] C. W. FELLENSTEIN, C. P. GUSLER, R. A. HAMILTON, II, and M. IBRAHIM, "System and method for achieving autonomic computing self-healing, utilizing meta level reflection and reasoning," Patent, 08 2007, uS 7260743. [Online]. Available: http://www.patentlens.net/patentlens/patent/US_7260743/en/

[5] R. Gioiosa, J. Sancho, S. Jiang, and F. Petrini, "Transparent, incremental checkpointing at kernel level: a foundation for fault tolerance for parallel computers," in *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, 2005, pp. 9–9.

[6] A. Oliner, L. Rudolph, and R. Sahoo, "Cooperative checkpointing theory," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, 2006, pp. 10 pp.–.

[7] M. Schulz, G. Bronevetsky, R. Fernandes, D. Marques, K. Pingali, and P. Stodghill, "Implementation and evaluation of a scalable application-level checkpoint-recovery scheme for mpi programs," in *Supercomputing,*

2004. Proceedings of the ACM/IEEE SC2004 Conference, 2004, pp. 38–38.