Autonomic Computing and Reliability Improvement

Yuan-Shun Dai
Department of Computer and Information Science, Indiana University, Purdue University, Indianapolis, IN, USA. (Email: ydai@cs.iupui.edu)

Abstract

The rapidly increasing complexity of systems is driving the movement towards autonomic systems that are capable of managing themselves without the need for human intervention. Without autonomic technologies, many conventional systems suffer reliability degradation due to the accumulation of errors. The autonomic management techniques break the traditional reliability degradation trend. This paper comprehensively describes the roles and functions of various autonomic components, and systematically reviews past and current technologies that have been developed to address the specific areas of the autonomic computing environment. A new mechanism of model-driven autonomic management is further presented, which is more intelligent and efficient.

1. Introduction

The term *autonomic computing* [1] is intentionally chosen because the idea is to mimic the autonomic nervous systems found in biology. The systems self-adapt, self-heal, and self-protect. Because of the increasing complexity of computing systems, human systems management is rapidly becoming obsolete. Humans are simply not able to optimally configure these large, complex, heterogeneous, and dynamically evolving systems in an effective manner. Therefore, there is a strong need to move away from human managed systems to autonomic managed systems.

Reliability is a measure of trustworthiness of a computing system, which can be defined as the probability for component, communication, service or system to successfully achieve their tasks/objectives. The reliability has been extensively studied for about half century and many theories and models have been presented for various conditions, see e.g. Xie *et al.* [2].

The autonomic computing is related to reliability enhancement, and also the reliability models will further improve current autonomic technology to a more predictable, measurable, and effective manner.

This paper introduces and classifies the autonomic technologies that will enhance the reliability, including the self-adaptation, self-healing and self-protection. Following that, we further propose a new autonomic management mechanism driven by reliability models. This model-driven mechanism depicts an outline sketch to further improve the autonomic technology.

2. Reliability and Autonomic Computing

This section identifies the technologies and theories that have been proposed to be useful in an autonomic computing environment focusing on those things that will enhance the reliability of such a system. The three areas in autonomic computing related to reliability improvement will address are: (1) Self-adaptation, (2) Self-healing, and (3) Self-protection.

2.1. Self-Adaptation

Self-adaptation refers to the ability of the system to automatically adapt to changes in the physical topology of a system as well as changes in the applications that run on the system. Many techniques have been presented for the purpose of self-adaptation, which can be classified by adaptive domains, aspect-oriented programming, and utility optimized adaptation.

Adaptive domains are actually a hierarchical system. Each domain is separately managed based on policies loaded from a parent domain during runtime. A Host Manager oversees the operation of the Domain Manager which manages parent and child nodes of a specific domain. The domain manager then is tied to a specific Managed Object which may be a sensor that is responsible for one or more domains.

The idea using Aspect-Oriented Programming is to implement a monitor system outside the OS kernel or applications. Application being monitored is connected to sensors and effectors of the autonomic manager via an aspect crosscut layer. The effectors self-adapt if the sensors suggest corrective action need occur.

Utility Optimized Adaptation uses the economic theory of utility to determine how resources should be allocated on competing resource consumers. Utility theory analyzes what the consumer is willing to give up something for his need. Individual utility curves are

Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)
0-7695-2356-0/05 $ 20.00 **IEEE**

constructed to form an aggregate utility curve which can be fitted against the available resources to best adapt configurations that can maximize overall utility.

The self-adaptive techniques improve system reliability by eliminating or dramatically reducing the need for human operations, as human configuration of complicated systems can often be error prone. Moreover, the human configuration for large and complex systems may be much ineffective, which can cause timing failures due to the dynamic property of these systems, so the self-adaptation is also beneficial to the reliability from this perspective. The self-adaptation can also improve the fault-tolerance of the system, e.g. if a component used by a service is in an erroneous state, then the self-adaptive scheme will automatically configure some similar components found online as backups in maintaining this service.

2.2. Self-Healing

Self-healing is concerned with the ability of the system to automatically recover from faults, which is directly related to reliability growth. Accurate detection is critical to designing an effective self-healing system. Fault detection is accomplished by some form of monitoring. The question is what parameters to monitor, how to determine accurately if a fault has occurred or is likely to occur, and what corrective measures can be taken to repair the system unobtrusively. The idea is to repair only the component that has failed without bringing down the entire system so that resource availability is maintained even if the QoS is somewhat degraded. Some self-healing methods are introduced in the following.

Aspect-Oriented Programming can also be used for self-healing in addition to self-adaptation. The basic approach is same that source code need not be accessed to implement the system. For self-healing application, the monitored parameters are those that are associated with fault-detection rather than configuration. Corrective action takes place if a monitored parameter indicates that a fault is occurring or about to occur.

Patterson *et al.* [3] presented the Recovery Oriented Computing which focuses on MTTR rather than MTTF in order to provide higher system availability. The use of data clustering to analyze successes and failures provides for the discovery of the combination of components that are most highly correlated with the failed requests. Once failures have occurred how is recovery initiated? Rather than perform a hard reboot on the system, a recursively restartable system that gracefully tolerates successive restarts at multiple levels is used. Also, fine grain partitioning of the system enables bounded, partial restarts that recover a failed system faster than a full reboot. It also enables strong fault containment and diagnosis providing for enhanced system reliability.

Another method is *remote healing* whereby the system utilizes an architecture that supports monitoring and repair actions on a remote operating system or application memory without using the processor(s) of the target machine. To support remote healing a computer system must be equipped with a backdoor, a specialized network interface that allows external accesses to its resources without involving its processors. Backdoors enable intervention on a system even when it is "dead." There must be generic support by the OS to enable remote healing via a backdoor to allow remote access to system memory. Backdoors can be implemented using remote memory communication.

Machine leaning is also proposed as a self-healing approach for the system to "learn" series of events that lead to failures thereby providing prediction and system manage process control. Time-series algorithms, rule-based algorithms and Bayesian network techniques are mainly used for the machine learning.

2.3. Self-Protection

The self-protection technique of an autonomic computing system is concerned with protecting the system from malicious attacks. Most of these attacks are assumed to be generated from external sources, but the system must be prepared to address attacks that are initiated from within the system as well. Examples of types of attacks that should be monitored fall into three basic categories: (1) Denial of Service; (2) Viruses and Hackers; and (3) Application level attacks or failures.

One way of self-protection is to dynamically modify ones' defensive posture in an attempt to hinder the adversary's intelligence gathering process to decrease system vulnerability. Experiments sought to substantiate, "autonomic response can improve overall assurance by thwarting an attack while it is underway."

A feedback mechanism is also used for self-protection, considering the tradeoffs of compromised information systems resulting from "false negatives" and the maintenance costs of unnecessary ongoing defensive countermeasures that result from "false positives." It combines online implementation using sensors to detect intrusions and an offline component storing models and numerical optimization utilities. The approach makes heavy use of probabilistic models for decision making. The offline component performs the mathematical, probabilistic computations for decision making that is then stored in an online module to determine if the system state is normal.

The self-protection techniques are good for both reliability and security. To protect the system from inner faults of components is related to reliability

issues while to protect the system/information from outer attacks is related to security issues.

3. Model-Driven Autonomic Management

Here, a new model-driven scheme for autonomic management is further presented, based on various reliability models, see e.g. Xie *et al.* [2].

3.1. Why Model-Driven?

Most prior autonomic computing techniques rely on monitoring effort evenly distributed all over the system. In fact, different services are not equally likely to fail at a same time, so evenly distributing the monitoring effort is neither economic nor efficient.

The model-driven autonomic management can better allocate the resources by using the reliability models to predict and direct the distribution of monitoring efforts. By the models, if certain services or components are predicted to have high reliability at a time, then there is no need for intensive monitoring during that period. On the other hand, those with low reliability require more intensive monitoring. Combining with the models, the autonomic computing becomes more intelligent.

The models can also guide some automated optimization. The evaluated/predicted reliability can be input into an optimization model for system design, control, configure, allocation problems, in order to maximize reliability, minimize cost, or other objectives.

The models can be further combined with the machine learning process to predict the failure events and the causes related to them. The correctness or errors of prediction can feedback to adjust the models and parameters as a learning process. Based on the prediction, certain preventative maintenance can be conducted to avoid the failures from real occurrence.

3.2. The Model-Driven Mechanism

To implement the model-driven autonomic manage, it proposes to include a library of reliability models, hierarchical modeling scheme, real-time and long-term data collection, model selection and adjustment, and finally combined with other autonomic technology.

For the model-driven mechanism, a library of reliability models should be preset. This library will contain many reliability models at various levels, such as component-level (such as software/hardware), network-level (communication channels), service-level (combination of multiple components and communication channels), and system-level (involve multiple services). This library is used to provide sources for any component/service/system to select suitable models based on the goodness of fit.

The hierarchical modeling scheme will be applied, which means models of the higher service/system-level can be integrated from the lower component-level models. Such hierarchical modeling will be much efficient because the component models can be built in parallel by sensors when monitoring the components, and the service/system-level model can be derived by just substituting the lower-levels' models/parameters in.

The data need to be collected for two purposes: real-time system monitoring and optimization; and long-term observation to discern and archive patterns of model types and parameterization. The goodness of fit will be used to select/adjust models.

The hierarchical models are corresponded to the hierarchical control of autonomic computing. The component-level models can help sensors to monitor and report more effective and intelligent. E.g., highly reliable components are not required to be intensively monitored. The models also prepare for the sensors not to always report, which saves network bandwidth. The sensors will not report to higher-level agents if the data being monitored is in a range of the model prediction. If abnormal data appear, the sensors will report it and then analyze the causes. If such abnormal condition is persistent, which means the models are not fitted, then model adjustment is required.

For higher-level control, the higher-level models are useful not only to direct the monitoring but also to help make optimal decisions. Moreover, if the possible causes for the latent failures can be determined as well, some preventative maintenance can be conducted and remove the causes in advance. To make optimal decisions [2: pp. 239-274].

4. Conclusion

This paper has presented a variety of existing ideas, models, and technologies that encompass the self-adaptation, self-healing, self-protect components of such a system in an effort to identify promising areas that will improve reliability. Furthermore, this paper presented a new model-driven autonomic management mechanism which makes the autonomic computing more intelligent, efficient, and economic.

References

[1] J. Kephart, and D. Chess, "The Vision of Autonomic Computing", *IEEE Computer,* January 2003, pp. 41-50.

[2] M. Xie, Y.S. Dai, K.L. Poh, *Computing Systems Reliability: Models and Analysis*, Kluwer Academic Publishers: New York, U.S.A., 2004.

[3] D. Patterson, A. Brown, P. Broadwell, *et al.*, "Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies", *Technical Report CSD-02-1175,* Univ of California-Berkeley, March 2002, pp. 1-25.