COLLECTION, ASSOCIATION FOR PROBLEM DETERMINATION, AND KNOWLEDGE DISSEMINATION IN AUTONOMIC COMPUTING

Mohammad A. Munawar and Paul A.S. Ward

November 13, 2003

Autonomic computing requires that systems be self-configuring, self-healing, self-optimizing, and self-protecting [8]. To achieve such capabilities systems must monitor their own operation and reflect on the data thus collected. To be effective, the level of monitoring must be feasible, yet sufficient for problem determination. Once problems have been identified, the knowledge thus gained needs to be shared among elements participating in a system.

In the remainder of this proposal we motivate the need for effective data collection, problem determination, and knowledge dissemination. We then outline our approach to the various problems that these requirements entail, indicating possible solutions and their integration with existing IBM solutions. Finally we describe our proposed course of action.

1 MOTIVATION

Present computer systems comprise a large number of components, both software and hardware, each of which may generate data about its normal operation, erroneous conditions, exceptional circumstances, *etc*. This data can take the form of events, entries in log files, or other representations such as core dumps. Additional information originates from the environment in which the component is running and from external sources like domain experts, problem tracking databases, *etc.*.

The collection of this information is a pre-requisite to autonomic computing. This collection of data is problematic at several levels. First, there is potentially an enormous volume of data to collect and process. Second, while various parts of this data are currently collected, the data as a whole are not yet well-integrated, though the log and trace utility [6] goes someway to alleviating this problem. Third, a significant portion of this data is continuous, not discrete, and thus requires careful consideration when placing it in the Common Base Event (CBE) format [5]. Finally, if there are multiple autonomic managers requiring access to the monitored data, we must suitably distribute it.

Data gathered is of no use if it cannot be analyzed and transformed into valuable information. While a number of tools and techniques have been developed to address correlation and problem determination, such as IBM's Active Correlation Technology, various aspects still need to be investigated. We are interested in addressing a few focused issues in this particular area. First, the timebase of the various data collected may need to be corrected. Specifically, the disparate elements that compose the system will not typically operate under a single clock, nor will the clocks necessarily be synchronized. Second, while time is an important correlator, it is not the only one. In particular, there is a significant class of problems for which time is a very poor correlator. For example, the incorrect initialization of data may only much later cause a problem

in an information system. Third, the determination of anomalies based on both continuous and discrete data is a problem that has yet to be studied.

Finally, having identified the cause of problematic symptoms, there is a need to disseminate that knowledge to all interested elements. Such distribution is necessary to allow systems to respond more rapidly when faced with the same situation.

2 METHODOLOGY

Our motivation implies a three-phase project, addressing problems in the following areas:

1. Collection of all relevant system data into common format
2. Association of data for problem determination
3. Knowledge dissemination

2.1 COLLECTION

We aim to address two key issues with regard to the collection phase of our project:

1. Collection of continuous data
2. Scalability through dynamic filtering

As noted in our motivation, we see four key issues with regard to data collection: scalability, integration, continuous data, and dissemination to multiple managers. We plan to focus our efforts on the first and third of these, using Hyades as the framework [3] within which we solve problems so as to provide an integrated collection in the Log and Trace utility [6]. The collection of continuous data consists of both environment data (*e.g.* CPU utilization, memory utilization, *etc.*) and data on the internal status of elements (*e.g.* thread-pool size, queue length, *etc.*). We presume we have access to this data *via* existing monitoring tools and/or APIs (*e.g.* the Tivoli Monitoring Engine collects runtime environment data, WebSphere has a Performance Monitoring Interface (PMI) API, *etc.*) Where it is lacking we will use remote agents controllers (RAC) [9] to instrument the element that needs to be observed.

Once we have access to the data, there are two problems that we must address: it must be changed from a continuous format to a discrete one and there must be a single format for that discrete data. The second problem is relatively easy to deal with. We expect to use the CBE REPORT type for recording such data, though we may need to augment the associated event to indicate various forms of continuous data collected. Further, we expect to specify generic types for the various data collected. The first problem, however, is trickier to deal with. We are currently considering a range of possibilities. We initially expect to record significant changes only, where what is significant will be dependent on the resource being monitored. We then expect to use filtering techniques to increase or decrease the quantity of data collected, as needed for any given situation. Alternatively, we intend to evaluate technique whereby past data can be obtained on-demand, only when their use would help problem determination.

We intend to tackle the scalability problem through data filtering at the source. Work at IBM is in line with this approach and is reflected in efforts such as the Common Situation Filter [7], a filter specification for disabling particular fields of events arranged in the CBE format, and the Symptom Service [7] that determines the cause of a problem by querying sources for only the information required, and only when need

arises. In this research, we aim at complementing these efforts by studying dynamic techniques to reduce the burden of dealing with vast amount of data while still achieving as precise diagnostics as possible. By reducing data that needs to be analyzed we enable systems to scale better and improve their responsiveness. We note, however, that much of the data collected may only be relevant under circumstances that are not easily identifiable in advance.

We are particularly interested in studying filtering capabilities deployed close to autonomic components. While a common format for describing events (Common Base Event) and an associated format for filtering these events (Common Situation Filter) have been defined, we still need to develop mechanism to deploy these in a flexible way. IBM's vision in this respect is being able to deploy filtering on-demand [7]. On-demand filtering refers to the ability to dynamically set up new filters at run-time, and remove them when need be. This represents a non-trivial task as the system has to decide when data needs to be captured and when it is no longer required. Given that present autonomic systems are reactive by nature, enabling filters is a sensitive task, as it may cause loss of information and consequently hinder the ability to diagnose problems. At the same time, filtering is required to reduce the amount of data that has to be dealt with. Determining a trade-off between precision and responsiveness is challenging, and requires system intelligence. Creating this intelligence is hard, and is the main thrust of this research.

2.2 Association for Problem Determination

As noted in out motivation, there are three key issues that need to be addressed with regard to the association phase of our project:

1. Timebase correction
2. Non-time correlated problem determination
3. Anomaly determination in the presence of continuous data

Associating data from logs, resource usage, and internal element status requires that we adjust the timebase to ensure that all records in the distributed environment have a consistent view of the time. While in principle a relatively straightforward task, it is not currently available. There are three approaches that can be taken to this task. First, the partial-order information can be collected to ensure that the time is consistent, if not correct. The problem with this method is that is it heavyweight. The second technique is to require all clocks be synchronized. This, while ideal, cannot be guaranteed in practice. The third method, and the one we propose to adopt, is to monitor time when monitoring system resources, and so keep track of any time variations between systems. This will then be recorded as an additional event within the resource environment log.

Time is an important factor used for correlating events and monitoring data. Associating continuous data with events in logs can be readily made based on time. However, the complexity of present systems is such that dependency between events and status data may not be directly evident based on strict timing. This is more-so with data of a continuous nature such as memory usage. For example, a low memory availability may cause a component to fail later on because some parameters of the latter were not properly initialized. The dependency may not be evident as the time elapsed between the events may be arbitrarily long. Other dependencies between data may be the result of implementation decisions. Detecting and incorporating these possibilities represent an important task that we intend to achieve.

Finally, an important requirement for data association is a clear concept of what constitutes anomalous conditions. For discrete events, it is sufficient to have a rule base that identifies what event represents an anomaly. When dealing with continuous data, thresholds are commonly used to identify deviations. Given that normal behaviour is a function of conditions prevailing in the system, simple rules are no longer sufficient. The baseline performance is adjusted at runtime, and as such, rules need to be updated dynamically. An alternative is to leverage machine learning techniques such as neural networks that have been used for detecting intrusion and various types of network attacks. Identifying the most appropriate approach and combining it with the already existing rule-base for discrete events is necessary in order to provide a more comprehensive problem analysis.

By being able to associate events and data from various sources, we create more awareness about the source of a problem. Once the original cause have been identified using root-cause analysis tools, a piece of knowledge is created. The next segment of this research studies how best to maximize benefit from it.

2.3 KNOWLEDGE DISSEMINATION

In the knowledge dissemination phase of our project, we will address the following issues:

1. Customization and distribution of problem-related knowledge
2. Conflict resolution in case of differing knowledge

Having identified the cause of problematic symptoms, there is a need to push out the knowledge such gained to all interested elements. By knowledge, we mean facts describing particular relationships between elements (*e.g.* the occurrence of conditions a in A, b in B, and c in C imply a high likelihood of failure in component A). Distributing lessons learned, allows the system to respond more promptly the next time it is faced with the same situation. Initially, we started looking at this problem for the purpose of continuous data, but we realized that the knowledge dissemination issue is a general one, and requires special attention. More specifically, the question that arises is what piece of information should an element receive after a problem is identified. With regard to the wider context of utility computing, it is reasonable to expect that system configurations are likely to change from time to time. For example, a component may be combined new ones to fulfill a different task. Distributing information to the system elements is an effective way of disseminating knowledge.

When pondering on the issue of who needs to know what, we quickly realize that it is useless for an element to only learn information about itself. Such information without a context may not be an indication of an anomaly on its own, but in combination with information from other components may correspond to problems. As such, it is more likely that lessons learned by each element will comprise reference to conditions related to other elements. Besides, each element will differ in what it would learn depending on its dependencies on other elements. The distribution requirement can be avoided if all knowledge is kept by a single central service (e.g. an autonomic manager). While we believe that analysis can be performed more effectively with a centralized approach, storing knowledge in a single place is not appropriate. A central entity not only reduces the autonomy of elements, but also creates a single point of vulnerability. We also note that some conditions that deviate from normal can be due to specific environmental conditions, and to not any particular element fault. For example, the current system load may simply be high. In such cases, the coordinating autonomic manager is the prime candidate for holding the new piece of information.

Once we have decided what each element will know, this information has to be conveyed in a suitable form. Assuming that we are able to translate this knowledge into rules, new concerns are raised. The receiving element being an autonomous entity should be able to accept, reject or temporarily cache the information. On what basis should the element decide whether it should keep the piece of information or reject it? An element can decide that more mature information, insofar as it is old and validated, has precedence or else decide that new information is better. It is necessary to deal with such cases because we are likely to encounter existence of conflicting rules. These conflicts will need to be resolved and knowledge adjusted accordingly.

Finally, we observe that there exist a wealth of information that is very pertinent for problem determination and as knowledge but which is not suitable for direct use in the monitoring system. One such example is the Problem Management Report (PMR) database which contains information about problems encountered by customers and how they have been solved. Converting such information into a form that can be readily used in the problem analysis process is difficult, but will result in increased effectiveness. Building on this idea, we believe that having an expert knowledge database that periodically captures system knowledge is another effective way to disseminate knowledge. This information is not necessarily acquired by the same component or even the same system. Organizing these into hierarchies further extends the benefits gained. Moreover, data from external sources such as the PMR database can eventually be accommodated in these expert knowledge databases. As these expert services are dedicated to knowledge dissemination, we believe that these are complementary to the autonomic managers hierarchy envisioned in [4].

3 Research Problems This Proposal Addresses

There are several research problems that arise in our proposed work. Although work has been done, and is still underway in the areas of collection, association and knowledge dissemination, we intend to tackle precise problems that are complementary to previous and current efforts. For collection, the question that we aim to answer is how do we manage data of a continuous nature. Do we collect such data on-demand and if so, how much of this data must be buffered, or else do we capture the data only when some conditions are met? When managing collection of data, the other issue that needs to be addressed is scalability. We aim to answer the question of when to collect what data, and what is the best mechanism and most suitable strategy for dynamically enabling and disabling filters on events captured by the system? As far as association is concerned, we aim to research techniques to associate data items that cannot be directly related based only on strict time factor. As mentioned earlier, in complex systems cause and effects relationships may not be straight-forward, and more advanced techniques are needed to relate events or facts that appear to be unconnected, but indeed are. Finally, in the knowledge dissemination area, we are interested in studying which elements of the system learn what as a result of the problem determination process. In addition, we plan to investigate how to deal with diverging knowledge after the problem identification process has taken place. While problem determination can be best performed centrally, we argue that distribution of knowledge is most beneficial for the system. The benefits of this approach are yet to be demonstrated. As can been seen, while the areas we will work in are vast, our objective is to deal with specific issues. However, lessons learned as a result will have far reaching benefits extending beyond the scope of this work.

4 PROPOSED ACTION

The proposed project is organized in three phases corresponding to the three main problem areas: collection, association, and knowledge dissemination. The deliverables for each phase will consist of a software testbed, and a technical report describing lessons learned during the phase.

Given that a significant amount of work has already been done by IBM, and the open source community, we intend to concentrate on the added-value. As such, our software will be based on the Hyades Framework for the Eclipse platform [2], and will subsequently be integrated with the Log and Trace tool.

In the first phase, we first intend implement different mechanisms for collecting continuous data (*e.g* memory and cpu usage), and store them in a format suitable for integration in our testbed. Ideally, such information will be stored in a compact form using the Common Base Event format [5]. Where data collection facilities are not available, will use remote agents controllers (RAC) [9] to instrument the element that needs to be observed. The best approach for collecting continuous data will be selected for use in the subsequent parts of the project. The next step will consist of implementing dynamic filtering capability in the monitoring system. To this effect, both the monitoring platform and the remote system being observed will need to be augmented. We will implement a Filter Manager using the Hyades Framework which will dynamically deploy and remove filters across distributed components. The Hyades framework is ideal for our purpose as it already provides for facilities to plug-in correlation and analysis tools, and offers a range of core functionalities. Remote agent controllers will monitor the managed elements and implement mechanisms for the dynamic reception of filtering instructions, their installation, and their removal. Having this minimal operational prototype will allow us to evaluate different strategies for deploying and enabling filters. For example, a basic strategy would be to monitor every parameter that has a high variance when compared to its preset threshold, and filter out all others that appear to be stable. Other strategies (*e.g.* using time-series analysis) that span over longer periods of time will be implemented.

In the second phase, our objective is to make the best use of data that can be collected with our tool. To this effect, suitable correlation techniques will be integrated in the tool, and studied. The work will consist of associating continuous data with other events captured by the system not only based on strict timing but under more relaxed constraints. Data collected will be studied to characterize abnormal behaviour. We plan to leverage the learning and inferencing work done in the context of the ABLE project [1] for this purpose. Furthermore, we intend to integrate this tool with problem determination work that has been done previously to enable complete root-cause analysis. While we expect to make changes to the tool developed, most of the work at this stage will consist of experimentation and analysis.

In the third stage, we will build on the previous two stages by manipulating knowledge gained through the tool. More precisely, problem-related cause and effect information, will be distributed to elements of the system. In case of a rule-based system, rules for each individual elements will have to be determined and propagated back to them. Rule distribution will also use the remote agent controllers previously put in place for monitoring. We need to devise techniques to deal with stale knowledge and conflicting rules. Again, we believe that techniques developed in the ABLE project, and the academia can be leveraged to this effect. However, without suitable strategies, these tools and techniques will serve no purpose. This work will add another component to the software, but still most effort will be focused on the design and analysis of alternative strategies.

5 SUMMARY

Addressing data collection, problem determination, and knowledge dissemination issues is critical to build autonomic systems that are prompt in adapting their behaviour when change occurs.

The specific problems we will address are the following:

1. Integration of disparate monitored data, especially continuous data
2. Dynamic filtering of data collected
3. Timebase correction for data collected
4. Non-time correlation of data for problem determination
5. Dissemination of knowledge acquired from the problem determination step

These problems are clearly of direct relevance to IBM and to the IBM Toronto Lab in particular. Our solutions to these problems will be created in the framework of existing IBM technology, and in particular will be tied to the Hyades framework and the Log and Trace utility.

REFERENCES

[1] J. P. Bigus, D. A. Schlosnagle, J. R. Pilgrim, W. N. Mills, and Y. Diao. ABLE (Agent Building and Learning Environment): A toolkit for building multiagent autonomic systems. Available at: http://researchweb.watson.ibm.com/journal/sj/413/bigus.html.

[2] Eclipse Consortium. The Eclipse Platform. Available at: http://www.eclipse.org.

[3] Norman M. G., Guckenheimer S., Erickson M. R. Sluiman H, and Mariani S. The Hyades Project: Automated Software Quality for Eclipse. Available at: http://www.eclipse.org/hyades.

[4] IBM. An Architectural Blueprint for Autonomic Computing, April 2003. Available at: http://www-3.-ibm.com/autonomic/library.shtml.

[5] IBM. Canonical Situation Data Format: The Common Base Event ACAB.BO0301.1.1, August 2003.

[6] IBM. Log and Trace Analyzer for Autonomic Computing, October 2003. Available at: http://www.-alphaworks.ibm.com/tech/logandtrace.

[7] IBM and CISCO Systems. Adaptive Services Framework white paper, October 2003. Available at: http://www3.ibm.com/autonomic/library.shtml.

[8] Jeffrey O. Kephart and David M. Chess. The Vision of Autonomic Computing. In *IEEE Computer Magazine*, January 2003.

[9] Chan Terry. The Hyades Logging Framework. Available at: http://www.eclipse.org/hyades.