Run-Time Correlation Engine for System Monitoring and Testing

Viliam Holub
Lero, System Research Group
University College Dublin,
Ireland
viliam.holub@ucd.ie

Trevor Parsons
Performance Engineering Lab.
University College Dublin,
Ireland
trevor.parsons@ucd.ie

Patrick O'Sullivan
WPLC SVT
IBM Software Group
Dublin, Ireland
patosullivan@ie.ibm.com

John Murphy
Performance Engineering Lab.
University College Dublin,
Ireland
j.murphy@ucd.ie

ABSTRACT

We present an approach and implementation for run-time correlation of large volumes of log data and symptom matching of know issues in the context of large enterprise applications. Our solution provides for (a) automatic data collection, (b) data normalisation into a common format, (c) run time correlation and analysis of the data to give a coherent view of system behaviour at run-time and (d) a symptom matching mechanism that can identify known errors in the correlated data on the fly. We have implemented our approach in the form of the *Run-Time Correlation Engine* (RTCE) and have applied it in a real industry test.

Categories and Subject Descriptors

D.2.5 [**Testing and Debugging**]: Diagnostics; C.4 [**Performance of Systems**]: Reliability, availability, and serviceability

General Terms

Performance, Reliability, Verification

1. INTRODUCTION

Current enterprise systems are often very complex, consisting of large numbers of software components that can be physically distributed across different hardware devices. Each software component (e.g. web server, application server, database etc.) will in general produce numerous logs files containing information on the system level events. The volume of information produced for typical enterprise applications can be extremely large. This is especially the case where systems are expected to cater for high numbers of simultaneous users/requests. Furthermore, while logging facilities are common practice today, most are application or vendor specific. Events from different components can differ in their format and level of detail. This can be particularly problematic for manual or automatic correlation and analysis of the data. Very often the correlation of such data is required when issues arise during testing to obtain an understanding for global system behaviour. Aggregating several sources of information represented in various formats can be very time consuming and often requires the attention of highly skilled engineers. Due to the time required for correlation and analysis, this activity is generally performed manually, after test runs have completed. As a result system testers often do not have a global, correlated view of these events as the system executes. This can be particularly problematic for timely root cause analysis which can require analysis of an issue within its running context.

We address the problem by aggregating, correlating and analysing events from distributed system logs in run-time. Also, using a symptom database, we augment those parts of the log which corresponds to symptomatic problems. We present a solution to this problem: the Run-time Correlation Engine (RTCE).

Goals.

In order to fully understand the requirements for real time correlation and analysis of event logs of enterprise applications we have held discussions with numerous industry testing teams that are responsible for performance and reliability tests carried out on real enterprise systems on a daily basis. From these discussions we have produced a number of evaluation criteria for assessing our work. The evaluation criteria are the attributes which we (and industry test teams) believe are required, in order for our approach to be considered useful.

The evaluation criteria are as follows. *Scalability* in terms of the number of components that the system can handle, the amount of information contained in the variety of logs, and a high capacity for end users. *Responsiveness* in terms of the run-time correlation, runtime log searching, and run-time symptom identification. *Overhead on monitored environment* in terms of the CPU usage and memory requirements.

Along with the main attributes, we have identified several desirable features. Events can be *filtered* per component as well as in centralized, correlated log. There are many potential filtering properties, for example component ID, severity, pattern-match, etc. The system should present to the user *statistics* and performance counters such as number of events occurred, their count, etc. For ease analysis even when the whole network is not available or the application is not running, the system should work in *off-line* mode with information stored during previous monitoring. When switched to off-line mode, the real time correlation may not be performed at all. User can create a *summary log* from the correlated log by applying specified filters. A summary log would contain only the most important events and used for as a backup.

ICAC'09, June 15–19, 2009, Barcelona, Spain.
ACM 978-1-60558-564-2/09/06.

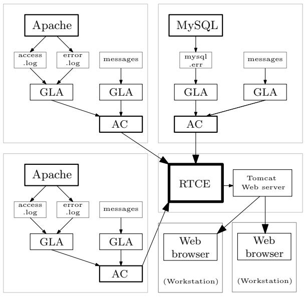

Figure 1: Application and RTCE deployment example

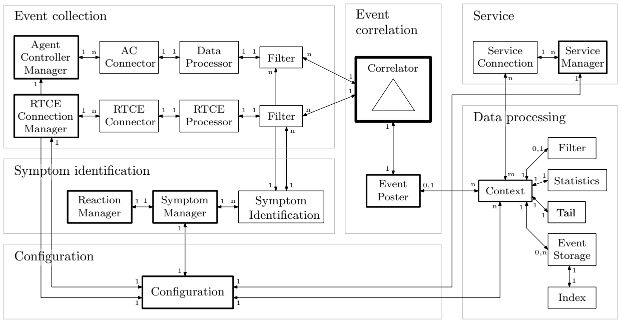

Figure 2: RTCE internal structure

Background.

The *Run-Time Correlation Engine* implementation builds upon a number of current technologies, which have already been built as a part of IBM Autonomic Computing Toolkit [1] (AC Toolkit). Common Base Events [2] (CBE) is a common XML format that can be used for the representation of system events. The Generic Log Adapter (GLA) technology [3] overcomes problems with proprietary event formats by automatically converting them to the CBE format. The IBM Autonomic Computing Toolkit provides GLA configuration adapters to support over 280 log types. The symptom database [4] provides a mechanism for matching known problems in large volumes of data. It allows for the documentation of expert knowledge and for the analysis of logged events using this expert knowledge.

2. RTCE ARCHITECTURE

An example of the RTCE architecture is depicted in Figure 1. Events from the various logs are converted to the CBE format by Generic Log Adapters (GLA). The Agent Controller (AC) is an intermediate between local GLAs and the RTCE core; it routes events from specified GLAs to the RTCE core (Figure 2). The RTCE core maintains an open connection with all available ACs, collects incoming events, and processes them. Together with the RTCE core, we deploy the RTCE presentation component. The role of the RTCE presentation component is to accept user requests, obtain required information from the RTCE core, and present this information in a convenient way through dynamic web pages.

The internal structure of the RTCE core is displayed in Figure 2. The RTCE core is responsible for managing connections with Agent Controllers and collecting events from the adapters. Filtering is applied in the form of rules specified for each particular application component. Subsequently, events are compared to the list of known symptoms in the symptom database using a specialized pattern-matching algorithm for fast symptom identification. When the event is matched with a particular symptom situation, it is augmented with a corresponding explanation and a solution. Events matched against specified critical symptoms are passed to the reaction manager for further immediate reaction.

Finally, the event is passed to the correlator which correlates incoming events according to their timestamp. For situations where the synchronisation of the host application's hardware clocks is not possible, a time delay resolution mechanism is being developed. The time delay resolution mechanism will reside with the centralised RTCE core and will resolve the delay between the different clocks by polling each of the hardware devices periodically. Furthermore due to the possibility of delays when passing events over the network, events must be kept in the Correlator for a specified amount of time before correlation occurs to stabilize the event's position within the correlated log (otherwise events which were delayed may require the logs to be re-correlated).

3. PERFORMANCE

Since RTCE processes events is real-time, a special care has been taken for the use of effective data structures and algorithms. To evaluate RTCE viability in real scenarios, we have carried out series of stress performance tests. Especially, we addressed a number of performance issues that can arise when gathering, normalising, correlating and analysing large volumes of information. From discussions with experts from the domain of system testing and diagnosis, we conclude that the RTCE run on a dedicated hardware is able to safely handle more than 10 typical hosted enterprise application simultaneously.

4. CONCLUSION

Although our solution has been developed as a result of a real industry requirement in the context of system testing of large enterprise applications, we believe it is applicable in a wide range of domains (e.g. for live system administration, monitoring and problem determination). Furthermore we believe it is particularly relevant in the context of autonomic computing where large volumes of information may need to be correlated and analysed on the fly. Autonomic management systems, for example, require an understanding of global and local system behaviour such that they can self-manage and adapt to system events in a timely manner.

Acknowledgments.

This work was supported, in part, by Science Foundation Ireland grant 03/CE2/I303_1 to Lero - the Irish Software Engineering Research Centre. Viliam Holub and Trevor Parsons are supported by the IRCSET Embark Postdoctoral Fellowship Scheme.

5. REFERENCES

[1] B. Jacob, R. Lanyon-Hogg, D. K. Nadgir, and A. F. Yassin, *A Practical Guide to the IBM Autonomic Computing Toolkit*. IBM Redbooks, 2004. [Online]. Available: http://www.redbooks.ibm.com/redbooks/pdfs/sg246635.pdf

[2] "Understanding common base events specification v1.0.1." [Online]. Available: http://www.ibm.com/developerworks/autonomic/books/fpy0mst.htm#HDRAPPA

[3] G. Grabarnik, A. Salahshour, B. Subramanian, and S. Ma, "Generic adapter logging toolkit," in *ICAC'04*. IEEE Computer Society, 2004, pp. 308–309.

[4] "Symptoms deep dive, Part 1: The autonomic computing symptoms format." [Online]. Available: http://www-128.ibm.com/developerworks/autonomic/library/ac-symptom1/index.html