

OPTIMIZING APPLICATION MONITORING



INTRODUCTION

As the use of n-tier computing architectures become ubiquitous, the availability and performance of software applications is ever more dependent upon complex network interactions between hardware, network, and software components and thus more difficult to measure, monitor, and manage. At the same time, the importance of monitoring applications successfully and cost effectively becomes greater and greater especially for e-commerce and IT Service Management.

In the first part of this paper, a set of practical techniques for application monitoring are identified and described. In the second part, a method for value engineering the overall design for application monitoring is presented.

ALTERNATIVE METHODS OF APPLICATION MONITORING

The list below describes some of the basic methods that are commonly used to monitor the performance and availability of applications.

- » Application monitoring plug-ins
- » Multi-measure containers
- » Application admin console “scraping”
- » Web services based application monitors
- » Synthetic transaction monitors
- » Web services subscription
- » Port based “gets”

These methods differ substantially from each other in effectiveness, initial cost, and operating cost. The most mission critical applications are often monitored with more than one technique. These techniques can be categorized into either white box or black box monitoring as described in the following paragraphs. The use of network sniffers for application monitoring are not described below, however, the design methods described in the second part of the paper apply equally to all application monitoring techniques.

WHITE BOX TESTING

In white box testing, measurements are made of the availability and sometimes the performance of the internal software components of the application. The individual measures can then be combined and correlated to make inferences concerning the application’s overall availability and performance.

APPLICATION MONITORING PLUG-INS

Most of the monitoring platforms (HP OpenView, CA Unicenter, IBM Tivoli, BMC Patrol, etc.) have libraries of application monitoring agents that can be deployed on all of the servers on which the software of a particular application has been installed. The plug-in for a particular application communicates with a console-like application on the monitoring server. The plug-in correlates the information collected from the agents and then presents the overall availability of the application through a central console. Performance information can be extracted from these consoles provided that the owner/installer is willing and able to construct a complex dependency network which models the performance of the application based on its constituent parts.

Since this technique requires expensive monitoring software and also the creation and maintenance of complex architecture-specific correlation models, it is very expensive and thus commonly used only by the largest and most sophisticated organizations and even then only for their most important applications. Further, the technique is sufficiently technical and complicated that it lacks credibility for use in Service Level Management

MULTI-MEASURE CONTAINERS

Many monitoring tools allow the assembly of an arbitrary collection of different monitors into a “container” or virtual host that is then assigned the name of the application. The container is then associated with the availability of the particular application that is to be monitored. Some of the measures that may go into a typical container are the following:

- » SNMP providing Windows services availability and application procs
- » Perfmon providing Windows services availability and application procs
- » SNMP for an application specific MIB
- » Script-based monitors executing Unix shell commands for application procs status information
- » Port based scans for process availability
- » Log file parsing

Most monitoring tools can be set to use this technique. The idea is that if all of the monitors in the container are not in alarm, the application is considered to be available. It isn't practical to create performance measures from multi-measure containers.

APPLICATION ADMIN CONSOLE SCRAPING

Many of the enterprise class applications in use today are equipped with their own process monitoring consoles, often accessible as either web sites or secure web sites. The admin subsystems contain whatever set of white box internal monitoring checks that the application developer has provided for this purpose and clearly there are significant differences in the effectiveness between products. Indeed, not all applications are supplied with monitoring consoles. These consoles are often capable of providing extensive availability and crude performance monitoring capabilities.

It is quite feasible to use the system admin console function for application monitoring, and then import the information it develops into the centralized monitoring function. This is done by connecting to the admin console with http or http(s), and then parsing the html of the web pages to detect the presence of error conditions. When detected, a generic error message can be sent to responder personnel. The responder personnel then use the monitoring system console to gain secure access to the application admin console where the details of the incident are available for review and corrective action.

Generally this technique is useful to rapidly identify a subset of the problems causing a lack of system availability. Its limitation is that it does not consider or directly measure the interaction between application and network components.

WEB SERVICES MONITORING

Application Monitoring using web services is an exciting and rapidly evolving area for the application of industry standards. In short, the technique consists of using the “software bus” to subscribe to web services provided by enterprise applications. These applications publish relevant monitoring information in the format of xml documents using the SOA protocol. The technique may have some of the limitations of all white box testing, however, it promises to be much easier to install and maintain than many current generation white box techniques.

BLACK BOX TESTING

In black box testing, the entire application is tested from an outside probe, simulating the actions of an end user or interfacing application.

SYNTHETIC TRANSACTION MONITORS

Synthetic transactions are created using a macro recorder to generate scripts that emulate the actions of a real human being using the application at characteristic speeds. The alarm is based upon the time that it takes to complete the simulated transaction. Since the synthetic transaction is processed at the same time as any other real transaction that is in process, the response time is influenced by demand vs. capacity at the time the synthetic transaction is performed.

Therefore, the measurement gives a realistic picture of not only application availability, but also of system response performance. Behavioral studies have shown that slow performance is comparable to no performance to a user, regardless of whether the software processing the transaction is actually up or down. Performance of n-tier transactions is often path dependent and therefore multiple points of origination may be needed to completely measure the applications performance for all users.

There are two types of synthetic transaction macro-recorders; one is for thick windows clients and the other for browser based thin clients. These functions are available on a many standard monitoring tools but, more generally, are available from the software testing marketplace whose leaders are Mercury Interactive, Software Research, Rational, Quest, Compuware, and others.

PORT BASED SCANS

These are the most simple and direct methods of testing the availability of a server. Virtually all monitoring systems can easily be configured to perform a get command, using one of the characteristic ports used to access an application. For example a "URL get with string match" monitor, will exercise DNS and the other network utility servers, the NIC card of the server, the port daemon of the characteristic port(s), the OS, and the application being tested in a single monitor. This format of application monitor is often suitable for lower impact applications, or as a temporary placeholder for more comprehensive monitors that may be added in subsequent phases of multi-phase deployments.

WHITE BOX VS. BLACK BOX TESTING

The n-tier software architectures for today's enterprise applications qualify as complex systems. Not only are there interactions between components of the same systems, but interactions also occur with virtually all other systems, due to the shared system resource represented by the network, and common storage systems that are not under the control of a particular application. In complex systems, the subtle interactions between system elements and shared resources often dominate the target system performance. This reality establishes an upper limit to the effectiveness of white box testing for service level measurement. In more practical terms, IT customers seem prepared to accept direct simulations of user experience for measurement of delivered service level.

On the other hand, white box monitoring provides very valuable information for rapidly diagnosing, isolating, and resolving system failures and for doing performance tuning. For many mission critical applications, it may be worthwhile to employ both white box and black box testing techniques.

DESIGN PROCEDURES

The actual procedure for value engineering the application monitoring design is complicated and approached with step-wise iteration. The approach uses the concept of marginal analysis and expressly includes the life cycle costs of maintaining the monitoring application as well as initial deployment costs.

The first step of the application design procedure is to identify all of the important applications that justify being monitored. Then the applications are placed in rank order depending upon the business impact of outages. The diagram given below illustrates this:

The next step in the process is to select the set of techniques that can be considered for monitoring tools that are already deployed, or are likely to be purchased as a result of the design process. As part of this process, estimates are made of the performance, and cost of acquiring, any new tools needed to supply the various application monitoring methods. Conceptually, the result is a horizontal stack ranking of application monitoring tools based upon their relative cost as shown in the following figure:

Figure 1. Business Impact of Application Outages

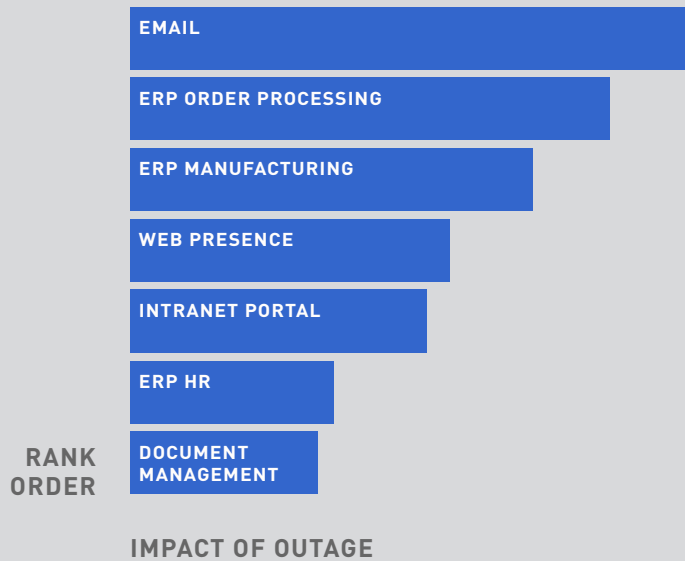
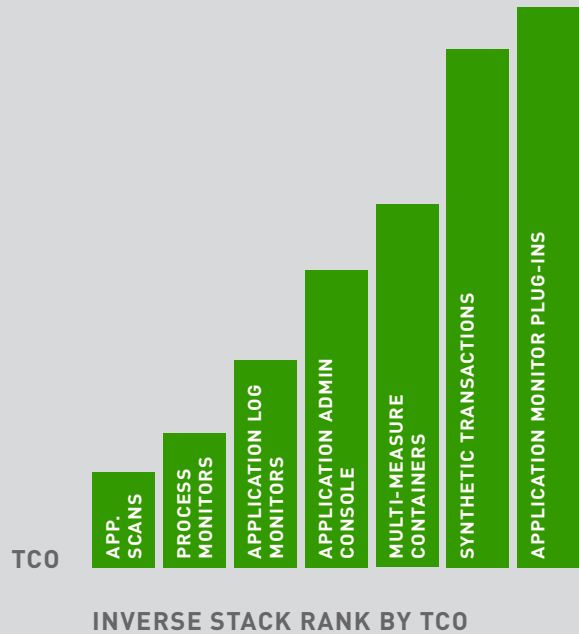


Figure 2. Application Monitoring Tools TCO



Experience shows that the optimal solution for synthetic transaction monitoring is to use free standing macro recorders operating under the control of a central monitoring system, rather than “built in” macro recorders from commercial monitoring tools. Accordingly, at GroundWork we estimate the cost of this approach, rather than to estimate the costs of using the macro recorders that are included with some monitoring systems. We then re-visit this design choice if the installed tools have the required capability.

The effectiveness of synthetic transaction monitors is substantially leveraged when the same testing environment is available for use in production monitoring, pre-release testing, and load testing. In addition to conserving license fees, the diagnostic information reported from production monitoring is familiar to both QA and development personnel.

It is far more likely to select a white-box monitoring tool where the platform products in which they are embedded are already installed and when the number of “tiers” in the application implementation are relatively few. This simplifies the installation and maintenance of the correlation network required for application monitoring. Multi-measure containers are more likely to be used where the application is contained within a single hardware server. For example, Microsoft’s recommendations for Exchange monitoring are easily implemented with this approach, but only the fearless would attempt it for the SAP infrastructure.

The final design of the monitoring system is based upon assigning multiple monitoring techniques of variable monitoring intensity. In the final design, the highest impact applications might use white box plug-ins from a monitoring system like HP Open View or CA Unicenter (if already licensed) and synthetic transactions with multiple test transactions and port based scans of characteristic ports. In contrast, a medium impact application might use multi-measure containers with a single synthetic transaction and port based scans.

Finally, a still lower impact application might only be monitored with a port-based scan with string match of the returned field. For those applications offering an admin console, scraping the console for error messages is usually recommended. The resulting assignment of monitoring techniques to applications can be shown to provide the optimal relationship of cost to utility.

CONCLUSION

After studying dozens of designs for application monitoring, we have found that the optimal solution is often achieved through simple synthetic transaction monitors, combined with multi-measure containers for simple and less critical applications. The cost effectiveness of this solution is leveraged substantially when the same tools are used for monitoring release testing, and load testing of n-tier applications. We find that complex white box testing techniques are of useful primarily for diagnostics.

Applying these principles to application monitoring, coupled with a well-managed project, can achieve significant business benefits. These include higher application availability and employee productivity, and lower capital and operating costs. Whether using expensive monitoring software or open source tools, the value of the monitoring system derives from the effectiveness of its design and implementation plan, and the business processes that use its output – not the features of the software. For many organizations, this is good news.

Contact us

1.866.899.4342

info@groundworkopensource.com

www.groundworkopensource.com

GroundWork Open Source, Inc.

139 Townsend Street, Suite 100
San Francisco, CA 94107

ABOUT GROUNDWORK

GroundWork Open Source, Inc. provides open source-based IT infrastructure management solutions. Groundwork's solutions enable IT management to leverage the flexibility and low cost of open source tools to achieve enterprise-level availability, performance and operational efficiency for a fraction of the cost of commercial software.