WHITE PAPER

now from symantec VERITAS™

VERITAS™ Application Saver

MANAGING APPLICATION RELIABILITY WITH VERITAS™ APPLICATION SAVER

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Software has become an intrinsic part of business over the past few decades. Virtually every business in the world, in every sector, depends on software to aid in the development, production, marketing, and support of its products and services. Advances in computers, software development tools and associated technologies have enabled the development of high performance enterprise quality software applications. Despite these advances, even thoroughly tested software applications can contain flaws and its reliability cannot be guaranteed.

Improving application quality is an important objective of the software industry. The complexity of the underlying software needed to support our computerized economy is increasing at a rapid rate. The size of software applications is no longer measured in terms of thousands of lines of code, but millions of lines of code. While traditional quality assurance methodologies, such as code review and unit testing, are an essential component of any reliability strategy, even the most exhaustive review and test can not ensure the detection of all critical bugs and will likely overlook many residual bugs. The National Institute of Standards and Technology (NIST) estimates the cost of finding and fixing these bugs during integration/system testing through post-deployment can range anywhere from 10 to 30 times more than those found during requirement specifications and coding/unit testing phases. Realizing defects reside in even the most thoroughly tested software applications, a proactive application monitoring and reliability solution with the ability to fix defects in real time emerges as an essential IT requirement to ensure application availability and business continuity.

This paper presents, **VERITAS™ Application Saver**, a comprehensive solution for managing the health of enterprise application software. Application Saver manages application health as it executes in distributed environments and hence complements and widens the scope of the VERITAS i^3™ Application Performance Management solution by improving the reliability of C/C++, Java, .NET and Visual Basic-based software. It addresses the common need to quickly resolve software problems in the event that an application experiences intermittent, inexplicable problems in the production environment.

INTRODUCTION

Businesses invest billions of dollars in application software to optimize business processes and increase workforce productivity. However, the full value of this investment may not be realized due to the impact of software defects. According to the NIST, software defects cost the U.S. economy an estimated $59.5 billion each year with 64% of that cost borne by end-users.

It's difficult to address end user problems that arise from software defects without insight into the real-world operation of the application software and its underlying software components. Often when problems occur in production environments, very little information is available to assist support teams in determining the root cause of the problem. Due to the complexity of modern application architectures and a lack of forensic detail, problem solving quickly degrades into *blamestorming* – a nonproductive, cyclical fray of finger pointing in which each group absolves itself of blame, and points to another, or a vendor as the culprit. During the time consuming and non-productive blamestorm process businesses that license commercial applications or software components can do little more than complain to their vendor while implementing labor-intensive workarounds.

Due to these realities, IT organizations are now searching for solutions to manage the health of their application deployments to ensure their reliability. The ideal solution should scale to production and provide rapid response to software problems by quickly identifying faulty software components and offering solutions to keep applications running reliably.

THE TYPICAL ENVIRONMENT

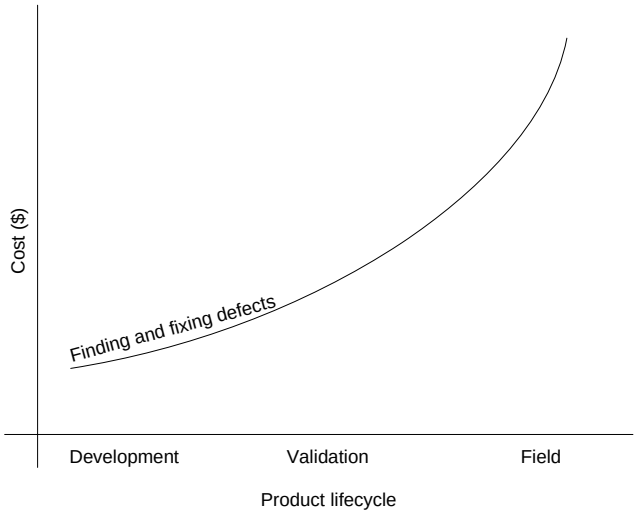
Modern enterprise applications are a complex hierarchy of software components. Software components range from those defined by a formal application model, such as Enterprise Java Beans (EJBs) in the Java 2 Enterprise Edition (J2EE) application model to less formal, although well defined, collections of C functions exported by shared libraries, such as the standard C Runtime Library. Many of today's applications are actually a mix of formal and informal software components that implement everything from business logic through system services. For example, J2EE compliant applications are deployed on application servers where ultimately a Java Virtual Machine (JVM), - a C/C++ software component that depends on the C Runtime Library - executes the business logic encapsulated in EJBs. Similarly, a Windows .NET application may depend on a hierarchy of formal and informal software components executed under the control of COM+ Component Services.

Modern component models allow for faster development and easier customization of powerful, enterprise-class applications. These application environments are however, very dynamic with many characteristics determined during deployment. This dynamic behavior creates a vulnerability for subtle configuration errors and makes runtime behavior hard to predict and difficult to replicate on developer desktops or in test environments (e.g., DLL hell). In addition, it is not uncommon for application programmers, in an attempt to deal with the side effects of this dynamic nature, to mask problems emanating from service layers, hiding the true source of defects and inefficiencies. In fact, a survey conducted by The Standish Group, which found that 37% all system downtime was caused by application bugs and errors (similar analysis by Giga and Gartner put application errors at 33% and 40%, respectively).

It is an acknowledged fact that most applications are put into production with hundreds, if not thousands, of undiscovered software defects. The frequency of software defects in deployed programs varies widely. However, studies by the Software Engineering Institute suggest that 0.8 defects per thousand lines of code (KLOC) in deployed programs is "a rather good level of quality by current industrial standards". Some estimates are as high as 20 defects per KLOC. In a million lines of code, these figures suggest that somewhere between 800 and 20,000 bugs can be expected. In modern application environments where applications are built-up from numerous layers of software components, it is not hard to exceed a million lines of code. Ultimately, regardless of the number lines of code, an application can be no more reliable than its least reliable software component.

THE DEFECT DICHOTOMY

Finding and correcting 100% of the defects in commercial enterprise strength applications is a statistically impossible task. Some bugs are easy to detect while others can only be detected under complex conditions. These conditions require a specific combination of time, data, resources, and usage and are generally very expensive, and sometimes impossible, to reproduce. Considering these limitations it is accepted by the software industry as economically inefficient and statistically impossible to remove all the defects in commercial enterprise strength applications.



NIST estimates that the cost of finding and fixing defects during the latter stages of a product's life cycle (integration/system testing through post-deployment) can range anywhere from 10 to 30 times more than those found during the early stages of the product's life cycle (requirement specifications and coding/unit testing phases).

Herein lies the defect dichotomy. A software developer cannot profitably afford to find and resolve all defects before release of a product. However, finding and fixing defects in the field is likely to negatively affect profitability. Despite this dichotomy, software is often released after a subjective review of its quality which takes into account the known and unknown defects and their effect on profitability.

APPLICATION RELIABILITY CHALLENGES

A primary challenge for IT organizations is to maintain the availability of applications that depend on unreliable software. Intermittent, recurring software problems that only manifest themselves in production environments are the most difficult to resolve – and the most expensive. Especially challenging are problems that appear over extended periods of time or only under the heaviest of workloads. Many common software errors, such as buffer overwrites and memory leaks, can cause intermittent application failure and performance degradation hours or even days before they are recognized as problems.

Reliability, scalability and maintainability are largely subjective and difficult to test prior to deployment. Typically, software developers do not truly know how their applications and software components will perform until they are exposed to real-world scenarios. The challenge becomes finding and fixing any residual software defects before they impact the business.

Attempting to solve these problems with software development tools, such as debuggers and profilers, introduces excessive overhead and security risks that are not appropriate for application management in most production environments. Furthermore, most IT organizations neither have access to the complete source code for all the layers of software components they use in their applications, nor do they have the skills to fix bugs that arise in many of the service layers even if source code is available.

Although IT organizations have a variety of management solutions for distributed systems, a quick review of capabilities will show an obvious gap. To ensure smooth deployments and reliability over the lifetime of an application, a secure, low overhead solution is needed that allows IT organizations to:

- Proactively identify application software design and coding defects that are often revealed under load.
- Prevent common problems in software components from limiting application availability or capacity.
- Reconstruct the chain of events leading up to any application failures – the first time they occur.

IT staff, including development, quality assurance, system and application administrators, all have important roles in managing application reliability. To eliminate blamestorming and empower all groups to be successful in their increasingly complex and demanding roles, an Application Reliability Management solution is an essential tool for overcoming the challenges of the modern application environment.

VERITAS™ APPLICATION SAVER

VERITAS Application Saver provides a centralized view of application reliability as it executes. VERITAS Application Saver uses unique Runtime Agent technology that, in addition to collecting critical reliability metrics, can be configured to:

- Correct several classes of common software defects at run time, preventing application failures.
- Collect detailed Forensics for a wide range of software faults, enabling rapid response to other failures.

VERITAS Application Saver is the first practical solution to give IT Operations the ability to monitor beyond the infrastructure, down into the individual software components that comprise an application and to offer run time suppression of software defects.

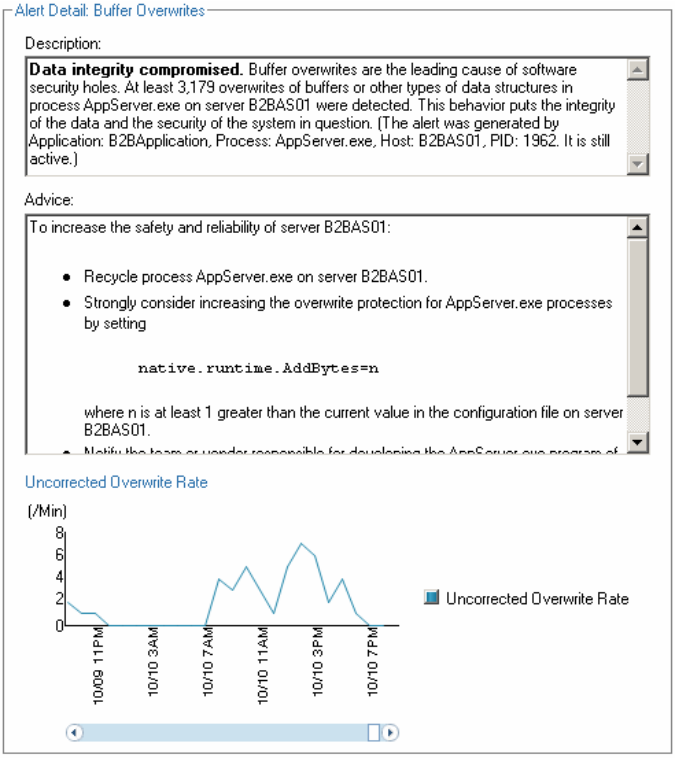
VERITAS APPLICATION SAVER FEATURES

VERITAS Application Saver is a comprehensive and heterogeneous solution for managing the reliability enterprise applications. Key features include:

- A centralized view of application software as it executes in distributed/clustered environments.
- Detection of destructive errors including buffer overwrites.
- Detection of chronic software defects including memory leaks
- Transparent correction of memory leaks in C/C++ applications.
- Transparent protection of memory from short overwrites.
- Dynamic optimization of memory resource use.
- Detection of subtle scalability bottlenecks such as excessive lock contentions and virtual memory use.
- Support for heterogeneous application environments – applications developed in Java, C/C++, .NET and Visual Basic and deployed in AIX, HP-UX, Linux, Solaris, Windows, .NET and JRE environments.
- Collection of extensive Forensics data for application failures, including per-thread, statement-level execution histories for Java, C/C++, .NET and Visual Basic applications (deployed in Linux, Solaris, Windows, .NET and JRE environments).
- A separate offline Forensics Viewer, designed specifically for software engineers, that enables rapid root cause analysis of application failures.

HEAL WITH SMARTUNE™ CORRECTIONS

SmarTune Corrections instruct VERITAS Application Saver's Runtime Agent to reduce application failures by automatically detecting and correcting destructive software errors in real-time. Additionally, the Runtime Agent can be configured to make more effective use of processing resources typically found on production servers.



Improving Robustness

Buffer overwrites are surprisingly common programming errors. Many well-publicized security issues in commercial software are buffer overwrite problems. Equally common are logic errors in the management of dynamic memory allocations, which can corrupt data or cause intermittent crashes. Most conventional runtime environments do little to protect application environments from either of these types of errors. Application Saver's Runtime Agent can be configured to provide the necessary protection from such errors by padding allocations and ensuring attempts to release memory are appropriate, which combined with leak correction helps to ensure information is correct and applications are available to serve the business.

Plugging Leaks

Memory leaks are particularly common in multi-threaded, event-driven applications, and particularly problematic in long running servers. A memory leak occurs when an application is finished using an object of memory, but doesn't return that object to the system for re-use. In this way, available memory is lost or "leaked". If the memory leak occurs as a repeated pattern within the application, the virtual memory footprint of the application grows. This footprint growth can extend until the operating system runs out of swap space. In this scenario, applications on the server will become unresponsive or crash.

VERITAS Application Saver's Runtime Agent not only reports the source of leaks, but can also be configured to automatically resolve this problem with a conservative garbage collector that safely and efficiently reclaims leaked memory and reduces the virtual memory footprint of applications. For applications with severe leaks where pre-emptive recycling is often necessary to prevent application failures, VERITAS Application Saver offers a solution by eliminating the need to recycle the application and hence improving overall server performance.

Increasing Scalability
Typical thread-safe runtime environments protect a single, monolithic heap with a lock or critical section. These synchronization primitives can nearly double the allocation times, and severely limit vertical scalability as the number of threads increase because of the increased risk that a thread may lose its time-slice while synchronizing. The poor vertical scalability of many applications is due primarily to this contention for heap resources. Application Saver's Runtime Agent can be configured to not only reduce this contention through use of multiple concurrent heaps, but many shared data structures are carefully implemented in machine code to use non-blocking operations that allow thread-safe modification without locks.

At the processor architecture level, whenever a processor writes to a memory location, the old copy of data existing in the caches of other processors must be invalidated. Cache coherence activities, due to this 'write-invalidate' protocol used by Symmetric Multiprocessors, not only increases cache miss rates, but also may incur unnecessary overhead from a condition known as *false sharing*. False sharing occurs when multiple processors attempt to update different memory objects allocated in the same *cache line* (a block of consecutive memory locations) at about the same time. Application Saver's Runtime Agent will also reduce the 'write-invalidate' induced bus traffic of false sharing by mapping memory objects likely to be used by different processors to distinct cache lines. This increases memory access speed throughout the entire object lifetime. For multi-threaded, component-based application models, these optimizations unleash the processing power of SMP-based servers and enables vertical scale up, increasing capacity.

FORENSICS ENABLE RAPID RESPONSE

VERITAS Application Saver enables rapid response to application failures through its unique ability to capture detailed Forensics for software faults that can be used by support teams to completely reconstruct the chain of events leading up to the fault that caused the failure.

As software instrumented with VERITAS Application Saver executes, optimized, fine-grain probes append trace data to in-memory buffers. Each trace buffer holds a bounded amount of data, wrapping circularly when the buffer is full. The result is, at any point in time, the trace buffers hold a recent execution history for each thread across the instrumented classes or libraries. In addition to the per-thread, statement-level trace data, Application Saver also captures the values of relevant system variables, version data and configuration data.

Forensics collected by VERITAS Application Saver are presented offline in an easy to use graphical interface that a software engineer can use to quickly determine the root cause of application failures. And Application Saver supports heterogeneous application environments providing a cross-language view of the entire application environment including applications that mix Java with native C/C++ code.

BENEFITS OF VERITAS APPLICATION SAVER

VERITAS Application Saver allows the benefits of existing IT investments to be fully realized by addressing the leading cause of application failures – software bugs and errors.

- Facilitates rapid response to software problems.
- Reduces downtime and application support costs.
- Prevents revenue and productivity losses.
- Improves service levels and end user satisfaction.
- Maximizes return on existing IT investments.

End users of business-critical applications will benefit from instantly improved stability and increased scalability and Application Managers will benefit from the health metrics for application software, providing a much-needed view into the real-world operation of applications. This helps to alleviate the anxiety senior managers experience when making expensive decisions based upon inaccurate or poorly gathered information.

QA, Production Acceptance, Operations and Support teams will benefit from increased visibility into potential problems and a better ability to communicate issues to software engineers. And Administrators will benefit from the ability to implement immediate solutions through SmarTune corrections

SUMMARY

VERITAS Application Saver is the industry's first Application Reliability Management solution that successfully combines fault management and fault tolerance for software into a practical solution for ensuring application reliability in production environments. With Application Saver, you can keep applications running with timely, detailed information about the overall health of application software, including reliability metrics and advance warning of potential problems. Administrators can create fault-tolerant software installations by enabling Application Saver's SmarTune Corrections to transparently heal software faults and prevent application failures, which can be a career saver while you wait for vendors or developers to deliver a fix. And Application Saver's Forensics feature gives Operations the ability to deliver detailed diagnostics to Technical Support the first time a problem occurs, without delay or interruption of service.