Appears in *Proceedings of the Second IEEE International Conference on Autonomic Computing (ICAC 2005)*, Seattle, WA, June 2005.

AutoPod: Unscheduled System Updates with Zero Data Loss

Shaya Potter and Jason Nieh
Computer Science Department
Columbia University
{spotter, nieh}@cs.columbia.edu

Abstract

Patching, upgrading, and maintaining operating system software is a growing management complexity problem that can result in unacceptable system downtime. We introduce AutoPod, a system that enables unscheduled operating system updates while preserving application service availability. AutoPod provides a group of processes and associated users with an isolated machine-independent virtualized environment that is decoupled from the underlying operating system instance. This virtualized environment is integrated with a novel checkpoint-restart mechanism which allows processes to be suspended, resumed, and migrated across operating system kernel versions with different security and maintenance patches. AutoPod incorporates a system status service to determine when operating system patches need to be applied to the current host, then automatically migrates application services to another host to preserve their availability while the current host is updated and rebooted.

1 Introduction

As computers become more ubiquitous in large corporate, government, and academic organizations, the total cost of owning and maintaining them is becoming unmanageable. As the amount of computers an administrator manages increases, the probability that one will suffer a hardware failure increases proportionally. Similarly, computers are increasingly networked which only complicates the management problem given the myriad of viruses and other attacks commonplace in today's networks. Security problems can wreak havoc on an organization's computing infrastructure. To prevent this, software vendors frequently release patches that can be applied to address security and maintenance issues that have been discovered. This provides a management nightmare for administrators who take care of large sets of machines. For these patches to be effective, they need to be applied to the machines. It is not uncommon for systems to continue running unpatched software long after a security exploit has become well-known [3].

We present AutoPod, a system that provides an easy-to-use autonomic infrastructure for operating system self-maintenance. Autopod uniquely enables unscheduled operating system updates of commodity operating systems while preserving application service availability of applications during system maintenance. AutoPod provides its functionality without modifying, recompiling, or relinking applications or operating system kernels. This is accomplished by combining three key mechanisms: a lightweight virtual machine isolation abstraction that can be used at the granularity of individual applications, a checkpoint-restart mechanism that operates across operating system versions with different security and maintenance patches, and an autonomous system status service that monitors the system for system faults as well as security updates.

2 AutoPod Virtualization and Migration

The AutoPod model is based on a virtual machine abstraction called a pod (PrOcess Domain). Pods were previously introduced in Zap [2] to support migration assuming the same operating system version is used for all systems. AutoPod extends this work to enable pods to provide a complete secure virtual machine abstraction in addition to heterogeneous migration functionality. A pod looks just like a regular machine and provides the same application interface as the underlying operating system. Pods can be used to run any application, privileged or otherwise, without modifying, recompiling, or relinking applications. This is essential for both easy-of-use and protection of the underlying system, since applications not executing in a pod offer an opportunity to attack the system. Processes within a pod can make use of all available operating system services, just like processes executing in a traditional operating system environment. Unlike a traditional operating system, the pod abstraction provides a self-contained unit that can be isolated from the system, checkpointed to secondary storage, migrated to another machine, and transparently restarted. Unlike Virtual Machine Monitors (VMMs), such as VMware [4], pods decouple process execution from the underling host. Therefore, while VMMs can enable processes to be moved off a faulty host machine, they don't allow for them to be moved off an insecure operating system that needs to be patched. The Microvisor [1] approach tries to remedy that, but depends on a system, such as AutoPod, to provide the decoupling infrastructure that's needed.

To support the AutoPod abstraction design of secure and isolated namespaces on commodity operating systems, we employ a virtualization architecture that operates between applications and the operating system, without requiring any changes to applications or the operating system kernel. This virtualization layer is used to translate between the AutoPod namespaces and the underlying host operating system namespace. It protects the host operating system from dangerous privileged operations that might be performed by processes within the AutoPod, as well as protecting those processes from processes outside of the AutoPod on the host.

The key pod virtualization mechanisms used are a system call interposition mechanism and the `chroot` utility with file system stacking to provide each pod with its own file system namespace that can be separate from the regular host file system. Pod virtualization support for migration is based on Zap [2]. Pod virtualization is very light weight, imposing on average less than 10% performance hit on microbenchmarks, while imposing less than 2% overhead for real application scenarios such as outlined by TPC-W.

To support migration across different kernels, AutoPod use a checkpoint-restart mechanism that employs an intermediate format to represent the state that needs to be saved on checkpoint. On checkpoint, the intermediate format representation is saved and digitally signed to enable the restart process to verify the integrity of the image. Although the internal state that the kernel maintains on behalf of processes can be different across different kernels, the high-level properties of the process are much less likely to change. We capture the state of a process in terms of higher-level semantic information specified in the intermediate format rather than kernel specific data in native format to keep the format portable across different kernels. AutoPod is able to checkpoint and restart real applications much faster than restart could take on its own. While starting a full desktop environment from scratch can take nearly 20 seconds, checkpointing and restarting take only 851 ms and 942 ms respectively.

3 Autonomic System Status Service

Many operating system vendors provide their users with the ability to automatically check for system updates and to download and install them when they become available. Example of these include Microsoft's Windows Update service, as well as Debian based distribution's security repositories. User's are guaranteed that the updates one gets through these services are genuine because they are verified through cryptographic signed hashes that verify the contents as coming from the vendors. The problem with these updates is that some of them require machine reboots; In the case of Debian GNU/Linux this is limited to kernel upgrades. We provide a simple service that monitors these security repositories. The autonomic service simply downloads all security updates, and by using the pod's checkpoint/restart mechanism enables the security updates that need reboots to take effect without disrupting running applications and causing them to lose state.

Commodity systems also provide information about the current state of the system. Subsystems, such as a hard disk's Self-Monitoring Analysis Reporting Technology (SMART), let an autonomic service monitor the system's hardware state. Similarly, the kernel on the machine monitors the state of the system, and if irregular conditions occur, such as DMA timeout or needing to reset the IDE bus, will log this occurrence. Our autonomic service simply monitors the kernel logs to discover these irregular conditions. When the hardware monitoring systems or the kernel logs provide information about possible pending system failures, the autonomic service simply checkpoint the pods running on the system. It then migrates the pod to a new system to be restarted on, ensuring that no state is lost.

4 Conclusions

The AutoPod system provides an operating system virtualization layer that decouples process execution from the underlying operating system, by running the process within a Pod. AutoPod can transparently migrate isolated applications across machines running different operating system kernel versions. This enables security patches to be applied to operating systems in a timely manner with minimal impact on the availability of application services. We have implemented AutoPod on Linux without requiring any applications changes.

References

[1] D. E. Lowell, Y. Saito, and E. J. Samberg. Devirtualizable virtual machines enabling general, single-node, online maintenance. In *Eleventh International Conference on Architectural Support for Programming Languages and Operating Systems*, October 2004.

[2] S. Osman, D. Subhraveti, G. Su, and J. Nieh. The Design and Implementation of Zap: A System for Migrating Computing Environments. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA, Dec. 2002.

[3] E. Rescorla. Security holes... Who cares? In *Proceedings of the 12th USENIX Security Conference*, Washington, D.C., Aug. 2003.

[4] VMware, Inc. `http://www.vmware.com`.