Final Report: Migration Mechanisms for Large-scale Parallel Applications

Jason Nieh (nieh@cs.columbia.edu), Principal Investigator
DOE grant DE-FG02-03ER25562, Aug. 15, 2003 - Aug. 14, 2007

A Executive summary

Process migration is the ability to transfer a process from one machine to another. It is a useful facility in distributed computing environments, especially as computing devices become more pervasive and Internet access becomes more ubiquitous. The potential benefits of process migration, among others, are fault resilience by migrating processes off of faulty hosts, data access locality by migrating processes closer to the data, better system response time by migrating processes closer to users, dynamic load balancing by migrating processes to less loaded hosts, and improved service availability and administration by migrating processes before host maintenance so that applications can continue to run with minimal downtime. Although process migration provides substantial potential benefits and many approaches have been considered, achieving transparent process migration functionality has been difficult in practice.

To address this problem, our work has designed, implemented, and evaluated new and powerful transparent process checkpoint-restart and migration mechanisms for desktop, server, and parallel applications that operate across heterogeneous cluster and mobile computing environments. A key aspect of this work has been to introduce lightweight operating system virtualization to provide processes with private, virtual namespaces that decouple and isolate processes from dependencies on the host operating system instance. This decoupling enables processes to be transparently checkpointed and migrated without modifying, recompiling, or relinking applications or the operating system. Building on this lightweight operating system virtualization approach, we have developed novel technologies that enable (1) coordinated, consistent checkpoint-restart and migration of multiple processes, (2) fast checkpointing of process and file system state to enable restart of multiple parallel execution environments and time travel, (3) process migration across heterogeneous software environments, (4) network checkpoint-restart and migration of distributed and parallel applications, (5) a utility computing infrastructure for mobile desktop cloud computing based on process checkpoint-restart and migration functionality, (6) a process migration security architecture for protecting applications and infrastructure from denial-of-service attacks, and (7) a checkpoint-restart mobile computing system using portable storage devices.

This work is primarily the result of two PhD students, Oren Laadan and Shaya Potter, who were supported through this DOE Early Career Award. A dozen papers, including two best paper awards, have resulted from this work [11, 3, 4, 10, 8, 7, 5, 6, 9, 12, 1, 2]. Furthermore, based on this work, operating system virtualization has already become widely-used and part of the mainline Linux kernel, and checkpoint-restart mechanisms are in the process of being integrated as well. Detailed technical project activities are discussed in the following sections.

B Consistent checkpoint-restart and migration of multiple processes

We developed mechanisms to transparently checkpoint multiple processes such that they can be restarted in a globally consistent state. We also address for the first time key design issues in building a transparent checkpoint-restart mechanism for multiple processes on commodity operating systems. We combine a kernel-level checkpoint mechanism with a hybrid user-level and kernel-level restart mechanism to leverage existing operating system interfaces and functionality as much as possible for checkpoint-restart. We introduce a novel algorithm for accounting for process relationships that correctly saves and restores all process state in a globally consistent manner. This algorithm is crucial for enabling transparent checkpoint-restart of interactive graphical applications and correct job control. We introduce an efficient algorithm for identifying and accounting for shared resources and correctly saving and restoring such shared state across cooperating processes. To reduce application downtime during checkpoints, we also provide a copy-on-write mechanism that captures a consistent checkpoint state and correctly handles shared resources across multiple processes.

We have implemented a checkpoint-restart prototype as a set of user-space utilities and a loadable kernel module in Linux that operates without modifying, recompiling, or relinking applications, libraries, or the operating system kernel. Our measurements on a wide range of real-world server and desktop Linux applications demonstrate that our prototype can provide fast checkpoint and restart times with application downtimes of less than a few tens of milliseconds. Comparing against using virtual machine monitors, we show up to 550 times faster checkpoint times and 1100 faster restart times than Xen, a hardware virtualization approach.

C Parallel execution environments and time travel

We have optimized checkpoint-restart and augmented it with efficient snapshotting of file system state. Checkpoint optimizations include shifting expensive I/O operations out of the critical path, and fast incremental and copy-on-write techniques. We combine these mechanisms together with logging and unioning file system mechanisms to capture the

file system state at any checkpoint. This is done by using log structured file systems to efficiently provide a read-only snapshot of the file system at checkpoint, then using file system unioning to join the read-only snapshot with a writable file system layer by stacking the latter on top of the former to create a read-write version of the snapshot. This ensures that applications revived from a checkpoint are given a consistent file system view corresponding to the time at which the checkpoint was taken. Our combination of unioning and file system snapshots provides a branchable file system to enable us to create multiple revived sessions from a single checkpoint. Since each revived session is encapsulated in its own virtual execution environment and has its own writable file system layer, multiple revived sessions can execute concurrently. Furthermore, by using the same log structured file system for the writable layer, the restarted session retains the ability to continuously checkpoint session state and later revive it.

We have implemented these mechanisms as a set of loadable modules for Linux to provide transparent operation without modifying, recompiling, or relinking applications or OS kernels. To demonstrate its effectiveness, we have evaluated its performance on a wide-range of real-world interactive desktop applications. Our results show that our approach can provide continuous checkpointing without any user noticeable performance degradation of the system. Downtime due to checkpoints when running desktop application benchmarks is less than 10 ms, a time delay much shorter than what humans can readily detect.

D Process migration across heterogeneous software environments

Patching, upgrading, and maintaining operating system software is a growing management complexity problem that can result in unacceptable system downtime. We have created a system that provides an easy-to-use autonomic infrastructure for operating system self-maintenance. The system uniquely enables unscheduled operating system updates of commodity operating systems while preserving application service availability during system maintenance. This is accomplished by combining three key mechanisms: a lightweight operating system virtualization 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 autonomic system status service that monitors the system for system faults as well as security updates. In particular, we provide a novel checkpoint-restart mechanism that uniquely decouples processes from dependencies on the underlying system and maintains process state semantics to enable processes to be migrated across different machines. The checkpoint-restart mechanism introduces a platform-independent intermediate format for saving the state associated with processes and operating system virtualization. We combine this format with the use of higher-level functions for saving and restoring process state to provide a high degree of portability for process migration across different operating system versions that was not possible with previous approaches. In particular, the checkpoint-restart mechanism relies on the same kind of operating system semantics that ensure that applications can function correctly across operating system versions with different security and maintenance patches.

We have implemented a prototype system as a loadable Linux kernel module and used this prototype to securely isolate and migrate a wide range of unmodified legacy and network applications. We measure the performance and demonstrate the utility of across multiple systems running different Linux kernel versions using three real-world application scenarios, including a full KDE desktop environment with a suite of desktop applications, an Apache/MySQL web server and database server environment, and a Exim/Procmail e-mail processing environment. Our performance results show that the system can provide secure isolation and migration functionality on real world applications with low overhead.

E Network checkpoint-restart and migration

We have created a novel system to provide transparent coordinated checkpoint-restart of distributed network applications on commodity clusters. It provides a thin virtualization layer on top of the operating system that decouples a distributed application from the operating system instances running on a set of cluster nodes. This frees the application from dependencies on the cluster nodes on which it is executing. This virtualization also exposes the underlying host operating system functionality directly to applications, enabling parallel applications to efficiently utilize existing multiprocessor operating system functionality.

By decoupling applications from the operating system instances running on a set of cluster nodes, we can checkpoint an entire distributed application across all nodes in a coordinated manner so that it can be restarted at the checkpoint on a different set of cluster nodes at a later time. In checkpointing and restarting a distributed application, our system separates the processing of network state from per node application state. It only requires synchronized operation in checkpointing the network state, which represents a small fraction of the overall checkpoint time. Checkpoint-restart operations on per node application state are designed to proceed in parallel with minimal synchronization requirements

among nodes, resulting in faster checkpoint and restart times. The system can also directly stream checkpoint data from one set of nodes to another, enabling direct migration of a distributed application to a new set of nodes without saving and restoring state from secondary storage.

Our system uniquely supports complete checkpoint-restart of network state in a transport protocol independent manner without application or library support. It leverages the socket abstraction and correctly saves and restores all socket state, including socket parameters, socket data queues, and minimal protocol specific state. We accomplish this in a portable manner using the standard socket interface without detailed knowledge of the underlying network protocol data structures. We account for network state in a protocol independent manner for reliable and unreliable network protocols, including TCP, UDP and raw IP.

We have implemented a prototype that runs across multiple Linux operating system versions, including both Linux 2.4 and 2.6 kernels, and demonstrated its effectiveness on cluster systems in checkpointing and restarting a number of real distributed application workloads, including both MPI and PVM applications. Our results show that our system imposes very low virtualization overhead and provides fast checkpoint-restart times for unmodified distributed network applications. We show that our approach uniquely provides distributed checkpoint-restart functionality without any application, library, kernel, or network protocol modifications.

F Desktop cloud utility computing infrastructure

Using our checkpoint-restart and migration mechanisms, we created MobiDesk, a mobile virtual desktop cloud computing infrastructure. With wire-speed network technologies scaling at faster Moore's exponents than silicon, MobiDesk leverages rapid improvements in network bandwidth, cost, and ubiquity to address the limitations of the current personal computing model. MobiDesk uses the network to decouple a user's desktop computing session from the end-user device by moving all application logic to hosting providers. In this manner, end-user devices are simply used to transmit user input and display application output, allowing them to be simple stateless clients. MobiDesk also decouples a user's desktop computing session from the underlying operating system and server instance, allowing a user's entire computing environment to be migrated transparently from one server to another. This enables a server to be brought down for maintenance and upgraded in a timely manner with minimal impact on the availability of a user's computing services. Once the original machine has been updated, the user's computing session can be migrated back and continue to execute even though the underlying operating system may have changed. MobiDesk ensures that any network connections associated with the user's computing session are maintained, even as the session is migrated from one machine to another. MobiDesk provides these benefits without modifying, recompiling, or relinking applications or operating system kernels. MobiDesk requires no changes to clients other than being able to execute a simple user-space application to process and display input and output.

MobiDesk provides a mobile virtual desktop computing environment by introducing a thin virtualization layer between a user's computing environment and the underlying system. MobiDesk focuses on virtualizing three key system resources: display, operating system, and network. MobiDesk virtualizes display resources by providing a virtual display driver that efficiently encodes and redirects display updates from the server to an end-user device. MobiDesk virtualizes operating system resources by providing a virtual private namespace for each desktop computing session. The namespace offers a host independent virtualized view of an operating system, enabling the session to be transparently migrated from one server to another. MobiDesk virtualizes network resources by providing virtual address identifiers for connections, and a transport-independent proxy mechanism. Together, they preserve all network connections associated with a user's computing session, even if it is migrated from one server to another inside the MobiDesk server infrastructure.

The MobiDesk hosted desktop computing approach provides a number of important benefits over current computing approaches, including (1) high-availability and reliable application services, (2) persistence and continuity of business logic, (3) secure, low-cost global access and transparent user mobility, and (4) on-demand access to application and computational resources.

G Process migration architecture to prevent DDOS attacks

Using our process migration mechanisms, we created a solution to the denial of service (DoS) problem that does not rely on network infrastructure support, conforming to the end-to-end (*e2e*) design principle. Our approach is to combine an overlay network, which allows us to treat authorized traffic preferentially, with a lightweight process-migration environment that allows us to move services easily between different parts of a distributed system. Functionality residing on a part of the system that is subjected to a DoS attack migrates to an unaffected location. The overlay network ensures that traffic from legitimate users, who are authenticated before they are allowed to access the service, is routed to the new location. We demonstrate the feasibility and effectiveness of our approach by measuring the performance of an

experimental prototype against a series of attacks using PlanetLab, a distributed experimental testbed. Our results show that the end-to-end latency remains at acceptable levels during regular operation, increasing only by a factor of 2 to 3, even for large overlays. When a process migrates due to a DoS attack, the disruption of service for the end user is in the order of a few seconds, depending on the network proximity of the servers involved in the migration.

H Mobile computing using portable storage devices

Using our checkpoint-restart mechanisms, we created present WebPod, a portable system that enables mobile users to use the same persistent, personalized web browsing session on any Internet-enabled device. No matter what computer is being used, WebPod provides a consistent browsing session, maintaining all of a user's plugins, bookmarks, browser web content, open browser windows, and browser configuration options and preferences. This is achieved by leveraging rapid improvements in capacity, cost, and size of portable storage devices. WebPod provides a virtualization and checkpoint/restart mechanism that decouples the browsing environment from the host, enabling web browsing sessions to be suspended to portable storage, carried around, and resumed from the storage device on another computer. WebPod virtualization also isolates web browsing sessions from the host, protecting the browsing privacy of the user and preventing malicious web content from damaging the host. We have implemented a Linux WebPod prototype and demonstrate its ability to quickly suspend and resume web browsing sessions, enabling a seamless web browsing experience for mobile users as they move among computers.

We also created DeskPod, a portable system that provides a highly reliable desktop computing environment for mobile users by leveraging rapid improvements in capacity, cost, and size of portable storage devices. DeskPod decouples a user's live computing environment from being tied to any single computer so that it can be suspended to portable storage, carried around, easily copied for fault-resilience, and resumed from the storage device to provide the user with the same persistent, personalized computing environment on another computer. DeskPod achieves this by providing a virtualization and checkpoint/restart mechanism that decouples a desktop computing environment from any single hardware device so that it can be stored and executed anywhere, improving desktop computing reliability by eliminating a potential single point of failure. We have implemented a Linux DeskPod prototype and demonstrate its ability to quickly suspend and resume desktop sessions, enabling a seamless computing experience for mobile users as they move among computers.

References

- [1] R. Baratto, S. Potter, G. Su, , and J. Nieh. MobiDesk: Mobile Virtual Desktop Computing. In *Proceedings of the Tenth Annual ACM International Conference on Mobile Computing and Networking (MobiCom 2004)*, Sept. 2004.
- [2] A. D. Keromytis, J. Parekh, P. N. Gross, G. Kaiser, V. Misra, J. Nieh, D. Rubenstein, and S. Stolfo. A Holistic Approach to Service Survivability. In *Proceedings of the 2003 ACM Workshop on Survivable and Self-Regenerative Systems*, pages 11–22, Fairfax, VA, Oct. 2003.
- [3] O. Laadan, R. Baratto, D. Phung, S. Potter, and J. Nieh. DejaView: A Personal Virtual Computer Recorder. In *Proceedings of the Twenty-first ACM Symposium on Operating Systems Principles (SOSP 2007)*, pages 279–292, Stevenson, WA, Oct. 2007.
- [4] O. Laadan and J. Nieh. Transparent Checkpoint-Restart of Multiple Processes on Commodity Operating Systems. In *Proceedings of the 2007 USENIX Annual Technical Conference*, pages 323–336, Santa Clara, CA, June 2007.
- [5] O. Laadan, D. Phung, and J. Nieh. Transparent Checkpoint-Restart of Distributed Applications on Commodity Clusters. In *Proceedings of the 2005 IEEE International Conference on Cluster Computing (Cluster 2005)*, Boston, MA, Sept. 2005.
- [6] S. Potter and J. Nieh. AutoPod: Unscheduled System Updates with Zero Data Loss. In *Proceedings of the Second IEEE International Conference on Autonomic Computing (ICAC 2005)*, pages 367–368, Seattle, WA, June 2005.
- [7] S. Potter and J. Nieh. Breaking the Ties that Bind: Process Isolation and Migration. *login*, 30(6):14–17, Dec. 2005.
- [8] S. Potter and J. Nieh. Reducing Downtime Due to System Maintenance and Upgrades. In *Proceedings of the Nineteenth Large Installation System Administration Conference (LISA 2005)*, pages 47–62, San Diego, CA, Dec. 2005.

- [9] S. Potter and J. Nieh. WebPod: Persistent Web Browsing Sessions with Pocketable Storage Devices. In *Proceedings of the Fourteenth International World Wide Web Conference (WWW 2005)*, pages 603–612, Chiba, Japan, May 2005.
- [10] S. Potter and J. Nieh. Highly Reliable Mobile Desktop Computing in Your Pocket. In *Proceedings of the IEEE Computer Society Signature Conference on Software Technology and Applications (COMPSAC 2006)*, pages 247–254, Chicago, IL, Sept. 2006.
- [11] S. Potter, J. Nieh, and M. Selsky. Secure Isolation of Untrusted Legacy Applications. In *Proceedings of the Twenty-first Large Installation System Administration Conference (LISA 2007)*, pages 117–130, Dallas, TX, Nov. 2007.
- [12] A. Stavrou, A. D. Keromytis, J. Nieh, V. Misra, and D. Rubenstein. MOVE: An End-to-End Solution To Network Denial of Service. In *Proceedings of the ISOC Symposium on Network and Distributed System Security (NDSS 2005)*, Feb. 2005.