ExtraVirt: Detecting and recovering from transient processor faults

Dominic Lucchetti Steven K. Reinhardt Peter M. Chen

Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109-2122
covirt@umich.edu

Reliability is becoming an increasingly important issue in modern processor design. Smaller feature sizes and more numerous transistors are projected to increase the frequency of transient faults [4, 5]. Our project, ExtraVirt, leverages the trend toward multi-core and multi-processor systems to survive these transient faults. Our goals are (1) to add fault tolerance without modifying existing operating systems, applications or hardware, (2) to minimize the time spent executing software that cannot tolerate faults, and (3) to minimize the time and space overhead needed to detect and recover from faults. We accomplish these goals by leveraging virtual-machine technology and by sharing memory and I/O devices across replicas. ExtraVirt extends prior work on VM-level fault tolerance[2] by detecting and recovering from non-fail-stop faults and by running multiple replicas efficiently on a single machine.

Detecting and recovering from processor faults requires running multiple replicas, comparing their outputs before they go to external devices (e.g., network, disk, monitor), and correcting faulty replicas before their output becomes visible. The unit of replication in ExtraVirt is a virtual machine; this accomplishes our first goal of enabling fault tolerance without modifying existing operating systems, applications or hardware[2]. ExtraVirt keeps replicated virtual machines consistent in the presence of non-deterministic input and events through virtual-machine logging and replay[2, 3], leaving only processor faults as non-determined input. Any divergence in output can thus be attributed to a processor fault. ExtraVirt manages its replicas and additional functionality through extensions residing in the Xen virtual-machine monitor[1].

Our second goal is to maximize the system's tolerance of faults by minimizing the time spent executing software that is not replicated. Since only software above the replication-management layer (RML) is replicated automatically, this goal dictates that we locate the RML below as much software as possible. Implementing the RML as an extension to a virtual-machine monitor is the first step toward this goal: all operating system and application software runs in a replicated virtual machine above the RML. Even with a virtual-machine approach, however, the virtual-machine monitor and RML remain vulnerable to faults. An open research question is how to tolerate faults that occur while executing outside the automatically replicated software. One possible approach being the use of a compiler based approach for replication of execution within the hypervisor[6].

Our third goal is to minimize time and space overhead needed to detect and recover from faults. To minimize time overhead, we leverage the fact that detecting a fault requires only two replicas, while identifying which replica is faulty requires a third. Thus when the two replicas used for detection diverge ExtraVirt dynamically create a third replica by replaying from a prior, known-good state. ExtraVirt periodically creates a known-good state by stopping the replicas at an identical point in their executions and verifying that their states remain identical.

To minimize the memory overhead of running multiple replicas, ExtraVirt shares memory between replica using a copy-on-write approach to ensure isolation. Using the copy-on-write technique creates a private copy of a page when either replica modifies it and thereby allows sharing while preserving the independence of failures between replicas[1]. In the absence of faults, the memory contents of one replica at a given point of execution should match the memory contents of the other replica at the same point of execution. ExtraVirt takes advantage of this similarity by combining opportunistically pages that have been verified to be identical between replicas[7]. ExtraVirt verifies output before sending it to external devices and so does not need to replicate disk storage, nor does it suffer increased network overhead.

We are currently implementing replica management in ExtraVirt. Open research issues include how to handle permanent faults, how to tolerate faults that occur while executing outside the automatically replicated software, and where the RML should be located relative to device drivers.

1. REFERENCES

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen

[1]A hardware memory fault to a shared page could affect both replicas, but we assume these are handled by ECC memory.

and the Art of Virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003)*, October 2003.

[2] T. C. Bressoud and F. B. Schneider. Hypervisor-Based Fault Tolerance. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP 1995)*, December 1995.

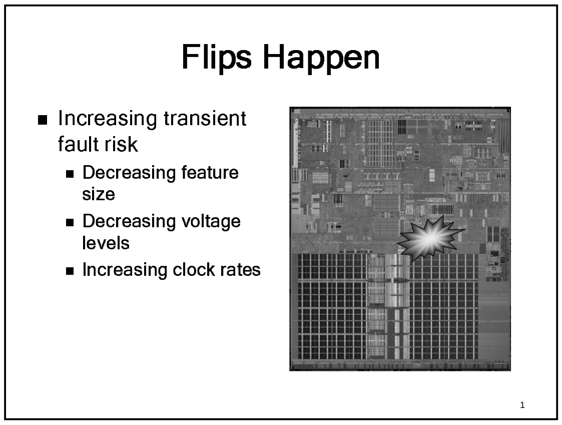
[3] G. W. Dunlap, S. T. King, S. Cinar, M. Basrai, and P. M. Chen. ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002)*, December 2002.

[4] S. S. Mukherjee, J. Emer, and S. K. Reinhardt. The Soft Error Problem: An Architectural Perspective. In *Proc. 11th Int'l Symp. on High-Performance Computer Architecture (HPCA)*, Februrary 2005.

[5] E. Normand. Single Event Upset at Ground Level. *IEEE Transactions on Nuclear Science*, 43(6), December 1996.

[6] George A. Reis, Jonathan Chang, Neil Vachharajani, Ram Rangan, and David I. August. SWIFT: Software implemented fault tolerance. In *Proceedings of the 3rd International Symposium on Code Generation and Optimization*, March 2005.

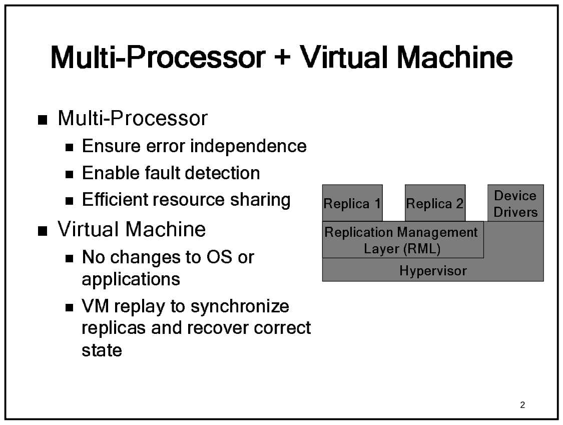
[7] C. A. Waldspurger. Memory Resource Management in VMware ESX Server. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002)*, 2002.



Flips Happen

- Increasing transient fault risk
  - Decreasing feature size
  - Decreasing voltage levels
  - Increasing clock rates

1

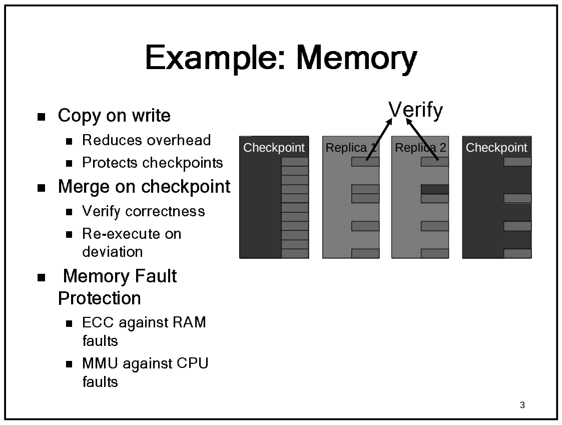


Multi-Processor + Virtual Machine

- Multi-Processor
  - Ensure error independence
  - Enable fault detection
  - Efficient resource sharing
- Virtual Machine
  - No changes to OS or applications
  - VM replay to synchronize replicas and recover correct state

Replica 1
Replica 2
Device Drivers
Replication Management Layer (RML)
Hypervisor

2

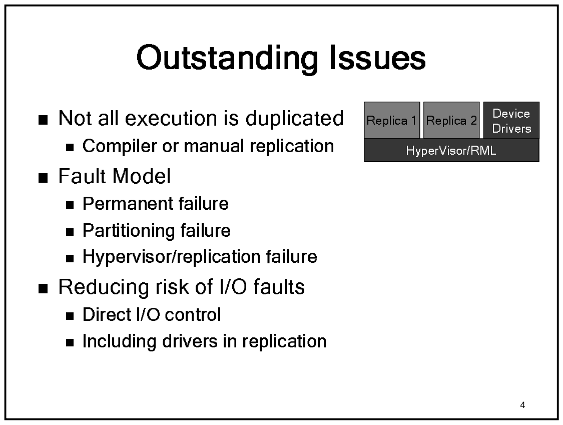


Example: Memory

- Copy on write
  - Reduces overhead
  - Protects checkpoints
- Merge on checkpoint
  - Verify correctness
  - Re-execute on deviation
- Memory Fault Protection
  - ECC against RAM faults
  - MMU against CPU faults

Verify
Checkpoint
Replica 1
Replica 2
Checkpoint

3



Outstanding Issues

- Not all execution is duplicated
  - Compiler or manual replication
- Fault Model
  - Permanent failure
  - Partitioning failure
  - Hypervisor/replication failure
- Reducing risk of I/O faults
  - Direct I/O control
  - Including drivers in replication

Replica 1
Replica 2
Device Drivers
HyperVisor/RML

4