


“Otherworld” - giving applications a chance to survive OS kernel crashes

*Alex Depoutovitch and Michael Stumm
University of Toronto*



A problem has been detected and windows has been shut down to prevent damage to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)

*** SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, Datestamp 3d6dd67c

Otherworld

- Mechanism that:
 - Reboots just OS kernel (microreboot)
 - Preserves state of running applications
 - Continues running applications so that they can:
 - save the latest state and restart
and/or
 - continue their execution

Existing fault tolerance mechanism

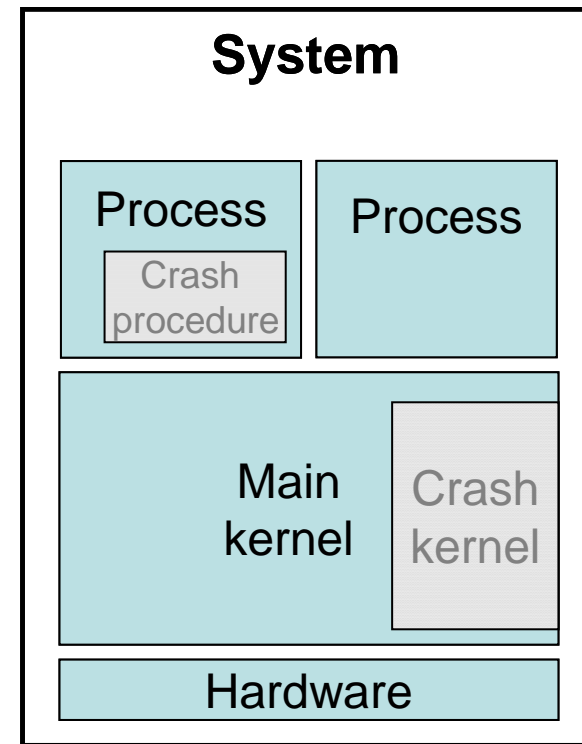
- Checkpointing
 - Introduces overhead
 - Most recent state is lost
- Redundant calculations
 - Increases system cost and complexity
- Continuous replication
 - Protects only from hardware faults
- Micro-kernels
 - Not directly applicable to production OSES
 - Error propagation
- Isolation (Nooks, etc.)
 - Introduces overhead
 - Protects from drivers and other kernel extensions failures

Otherworld benefits

- Works with existing OS architectures
- Protects from faults in any part of the kernel
- Does not require specialized or redundant hardware
- Preserves the very latest application state
- Minimal or no changes to applications
- Small or zero run-time overhead

Normal operations

- Main kernel
 - boots first
 - manages the system
- Crash kernel: cold standby
 - loaded into memory by main kernel
 - is inactive
 - kept protected and uninitialized
- Crash procedure (optional)
 - registered by the process
 - allows kernel to notify the process of a kernel microreboot



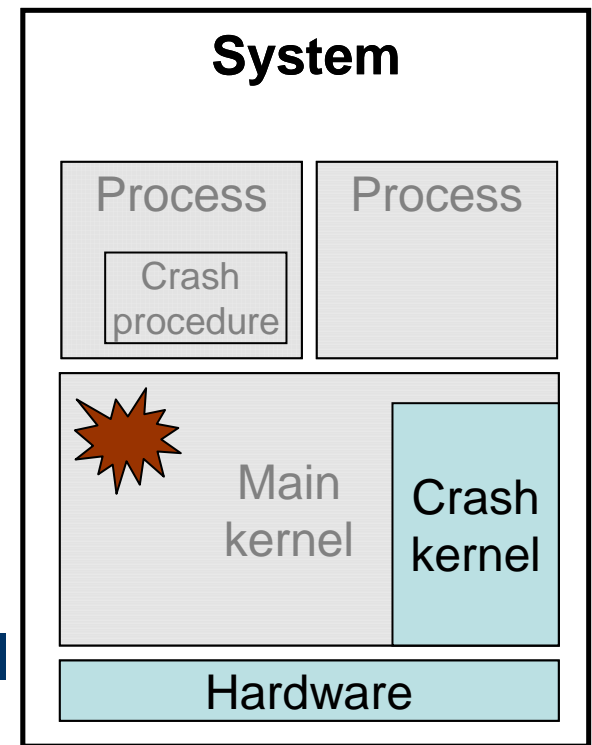
On kernel crash

Main kernel instead of panic:

1. saves user thread context
2. shuts down CPUs
3. passes control to crash kernel

Crash kernel:

1. initializes itself
2. recovers state from main kernel
3. resurrects applications
4. morphs itself into main kernel



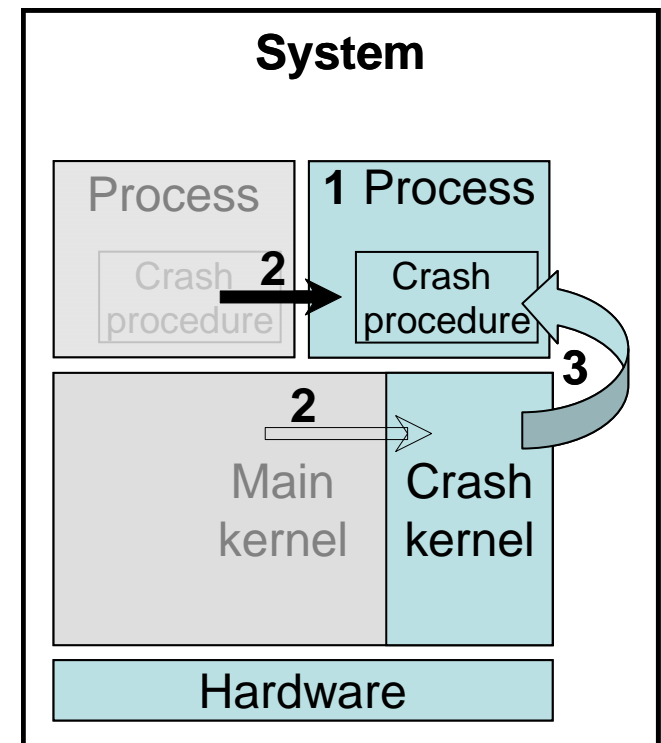
Recovery

Processes resurrection:

1. create new process
2. recover process information
 - Memory, open files, etc.
3. call process crash procedure and/or
4. continue process execution

Morphing into main kernel:

1. reclaim remaining memory
2. load new crash kernel
3. continue regular OS activities



Active

Inactive

Resources resurrection

Resources restored automatically by the Crash kernel	Currently we do not restore:
<ul style="list-style-type: none">•Application physical memory pages•Pages swapped to disk•Memory mapped files•Open files•File buffers•Physical screen content•Thread execution context•Signal handlers•Shared memory regions	<ul style="list-style-type: none">•Network sockets•Pipes•Pseudo terminals

That's where crash procedure is useful...

Crash procedure

- Role:
 - Called when application is resurrected and ready to run
 - Receives list of resources types not resurrected
- Functions:
 - Resurrects resources in application-specific way
 - Identifies application data corruption
 - Decides whether to :
 - save data and restart the application
 - continue application execution

Resurrection outcome

Resource resurrection	No crash procedure registered	Crash procedure registered
complete and successful	continue the process execution	crash procedure is called It can: <ul style="list-style-type: none">➤ Save application state and restart application
partial	resurrection will fail and process is killed	- or - <ul style="list-style-type: none">➤ Continue process execution

Continuing process execution

- Process was executing system call:
 - System call returns failure code
 - System call will have to be re-executed
- Process was running in user space:
 - Thread context was saved on the stack
 - Process execution continues from where it was stopped by the crash

Application state protection

- Separate page table sets for execution in user and kernel modes
- Kernel page tables do not map application space, protecting it from unauthorized access
- Page tables are switched on every system call
- Kernel is allowed to write to user space only through explicit function calls
- Introduces overhead (3%-12%)

Applications that benefit from Otherworld

- Database:
 - Up to 140 times faster than disk resident
 - More reliable in-memory databases
- Web server:
 - In-memory session is 25% faster than disk persistent
 - More reliable in-memory web session data
- Checkpointing
 - More reliable in-memory checkpointing
- Desktop application
 - Editors that can restore a document up to the last symbol entered

Evaluation

- OS
 - Linux kernel 2.6.18
 - KDump is used for loading the crash kernel
 - 2,700 lines of added and modified code
- Applications

Application	Crash procedure	Modified LOC
JOE	Not required	1
vi	Not required	0
BLCR	Not required	0
MySQL	Required	75
Apache/PHP	Required	115





MySQL

- Popular open source database server
 - 700,000 lines of code
 - Supports memory-resident tables
 - Amount of required changes: 75 lines of code
- Crash procedure (50 lines)
 - Calls MySQL functions to retrieve in-memory data
 - Saves the data to disk
 - Restarts the server
- Start-up code (25 lines)
 - Reads saved data
 - Populates in-memory tables
 - Continues normal server execution

Fault injection

- Reused fault injection tool (Rio cache, Nooks)
- Different types of faults
 - Stack corruption
 - Incorrect branch
 - Delete/change random instruction
 - Incorrect operands
 - Random writes
 -
- Tests ran in virtual machine
 - 30 faults injected during each experiment
 - For each application - 800 experiments with failures
 - 400 with application space protection and 400 without
 - 4,000 experiments in total
 - 97% – 98% success rate

Fault recovery results

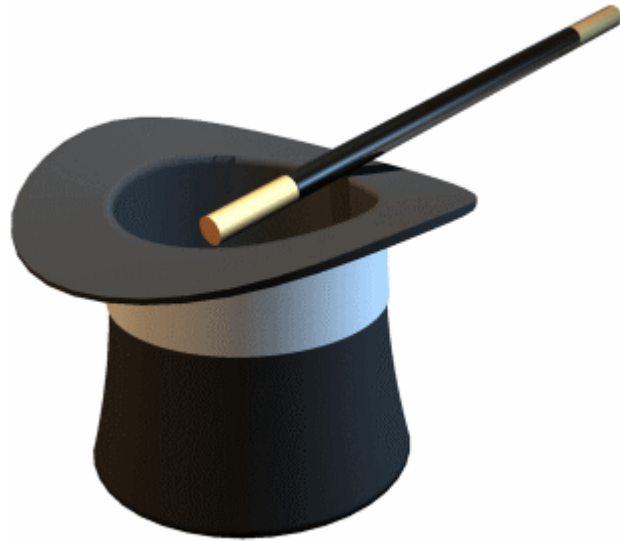
	Outcome	Frequency
	Crash kernel doesn't boot	2%-3%
	Application resurrection failure	Up to 0.5%
	Application data corruption: <ul style="list-style-type: none">➤ without protection➤ with protection	Up to: 0.5% (5 cases) 0.25% (1 case)
	Successful application resurrection	97%-98%

Conclusion

Otherworld is a fault recovery technique:

- Allows applications to survive OS kernel crash
- Requires only minor changes to the kernel and applications
- No or small run-time overhead
- Applicable to wide range of applications

Demo



Questions



Future work

- Automatic resurrection of network and IPC resources
- Reducing microreboot and resurrection times
 - Partial initialization at load time
 - Page remapping instead of copying
 - Exploiting device information from the main kernel
- Applying to hot kernel updates and system rejuvenation
- Detection and prevention of data corruption