Hot Patching

Advanced Operating Systems and Virtualization
Alessandro Pellegrini
A.Y. 2017/2018

Why hot patching?

- Huge costs of downtime → reduce cost of planned downtime

- Common tiers of change management:
  - incident response
    - "we are down and/or exploited"
  - emergency change
    - "we could go down: we are vulnerable"
  - scheduled change
    - "time is not critical, we keep safe"

Hot Patching
is handy here!

Why is Rebooting a Problem?

- Disruption to users/applications
- Sysadmins don't always have control of users or applications
- Many applications aren't distributed
- Re-architecting can be expensive or impractical
- Distributed systems need to reboot too
- (Up)time is money
- Hardware reboot failures

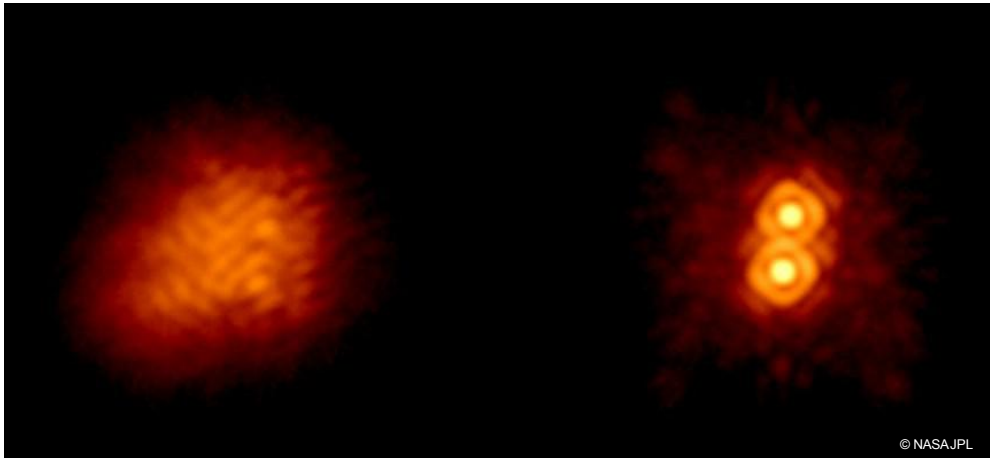# Barcelona Supercomputing Center
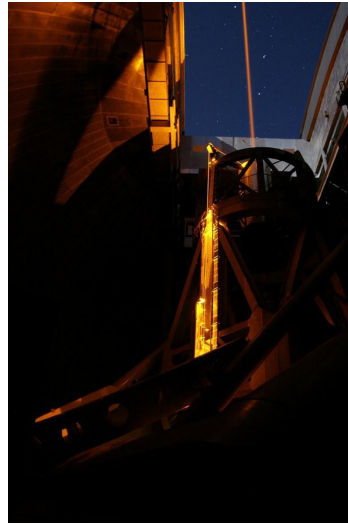
Mare Nostrum Supercomputer

- **50k Sandy Bridge cores**

- **The most beautiful supercomputer in the world**

- **Terabytes of data**

- **Reboot?**

NASA JPL

Hale telescope PALM-3000 Adaptive optics

© NASA JPL

- **5m telescope with adaptive optics on Mount Palomar**

- **Avoid atmospheric blurring in Real Time**

- **Control 3888 segments of a deformable mirror with a latency <250 µs**

- **Reboot?**

SAP HANA

In-memory database and analytics engine

HP DL980 w/ 12 TB RAM

- **4-16 TB of RAM**

- **All operations done in memory**

- **Disk used for journalling**

- **Active-Passive HA**

- **Failover measured in seconds**

- **Reboot?**

Not a New Idea:1943 Manhattan Project

- IBM punchcard automatic calculators were used to crunch the numbers

- A month before the Trinity nuclear device test, the question was: "What will the yield be, how much energy will be released?"

- The calculation would normally take three months to complete – recalculating any batches with errors

- Multiple colored punch cards introduced to fix errors in calculations while the calculator was running

Trinity test site, 16ms after initiation

Windows Hot Patching (2003)

- Windows Server 2003 SP1

- Stops Kernel execution for activeness check
  - Schedule procedures on all but current CPUs and keep them busy

- Uses short jumps patched into functions for redirection
  - The second redirection jumps to a new function

- Removed in Windows 8 and later versions

Linux kpatch (2014)

- Tries to overcome the costs and problems of rebooting systems to apply patches

- Based on two "simple" steps:
    1. Build the patch module (`kpatch-build foo.patch`)
    2. Apply the patch (`kpatch load kpatch-foo.ko`)

Building the Patch Module

- Much harder than patching the kernel
- Compile kernel with/without patch
- Compare binaries
- Detect which functions have changed
- Extract object code of changed functions into patch module

Patching the Kernel

- Load new functions into memory

- Link new functions into kernel

  - Allows access to unexported kernel symbols

- Activeness safety check

  - Prevent old & new functions from running at same time

  - `stop_machine()` + stack backtrace checks
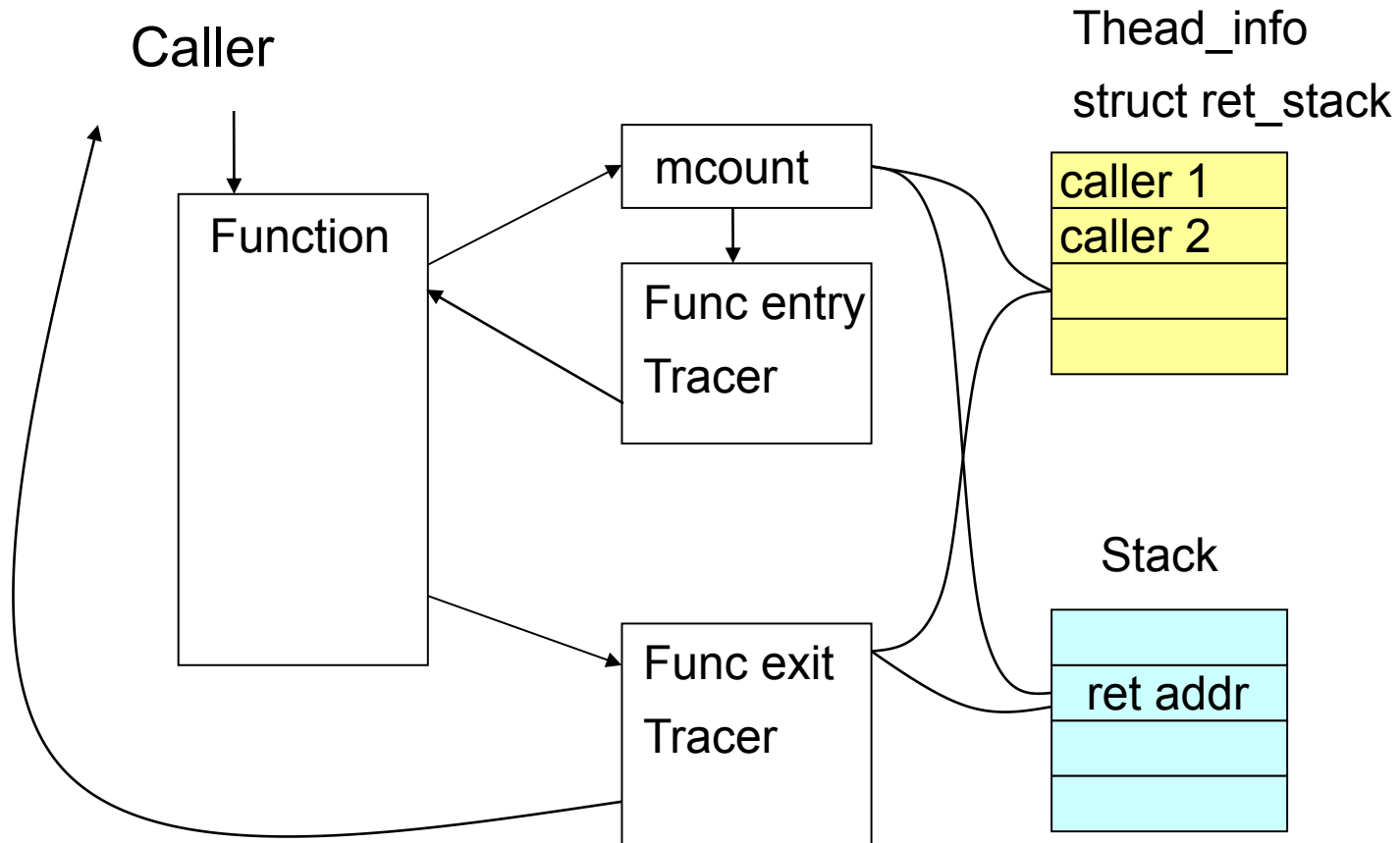
- Patch it!

  - Uses ftrace

What is ftrace?

- Ftrace is the first generic tracing system to get mainlined
  - Mainlined in 2.6.27
  - Derived from RT-preempt latency tracer
- Provides a generic framework for tracing
  - Infrastructure for defining tracepoints
  - Ability to register different kinds of tracers
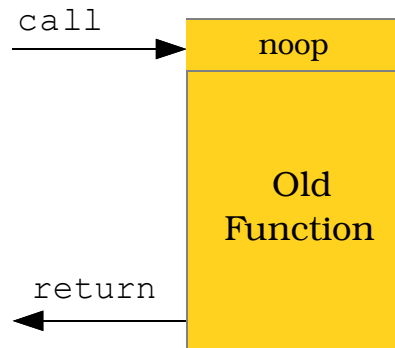 - Specialized data structure (ring buffer) for trace data storage

ftrace Schematization

Caller

Function

mcount

Func entry
Tracer

Func exit
Tracer

Thead_info
struct ret_stack

| caller 1 |
| caller 2 |
| |
| |

Stack

| |
| ret addr |
| |
| |

Patching with ftrace

Before patching:

After patching: