



Solaris FMA and Xen

- Frank van der Linden
- Sun Microsystems

Overview

- What is FMA?
- Requirements to implement FMA
- Changes made to Xen
- Changes made to Solaris
- Status / future work

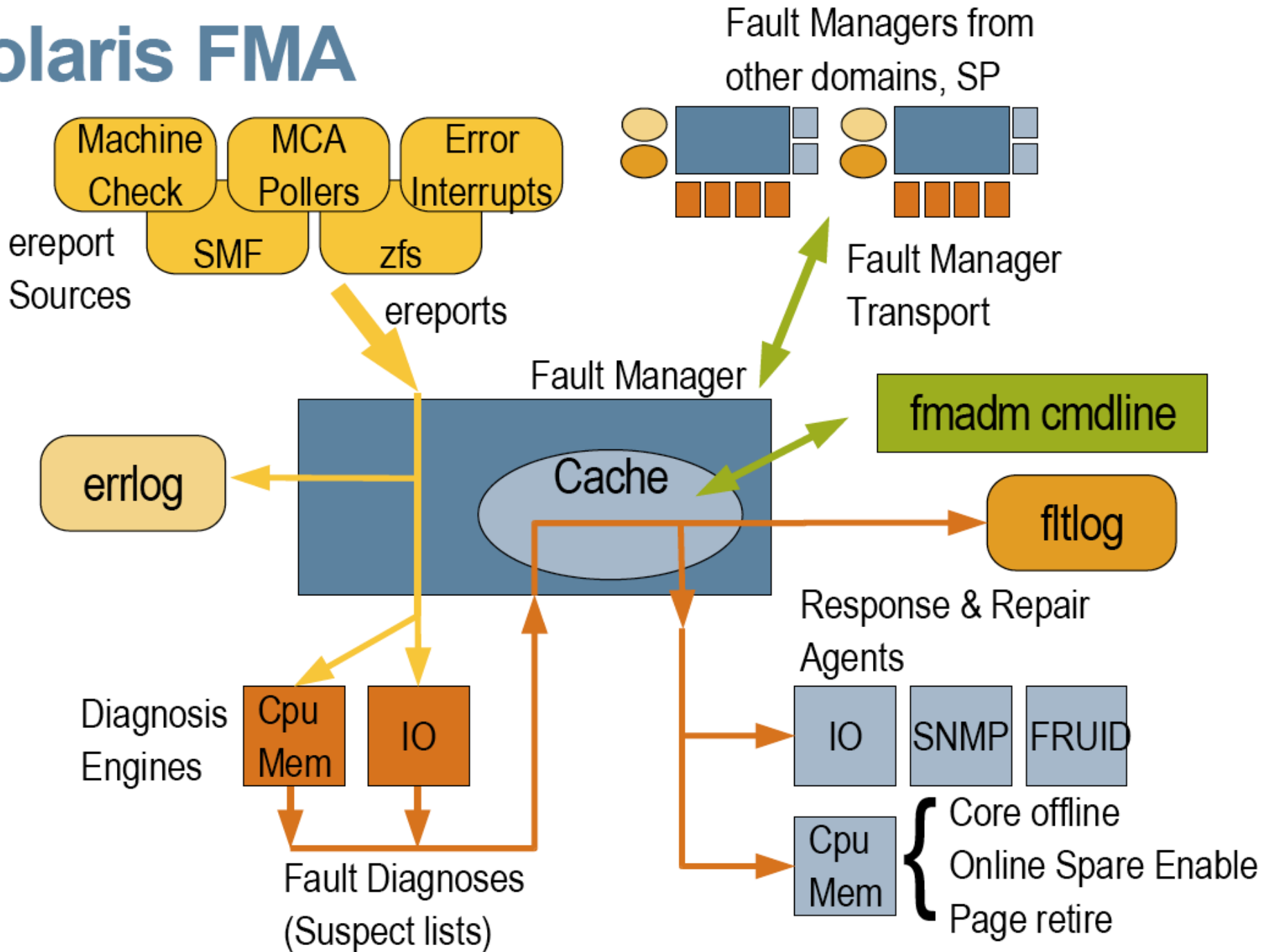
What is FMA?

- Fault Management Architecture
 - > Also known as Solaris Fault Management
- Provides fault diagnosis and recovery
- Without FMA:
 - > Random panics
 - > Cryptic error messages
 - > Needless debugging or hardware swapfests
- With FMA:
 - > Detailed reports on failure
 - > If possible, recovery

What is FMA? (cont.)

- Hardware fault handling for:
 - > CPU
 - > Cache
 - > Memory
 - > I/O (PCI)
- Initial detection through traps (MCE) or polling
- How to work around fatal errors?
 - > CPU offline
 - > Page retire

Solaris FMA



Requirement: hardware access

- In the plain Solaris implementation, needs access to the actual hardware to implement FMA
 - > Trap handlers
 - > PCI interrupts
 - > Reading MSR for polling and diagnostic
 - > Writing MSR for testing / error injection
- But, this is a virtualized environment
- Need to find a balanced way to provide the access that FMA needs, and the abstractions of the hypervisor.

General design

- Hypervisor does the low-level grunge work
 - > Immediate retrieval of error telemetry for CPU and memory after an MCE / NMI
 - > Hypercall interface to retrieve telemetry and to perform recovery actions
- Solaris runs as dom0
- DomUs will be informed of fatal errors
- Recoverable errors handled by dom0

Xen changes: trap handling and telemetry collection

- Based on work by Christoph Egger @ AMD
- Extend the MCE handling to collect telemetry
 - > Read the MSR
 - > Store telemetry in a global structure
- Domains can install an MCE/NMI trap handler for reporting on unrecoverable errors.
- Dom0 can install a VIRQ_MCA handler to be informed about recoverable errors (detected by polling)
- Hypercalls to retrieve telemetry

Xen changes: fault injection

- Fault injection is used for testing purposes
- Two ways to inject faults:
 - > Interpose fault data (MSR values) that are “read” later.
 - > Actually write MSRs
- Add hypercall to either interpose data or actually write the MSR values
 - > Only a subset of MSRs relating to MCA can be written

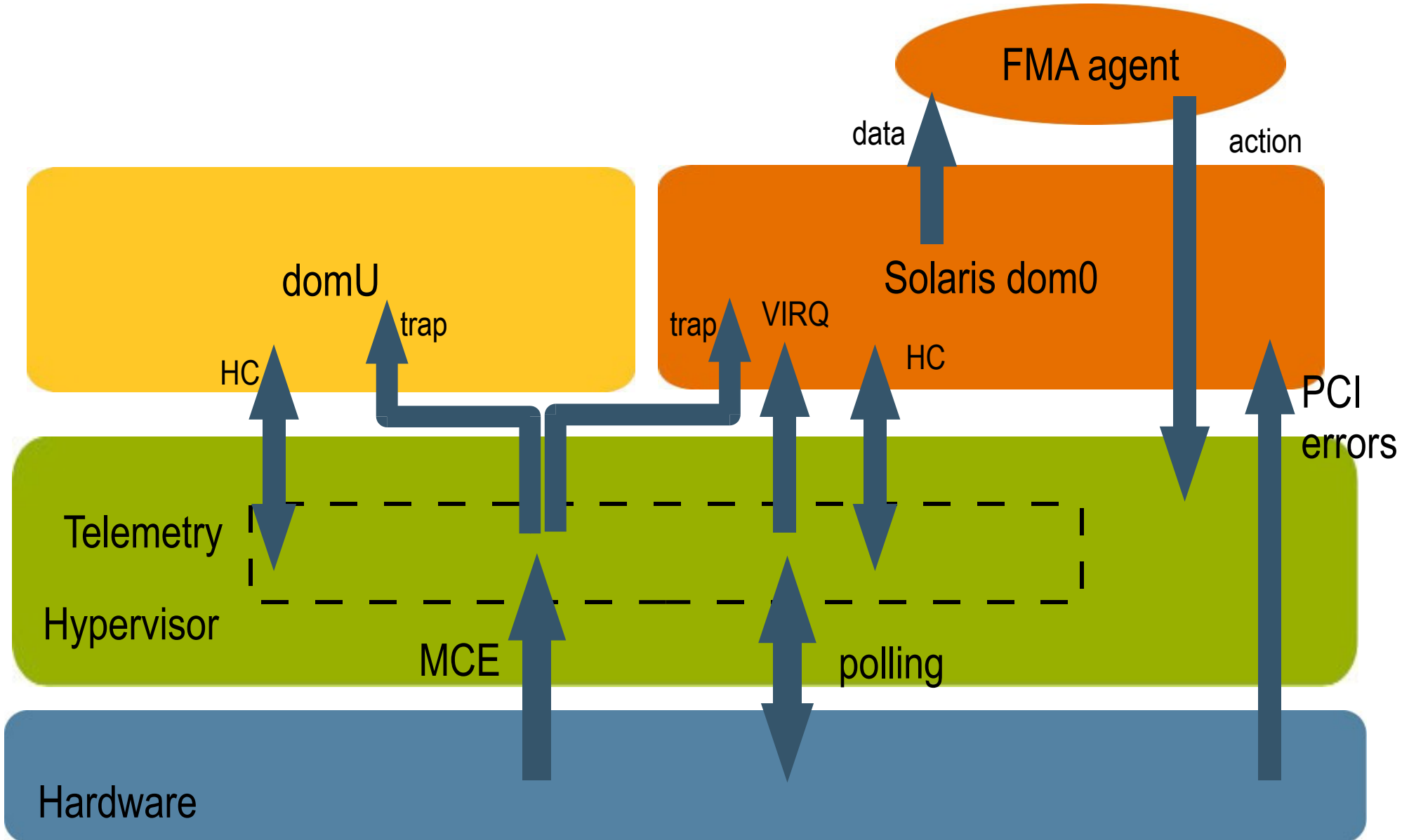
Xen changes: exposing CPU data

- FMA needs to know a little more about the actual CPUs than is available right now:
 - > Hardware CPU/Core/Thread ID
 - > Number of (active cores)
 - > APIC id
 - > CPUID info
- Hypercall to expose this information
- Information is versioned; for an event that changes this information (e.g. CPU offline or online), the version is incremented.

Xen changes: CPU offline

- The decision to offline a CPU is made by the FMA recovery code, running on Solaris/dom0
- It can take a physical CPU offline through a hypercall.
- There is also a hypercall to bring a CPU back online
- Restriction: physical CPU 0 seems to have some special tasks in the hypervisor, can not currently be offlined.

Overview



How to take a CPU offline (1)

- Stop scheduling VCPUs on it
 - > Scheduler picks from a new cpu set, `cpu_sched_map`
 - > Take CPU out of `cpu_sched_map`
- Force all VCPUs off the CPU
 - > Go through all domains and force a migration of all their VCPUs onto a new physical CPU
 - > Also migrate timers
- This might break CPU affinity or dom0 pinning
 - > Nothing we can do about that; just log the event

How to take a CPU offline (2)

- Interrupts must be re-routed
 - > There is some existing code for this
- Race condition: PIRQ acks
 - > May come in from a domain after the interrupt was re-routed.
 - > Can't wait for a domain to do the ack
 - > So, just do the hardware ack if one is needed, and turn the PIRQ ack from the domain into a no-op when it comes in.
- Finally, put the CPU into a sleep loop

Solaris changes

- Don't depend on the hardware access interface
 - > Make FMA modules, such as CPU handling ocde, use an abstraction layer for MSR reads, topography info, etc
 - > Abstraction layers and libraries also come in handy for the Sparc architecture and LDOM virtualization
- Don't depend on MSI functionality for PCI error reporting
 - > Fall back on normal PIRQ if MSI not available
- Retrieve actual physical CPU info when coming up.
- Set up NMI and MCE handlers for dom0 case.

Status / Future work

- Done:
 - > Telemetry retrieval
 - > Physical CPU info retrieval
 - > CPU offline
- Tested both for simulated errors and actual hardware errors
- To be done:
 - > CPU online
 - > Page retire
 - > Cleanup
 - > Feed back the code
- FMA domain?



Questions?

- **Frank van der Linden**
– fvdl@sun.com