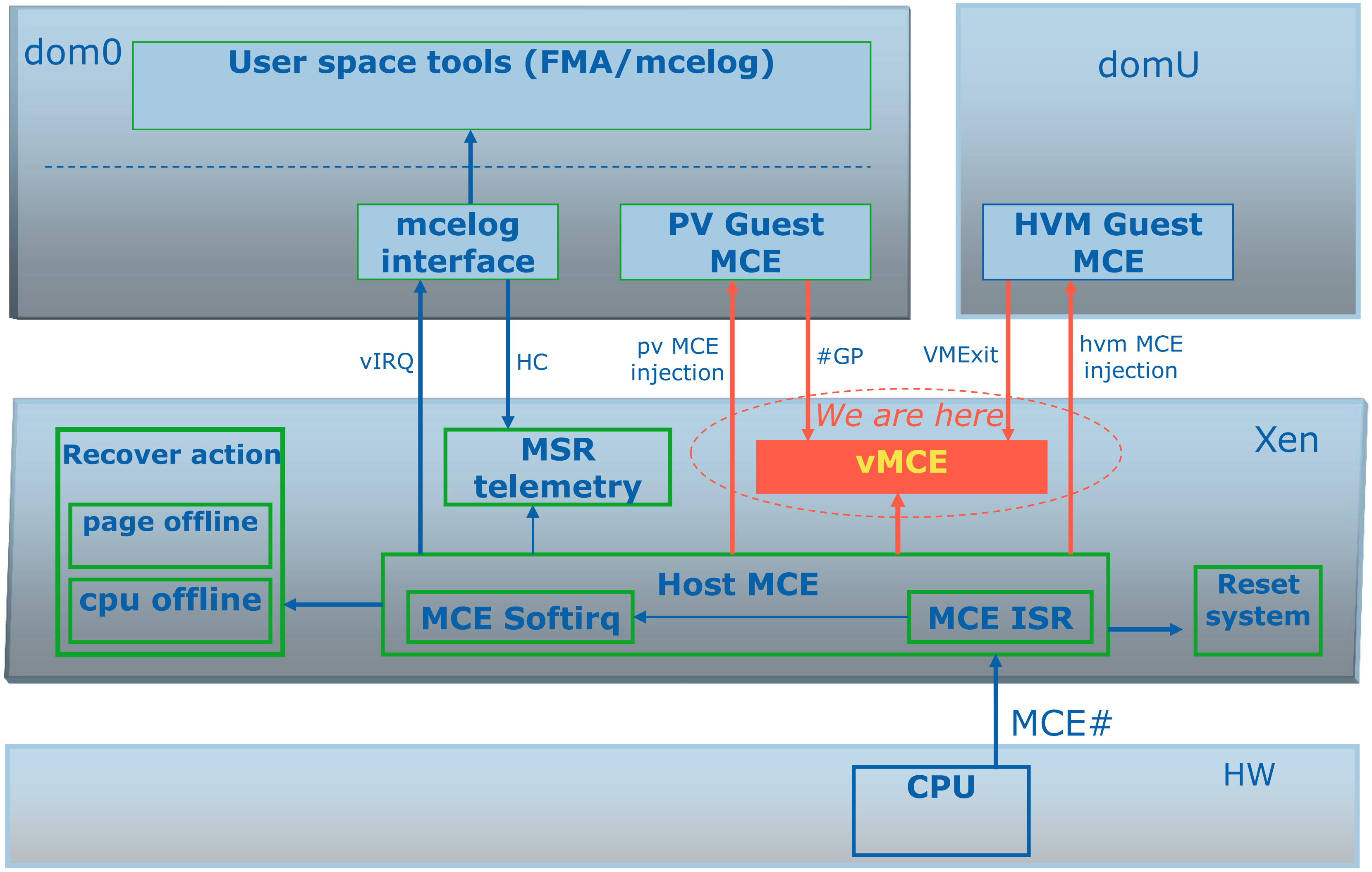
Xen vMCE Design

Liu, Jinsong <jinsong.liu@intel.com>

Jiang, Yunhong <yunhong.jiang@intel.com>

Auld, Will <will.auld@intel.com>

Xen MCE Architecture



Background

- Xen mce components
  - Xen MCE ISR
  - Xen MCE softirq
  - vMCE
  - Dom0 mcelog interface

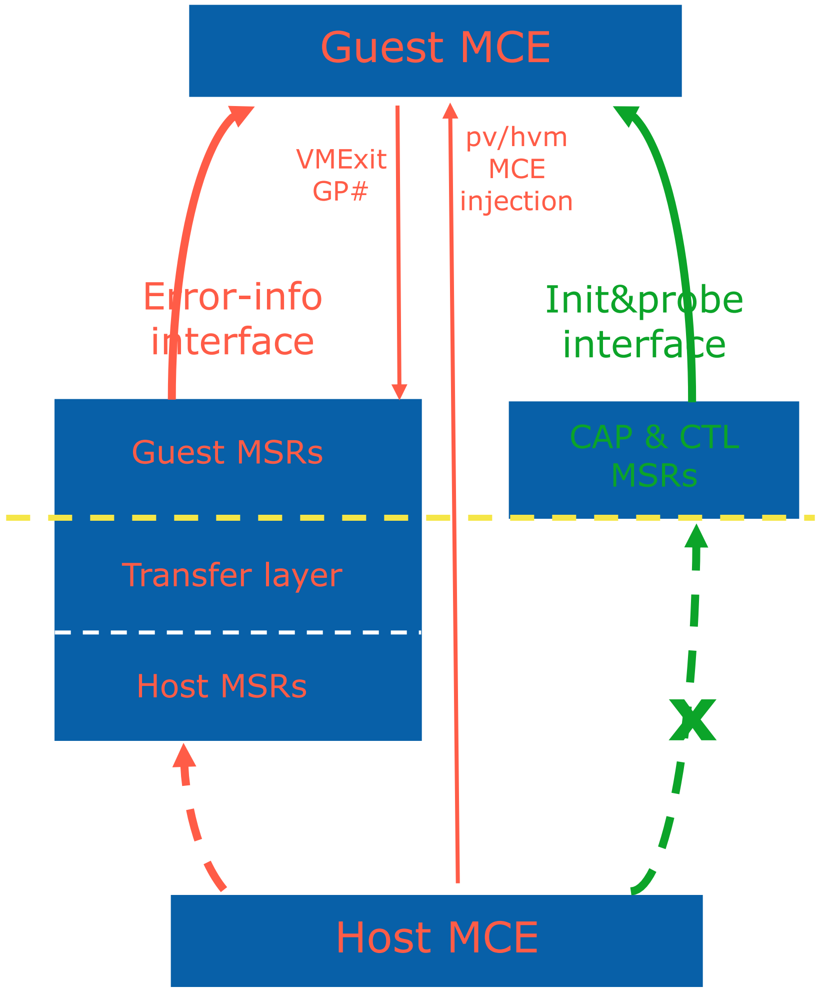
- 3 major points of vMCE logic
  - How to inject MCE
  - How to emulate MSRs
  - How to live migrate

- Major constraints of current vMCE
  - Bound to host
    - Host heterogeneous break migration, cannot control via cpuid
    - Non-arch issues of MSRs emulation
  - Weird per-domain MSRs
  - Unnatural MCE injection semantics

Xen vMCE approach (1)

- Separate guest MCE w/ host MCE
  - ➢ Unlike cpuid
    - vMCE can fake capabilities to guest if need
    - vMCE can mask capabilities if unnecessary
  - ➢ Emulate a well-defined MCE interface to guest, consists of
    - Init&probe interace
      - Basically for guest init & probe
      - disable UCR-unrelated capabilities
        - – Pointless to inject those error to guest
      - enable UCR-related capabilities
        - – No matter host support or not
      - disable MCG_CTL, stick all 1's for MCi_CTL
        - – Generally for h/w debug purpose
        - – Avoid model specific issue
    - Error-info interface
      - Only meaningful when real error generated
      - Filter & deliver to guest on demand
  - ➢ Leave host flexibility

- Error inject to guest
  - ➢ A transfer layer
    - Filter non-SRAO/SRAR banks
    - Translate host address to guest address
    - Deliver
      - Evaluate and deliver the most severe error



Xen vMCE approach (2)

- MSRs emulation
  - ➢ Pure s/w emulation
  - ➢ No non-arch issues again
    - For family-model issue root from legacy processor
      - Don’t care
      - Guest probe mce capabilities via MCG_CAP, not cpuid
    - For model-specific issue
      - For MCi_CTL, stick to all 1’s and treat guest write bit as ‘not implement’
      - For MSCOD of MCi_STATUS, expose all 0’s

- vMCE live migration
  - ➢ Error-info MSRs:
    - pointless to be migrated
  - ➢ Init&probe MSRs:
    - save/restore MCG_CAP, for the sake of future cap extension
    - save/restore MCi_CTL2

- MCE injection
  - ➢ Broadcast to all vcpus
  - ➢ Corner case
    - vcpus > pcpus, injection occur at a long time window
      - Tolerate it

Details

Point 1: MSR emulation (1)

- Init&probe interface:
  - MCG_CAP
    - Read-only, emulated as 'write not changes' since write effect is undefined
    - Disable UCR-unrelated cap (0)
      - MCG_CTL_P=0 to disable MCG_CTL
      - MCG_EXT_P=0, no ext regs, rdmsr/wrmsr for 180H-185H/188H-197H cause GP#
    - Enable UCR-related cap (1)
      - Enable MCG_TES_P to avoid model specific
      - Enable MCG_SER_P to support s/w recovery
      - Enable MCG_CMCI_P for sake of performance
      - Emulate 2 banks to guest
        - Bank0 to walk around old 'bank0 quirk', while bank1 to log most severe error
        - Migration would not blocked
        - If really 2 physical error occur at 1 MCE#, hypervisor would crash;
  - MCG_CTL: disabled via MCG_CTL_P=0
  - MCi_CTL
    - Stick all 1's, avoid model specific issues
      - Natively most MCi_CTL bits are 'not implement'. Processor does not write changes to bits that are not implements
      - MCi_CTL bits are usually for h/w debug purpose, os normally set them all 1's.
      - If guest crazy clear any bit, just treat the bit as 'not implement'. Guest would not surprise.
  - MCi_CTL2
    - Set CMCI_EN and error count threshold
    - Never inject CMCI to guest, just to avoid guest's periodic polling
    - Save/restore when migrate

MCG_CAP capabilities	Description	Defined interface for guest
Count field	CPU bank number	0000,0010
MCG_CTL_P	disable MCG_CTL when clear	0
MCG_EXT_P	extended regs	0
MCG_CMCI_P	CMCI	1
MCG_TES_P	MCi_status bit56:53 are arch when set	1
MCG_EXT_CNT	meaningful when MCG_EXT_P set	0000,0000
MCG_SER_P	support s/w error recovery	1

Point 1: MSR emulation (2)

- Error-info interface
  - ➢ A transfer layer
    - Filter non-SRAO/SRAR banks
    - Translate host address to guest address
    - Deliver:
      - Evaluate and deliver the most severe error to vcpu0
  - ➢ MCG_STATUS
    - Bit63:3 reserved, wrmsr should be same as rdmsr otherwise GP#
  - ➢ MCi_STATUS
    - Clear by writing 0s, writing 1s causes GP#
  - ➢ MCi_ADDR/MCi_MISC
    - Emulated as always implemented (no rdmsr/wrmsr GP# issue), ADDRV/MISCV=0 means no addr/info
    - Clear by writing 0s, writing 1s causes GP#
    - Translate host address to guest address

Point 1: MSR emulation (3)

- No non-arch issues again
  - ➢ For family-model issues root from legacy processors
    - Don't care, guest would not surprise
    - Guest probe mce capabilities via MCG_CAP, not cpuid
  - ➢ For model-specific issues
    - 2 model-specific fields
      - MCG_CTL/MCi_CTL
      - MCi_STATUS – MSCOD model specific error code
    - For MCG_CTL/MCi_CTL
      - For MCG_CTL, disable it via MCG_CTL_P = 0
      - For MCi_CTL
        - – Stick all 1's
        - – If guest crazy write bit to 0, treat it as 'not implement' bit and simply ignore write
    - For MCi_STATUS 'MSCOD' model specific error code
      - Stick all 0's to guest

Point 2: live migration

- live migration issue:
  - if migrate from A (large bank) to B (small bank)
    - will GP# at B when access MCi_CTL
    - Jan’s patch partly solve problem by save/restore MCG_CAP, avoid GP#
    - But B can only r/w some MCi_CTL after migrate

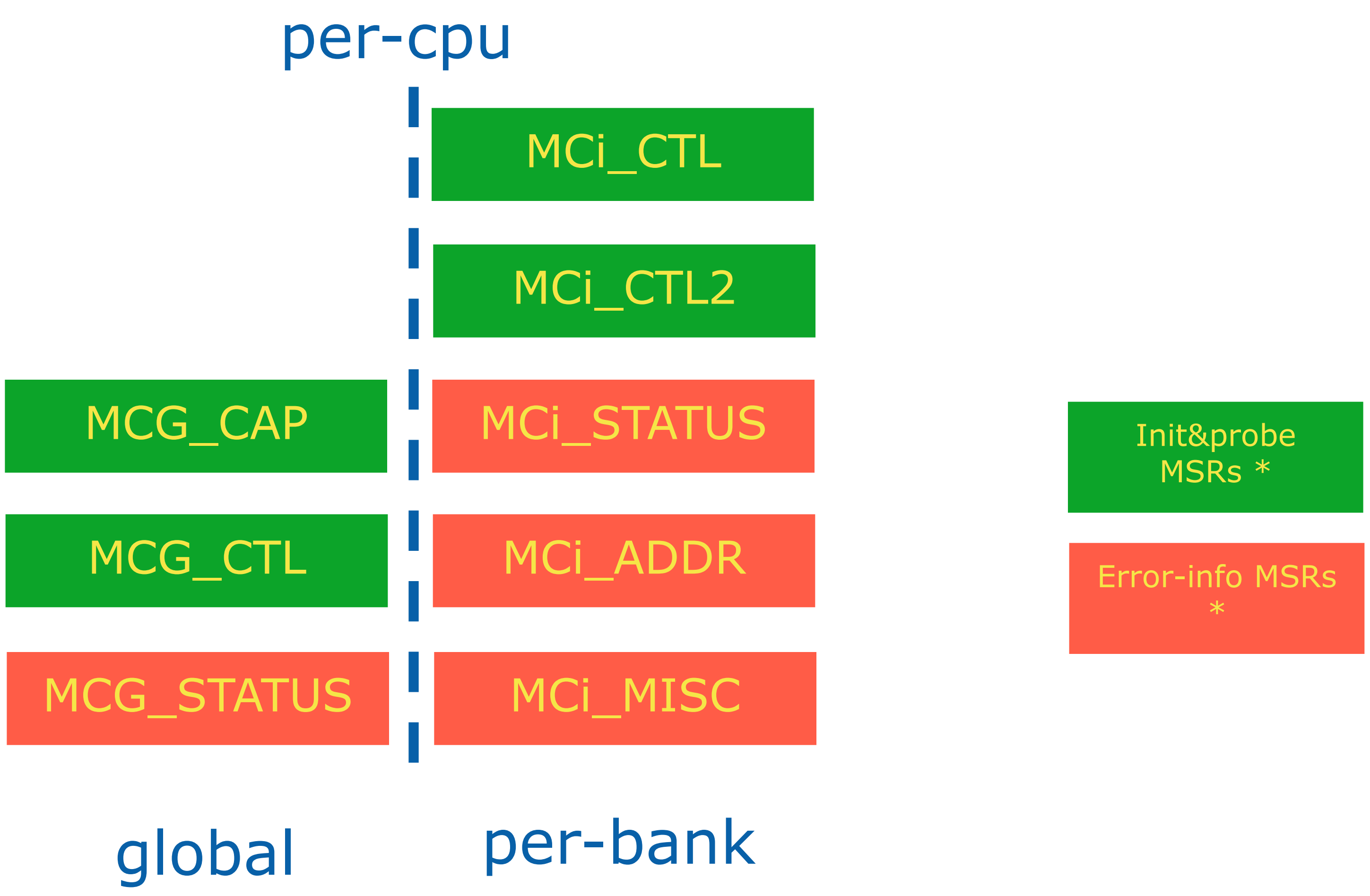
- approach:
  - For error-info MSRs
    - pointless to be migrated
  - For init&probe MSRs
    - For MCG_CAP
      - 2 banks, no GP# problem caused by bank number
      - Still save/restore when migrate, considering future CAP extension
      - command line option to allow admin to restrict the common CAP set among machine pool
    - For MCi_CTL2
      - Save/restore when migrate, set by guest so keep same to guest before/after migration
    - For MCG_CTL/MCi_CTL, unified at all platform
      - MCG_CTL disabled
      - MCi_CTL stick to all 1’s
  - Corner case: live migration while vMCE occur
    - abort migration
    - notify tools so that it got abort status & reason

Point 3: MCE injection

- Inject vMCE to all vcpus
  - ➢ Any risk when vcpus > pcpus, and injections occur at a long time window?
    - For win8, test indicate ok
    - For linux, it can tolerate this case (Tony Luck)
    - For Solaris, it can tolerate this case (Ashok)
    - Concern: hypervisor should be guest agnostic
  - ➢ Approach for vcpus > pcpus
    - Tolerate it
      - Worst case is guest kill itself

Backup

MCE MSRs


per-cpu
MCi_CTL
MCi_CTL2
MCG_CAP
MCi_STATUS
MCG_CTL
MCi_ADDR
MCG_STATUS
MCi_MISC
global
per-bank
Init&probe MSRs *
Error-info MSRs *

Xen RAS status

Item	Status	Comments
MCA infrastructure	**supported**	**Move from dom0 to hypervisor**
CE and UCNA	**supported**	**Userspace tools logging and analysis**
Uncore error recovery	**supported**	**Memory scrubbing error** **L3 explicit write-back error**
Core error recovery	**supported**	**Data load error** **Instruction fetch error**
APEI BERT	**WAIT**	**Dom0 own, wait kernel ready**
APEI ERST	**supported**	**Dom0 and hypervisor co-work**
APEI EINJ	**supported**	**Dom0 own**
APEI HEST/GHES	**N/A**	**Dom0 and hypervisor co-work**
Dom0 Xen RAS interface	**WIP**	**mcelog, cpu online/offline done;** **cpu hotadd, memory hotadd ongoing**