CPUID handling for guests

Andrew Cooper

Citrix XenServer

Friday 26th August 2016

The CPUID instruction

- Introduced in 1993
  - Takes input parameters in `%eax` and `%ecx`
  - Returns values in `%eax`, `%ebx`, `%ecx` and `%edx`

The CPUID instruction

- Introduced in 1993
  - Takes input parameters in `%eax` and `%ecx`
  - Returns values in `%eax`, `%ebx`, `%ecx` and `%edx`
- Return information about the processor
  - Identifying information (`GenuineIntel`, `AuthenticAMD`, etc)
  - Feature information (available instructions, MSRs, etc)
  - Topology information (sockets, cores, threads, caches, TLBs, etc)

The CPUID instruction

- Introduced in 1993
  - Takes input parameters in `%eax` and `%ecx`
  - Returns values in `%eax`, `%ebx`, `%ecx` and `%edx`
- Return information about the processor
  - Identifying information (`GenuineIntel`, `AuthenticAMD`, etc)
  - Feature information (available instructions, MSRs, etc)
  - Topology information (sockets, cores, threads, caches, TLBs, etc)
- Unpriveleged
  - Useable by userspace
  - Doesn't trap to supervisor mode

OS expectations

- CPUID information don't change after boot

OS expectations

- CPUID information don't change after boot
- Some kernels binary patch themselves (e.g. Alternatives framework)
  - e.g. New features in fastpaths (SMAP)
  - e.g. Errata workarounds

OS expectations

- CPUID information don't change after boot
- Some kernels binary patch themselves (e.g. Alternatives framework)
  - e.g. New features in fastpaths (SMAP)
  - e.g. Errata workarounds
- Userspace function dispatching at process start
  - e.g. Most efficient `memset()`
  - e.g. Hardware crypto if available
  - e.g. Audio or video processing

OS expectations

- CPUID information don't change after boot
- Some kernels binary patch themselves (e.g. Alternatives framework)
  - e.g. New features in fastpaths (SMAP)
  - e.g. Errata workarounds
- Userspace function dispatching at process start
  - e.g. Most efficient `memset()`
  - e.g. Hardware crypto if available
  - e.g. Audio or video processing
- Migration modelled as suspend/resume
  - Not a reboot

OS expectations

- CPUID information don't change after boot
- Some kernels binary patch themselves (e.g. Alternatives framework)
    - e.g. New features in fastpaths (SMAP)
    - e.g. Errata workarounds
- Userspace function dispatching at process start
    - e.g. Most efficient `memset()`
    - e.g. Hardware crypto if available
    - e.g. Audio or video processing
- Migration modelled as suspend/resume
    - Not a reboot
- Guest must not observe a loss of dependent features

How compatible does hardware need to be?

- All features previously used need to continue functioning
  - All instructions, MSRs, etc

How compatible does hardware need to be?

- All features previously used need to continue functioning
  - All instructions, MSRs, etc
- Available features controlled by:
  - CPU family, model and stepping
  - Firmware version and settings
  - Microcode version
  - Xen's hardware support and administrator choices

How compatible does hardware need to be?

- All features previously used need to continue functioning
  - All instructions, MSRs, etc
- Available features controlled by:
  - CPU family, model and stepping
  - Firmware version and settings
  - Microcode version
  - Xen's hardware support and administrator choices
- Typically, "identical" servers are not
  - Different CPUs, especially on RMA'd hardware
  - Configuring firmware is tedious and error prone

How compatible does hardware need to be?

- All features previously used need to continue functioning
    - All instructions, MSRs, etc
- Available features controlled by:
    - CPU family, model and stepping
    - Firmware version and settings
    - Microcode version
    - Xen's hardware support and administrator choices
- Typically, "identical" servers are not
    - Different CPUs, especially on RMA'd hardware
    - Configuring firmware is tedious and error prone
- Must lie to guests (for their own good)

Controlling a guests view of information

- HVM guests
  - `CPUID` instruction traps to hypervisor
  - Xen can control all information seen

Controlling a guests view of information

- HVM guests
  - CPUID instruction traps to hypervisor
  - Xen can control all information seen
- PV guests
  - CPUID instruction doesn't trap
  - Xen has no direct control

Controlling a guests view of information

- HVM guests
  - CPUID instruction traps to hypervisor
  - Xen can control all information seen
- PV guests
  - CPUID instruction doesn't trap
  - Xen has no direct control
- PV Emulated CPUID
  - Software can opt in to Xen's control
  - Adhoc use in PV guest kernels, but not by userspace

Controlling a guests view of information

- HVM guests
  - `CPUID` instruction traps to hypervisor
  - Xen can control all information seen
- PV guests
  - `CPUID` instruction doesn't trap
  - Xen has no direct control
- PV Emulated `CPUID`
  - Software can opt in to Xen's control
  - Adhoc use in PV guest kernels, but not by userspace
- `CPUID` Faulting
  - Non-architectural, but available in Intel IvyBridge and later
  - Causes `CPUID` to fault with `#GP(0)`
  - Xen can control all information seen

Controlling a guests view of information (2)

- CPUID Masking
  - Non-architectural, “documented” only in whitepapers

Controlling a guests view of information (2)

- CPUID Masking
  - Non-architectural, “documented” only in whitepapers
- AMD (All hardware Xen can run on)
  - Override MSRs
  - Must be careful not to advertise features unsupported in silicon
  - Covers basic and extended feature leaves

Controlling a guests view of information (2)

- CPUID Masking
  - Non-architectural, “documented” only in whitepapers
- AMD (All hardware Xen can run on)
  - Override MSRs
  - Must be careful not to advertise features unsupported in silicon
  - Covers basic and extended feature leaves
- Intel (only some Nehalem/Westmere processors, SandyBridge)
  - AND mask against the real hardware values
  - Different MSRs on different hardware
  - Covers basic, extended and xsave feature leaves

Controlling a guests view of information (2)

- `CPUID` Masking
  - Non-architectural, “documented” only in whitepapers
- AMD (All hardware Xen can run on)
  - Override MSRs
  - Must be careful not to advertise features unsupported in silicon
  - Covers basic and extended feature leaves
- Intel (only some Nehalem/Westmere processors, SandyBridge)
  - AND mask against the real hardware values
  - Different MSRs on different hardware
  - Covers basic, extended and xsave feature leaves
- Magic `CPUID` bits
  - `APIC` and `OSXSAVE` bits fast forwarded from other state
  - Interaction with masking completely undocumented
  - Behaviour reverse engineered, hopefully right

Problematic areas to control

- Guests can probe for some features
  - 1GB Superpage support
  - AES instructions

Problematic areas to control

- Guests can probe for some features
  - 1GB Superpage support
  - AES instructions
- Some features don't have feature bits
  - AMD's Long Mode Segment Limit Enable

Problematic areas to control

- Guests can probe for some features
  - 1GB Superpage support
  - AES instructions
- Some features don't have feature bits
  - AMD's Long Mode Segment Limit Enable
- FPU pipeline behaviour exposed directly to guests
  - `MXCSR_MASK`
  - Intel's `FPDP` and `FPCSDS`

Problematic areas to control

- Guests can probe for some features
  - 1GB Superpage support
  - AES instructions
- Some features don't have feature bits
  - AMD's Long Mode Segment Limit Enable
- FPU pipeline behaviour exposed directly to guests
  - `MXCSR_MASK`
  - Intel's `FPDP` and `FPCSDS`
- Some feature bits affect the interpretation of other leaves
  - `CMP_LEGACY`, `HTT` and `X2APIC` affect the topology interpretation
  - Can't control topology information with masking

CPUID handling in Xen

- No real develoment since the uni-vcpu guest days
  - No concept of topology information

CPUID handling in Xen

- No real develoment since the uni-vcpu guest days
  - No concept of topology information
- No auditing of the toolstack-provided information
  - Lots of runtime adjustment to regain plausibility

CPUID handling in Xen

- No real develoment since the uni-vcpu guest days
    - No concept of topology information
- No auditing of the toolstack-provided information
    - Lots of runtime adjustment to regain plausibility
- Inconsistent whitelist/blacklist approach
    - Different approaches between different leaves
    - Different approaches between different guest types
    - Some information passed straight through from hardware

CPUID handling in Xen

- No real develoment since the uni-vcpu guest days
  - No concept of topology information
- No auditing of the toolstack-provided information
  - Lots of runtime adjustment to regain plausibility
- Inconsistent whitelist/blacklist approach
  - Different approaches between different leaves
  - Different approaches between different guest types
  - Some information passed straight through from hardware
- No knowledge of feature dependency
  - Buggy assumptions in guests result in crashes

CPUID handling in Xen

- No real develoment since the uni-vcpu guest days
  - No concept of topology information
- No auditing of the toolstack-provided information
  - Lots of runtime adjustment to regain plausibility
- Inconsistent whitelist/blacklist approach
  - Different approaches between different leaves
  - Different approaches between different guest types
  - Some information passed straight through from hardware
- No knowledge of feature dependency
  - Buggy assumptions in guests result in crashes
- No way for the toolstack to evaluate the hardware capability
  - Feature internals exposed in the ABI, with no API

CPUID handling in Xen

- No real develoment since the uni-vcpu guest days
  - No concept of topology information
- No auditing of the toolstack-provided information
  - Lots of runtime adjustment to regain plausibility
- Inconsistent whitelist/blacklist approach
  - Different approaches between different leaves
  - Different approaches between different guest types
  - Some information passed straight through from hardware
- No knowledge of feature dependency
  - Buggy assumptions in guests result in crashes
- No way for the toolstack to evaluate the hardware capability
  - Feature internals exposed in the ABI, with no API
- PV dependence on leaked `CPUID` information
  - Hardware domain for C/P states, MTRRs
  - Control domain for building guests

CPUID-related improvements in Xen 4.7

- Introduce “a featureset” in the API/ABI
  - Keys as specified in architecture manuals
  - Can be treated as opaque by higher level software

CPUID-related improvements in Xen 4.7

- Introduce “a featureset” in the API/ABI
  - Keys as specified in architecture manuals
  - Can be treated as opaque by higher level software
- At boot, Xen calculates featuresets for itself and the toolstack:

raw	Features supported by hardware
host	Features used by Xen (after errata, command line, etc)
pv	Maximum featureset available for PV guests
hvm	Maximum featureset available for HVM guests

CPUID-related improvements in Xen 4.7

- Introduce "a featureset" in the API/ABI
  - Keys as specified in architecture manuals
  - Can be treated as opaque by higher level software
- At boot, Xen calculates featuresets for itself and the toolstack:

raw	Features supported by hardware
host	Features used by Xen (after errata, command line, etc)
pv	Maximum featureset available for PV guests
hvm	Maximum featureset available for HVM guests

- Shared Xen/libxc algorithm for feature dependencies
  - Provides consistent logic between Xen and libxc
  - Build-time calculations to avoid complicated runtime logic

Future work

- Extend the featureset concept to a full `CPUID` policy
  - Needs guest topology information
  - Will allow control of `MAXPHYSADDR`

Future work

- Extend the featureset concept to a full CPUID policy
  - Needs guest topology information
  - Will allow control of MAXPHYSADDR
- Improved hypercalls for policies
  - XEN_DOMCTL_get_cpuid_policy
  - Xen to audit policy plausibility at hypercall time, not runtime

Future work

- Extend the featureset concept to a full CPUID policy
  - Needs guest topology information
  - Will allow control of MAXPHYSADDR
- Improved hypercalls for policies
  - XEN_DOMCTL_get_cpuid_policy
  - Xen to audit policy plausibility at hypercall time, not runtime
- More migration stream work
  - CPUID policy should be in the migration stream
  - Avoid mistakes when regenerating
  - Can finaly audit RAM/CPU state against reality
  - Allows for a pre-migrate check at the destination

Future work

- Extend the featureset concept to a full `CPUID` policy
  - Needs guest topology information
  - Will allow control of `MAXPHYSADDR`
- Improved hypercalls for policies
  - `XEN_DOMCTL_get_cpuid_policy`
  - Xen to audit policy plausibility at hypercall time, not runtime
- More migration stream work
  - `CPUID` policy should be in the migration stream
  - Avoid mistakes when regenerating
  - Can finaly audit RAM/CPU state against reality
  - Allows for a pre-migrate check at the destination
- MSRs

CPUID handling for guests

Any Questions?