Live control of (most) CPU features via hybrid vCPU model

KVM Forum 2023

Like Xu

Tencent Cloud

Disclaimer on the status

- What started out as an original, crude and naive idea
    - is just one of my side projects.
- Still at the path finding stage,
    - but the POC functionality of vPMU features does have a heartbeat.
- Fully agree that supporting all CPU features is a daydream as you expect.
- Current changes involving almost the entire virt-CPU features stack;
    - allow to offline vCPU0, enable per-CPU drivers and a KVM specific eBPF scheduler.
- It has apparently not yet been deployed in a production environment,
    - however most administrators may not refuse to have such a knob.
- Stay tuned.

Outline

Hybrid CPU Stage 1

- Same architecture but different micro-architectures
  - Performance-Power Tradeoff
    - Different instruction-per-cycle (IPC) capabilities
  - SMPs, no ISA diff (cache hierarchy diff)
  - Little CPUID/MSR Incoherence
  - Different undefined behavior
    - Bitwise operations (AND, OR, XOR, NOT, rotate)
    - Shifts and multiplies, single-bit shift
- Vendor-driven
  - ARM: big.LITTLE (2011) Cortex-X Custom (CXC) Processors
  - Intel: 12th Gen Core (2021)
    - Golden Cove (Core, P) + Gracemont (Atom, E)
    - ALDERLAKE, RAPTORLAKE, METEORLAKE, LUNARLAKE, ARROWLAKE
  - AMD:
    - Zen 4c & standard Zen 4 core
    - Family 19h Model 70h Core (202X)
      - Performance/Efficiency Core
      - Different bank layouts between cores

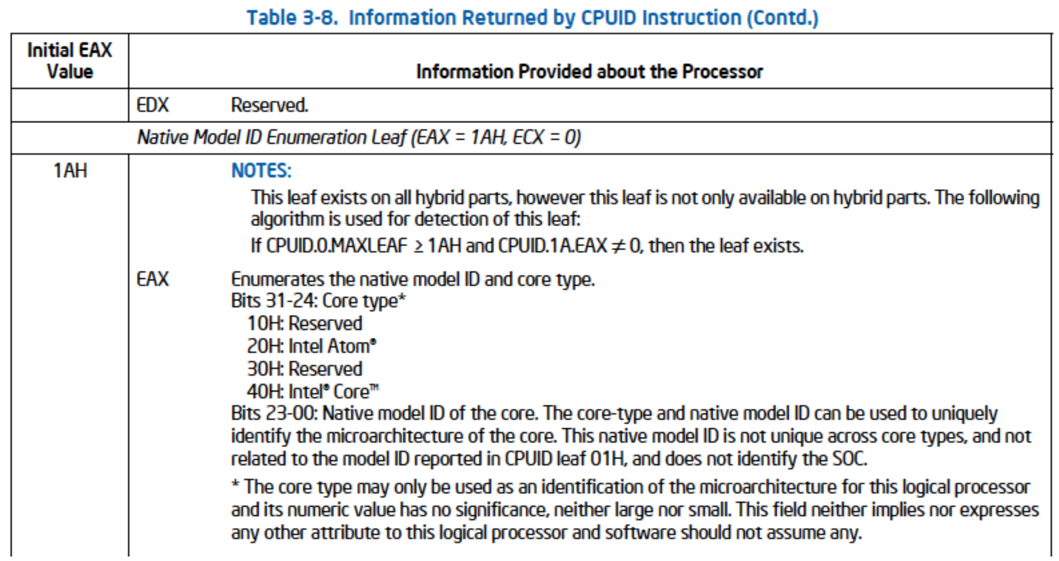
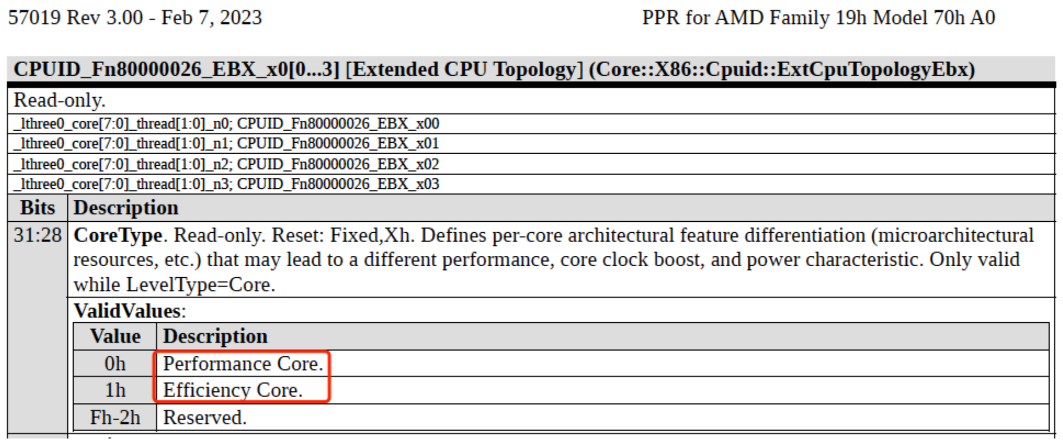


Table 3-8. Information Returned by CPUID Instruction (Contd.)

Initial EAX Value	Information Provided about the Processor	
	EDX	Reserved.
	Native Model ID Enumeration Leaf (EAX = 1AH, ECX = 0)	
1AH		NOTES: This leaf exists on all hybrid parts, however this leaf is not only available on hybrid parts. The following algorithm is used for detection of this leaf: If CPUID.0.MAXLEAF ≥ 1AH and CPUID.1A.EAX ≠ 0, then the leaf exists.
	EAX	Enumerates the native model ID and core type. Bits 31-24: Core type* 10H: Reserved 20H: Intel Atom® 30H: Reserved 40H: Intel® Core™ Bits 23-00: Native model ID of the core. The core-type and native model ID can be used to uniquely identify the microarchitecture of the core. This native model ID is not unique across core types, and not related to the model ID reported in CPUID leaf 01H, and does not identify the SOC. * The core type may only be used as an identification of the microarchitecture for this logical processor and its numeric value has no significance, neither large nor small. This field neither implies nor expresses any other attribute to this logical processor and software should not assume any.



57019 Rev 3.00 - Feb 7, 2023 PPR for AMD Family 19h Model 70h A0

CPUID_Fn80000026_EBX_x0[0...3] [Extended CPU Topology] (Core::X86::Cpuid::ExtCpuTopologyEbx)

Read-only.
_lthree0_core[7:0]_thread[1:0]_n0; CPUID_Fn80000026_EBX_x00
_lthree0_core[7:0]_thread[1:0]_n1; CPUID_Fn80000026_EBX_x01
_lthree0_core[7:0]_thread[1:0]_n2; CPUID_Fn80000026_EBX_x02
_lthree0_core[7:0]_thread[1:0]_n3; CPUID_Fn80000026_EBX_x03

Bits	Description
31:28	**CoreType**. Read-only. Reset: Fixed,Xh. Defines per-core architectural feature differentiation (microarchitectural resources, etc.) that may lead to a different performance, core clock boost, and power characteristic. Only valid while LevelType=Core. **ValidValues**:

Value	Description
0h	Performance Core.
1h	Efficiency Core.
Fh-2h	Reserved.

Stage 2: sprinkle a little ISA heterogeneity

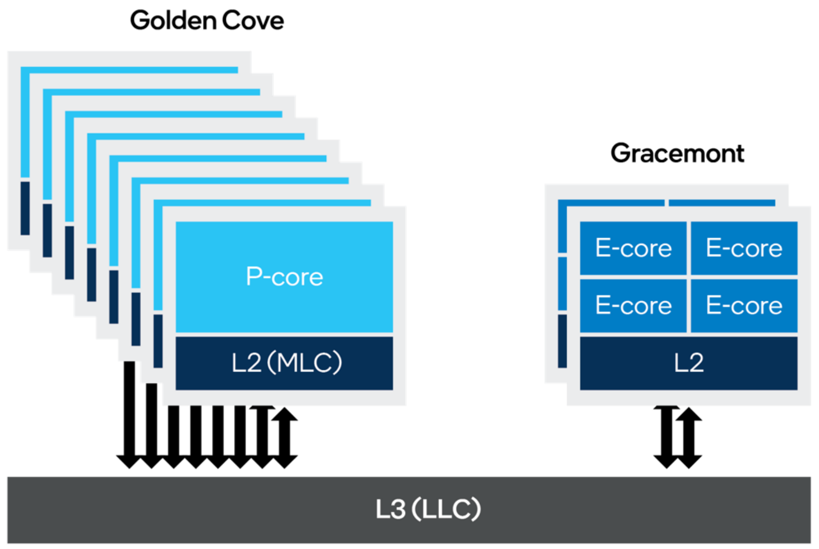
- Asymmetric cores in one CPU model
- Drop ISA's fully aligned commitment
  - to further implement more electrical optimization
  - reduce manufacturing costs
  - IP reuse/reconfigure
- AVX512 on Intel ADL P-cores
  - AVX-512 was not fuse-disabled on early ADL desktop products
  - some motherboard allow people to re-enable AVX-512 support
  - may be more instruction set differences between the P/E cores
- AMD patent US10698472B2
  - "A heterogeneous processor system includes a first processor ... and a second processor ..., wherein the first subset and the second subset of the set of ISA features are different from each other."

Stage 3: multi ISA hybrid CPU model

- Chiplet design & ecosystem
    - x86, ARM, and RISC-V all co-exist on the same physical silicon
    - each IP devoted to a particular type of best-fit workload
- Intel IDM 2.0
    - "pledged to license x86 designs to customers"
    - "we've made it such that either Arm or RISC-V cores would work in and also we've made it such that those connect seamlessly overcome design challenges for chips in which x86, Arm, and RISC-V co-exist."
- Wait until the HW is out before making SW changes?
    - Shift left software emulation/validation/design

Demand for hybrid vCPU stage 1/2/3

1. Hybrid physical CPUs are already available on client platforms
   - CPUs with NUMA topology actually is part of hybrid properties
   - Get ready/lessons for virtualized heterogeneous computing
   - More hybrid stage 1 architectures are emerging, such as CLX.mem
2. For system research and pure emulation
   - explore cache cfg and consistency proto for hybrid pCPU before release, x86-Simlify Arch
   - creating non-exist virtual CPU types is more attractive and flexibility
3. For users of hardware-assisted virtualization
   - Achieve "Performance-Power Tradeoff" goals for workloads on vCPUs
     - treating vCPU as an ordinary host thread is not effective enough
   - Prerequisite: a true (or at least partial) topological view
     - of any details related to performance
   - Emulate hybrid cpu/cache/isa topology
     - AMD Genoa-X offers up to 12 CCDs and 1.1GB L3 cache per socket
     - physical topo has a significant impact on performance
       - HT/Cluster/Energy/Fault aware scheduling



Link multiple VM pools

One vCPU model for all host, the most shared common basic vCPU model

- e.g. x86 model, migrating Intel VMs to AMD platforms
- then add or remove any (idealistically) specific suspicious or interesting features
- CSP define CPU features set per-vCPU on demands
    - Instead of CPU vendors, like 'AWS' x86 CPU
    - Change features from the boot time to the run-time.
- Choke the endless vCPU model version increase
    - Version numbers mean more restrictions on model compatibility checks
    - Icelake-Server version 1/2/3/4/5/6 or Cascadelake-Server version 1/2/3/4/5

Enhance the host maintainability

1. Indiscriminately isolating a partially faulty CPU is a waste
   - part of physical cores are partially characterized as broken
   - but not completely out of service
     - “Cores that don’t count”
   - no longer ignore the rising curve of core-level failure rate
     - more and more CPU Errata (w/ inactive advanced features)
   - continue to run a restricted set of a well-proven features for well-tested workload
2. Timely reduce hypervisor attack surface
   - by limiting vCPU features being exposed to the minimum required
   - avoid known issue
     - side channel or architecture data leakage, performance (including power budget) attack

Comparison with current solution

Simply deploy applications with different cap requirements to another vCPU pool through the control plane and take a distributed approach.

1. Programming model will be more simplified
   - Keep the shape of the monolithic application (mm layout/addr space)
   - No extra RPC intertwining
2. Performance is significantly improved
   - The overhead on cross-guest synchronization, data movement is removed
3. Less vCPU/mem will be required
   - Potential CPU utilization improvement
   - fewer infrastructure costs (mm allocation per guest)

Stage 1 case: Define an Intel ADL vCPU

VMM Scope

- Logical cores number
    - cpus/max_cpus/type, sockets/dies/modules/clusters/cores/threads
- Processor (base/maximum) Frequency
- Cache topology
    - L1/2/3/4 cache size
    - Ways of associativity (associativity field)
    - Prefetch size (64-Byte)
- TLB
    - 4K/2M/4M/1G page size entries
    - Partitioning or shared between the logical processors
    - Ways of associativity && Number of Sets
    - Translation cache type/level

KVM Scope

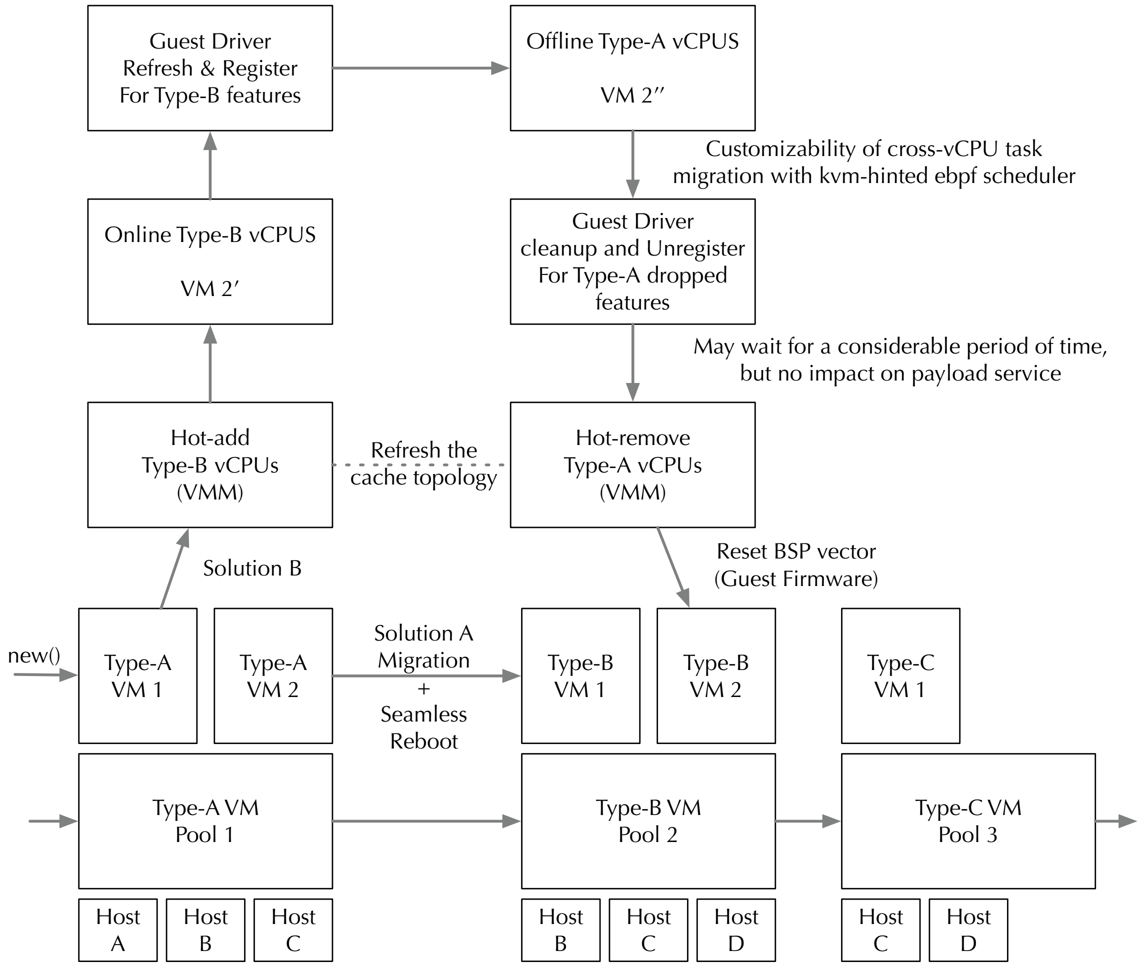
- PMU features (my starting point)
    - Num of GP counters and topdown perf_metrics
    - Cache event mapping table event_select/umask
    - PEBS latency and event_constraints
    - Arch LBR
        - LBR IP values contain LIP or Effective IP
    - PT
        - IP payloads have LIP values + CS
        - pebs_output_pt_available
    - IRQ
        - late_ack && mid_ack
- Model Specific MSR values
    - IA32_HWP_CAPABILITIES/REQUEST (unsupported)
    - ~~MSR_IA32_ARCH_CAPABILITIES~~
    - ~~MSR_IA32_SPEC_CTRL~~

Definition space of hybrid vCPU

- Configure guest hybrid CPU/Cache topology
  - TopologyState & MachineState
    - cpus/max_cpus/core_type/sockets/dies/clusters/cores/thread
  - Keep the topo consistency consistent with the current host
    - Whether to support HT, whether to share cache
  - i386: Introduce hybrid CPU topology
  - ARM virt: Support CPU cluster topology
  - ARM Virtual cache topology
- Add KVM emulation and guard for per-vCPU features
  - Add KVM emulation based on current pCPU capability
  - Active request for pCPU rescheduling
  - Deny user space request to change vCPU features after VCPU_RUN
- Reset guest BSP0 selection and delay guest initialization for AP bringup

Hybrid vCPU in multiple VM pools

- Only one basic X86CPUDefinition is needed
  - Still follow the TopologyState & MachineState
  - query-hotpluggable-cpus
  - device_add/del CPUs {w/ or w/o} features
  - ACPI event and CONFIG_ACPI_HOTPLUG_CPU
- Reset bootstrap core (BSP) for next cold reset
  - physically dependent on power sequence
  - virtually on BSP flag of APIC_BASE MSR
    - BSP APIC ID can be a non-zero value, e.g kexec()
  - Atomically modify SRAT, MADT table in flight
- Hot remove vCPU0 on the VMM side
  - eliminate guest ACPI driver warning functionally
    - ACPI: _SB_.CPUS.C000: No _EJ0 support for device
  - then hot-add a new CPU0 device w/ new features
    - via another not-yet-running vCPU thread w/ BSP ctx
- VM always has at least one vCPU always online

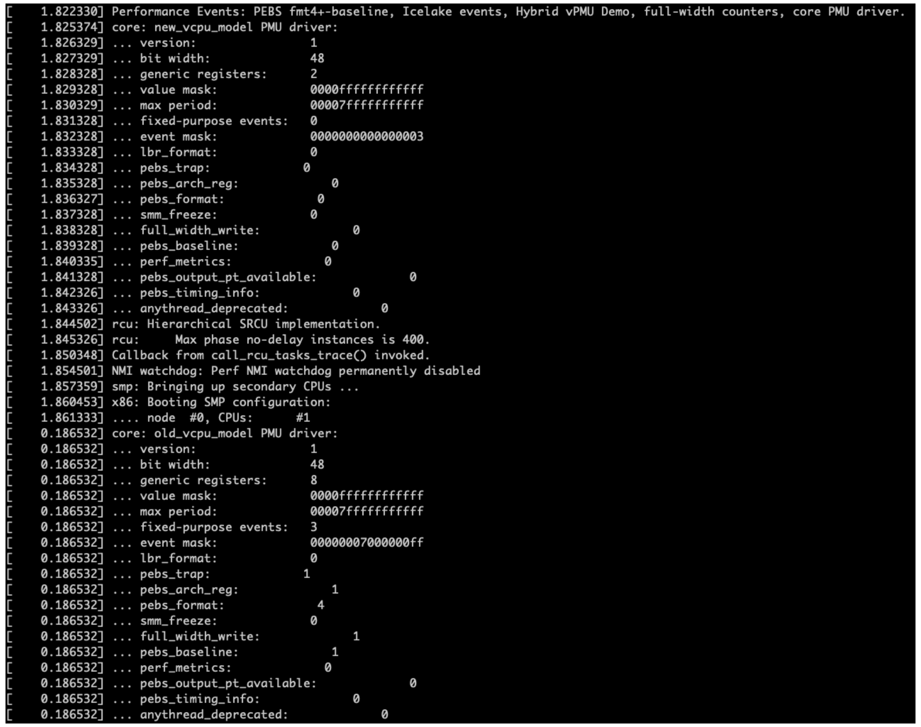


Guest Driver Refresh & Register For Type-B features
Offline Type-A vCPUS VM 2''
Customizability of cross-vCPU task migration with kvm-hinted ebpf scheduler
Online Type-B vCPUS VM 2'
Guest Driver cleanup and Unregister For Type-A dropped features
May wait for a considerable period of time, but no impact on payload service
Hot-add Type-B vCPUs (VMM)
Refresh the cache topology
Hot-remove Type-A vCPUs (VMM)
Solution B
Reset BSP vector (Guest Firmware)
new()
Type-A VM 1
Type-A VM 2
Solution A Migration + Seamless Reboot
Type-B VM 1
Type-B VM 2
Type-C VM 1
Type-A VM Pool 1
Type-B VM Pool 2
Type-C VM Pool 3
Host A
Host B
Host C
Host B
Host C
Host D
Host C
Host D

Live control of hybrid vPMU features

- Prerequisites: ADL Hyprid PMU native support
  - relevance to pCPU microarchitecture
  - hybrid_pmu = &x86_pmu.hybrid_pmu[i];
  - callback registration for CPU hot-plugging features
    - cpuhp_setup_state/cpuhp_remove_state
    - x86_pmu_prepare/online/dying_cpu
  - also with rollback handling for any failure
- Server Platform Porting
  - Refresh model specific data based on server model
  - Register a NULL PMU driver for enable_pmu=false
    - Add X86_FEATURE_PERFMON_NULL
  - Expose /sys/devices/cpu_{core/atom}
  - Porting perf tools to support hybrid PMU on the server

- Finally, host can control the guest to achieve from ”pmu disable” to “base counters available” to “LBR feature available”, and also support feature removal (as long as it is not being used).



```
[    1.822330] Performance Events: PEBS fmt4+-baseline, Icelake events, Hybrid vPMU Demo, full-width counters, core PMU driver.
[    1.825374] core: new_vcpu_model PMU driver:
[    1.826329] ... version:                1
[    1.827329] ... bit width:              48
[    1.828328] ... generic registers:      2
[    1.829328] ... value mask:             0000ffffffffffff
[    1.830329] ... max period:             00007fffffffffff
[    1.831328] ... fixed-purpose events:   0
[    1.832328] ... event mask:             0000000000000003
[    1.833328] ... lbr_format:             0
[    1.834328] ... pebs_trap:             0
[    1.835328] ... pebs_arch_reg:              0
[    1.836327] ... pebs_format:              0
[    1.837328] ... smm_freeze:             0
[    1.838328] ... full_width_write:             0
[    1.839328] ... pebs_baseline:              0
[    1.840335] ... perf_metrics:             0
[    1.841328] ... pebs_output_pt_available:             0
[    1.842326] ... pebs_timing_info:             0
[    1.843326] ... anythread_deprecated:             0
[    1.844502] rcu: Hierarchical SRCU implementation.
[    1.845326] rcu:     Max phase no-delay instances is 400.
[    1.850348] Callback from call_rcu_tasks_trace() invoked.
[    1.854501] NMI watchdog: Perf NMI watchdog permanently disabled
[    1.857359] smp: Bringing up secondary CPUs ...
[    1.860453] x86: Booting SMP configuration:
[    1.861333] .... node  #0, CPUs:      #1
[    0.186532] core: old_vcpu_model PMU driver:
[    0.186532] ... version:                1
[    0.186532] ... bit width:              48
[    0.186532] ... generic registers:      8
[    0.186532] ... value mask:             0000ffffffffffff
[    0.186532] ... max period:             00007fffffffffff
[    0.186532] ... fixed-purpose events:   3
[    0.186532] ... event mask:             00000007000000ff
[    0.186532] ... lbr_format:             0
[    0.186532] ... pebs_trap:             1
[    0.186532] ... pebs_arch_reg:              1
[    0.186532] ... pebs_format:              4
[    0.186532] ... smm_freeze:             0
[    0.186532] ... full_width_write:             1
[    0.186532] ... pebs_baseline:              1
[    0.186532] ... perf_metrics:             0
[    0.186532] ... pebs_output_pt_available:             0
[    0.186532] ... pebs_timing_info:             0
[    0.186532] ... anythread_deprecated:             0
```

Flexible scheduler for hybrid nature

- Differential scheduling (required on both hybrid host or guest)
  - Idea: update scheduling context based on execution context
- Basic functions
  - feature(s) matching, p/v-CPU binding in a generic way
  - core load balancing as usually
- HW-assisted solution: Intel Thread Director
  - partition its instruction set into an arbitrary number of classes
  - dispatch workload based on its ISA classes
  - [PATCH v3 00/24] sched: Introduce classes of tasks for load balance
- Software solution: BPF extensible scheduler
  - [PATCHSET v3] sched: Implement BPF extensible scheduler class
  - easy to learn and implementing a taskset-alike scheduler
  - More BPF helpers added to leverage KVM APIs to check saved CPUID[] and MSRs

CPU feature checklist in live migration case

- To narrow the scope
  - only focus on the features that will be added or removed from the target CPU model.
- CLX -> ICX
  - Remove
    - cdp_l3、hle、rtm、mpx
  - Add:
    - avx512ifma, sha_ni, ~~split_lock_detect,~~ wbnoinvd, ~~hwp, hwp_act_window, hwp_epp, hwp_pkg_req~~, avx512vbmi, umipavx512_vbmi2, gfni, vaes, vpclmulqdq, avx512_bitalg, tme、avx512_vpopcntdq, ~~la57~~, rdpid, fsrm, ~~pconfig~~
- ICX -> SPR
  - Remove
    - dca
  - Add
    - tsc_known_freq, cat_l2, cdp_l3, cdp_l2, ~~vnmi~~, arch_lbr,avx_vnni, avx512_bf16, avx512_vp2intersect, avx512_fp16, amx_bf16, amx_tile,amx_int8, hfi, ~~waitpkg~~, bus_lock_detect, cldemote, movdiri, movdir64b ,~~enqcmd~~, serialize, tsxldtrk, ibt
- Rough conclusion: we can do it for (most) CPU features
  - based on spec/driver review
  - very unprofessional quick check

Generic changes for most CPU features

- 2023 Reality:
    - already support hybrid PMU, MCE, power mgmt
- Daydream recipes (enlighten guest drivers)
    - allow to offline BSP via ACPI event
    - cleaning up the x86_64 boot process
        - refactor the smp initialization of infrastructure
        - move what is not required during early boot out into a later phase of the boot process
        - wrapper boot_cpu_data, boot_cpu_has(), static_cpu_has()
    - support kernel custom command line parsing
        - kernel can still filter out features
    - support per-vCPU initialization and registration
        - refer to the KVM-hinted hybrid bit
    - refresh callbacks on cpu-hp path per-feature
- Workload adaptive migration
    - supports multiple ISA in one binary
    - /proc/cpuinfo or arch_ctrl() still reports all flags
- More user-friendly knobs to enable/disable guest features
    - many dependencies between CPU features
    - determine the minimum set of CPU features

Feature limitations and known drawbacks

- Characteristics of unsupported features
  - global features that rely on early initialization on BSP
    - global paging mode, protected mode transactions, (x2)apic mode
    - firmware-assisted features: SGX、TDX、SEV-*
  - synchronization operations related to hardware details, memory barriers
- Applications migrating across hybrid cores should have the same view of the memory view
- Break applications based on a stable constant CPU model: digital rights management
- Online/offline may hang for a long time and fall into failure
  - revert due to offline failure may bring unnecessary troubles and hard to debug
  - especially when guests are overloaded (severely CPU over-commits).
- The num of pluggable vCPU is limited by the pre-defined max topology
  - The feature type of vCPU that can be added also depends on the guest's a priori knowledge and driver preparation
- Tracking the different features of each vCPU adds an extra burden to the control panel

Summary and Reflection

- This is a POC-quality practice that allow VMM admin to live control vPMU features by hot plugging or removing a class of vCPUs based on a hybrid vCPU model to or from an *enlighten* guest.
- Rolling out from PMU features to most CPU features requires more case by case changes, especially adding guest driver support on per-cpu refreshment.
- Time consuming and impractical to have every feature be converted.
  - make sense for CPU features used only at the local core scope, such as new instructions after LM.
- Upstream timing for hybrid vCPU stage 2/3 is not yet mature.
- Applications need to gradually migrate to a new programming model
  - based on CPUs that have predictable differences in ISA.
- More roadblocks ?
- Or "live control" is a bit like a poisoned apple. Do we really need it ?