KVM Forum 2024

The Road to Optimal CPU Virtualization on Hybrid Platform

Zhao Liu, Zhenyu Wang

intel®

Legal Disclaimers:

- Intel provides these materials as-is, with no express or implied warranties.
- All products, dates, and figures specified are preliminary, based on current expectations, and are subject to change without notice.
- Intel processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No product or component can be absolutely secure. Check with your system manufacturer or retailer or learn more at http://intel.com.
- Some results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.
- Intel and the Intel logo are trademarks of Intel Corporation in the United States and other countries.
- *Other names and brands may be claimed as the property of others.
  © Intel Corporation 2024

Agenda

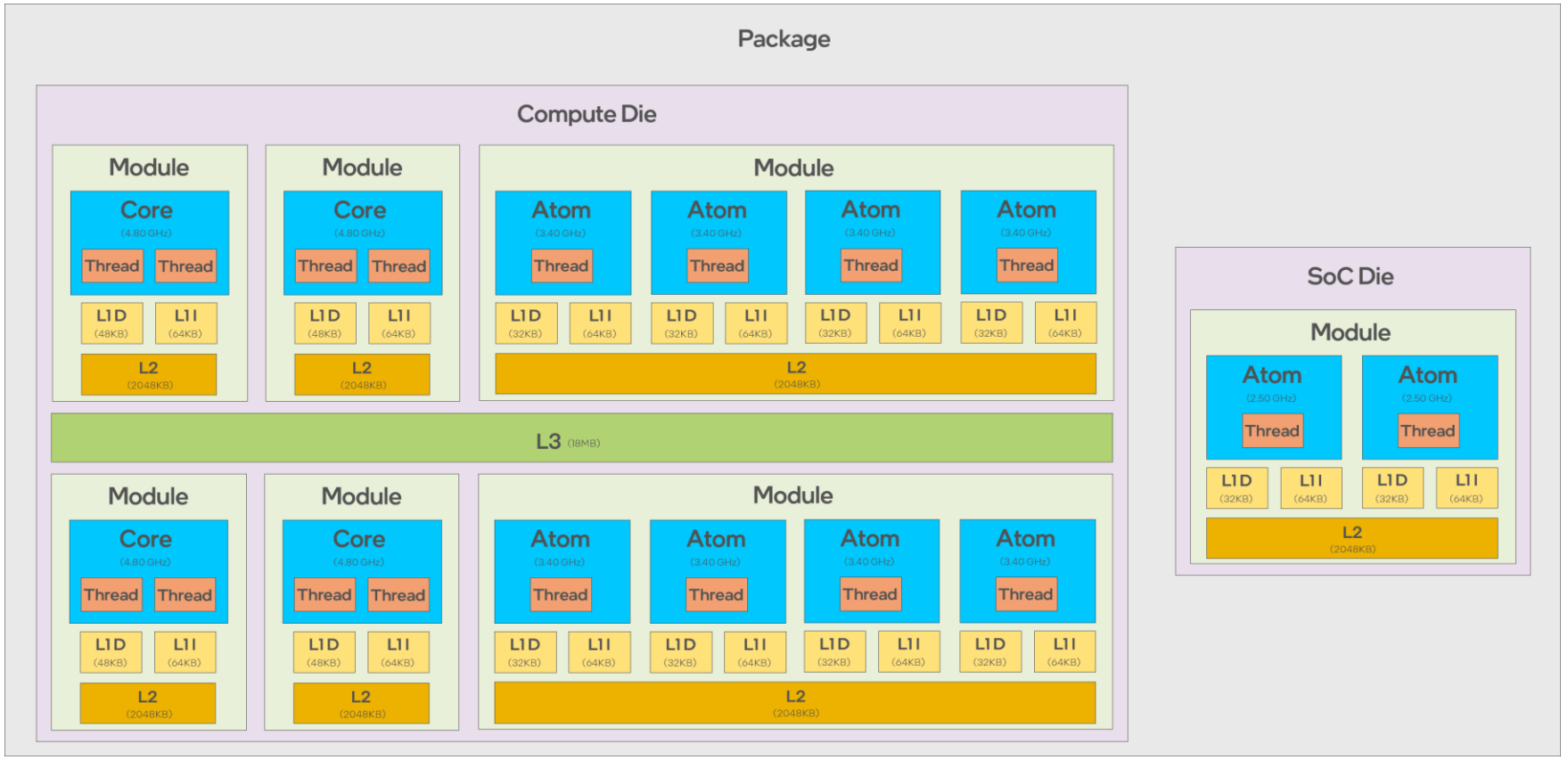
- Overview of Intel hybrid topology
- Hybrid CPU topology in QEMU
  - QOM CPU topology
  - Hybrid CPU topology
- Hybrid CPU scheduling improvement
  - Intel Thread Director (ITD) virtualization for Windows Guest
  - Intel Turbo Boost Max Technology (ITMT) scheduling for Linux Guest

Overview of Intel hybrid topology

CPU & cache topology

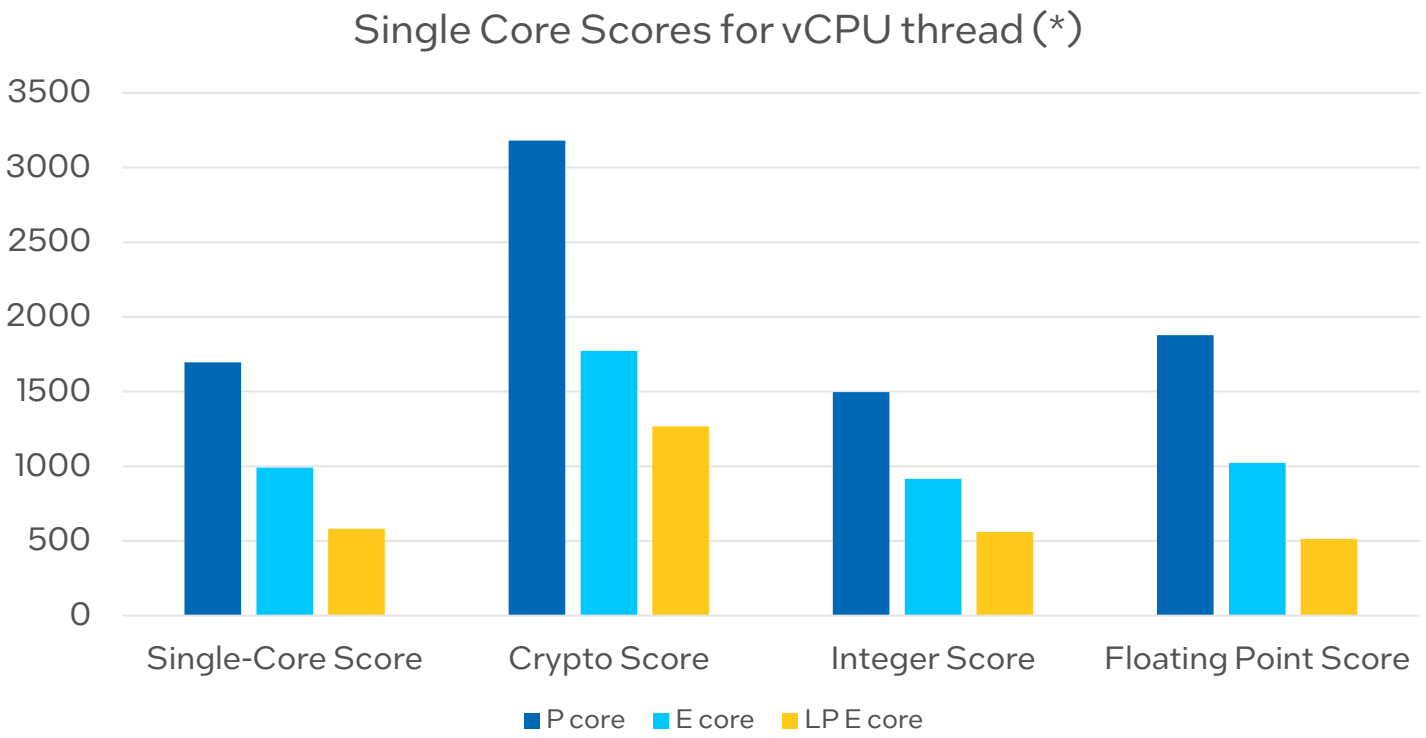
Topology Example of Meteor Lake (MTL)

Heterogeneous CPU topology and cache topology



Why does Guest need to care about hybrid CPU?

- Performance
  - vCPU performance varies greatly across cores.



- P core has the best performance on all single thread cases.
- Low power E core has the worst performance for all cases.

* Geekbench v5.3.1 on Intel(R) Core(TM) Ultra 5 1003H, 14 Cores (4+8+2), 18 Threads

Why does Guest need to care about hybrid CPU?

- Performance
  - vCPU performance varies greatly across cores.
  - Guest OS has special optimization for hybrid CPUs: Windows 11 - Intel Thread Director (ITD).
- Feature
  - Hybrid PMU: Different cores have different PMU events.
- Main use case
  - gaming

About vCPU pinning

Pinning vCPUs –precondition of hybrid CPUs in Guest.

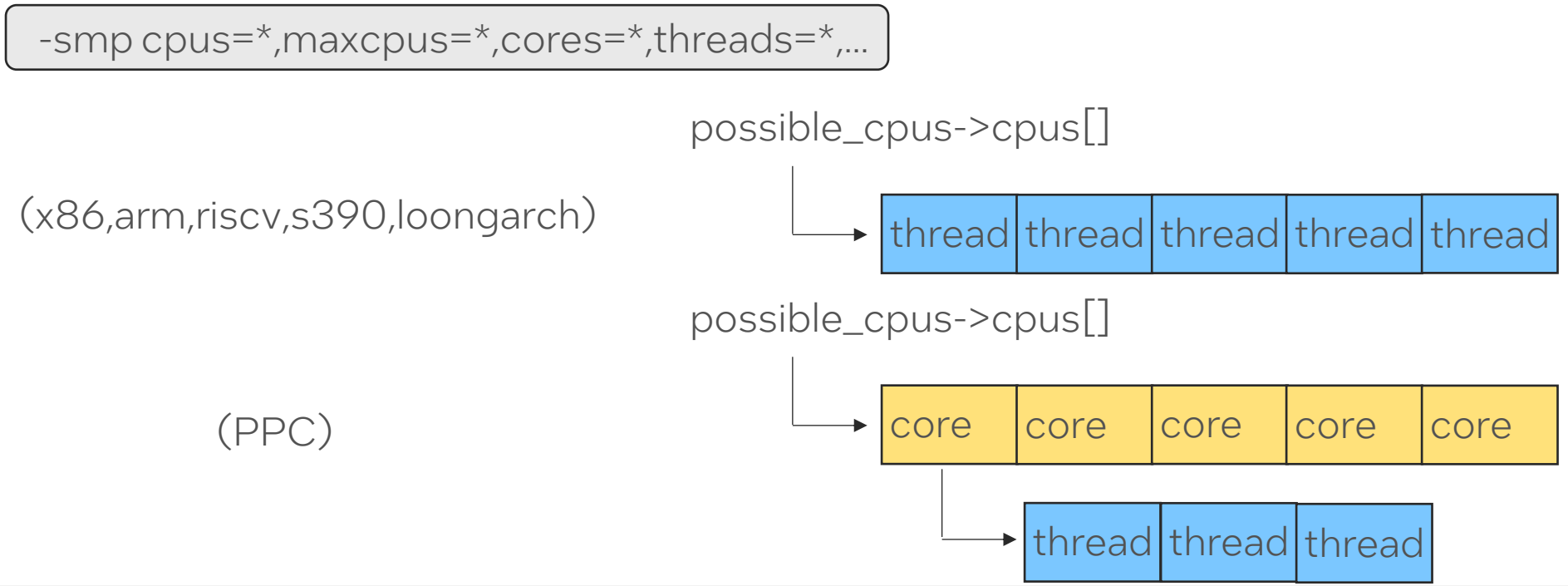
- Without pinning:
  - Synchronization issue of Hybrid CPU information:
    - when vCPUs are migrated to different cores, the hybrid information in Guest and Host will not match.
  - Double scheduling issue:
    - concurrent scheduling by the Guest and Host schedulers results in more task migrations and context switches.

Hybrid CPU topology in QEMU

From QOM CPU topology

SMP CPU Topology in QEMU

- From -smp, QEMU only creates CPU/core objects.
  - The other hierarchical information specified in -smp is only recorded.
  - No complete topology tree.



First step towards hybrid topology: QOM CPU topology

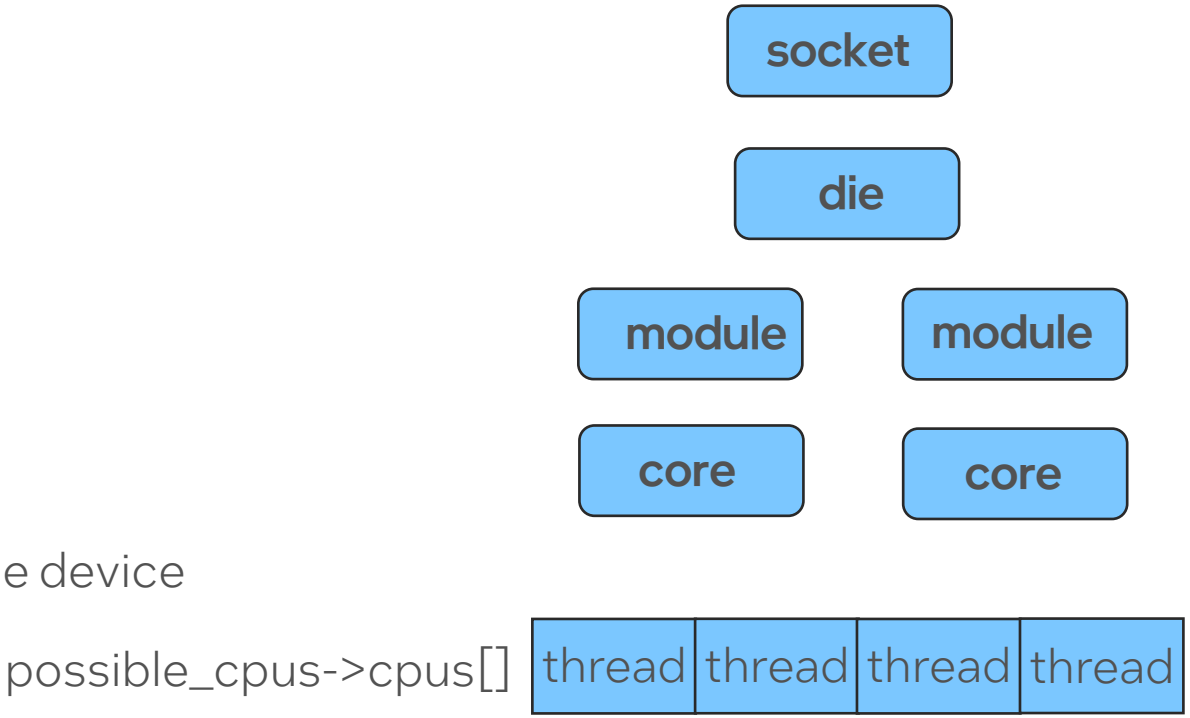
Goal: Abstract every CPU topology level as “device”

- Current CPU related devices:
  - CPU device.
  - Core device (for PPC).
  - Cluster device (for TCG).

First step towards hybrid topology: QOM CPU topology

Goal: Abstract every CPU topology level as “device”

- Current CPU related devices:
  - CPU device.
  - Core device (for PPC).
  - Cluster device (for TCG).
- We need to move forward...
  - Abstract more CPU topology devices:
    - For x86: socket device, die device, module device

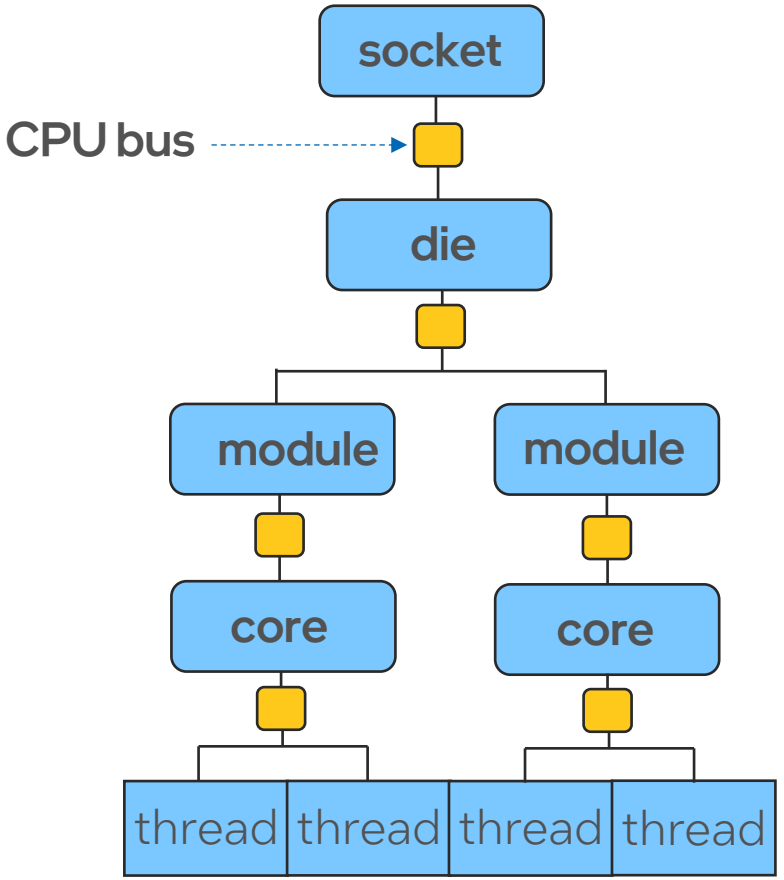


First step towards hybrid topology: QOM CPU topology

Goal: Abstract every CPU topology level as “device”

- Current CPU related devices:
  - CPU device.
  - Core device (for PPC).
  - Cluster device (for TCG).

- We need to move forward...
  - Abstract more CPU topology devices:
    - For x86: socket device, die device, module device
  - Build a complete topology tree:
    - How to connect topological hierarchies?
      - The abstract CPU bus is a proper choice.



Hybrid CPU topology

Proposal: build topology tree via -device

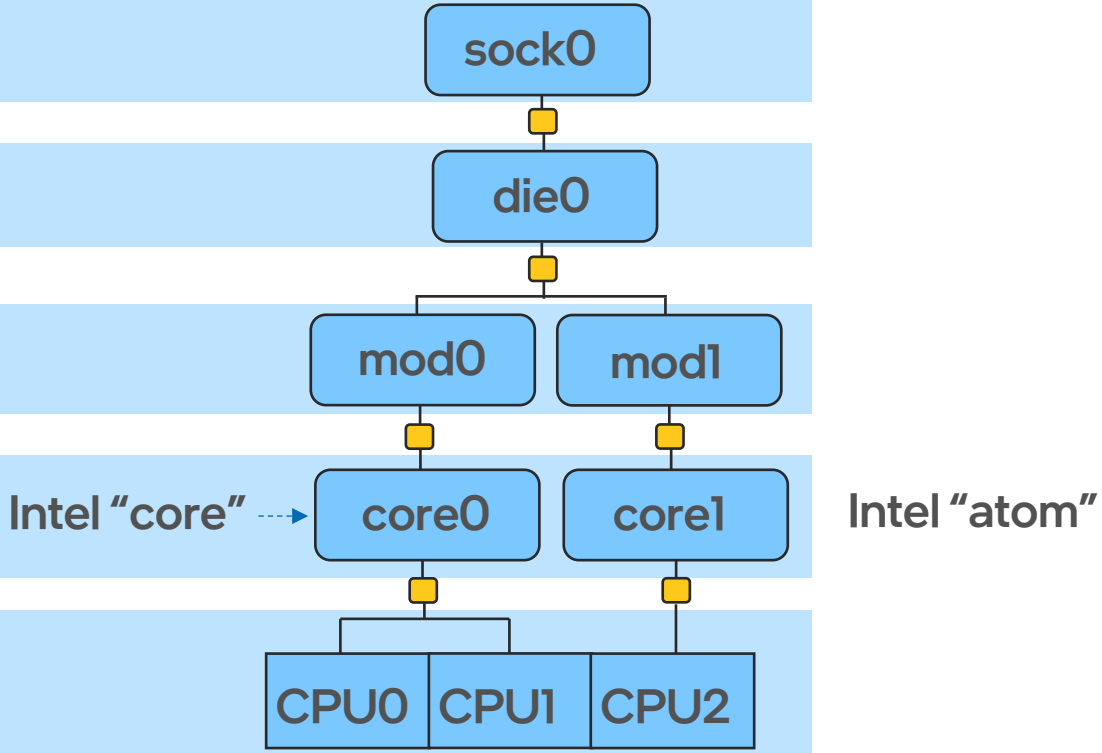
```
-device cpu-socket,id=sock0

-device cpu-die,id=die0,bus=sock0

-device cpu-module,id=mod0,bus=die0
-device cpu-module,id=mod1,bus=die0

-device x86-cpu-intel-core,id=core0,bus=mod0
-device x86-cpu-intel-atom,id=core1,bus=mod1

-device host-x86_64-cpu,id=cpu0, socket-id=0,die-id=0,\
                                 module-id=0,core-id=0,thread-id=0\
-device host-x86_64-cpu,id=cpu0, socket-id=0,die-id=0,\
                                 module-id=0,core-id=0,thread-id=1\
-device host-x86_64-cpu,id=cpu0, socket-id=0,die-id=0,\
                                 module-id=1,core-id=1,thread-id=0\
```



QOM CPU topology & hybrid CPU topology

Additional benefits

- Will obtain the QOM topology hierarchy tree.
  - "/machine/peripheral/cpu-slot/sock0/die0/mod0/core0/thread[0]“
  - People wanted for 9 years ago. [1]

- Maybe greater granularity of hotplug.
  - Socket (package) granularity.

- For x86, better emulation for the feature with different topology scopes:
  - Running Average Power Limit (RAPL): package/die scope MSRs.
  - Hardware Feedback Interface (HFI)/Intel Thread Director (ITD): package scope MSRs (registers).

Future work

QOM cache & Hybrid cache topology

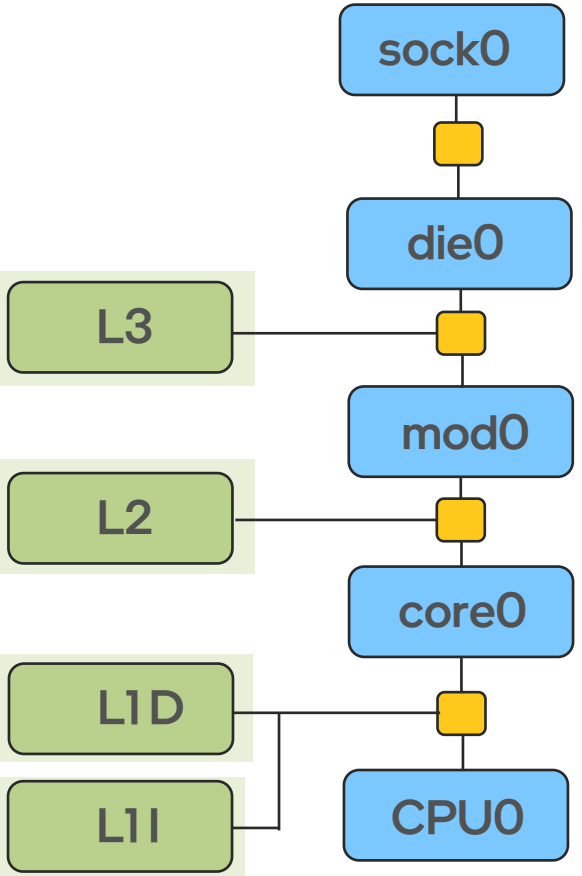
- Abstract cache to device, too.

-device cpu-cache,type=unified,level=3,bus=die0

-device cpu-cache,type=unified,level=2,bus=mod0

-device cpu-cache,type=data,level=1,bus=core0

-device cpu-cache,type=inst,level=1,bus=core0



QOM & Hybrid CPU topology

Patch link:

- QOM (CPU) topology:
  - [RFC v2 00/15] qom-topo: Abstract CPU Topology Level to Topology Device:
  - https://lore.kernel.org/qemu-devel/20240919015533.766754-1-zhao1.liu@intel.com/

- Hybrid (CPU) topology:
  - [RFC v2 00/12] Introduce Hybrid CPU Topology via Custom Topology Tree
  - https://lore.kernel.org/qemu-devel/20240919061128.769139-1-zhao1.liu@intel.com/

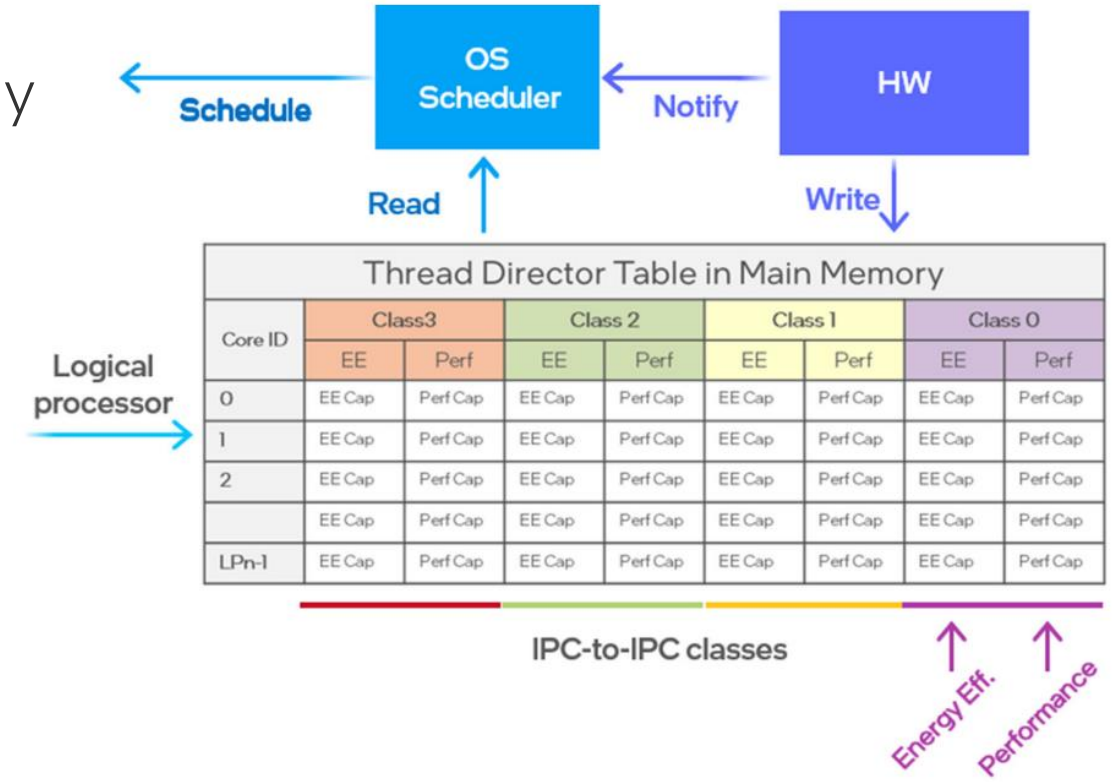
Hybrid CPU scheduling improvement

Intel Thread Director (ITD) virtualization for Windows Guest

Intel Thread Director (ITD) Introduction

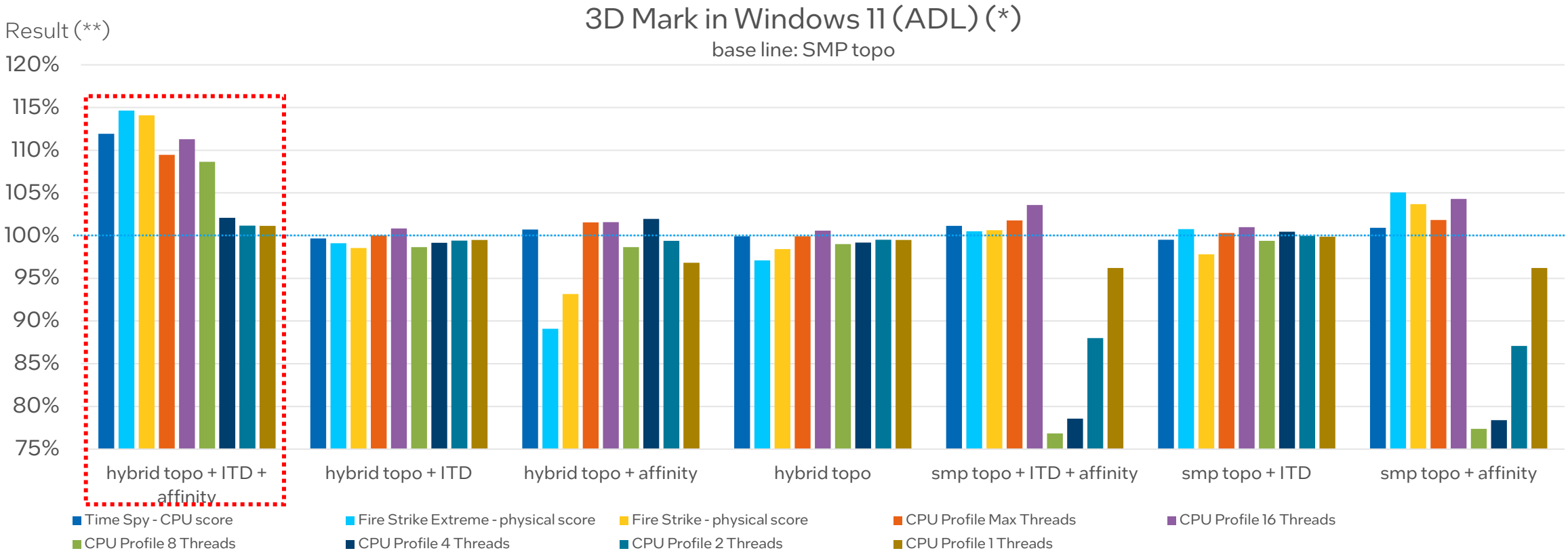
- Hardware provides a table (**Hardware Feedback Interface table**) to expose the performance and energy efficiency capabilities of CPUs to OS.

- HFI (Hardware Feedback Interface)/ITD (Intel Thread Director):
  - (Initial version) HFI supports only 1 class.
  - ITD is the extension of HFI, and supports more classes and is able to classify tasks



ITD performance in Windows Guest

Up to 14+% improvement in Windows VM.



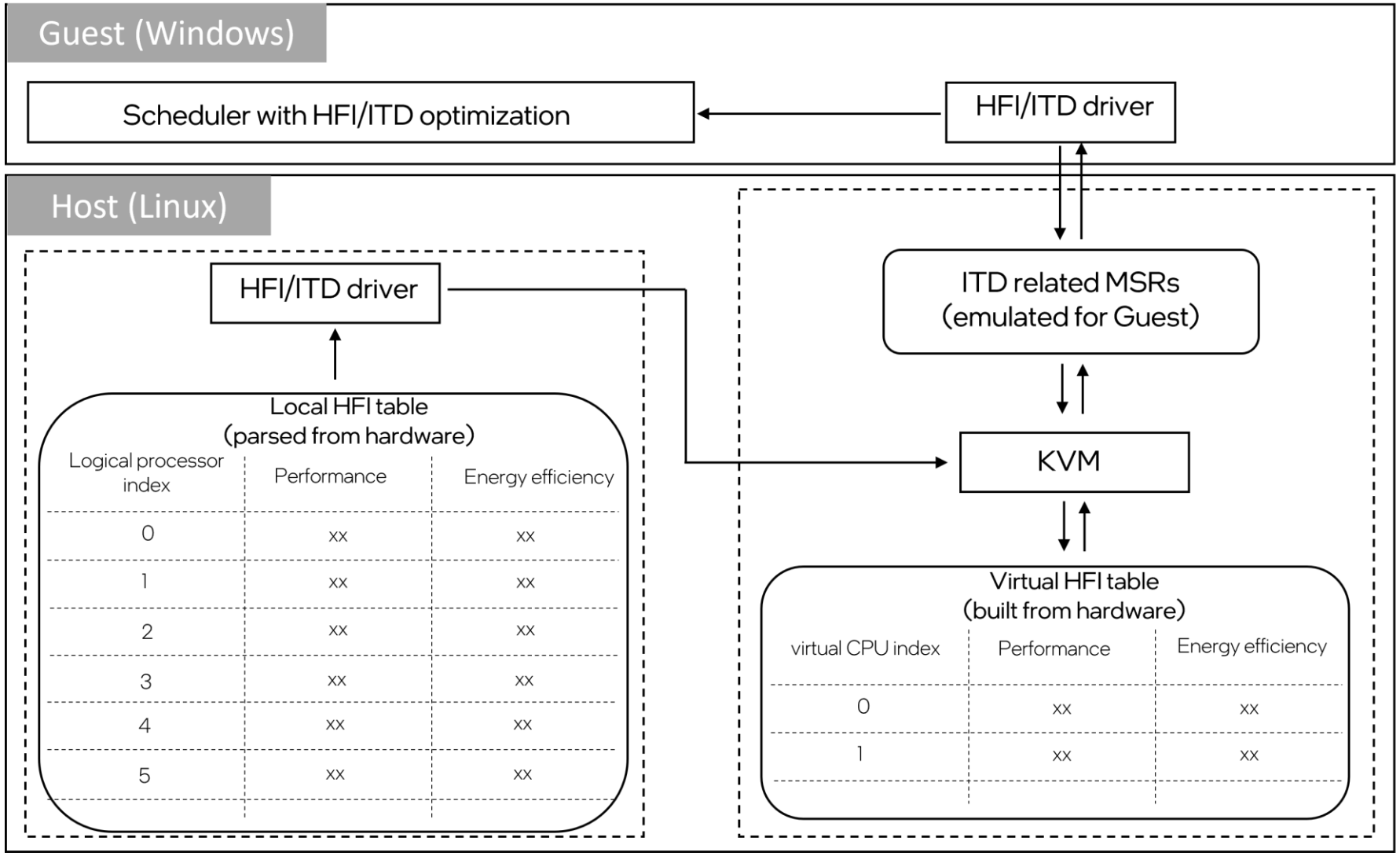
* 3D Mark on Intel(R) Core(TM) i9-13900K, 2995 Mhz, 24 Cores (8+16), 32 Threads
** Result = score / smp topo base line ("-smp cpus=total_cpus") * 100%

ITD performance in Windows Guest

Three Elements of Optimal CPU Performance:

- ITD enabling in Guest
- Hybrid CPU topology
- Pinning vCPUs (CPU affinity)

ITD virtualization solution



ITD virtualization solution

Opens & Future work:

- Thermal dependencies:
  - Thermal management (CPU scope & package scope).
  - Thermal interrupt.
- Windows dependencies:
  - ACPI Collaborative Processor Performance Control (CPPC) table...
- Future work:
  - Move dependencies emulation to user space...

ITD virtualization solution

Patch link:

- KVM part:
  - [RFC 00/26] Intel Thread Director Virtualization:
  - https://lore.kernel.org/kvm/20240203091214.411862-1-zhao1.liu@linux.intel.com/

- QEMU part:
  - [RFC 0/6] Intel Thread Director Virtualization Support in QEMU:
  - https://lore.kernel.org/qemu-devel/20240203093054.412135-1-zhao1.liu@linux.intel.com/

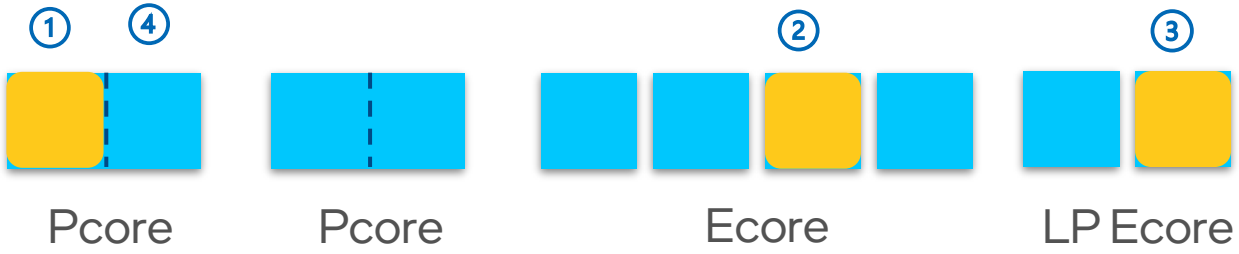
Hybrid CPU scheduling improvement

Intel Turbo Boost Max Technology (ITMT) virtualization for Linux Guest

Intel Turbo Boost Max Technology (ITMT) scheduling

- About ITMT:
  - Some CPU cores can reach higher turbo boost frequencies and thus have better performance.
  - ITMT scheduling detects the cores with the highest available turbo frequencies and prefers those cores to place tasks.
  - For hybrid platform, ITMT implements the "P core, E core, LP E core, P core-SMT" scheduling priority order.

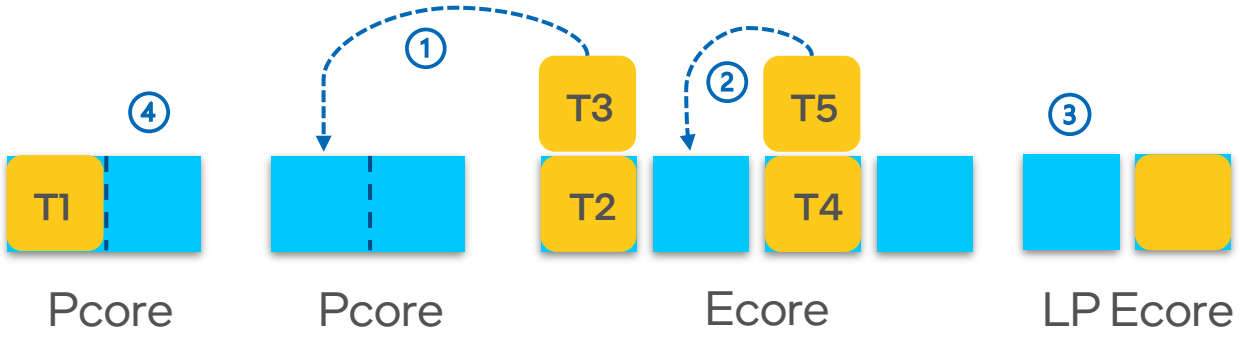
Order of task placement



Intel Turbo Boost Max Technology (ITMT) scheduling

- About ITMT:
  - Some CPU cores can reach higher turbo boost frequencies and thus have better performance.
  - ITMT scheduling detects the cores with the highest available turbo frequencies and prefers those cores to place tasks.
  - For hybrid platform, ITMT implements the "P core, E core, LP E core, P core-SMT" scheduling priority order.

Order of idle load balance



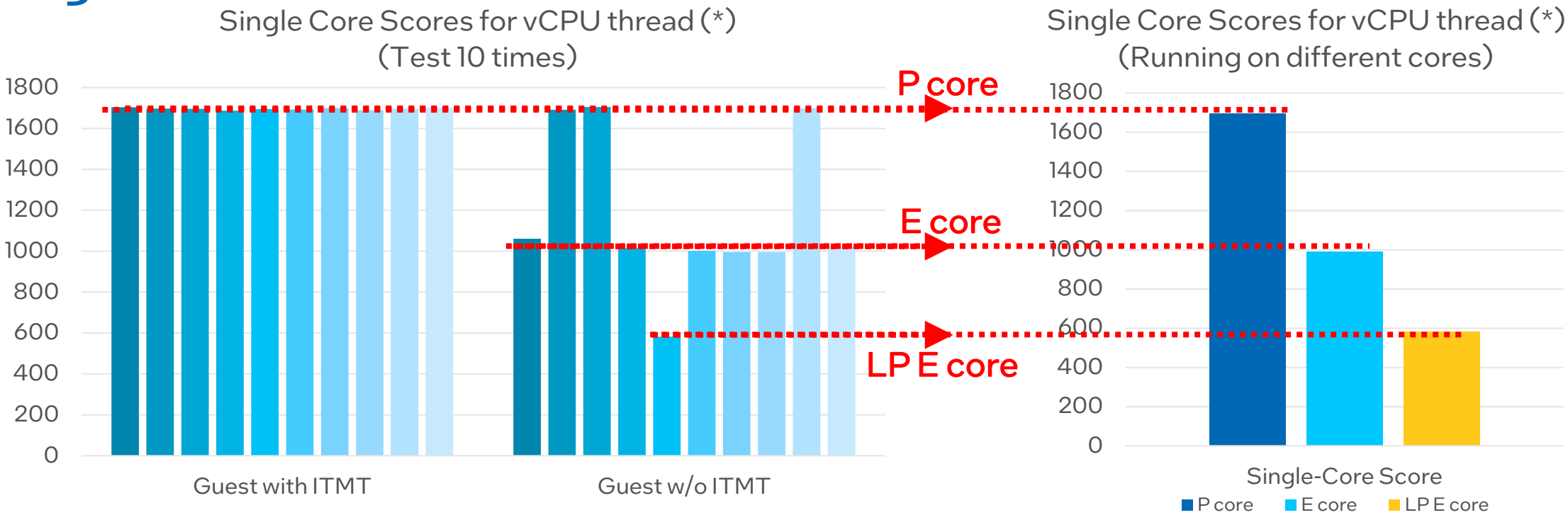
ITMT scheduling cases in virtualization

	No ITMT in Guest	Enable ITMT in Guest
vCPUs Pinning	 (aka, Guest w/o ITMT)	 (aka, Guest with ITMT)
vCPUs non-pining	 (aka, only ITMT on Host)	

- The wrong hybrid information can hurt performance!

Case 1: “Guest with ITMT” v.s. “Guest w/o ITMT”

(Pinning vCPUs on Host) “Guest with ITMT” is better for single thread.

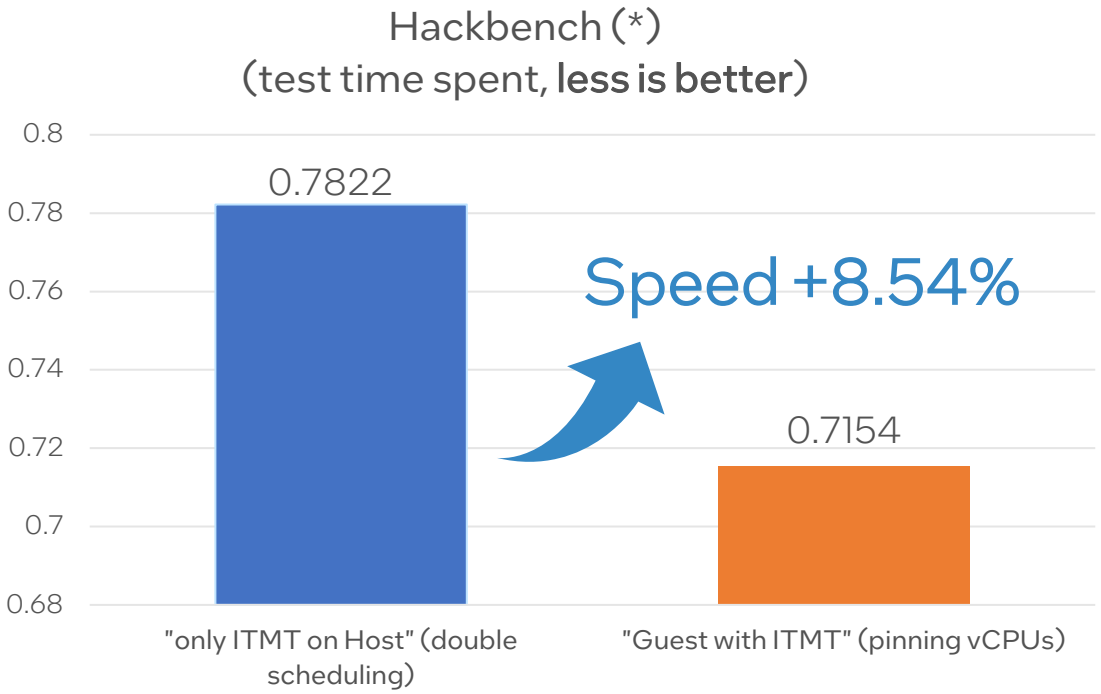


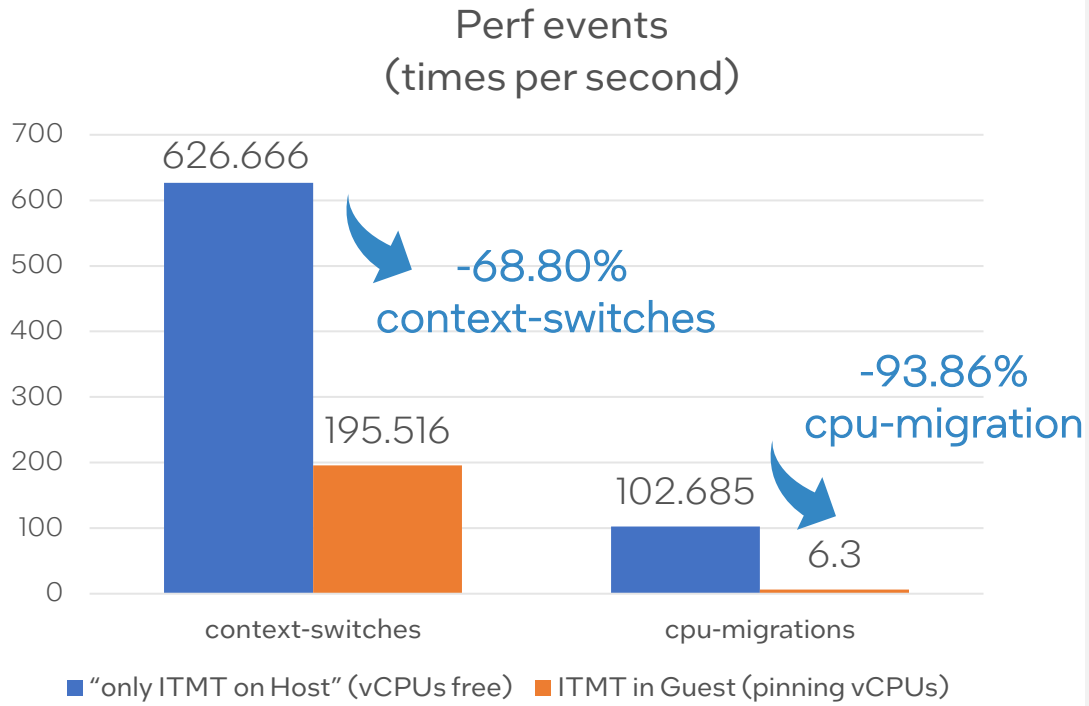
- If Guest has no knowledge about CPU difference, task will randomly run on P core/E core/LP E core without maximizing single thread performance

* Geekbench v5.3.1 on Intel(R) Core(TM) Ultra 5 1003H, 14 Cores (4+8+2), 18 Threads

Case 2: “Guest with ITMT” v.s. “only ITMT on Host”

“Guest with ITMT” (Pinning vCPUs on Host) is better on frequent scheduling.





- “Guest with ITMT” (pinning vCPUs), reduce the overhead of double scheduling.

* Hackbench v2.20 (-g 10 -T 18 -s 1024 -l 1000) on Intel(R) Core(TM) Ultra 5 1003H, 14 Cores (4+8+2), 18 Threads

ITMT virtualization

- Current POC:
  - Virtual MSRs to give Guest highest performance hints, then Guest sets the ITMT CPU priority based on hints.
- Future work:
  - PV sched framework?
    - PV sched framework is the more general way to improve Guest/Host scheduling.
    - We can apply ITMT to this framework.

Acknowledge:

Len Brown, Ricardo Neri,
Dapeng Mi, Yongwei Ma,
Yanting Jiang, Zhuocheng Ding


intel