intel

Game Developers Conference

Program Your Games Today. Prepare for Tomorrow.

Rudolph Balaz
Director/GM of Game Engineering

Agenda

- Brief History Lesson
- Performance Variability
- Impact on Games
- Best Practices
- Suggestions
- Summary

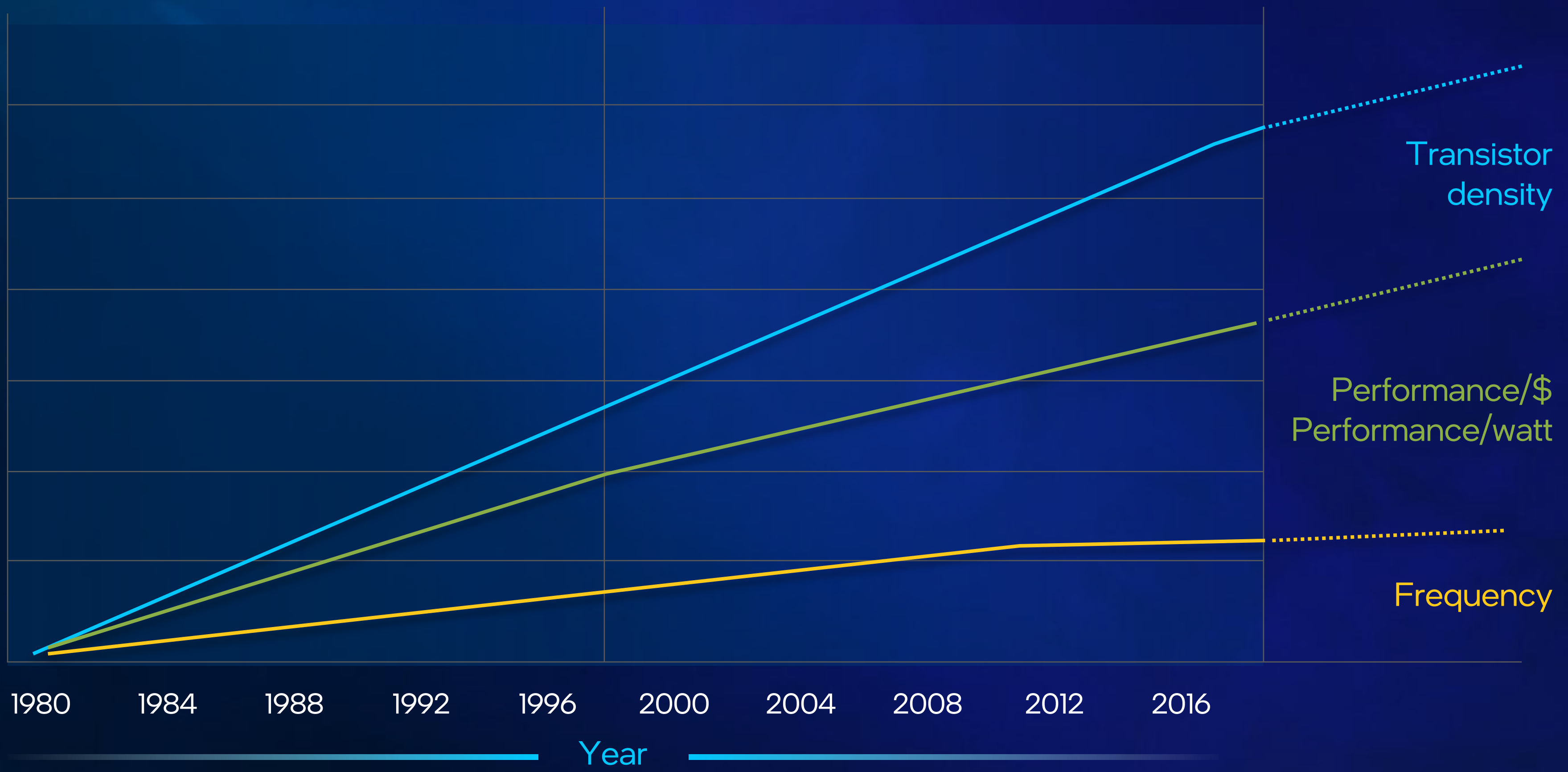
Brief History Lesson: Moore's Law & Architecture

Moore's Law and Architecture

MEGAHERTZ ERA

MULTI-CORE ERA

ARCHITECTURE ERA



CPU

AI

GPU

FPGA

Purpose Built Client

EXPERIENCE FIRST

SCALABLE

ENERGY EFFICIENT

OPTIMAL USE OF MOORE'S LAW

Corporate Employee

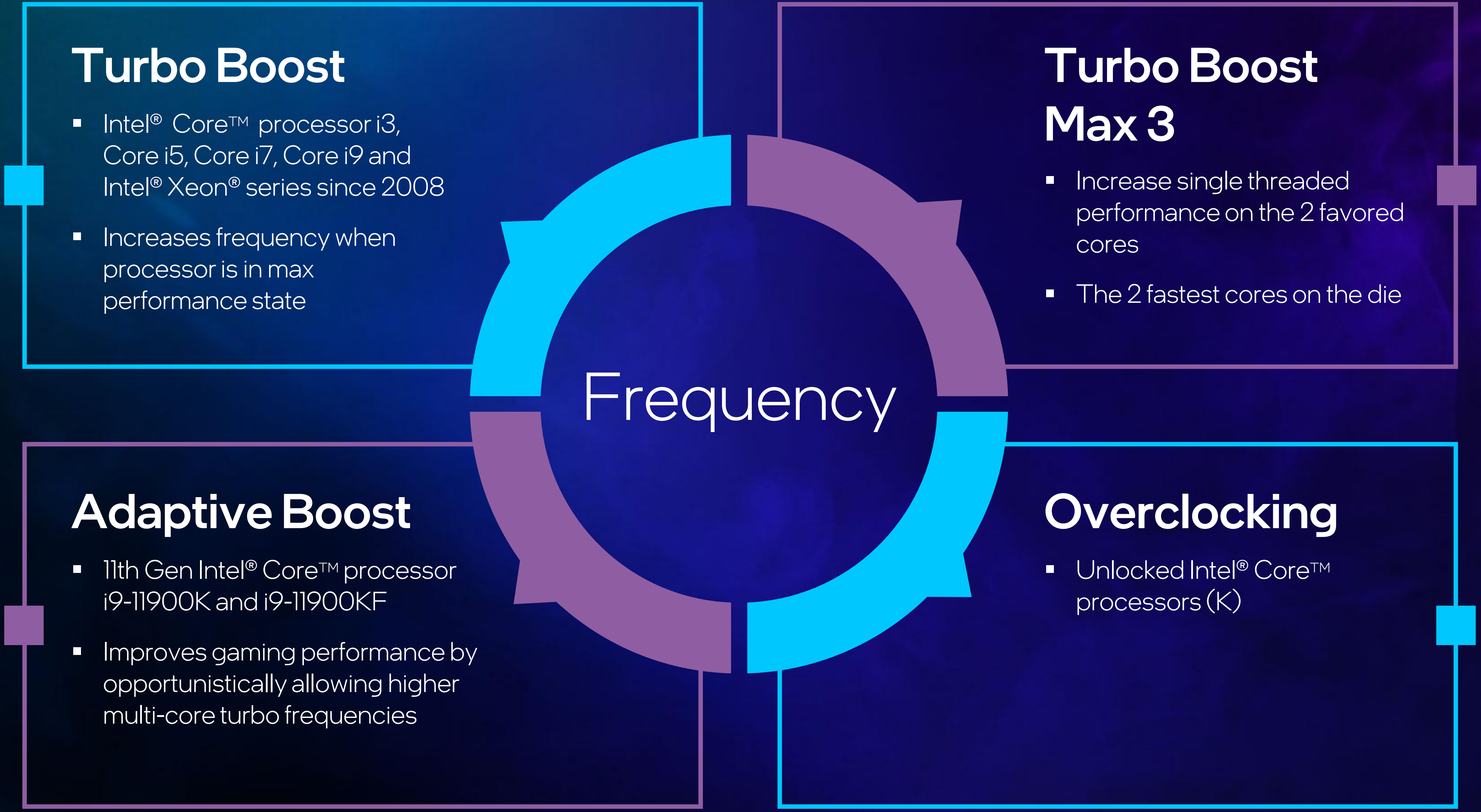
Mobile

Gamer

Creator

Focus on Performance for General Purpose Compute (CPU)
Focus on Density for scalable compute (GPU, AI etc.)

Performance Variability



Frequency

Turbo Boost

- Intel® Core™ processor i3, Core i5, Core i7, Core i9 and Intel® Xeon® series since 2008
- Increases frequency when processor is in max performance state

Turbo Boost Max 3

- Increase single threaded performance on the 2 favored cores
- The 2 fastest cores on the die

Adaptive Boost

- 11th Gen Intel® Core™ processor i9-11900K and i9-11900KF
- Improves gaming performance by opportunistically allowing higher multi-core turbo frequencies

Overclocking

- Unlocked Intel® Core™ processors (K)

Processor Count

Physical Processors

- **Desktop**
  - 65w to 150w
  - Intel® Core^{TM} i9 – 8 to 10 cores
  - Core i9 Extreme Edition 8 to 18 cores
- **Enthusiast Laptop**
  - 30w to 65w – 6 to 8 cores
- **Thin/Light laptop**
  - 12w to 28w – 4 cores

Logical Processors

- **Intel® Hyper-Threading Technology**
  - Allows more than one thread to run on each core
- **Typically, available on Core i5 and above**
- **Can be a performance boost on some workloads**
- **Available on more systems than ever before**
- **Trivia Question:**
  - Does Hyper-Threading apply to all processors on a package?

So, What’s the Problem?

Heat & Power

- Frequency
- Cores
- Threading
- Packaging
- Chassis

Not all workloads require max performance or max feature sets

- Games usually have a sweet spot around 8 cores or less
- Or various bottlenecks – Threading, Memory, I/O, etc.

Intel® Core™ Processor with Intel® Hybrid Technology

- **Launched in 2019 with 2 processors in heterogenous config**

- **High level goals:**
  - Balance of performance and power efficiency in small footprint
  - Enable design flexibility for mobile form factors, such as foldable
  - Always on, always connected, very low standby power

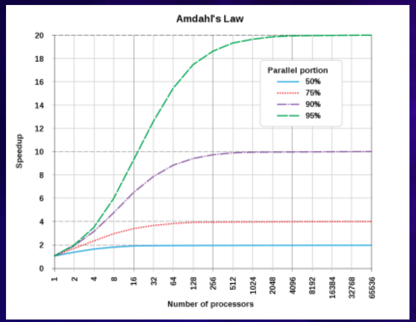
SUNNY COVE

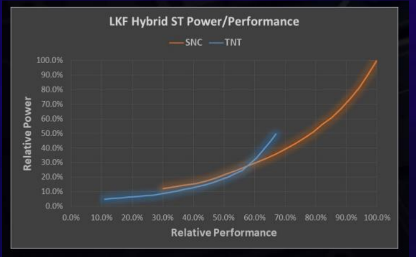
- Concentrate on single and limited threading scenarios
- Performance focused

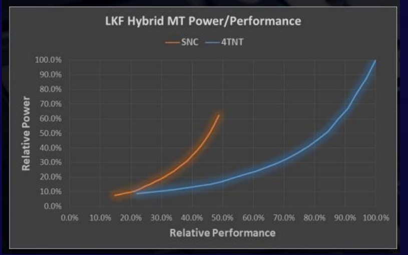
TREMONT

- Concentrate on throughput and power-limited scenarios
- Efficiency focused

- **Application runs on:**
  - 1 x Intel "Sunny Cove" core used for performance, serial, compute threads
  - 4 x Intel "Tremont" cores used for efficient, parallel, compute threads







Impact on Games

Assumptions You Should No Longer Make

There can be a significant performance delta between cores

- Even identical cores may run at different frequencies

There may be 1, 2, or more faster cores

The fastest core may move around the package

Hyperthreading may be available on only some cores in a package

- Logical core count may not equal 2x physical core count

The core topology layout may not be simple

- Performance, ordering or relationship between logical processors may change

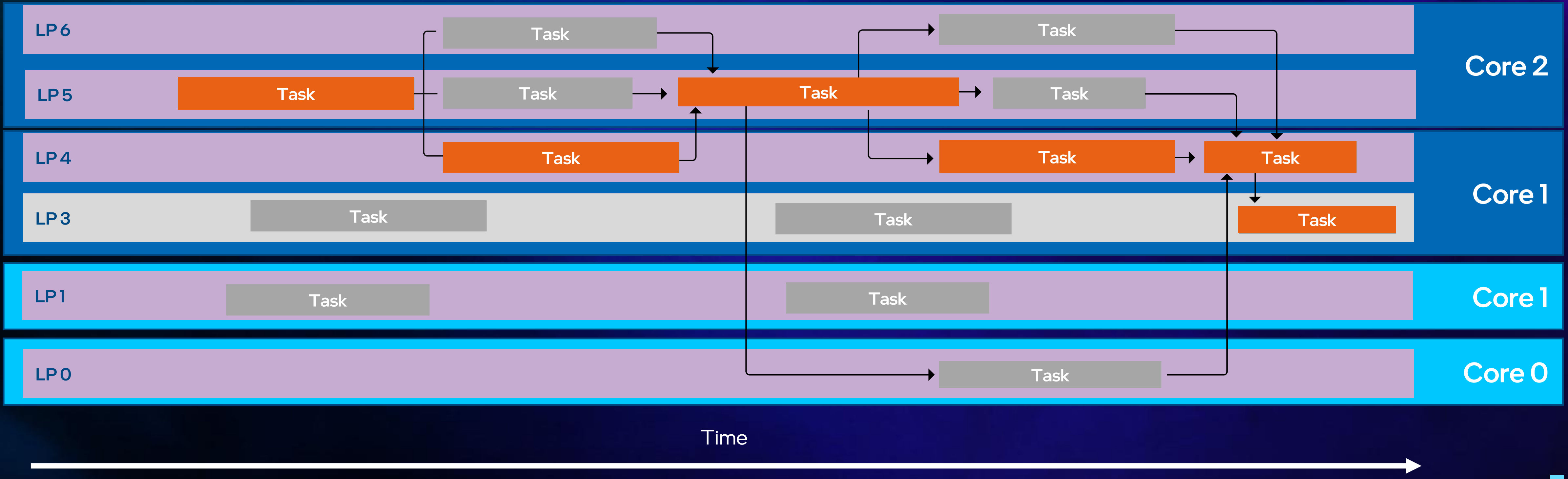
ISA may be identical, but specific performance of an instruction may vary

Running efficiently or slower may be overall faster

Power may be shared between GPU/CPU/Other -> frequency impact

The Critical Path

Definition: The extended critical path is the executed code segments of a program that, when reduced with a small ∑, will reduce the completion time on a given number of processors.



Best Practices

Profile your workload

Don’t oversubscribe your thread pool

- Don’t use sibling cores if your workload can’t benefit from hyperthreading
- Avoid unnecessary context switches

Avoid scheduling lower priority task on the same cores as your critical path

- Understand how your middleware uses threads

Avoid static partitioning, allow cores to steal work from other cores

- Do not use Processor Affinity

Avoid scheduling lower priority task on the same cores as your critical path

Understand how your middleware uses threads

Job systems need to dynamically balance based on core characteristics

Techniques for Maximizing Performance

Use SetThreadPriority(HANDLE, THREAD_PRIORITY_ABOVE_NORMAL) work that is

- Frequency/latency sensitivity
- Critical Path
- Render thread
- Needs Fastest ISA

Use SetThreadPriority(HANDLE, THREAD_PRIORITY_BELOW_NORMAL) work that is

- Secondary workloads
- Throughput workloads
- Async workloads
- IO threads
- Background threads/processes

Try implementing

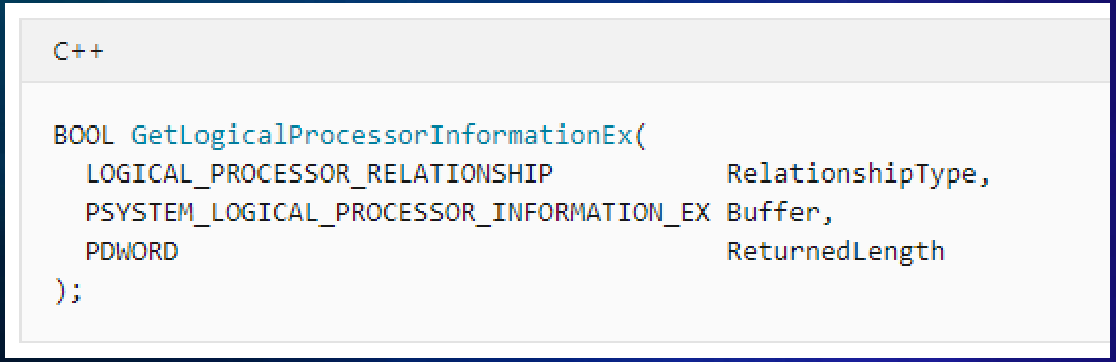
- A Primary and Secondary thread pool for different classes of work
- Decouple asynchronous workloads from primary thread pool
  - Shader Compilation, Audio Mixing, Asset Streaming, Decompression
- Offload none critical work to secondary thread pool
- Task stealing from primary to secondary?

Call To Action

- Verify your assumptions about the processor architecture
- Use Thread Priority and QoS APIs
- Make your code resilient to variations in core performance
- Allocate just enough threads for your workload
- Take advantage of the performance deltas by putting the right work on each core
- The only constant in the future is change - prepare for it

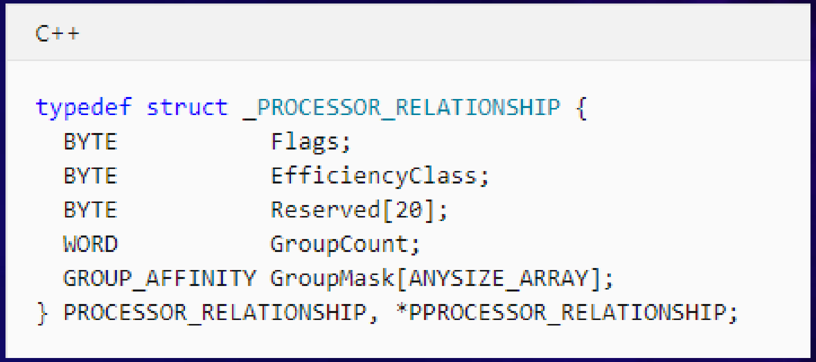
API Reference

Detecting The Cores



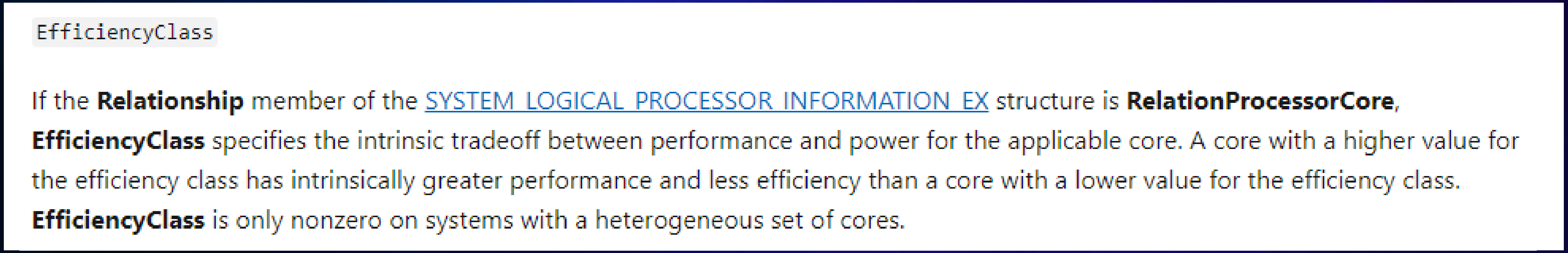
C++

```
BOOL GetLogicalProcessorInformationEx(
  LOGICAL_PROCESSOR_RELATIONSHIP           RelationshipType,
  PSYSTEM_LOGICAL_PROCESSOR_INFORMATION_EX Buffer,
  PDWORD                                   ReturnedLength
);
```



C++

```
typedef struct _PROCESSOR_RELATIONSHIP {
  BYTE           Flags;
  BYTE           EfficiencyClass;
  BYTE           Reserved[20];
  WORD           GroupCount;
  GROUP_AFFINITY GroupMask[ANYSIZE_ARRAY];
} PROCESSOR_RELATIONSHIP, *PPROCESSOR_RELATIONSHIP;
```



`EfficiencyClass`

If the **Relationship** member of the SYSTEM_LOGICAL_PROCESSOR_INFORMATION_EX structure is **RelationProcessorCore**, **EfficiencyClass** specifies the intrinsic tradeoff between performance and power for the applicable core. A core with a higher value for the efficiency class has intrinsically greater performance and less efficiency than a core with a lower value for the efficiency class. **EfficiencyClass** is only nonzero on systems with a heterogeneous set of cores.

https://docs.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-processor_relationship

Hard vs Soft Affinity

Hard affinity using SetThreadAffinityMask, is a contract with OS, prevents optimizations for power and performance

SetThreadIdealProcessor()

https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-setthreadidealprocessor

- You can use the GetSystemInfo function to determine the number of processors on the computer.
- You can also use the GetProcessAffinityMask function to check the processors on which the thread is allowed to run. Note that **GetProcessAffinityMask** returns a bitmask whereas **SetThreadIdealProcessor** uses an integer value to represent the processor.

Setting Quality of Service for a Process or Thread

- **MS provides 2 APIs to indicate importance of work done by thread/process**
  - **SetProcessInformation()**
    - https://msdn.microsoft.com/en-us/library/windows/desktop/hh448389(v=vs.85).aspx
  - **SetThreadInformation()**
    - https://msdn.microsoft.com/en-us/library/windows/desktop/hh448390(v=vs.85).aspx

```
BOOL WINAPI SetProcessInformation(
 _In_ HANDLE                                  hProcess,
 _In_ PROCESS_INFORMATION_CLASS               ProcessInformationClass,
 _In_reads_bytes_(ProcessInformationSize)     ProcessInformation,
 _In_ DWORD                                   ProcessInformationSize
);
```

ProcessMemoryPriority and
ProcessPowerThrottling

PROCESS_POWER_THROTTLING_STATE
Data structure

Legal Notices and Disclaimers

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

The products and services described may contain defects or errors known as errata which may cause deviations 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 computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

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.

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates.

Your costs and results may vary.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

intel®