

LOW LATENCY C++

C++ ON SEA 2019

DAVID GROSS
WALTHER ZWART

06/02/2019

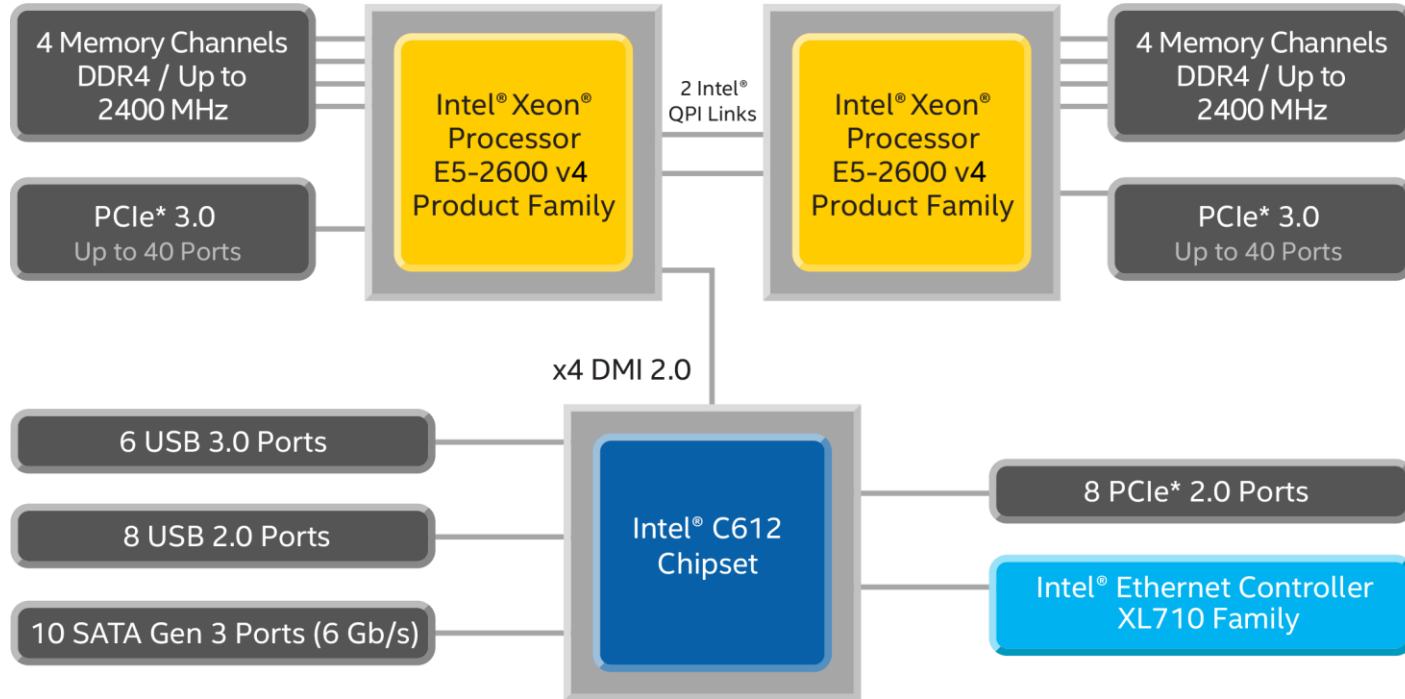


CPU AND MEMORY ARCHITECTURE

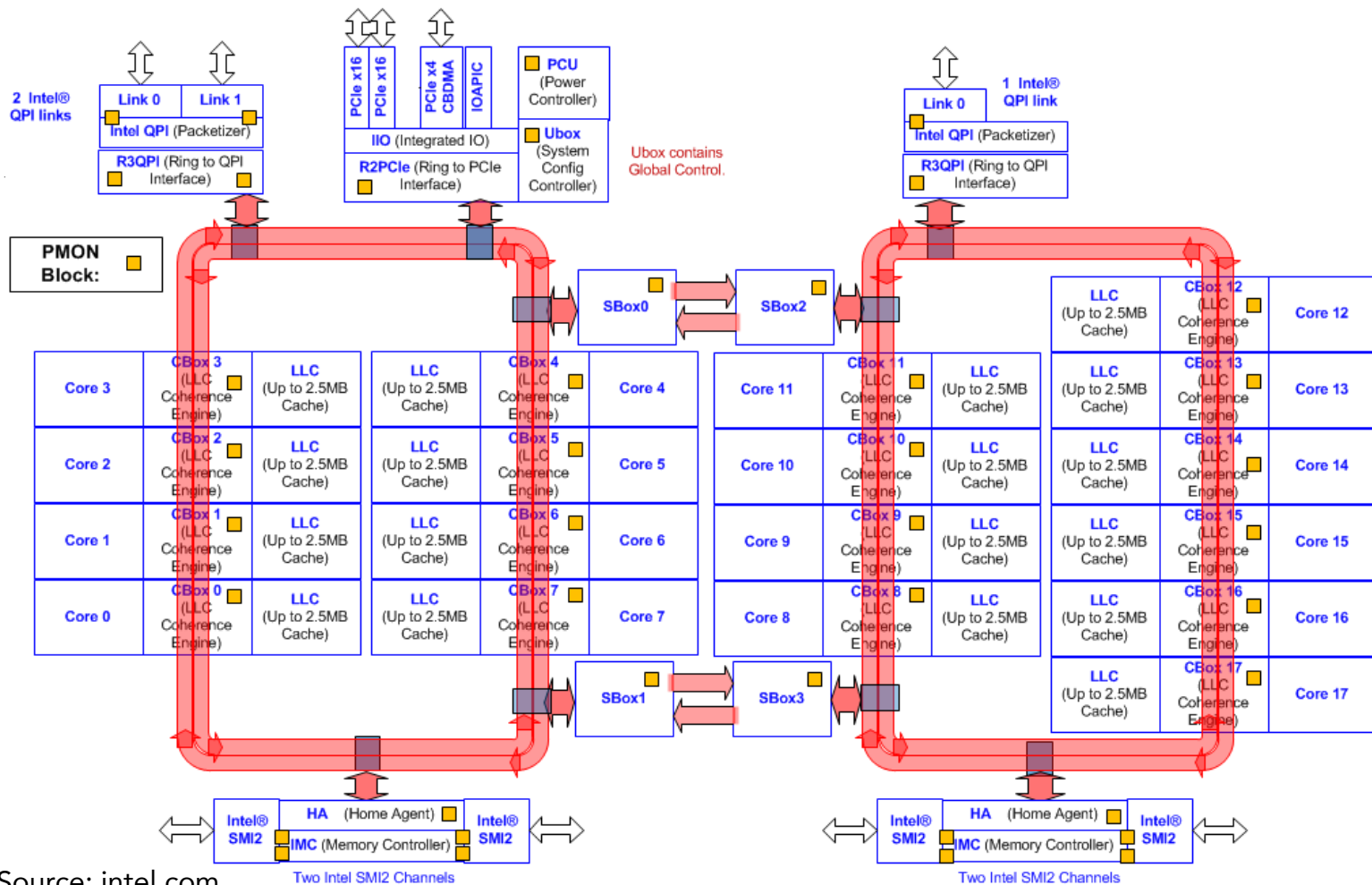
C++ ON SEA 2019



MODERN X86 ARCHITECTURE



Source: intel.com



Source: intel.com

Two Intel SMI2 Channels

Two Intel SMI2 Channels

MODERN X86 ARCHITECTURE

- △ Growing importance of the Uncore
 - △ Integrated memory controller (iMC)
 - △ Processor point-to-point Interconnect, e.g. Intel QuickPath Interconnect (QPI)
 - △ Snooping, e.g. Caching and Home agents
 - △ Last Level Cache (LLC)
 - △ On-die GPU
 - △ IOs controllers, e.g. PCIe

CPU CACHES LAYOUT



KNOW YOUR LATENCIES



KNOW YOUR LATENCIES

Latency Numbers Every Programmer Should Know

■ 1 ns

■ L1 cache reference: 0.5 ns

■ Branch mispredict: 5 ns

■ L2 cache reference: 7 ns

■ Mutex lock/unlock: 25 ns

■ = 100 ns

■ Main memory reference: 100 ns

■ = 1 μ s

■ Compress 1 KB with Zippy: 3 μ s

■ = 10 μ s

■ Send 1 KB over 1 Gbps network: 10 μ s

■ SSD random read (1 Gb/s SSD): 150 μ s

■ Read 1 MB sequentially from memory: 250 μ s

■ Round trip in same datacenter: 500 μ s

■ = 1 ms

■ Read 1 MB sequentially from SSD: 1 ms

■ Disk seek: 10 ms

■ Read 1 MB sequentially from disk: 20 ms

■ Packet roundtrip CA to Netherlands: 150 ms

Source: <https://gist.github.com/2841832>

BENCHMARKING C++

C++ ON SEA 2019



WHY BENCHMARKING?



WHY BENCHMARKING?

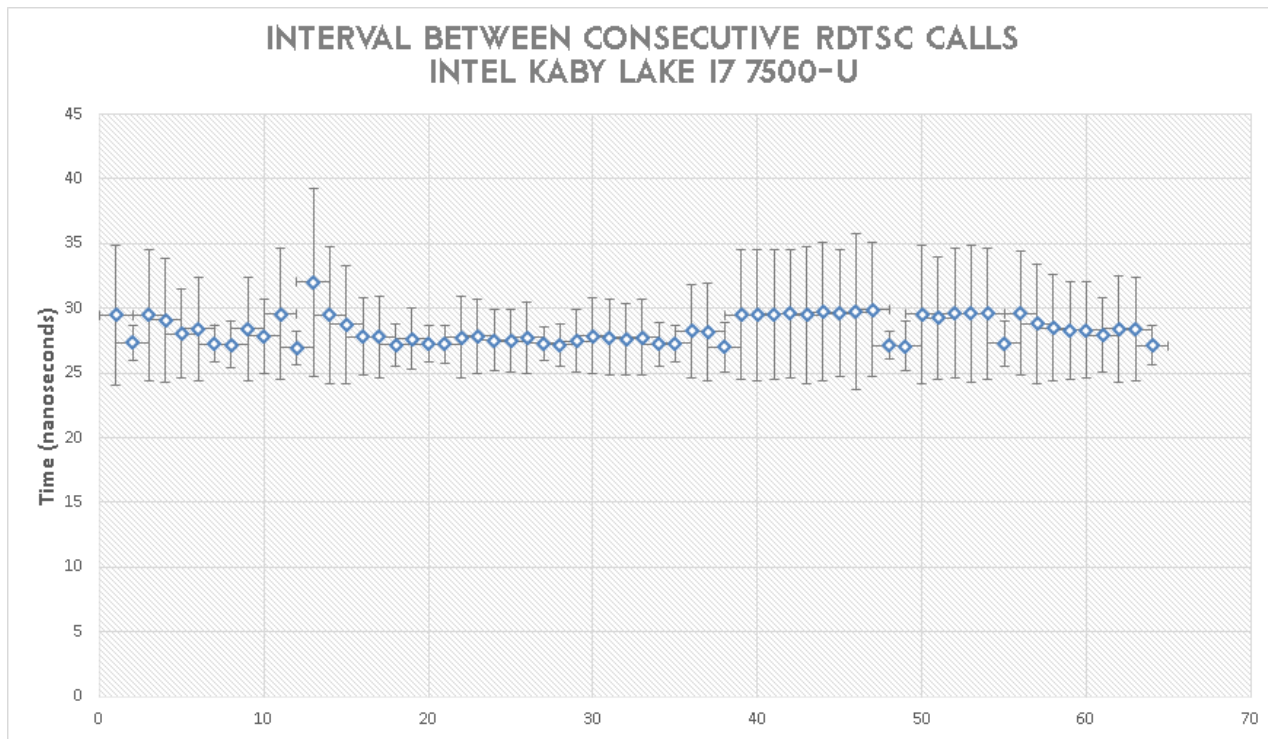
- △ If you don't benchmark your code, you don't care about its performance
- △ Iterative process
 - △ Measure: macro benchmarking, metrics, ...
 - △ Find : Pareto principle
 - △ Isolate and optimize: micro benchmarking
 - △ Repeat

LET'S BENCHMARK

C++ ON SEA 2019



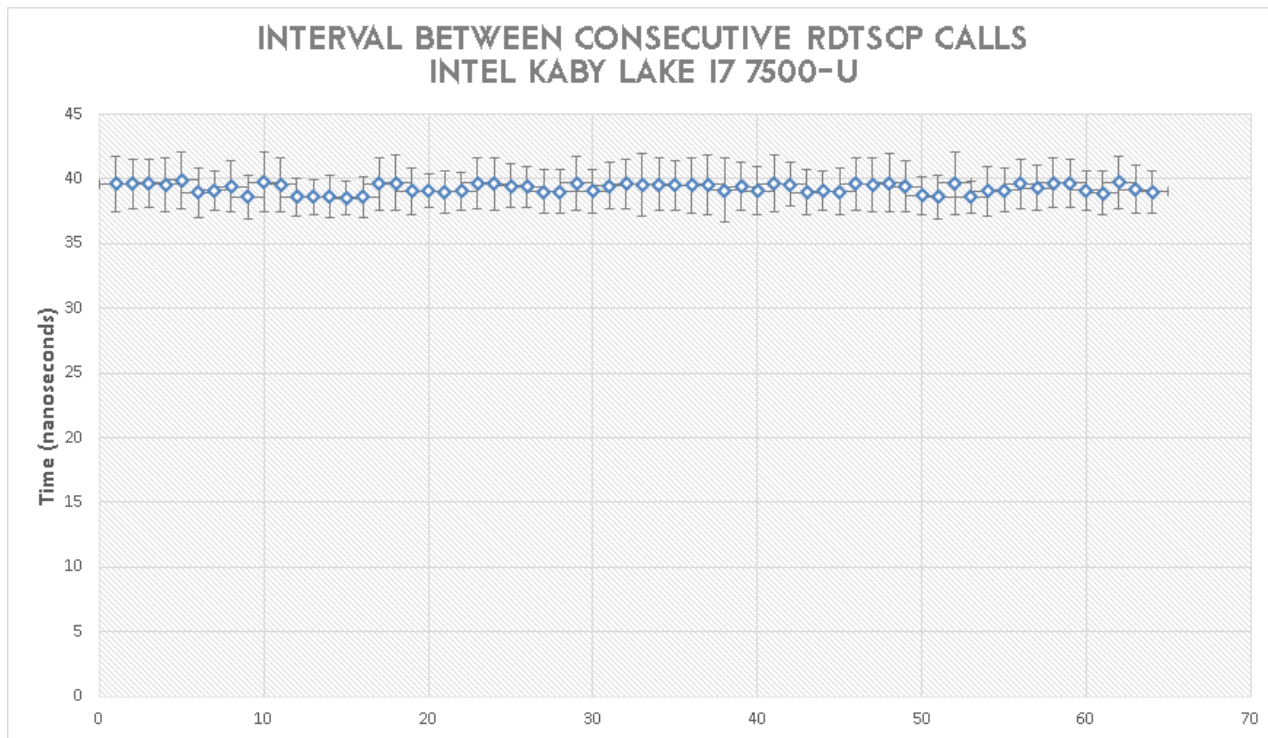
RDTSC, RDTSCP? FENCE, NO FENCE?



1 sample: 32-256 calls

Sample std dev: 3.49ns

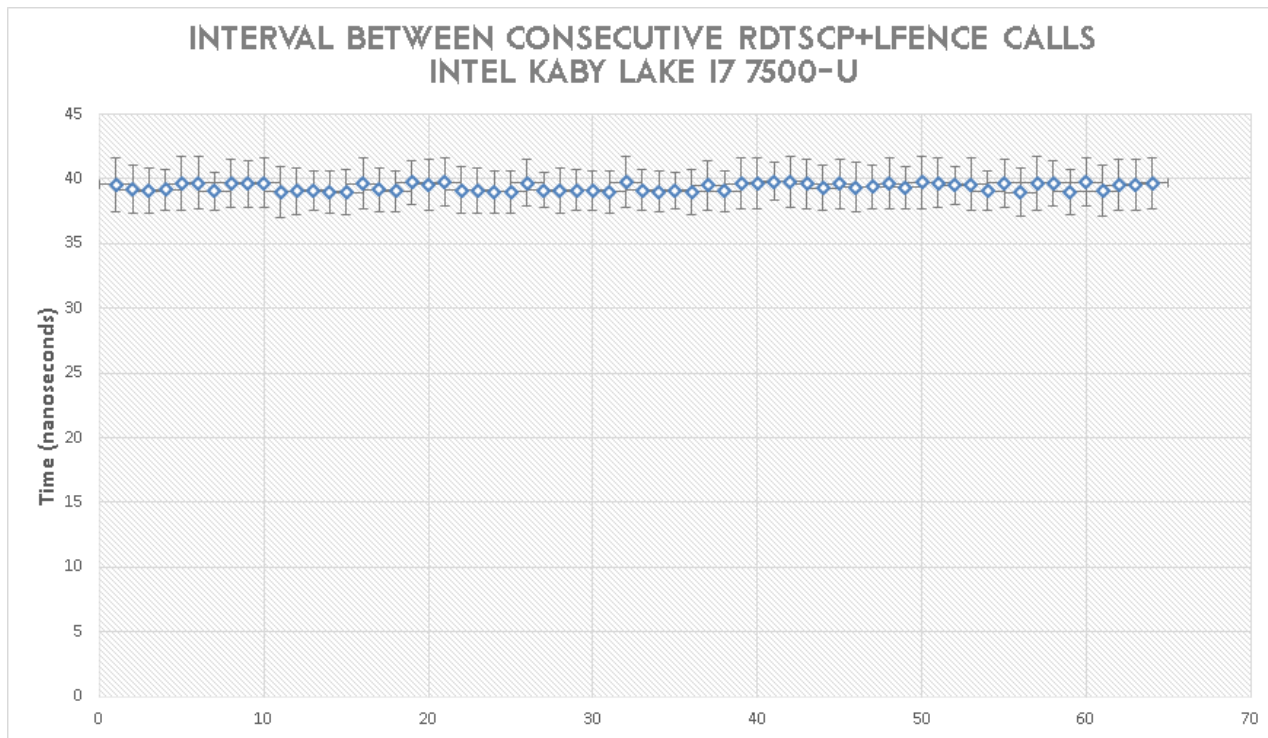
RDTSC, RDTSCP? FENCE, NO FENCE?



1 sample: 32-256 calls

Sample std dev: 1.85ns

RDTSC, RDTSCP? FENCE, NO FENCE?



1 sample: 32-256 calls

Sample std dev: 1.72ns