# Optimizing For Low Latency

Ramakrishna Karedla
Financial Services Lab
Parsippany, NJ

# The Challenges to Nano-Second Latencies

## Network

Data transfer on the network,
Network Hardware Latencies, Network Drivers Latencies, etc

## Software

Sync overhead and oversubscription of threads,
Interrupts, I/O, Parallel Programming, Software Tools, etc

## Compute

Frequency, Instructions per cycle,
Cache Sizes, Memory Speed & Bandwidth, RAS

# Achieving Low Latency

Maximizing Performance involves "Hardware & System Level" optimizations

- Implement low latency BIOS settings recommended by the OEM
  - OEM BIOS Settings: SMIs, HyperThreading, C-States- All Off.
  - Better Turbo stability in SandyBridge over previous version. Recommend enabling
  - Turn Memory Power Savings Off.

- On the application side: Maximize your resources by...
  - Pin Threads, Interrupts, and Processes to individual cores . CPUSets, isolcpus
  - Use the DPDK tool kit to optimize code www.intel.com/go/dpdk
  - Place "communicating" threads on adjacent cores. Avoid Global variables and partial cache line writes
  - Avoid use of locking algorithms; Keep retry section to at least 500 nsecs
    http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/xeon-lock-scaling-analysis-paper.pdf
  - Avoid mixing SSE instructions with AVX instructions in same code segment
    use separate compilation units or use vzeroupper
  - Evaluate IPP / SSE 4.2 for string comparison functions
    - ✓ http://software.intel.com/en-us/articles/using-avx-without-writing-avx-code

# Achieving Low Latency (contd)

- Determine how many cores your trading strategy requires. Avoid cross socket penalties. Match CPU+NIC as per DDIO strategy. Use polling to read data from L3 cache before data gets flushed !

- If using Red Hat, refer Red Hat suggestions for low latency profiles
  https://access.redhat.com/site/articles/221153

- If using acpi-cpufreq driver, suggest using performance scaling governor
  - ✓ https://access.redhat.com/knowledge/articles/221153

- MOVBE new instruction in Haswell. Suitable for network packet format conversions instead of BSWAP to reduce latency. Can use PSHUFB on SandyBridge

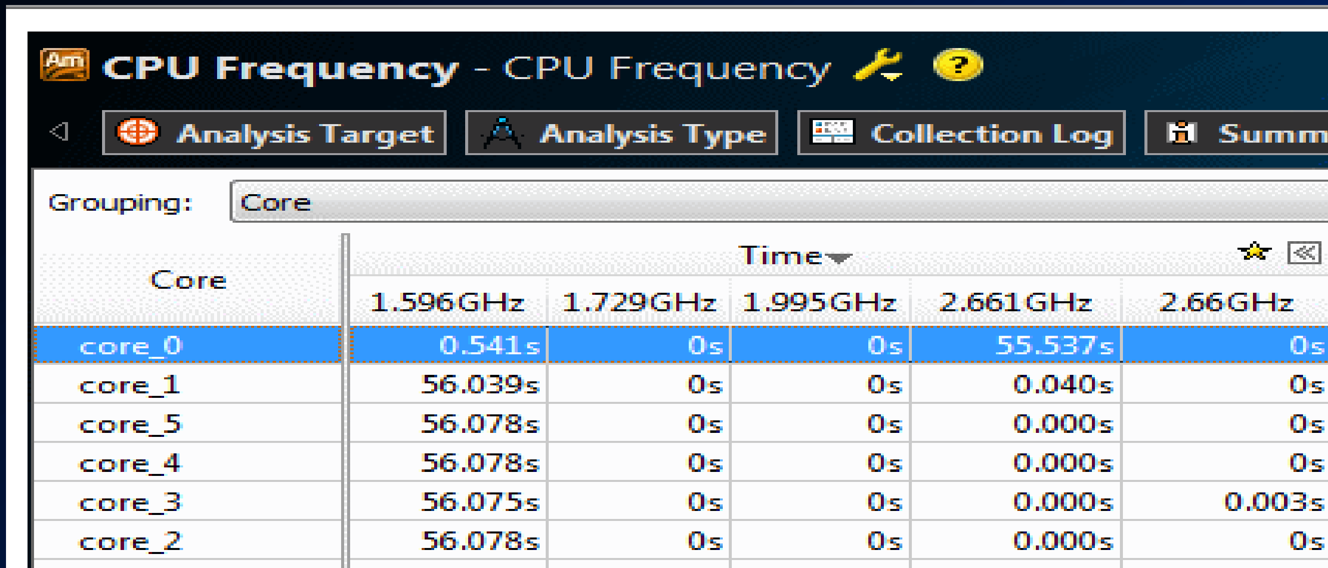
- Reduce kernel jitter by minimizing Ring Transitions. E.g Use NOPs (0.5 cycle) versus nanosleep()

- Read assembly (not difficult ! ). Compiler can always optimize code away !

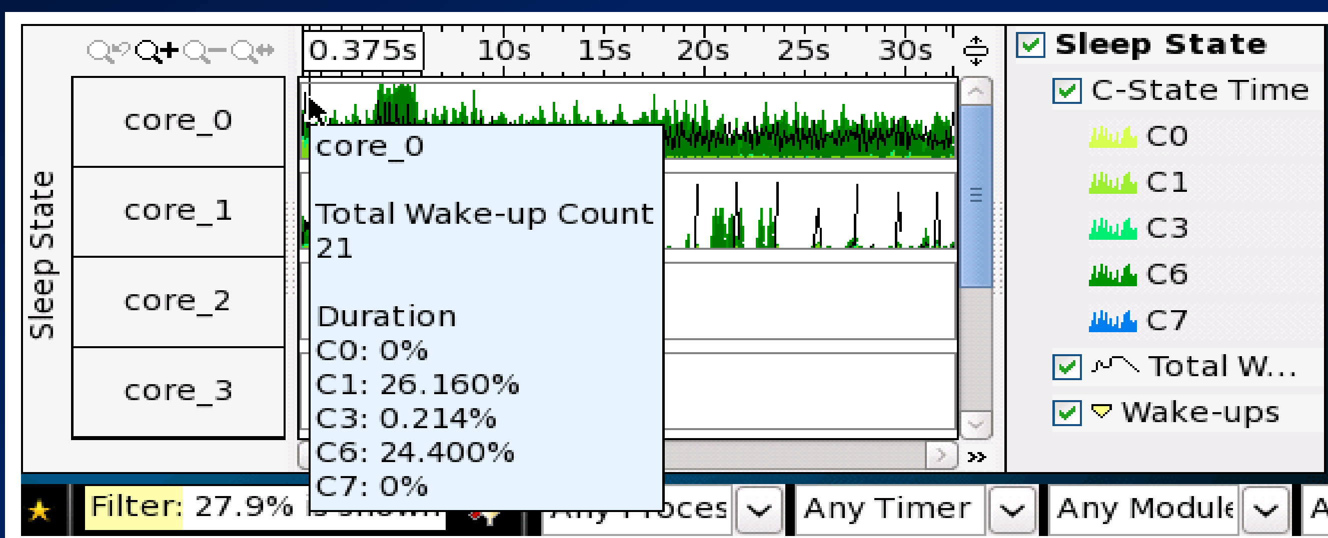
# Achieving Low Latency (contd)

- Linux, For C0 states Use idle=mwait instead of idle=poll. Same results. Less power.
  Windows, use powercfg.exe /setacvalueindex SCHEME_CURRENT SUB_PROCESSOR 5d76a2ca-e8c0-402f-a133-2158492d58ad 1
  powercfg.exe /setactive SCHEME_CURRENT
- Use RDTSCP instead of gettimeofday() to profile short code sections.
  RDTSCP invariant across cores and sockets in a box (with no node controllers)
- For specific code sections, use CPUID to serialize instructions to avoid Branch mispredictions that may cause jitter.
- Optimize NIC driver settings for low latency rather throughput
- At times, SystemTap scripts can give a better insight into kernel behavior
  http://sourceware.org/systemtap/
- VTune can now profile Java code. Consider using jhiccup to observe jitter
  http://jhiccup.com/
- If business case permits, consider using Intel Xeon PHI for low latency workloads
  http://software.intel.com/mic-developer

# VTune™ Amplifier XE 2013 – Power Analysis Functionality
# CPU Frequency & Sleep State Analysis – Timeline




| Core | Time | | | | |
|---|---|---|---|---|---|
| | 1.596GHz | 1.729GHz | 1.995GHz | 2.661GHz | 2.66GHz |
| core_0 | 0.541s | 0s | 0s | 55.537s | 0s |
| core_1 | 56.039s | 0s | 0s | 0.040s | 0s |
| core_5 | 56.078s | 0s | 0s | 0.000s | 0s |
| core_4 | 56.078s | 0s | 0s | 0.000s | 0s |
| core_3 | 56.075s | 0s | 0s | 0.000s | 0.003s |
| core_2 | 56.078s | 0s | 0s | 0.000s | 0s |



# Intel's Performance Counter Monitor – Real Time Tool

- Complements Intel's Flagship Profiler: Vtune Amplifier
- Can be used stand alone or embedded in code segments
- SOCH0; DRAMClocks: 665224950; Rank0 CKE Off Residency: 0.00%
- Core Frequency transition count: 0 ( useful to study Outliers)

| Core | (SKT) | IPC | FREQ | AFREQ | L3MISS | L2MISS | L3HIT | L2HIT | L3CLK | L2CLK | TEMP |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0.25 | 1.00 | 1.15 | 802 | 1557 | 0.48 | 0.45 | 0.00 | 0.00 | 56 |
| 1 | 0 | 0.25 | 1.00 | 1.15 | 3554 | 4671 | 0.24 | 0.38 | 0.00 | 0.0 | 62 |
| 2 | 0 | 0.25 | 1.00 | 1.15 | 49 | 608 | 0.92 | 0.45 | 0.00 | 0.00 | 60 |
| 3 | 0 | 0.25 | 1.00 | 1.15 | 86 | 312 | 0.72 | 0.28 | 0.00 | 0.00 | 61 |
| 4 | 1 | 0.25 | 1.00 | 1.15 | 805 | 6764 | 0.88 | 0.44 | 0.00 | 0.00 | 64 |

C0 (active,non-halted) core residency: 100.00 %

C1 core residency: 0 %; C3 core residency: 0 %; C6 core residency: 0 %; C2 package residency: 0 %; C3 package residency: 0 %; C6 package residency: 0 %

Also shows QPI Traffic and Memory Read /Write Performance Monitoring

SMI count via Turbostat ( MSR 0x34)

# Financial Services Lab - NJ

Provides testing of new H/W and Software technologies to financial vertical

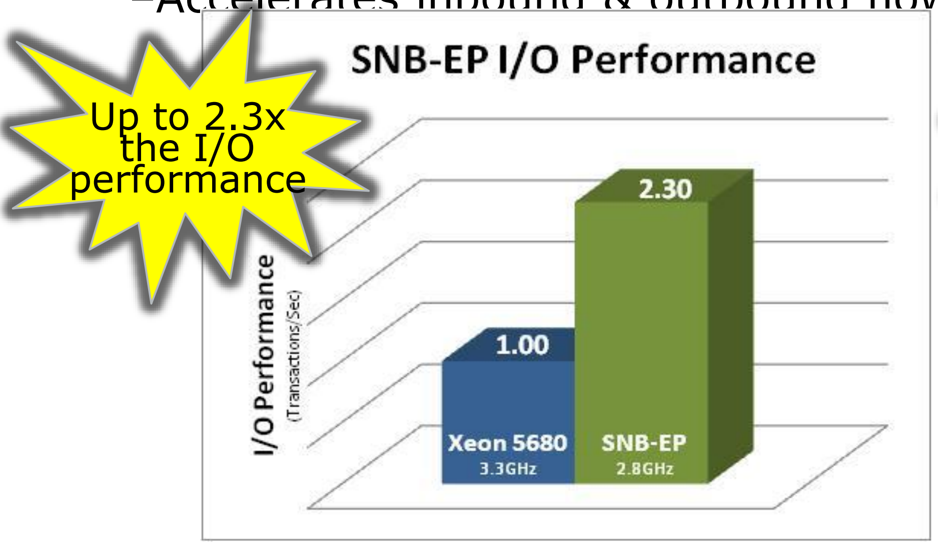
- Optimization of ISV and end user applications (banks, trading and market data firms on pre release & latest hardware: Processors, NICs, Switches etc & Intel's software tools
- Collaborate with OEMs and Industry benchmarks such as STAC T (Tick to Trade Latency), STAC N, STAC A2, STAC M3 etc
- Dissemination of non confidential best practices via http://financialservices.intel.com
- Investigations into Precision Time Protocol (PTP -1588) for clock synch in cluster.
- Recent STAC T (partner with STAC, Corvil, Cisco, Mellanox, Redline ISV, Dell, Datacom) highlighting Dell's DPAT Technology. A 13 % to 42 % reduction in latency over an earlier benchmark.

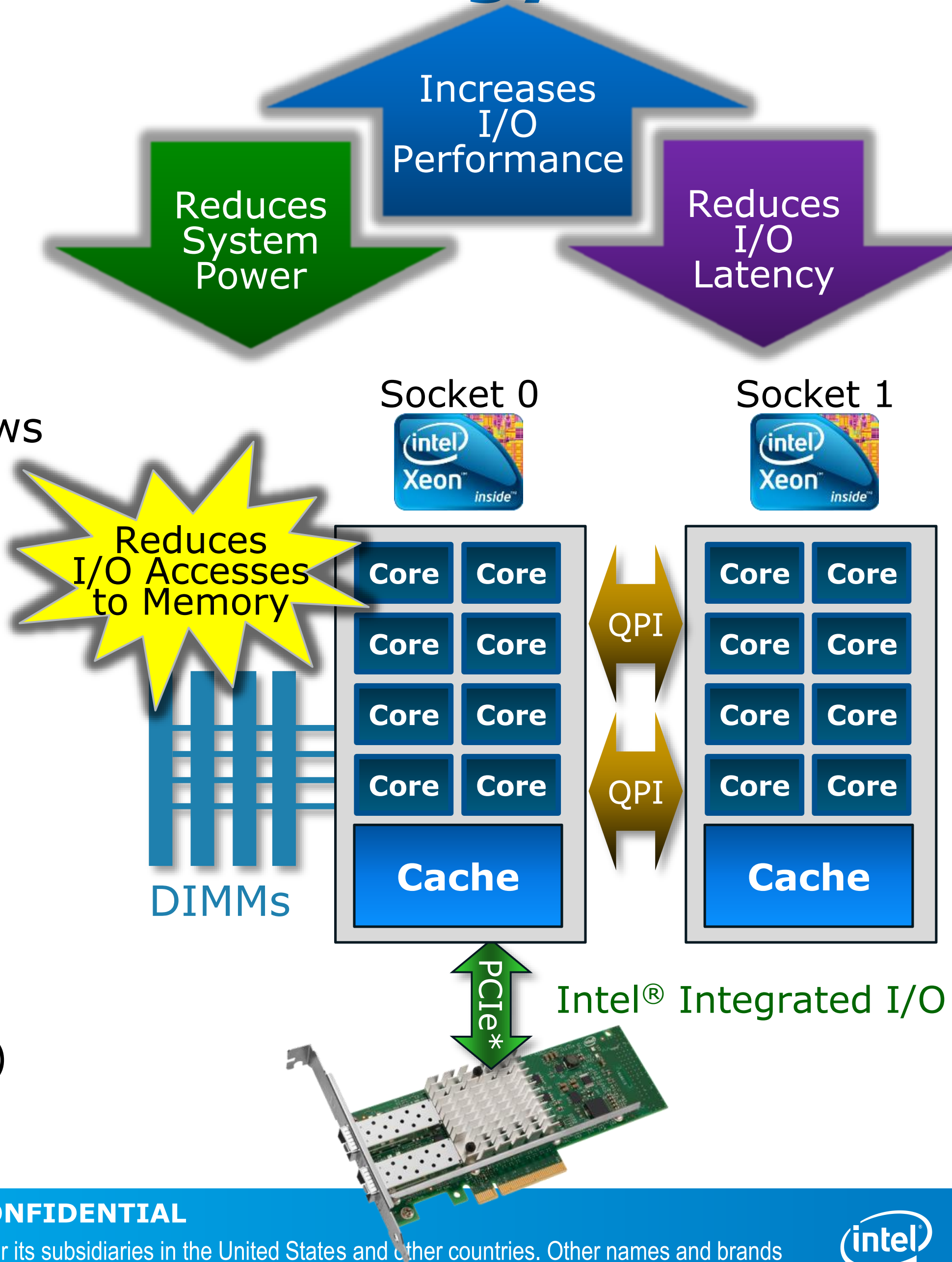
| Market Data | Mean | Median | $99^{th}$ % | Max | Std Dev |
|---|---|---|---|---|---|
| emini 1x | 5.4 | 5.0 | 10.7 | 16.2 | 1.3 |
| emini 8x | 5.2 | 4.8 | 10.5 | 16.8 | 1.2 |

# Backup

# Intel® Data Direct I/O Technology
## (Intel® DDIO)

- New Romley I/O architecture that leverages Intel® Integrated I/O
  - PCIe* lanes on the CPU
- Reduces memory accesses from I/O on local socket
  - Speeds up CPU data transfer
  - Accelerates inbound & outbound flows




- Romley Platform Ingredients
  - Sandy Bridge CPU (Skt-R or Skt-B2)
  - Intel Ethernet, PCIe* I/O devices

# New Instructions in Haswell

**ISA Spec is public http://software.intel.com/en-us/avx/**

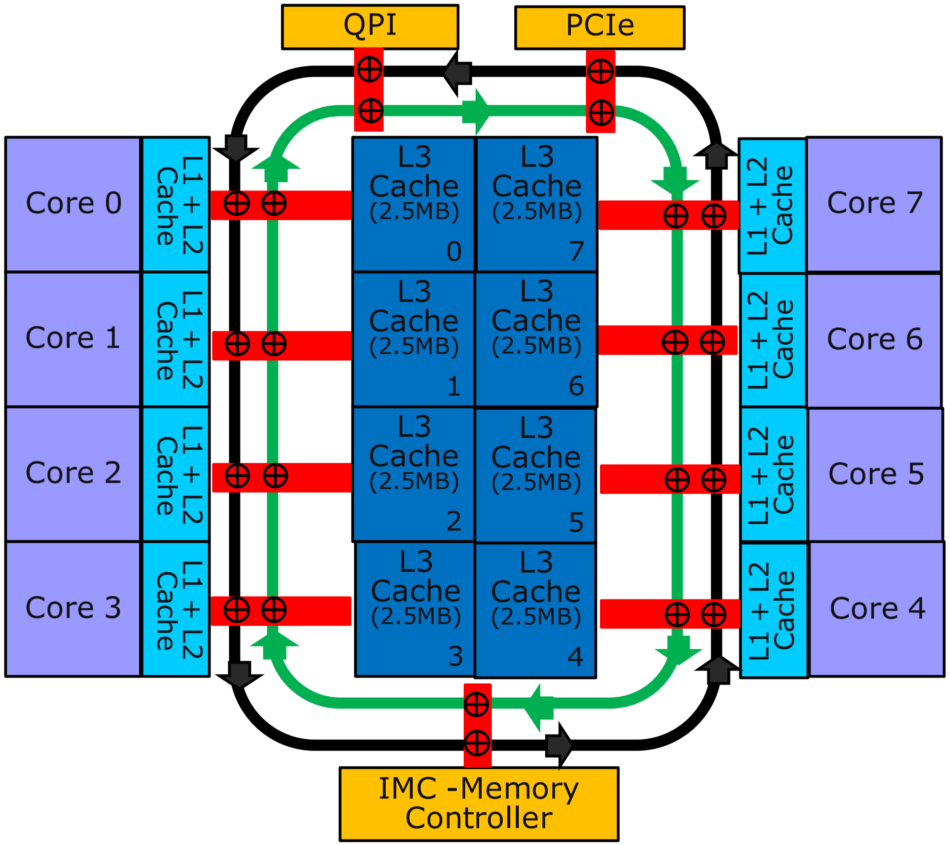
| Group | Description |
|---|---|
| Intel® AVX2 | Adding vector integer operations to 256-bit |
| FMA | Fused Multiply-Add operation forms |
| Gather | Load elements using a vector of indices, vectorization enabler |
| RTM | Restricted Transactional Memory |
| Bit Manipulation and Cryptography | 15 instructions improving performance of bit stream manipulation and decode, large integer arithmetic and hashes |
| MOVBE | Load and Store of Big Endian forms (previously introduced in Atom) |
| INVPCID | Invalidate processor context ID (Ring 0 instruction) |

**This presentation covers vector ISA additions to Haswell; other NIs covered in separate SES sessions**

# Sandy Bridge "Ring Bus" Optimizes Platform Data Movement

- Ring bus architecture delivers an efficient bi-directional highway for data movement

- Compared to Xeon 5500/5600:
  - Lower L3 cache latency (~20%)
  - Higher bandwidth between cores, L3, Memory & I/O
  - Up to 8x more L3 to core bandwidth




**Higher performance starts with The Ring!**