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
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](http://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
  - 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
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!

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
  [sourceware.org/systemtap/](http://sourceware.org/systemtap/)

- VTune can now profile Java code. Consider using hiccups to observe jitter
  [jhiccup.com/](http://jhiccup.com/)

- If business case permits, consider using Intel Xeon PHI for low latency workloads
  [software.intel.com/mic-developer](http://software.intel.com/mic-developer)
VTune™ Amplifier XE 2013 - Power Analysis Functionality
CPU Frequency & Sleep State Analysis - Timeline
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
- SOCHO; DRAMClocks: 665224950; Rank0 CKE Off Residency: 0.00%
- Core Frequency transition count: 0 (useful to study Outliers)

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

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](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.

<table>
<thead>
<tr>
<th>Market Data</th>
<th>Mean</th>
<th>Median</th>
<th>99th %</th>
<th>Max</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>emini 1x</td>
<td>5.4</td>
<td>5.0</td>
<td>10.7</td>
<td>16.2</td>
<td>1.3</td>
</tr>
<tr>
<td>emini 8x</td>
<td>5.2</td>
<td>4.8</td>
<td>10.5</td>
<td>16.8</td>
<td>1.2</td>
</tr>
</tbody>
</table>
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

SNB-EP I/O Performance

Up to 2.3x the I/O performance

Reduces I/O Accesses to Memory

Reduces System Power

Increases I/O Performance

Reduces I/O Latency
New Instructions in Haswell

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

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

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!