INTEL® VTUNE™ AMPLIFIER’S
APPLICATION PERFORMANCE SNAPSHOT:
PERFORMANCE OVERVIEW AT SCALE

Dmitry Prohorov,
VTune HPC Lead
ASPECTS OF HPC/THROUGHPUT APPLICATION PERFORMANCE

Intel Hardware Features

- Omni-Path Architecture
- MCDRAM
- 3D XPoint™
- Many-core Xeon Phi™ Multi-core Xeon™
- AVX-512

Distributed memory
- Message size
- Rank placement
- Rank Imbalance
- RTL Overhead
- Pt2Pt ->collective Ops
- Network Bandwidth

Memory
- False Sharing
- Latency
- Bandwidth
- NUMA

I/O
- File I/O
- I/O latency
- I/O waits
- System-wide I/O

Threading
- Threaded/serial ratio
- Thread Imbalance
- RTL overhead (scheduling, forking)
- Synchronization

CPU Core
- uArch issues (IPC)
- FPU usage efficiency
- Vectorization

Cluster

Node

Core

Cluster

Node

Core
### INTEL TOOLS COVERING THE ASPECTS

#### Intel® VTune™ Amplifier
- **Memory**
  - False sharing
  - Latency
  - Bandwidth
  - NUMA
- **I/O**
  - File I/O
  - I/O latency
  - I/O waits
  - System-wide I/O
- **Threading**
  - Threaded/serial ratio
  - Thread Imbalance
  - RTL overhead (scheduling, forking)
  - Synchronization
  - uArch issues (IPC)
  - FPU usage efficiency
  - Vectorization

#### Intel® VTune™ Amplifier
- **Intel Hardware Features**
  - Omni-Path Architecture
  - MCDRAM
  - XPoint
  - Many-core Processor
  - AVX-512

#### Intel® VTune™ Amplifier
- **Intel® VTune™ Amplifier**
  - Message size
  - Rank placement
  - Rank Imbalance
  - RTL Overhead
  - Pt2Pt ->collective Ops
  - Network Bandwidth

---

**Cluster**

**Node**

**Core**
BEFORE DIVING INTO A PARTICULAR TOOL ...

• How to assess that I have **potential in performance** tuning?
• **Which tool** should I use first?
• What to use on **large scale** avoiding being overwhelmed with huge trace size, post processing time and collection overhead?
• How to **quickly** evaluate environment settings or incremental code changes?
• Answer:
  Use VTune Amplifier’s Application Performance Snapshot 2018
PERFORMANCE OPTIMIZATION WORKFLOW BASED ON APS

Intel® VTune™ Amplifier’s Application Performance Snapshot

- MPI Bound
  - MPI Imbalance
- CPU Bound
  - Memory Bound
  - Thread-level scalability issues (OpenMP analysis)
- Thread-level serial time parallelization
  - FPU underutilization (vector efficiency issues)

CLUSTER
- Intel® Trace Analyzer and Collector
- Intel® MPI Tuner

NODE
- Intel® VTune™ Amplifier

CORE
- Intel® Advisor
  - Threading
  - Vectorization
APPLICATION PERFORMANCE SNAPSHOT AT A GLANCE (1/2)

- High-level **overview** of application performance
  - Detailed reports on MPI statistics
- Primary optimization areas and **next steps** in analysis with deep tools
- **Easy** to install, run, explore results with CL or HTML reports
  - No driver installation required working through perf
  - If SEP driver is available - will be additional advantage
- Multiple methods to obtain
  - Part of Intel® Parallel Studio XE, VTune Amplifier standalone
  - Separate **free** download (110Mb) from APS web page
APPLICATION PERFORMANCE SNAPSHOT AT A GLANCE (2/2)

- **Low** collection overhead – 5-10%
  - HW counters – counting mode only, no overtime
  - MPI and OpenMP tracing - trace aggregation in runtime, no overtime
    - Trace levels to collect more MPI details (potentially for the cost of overhead)
  - Ability to choose either tracing or HW counting in the case of interest in particular metric subset and avoid overhead (--collection-mode option)

- **Scales** to large jobs
  - Tested and worked on 32K ranks
  - Trace size on default statistics level – 4Kb per rank
APS WORKFLOW

Setup Environment
• >source <APS_Install_dir>/apsvars.sh

Run Application
• >aps <application and args>
• MPI: >mpirun <mpi options> aps <application and args>

Generate Report on Result Folder
• >aps –report <result folder>

Generate CL reports with detailed MPI statistics on Result Folder
• aps-report –<option> <result folder>
APS HTML REPORT

Application Performance Snapshot

Report examples (press the links to play): **MPI Bound, Memory Bound, OpenMP imbalance**
APS HTML REPORT BREAKDOWN - OVERVIEW

- Overview shows all areas and relative impact on code performance
- Provides recommendation for next step in performance analysis
- “X” collapses the summary, removing the flags (objective numbers only)

Your application is MPI bound.
This may be caused by high busy wait time inside the library (imbalance), non-optimal communication schema or MPI library settings. Use MPI profiling tools like Intel® Trace Analyzer and Collector to explore performance bottlenecks.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Execution</th>
<th>Impact</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI Time</td>
<td>52.09%</td>
<td>&lt; 10%</td>
<td></td>
</tr>
<tr>
<td>OpenMP Imbalance</td>
<td>0.59%</td>
<td>&lt; 10%</td>
<td></td>
</tr>
<tr>
<td>Memory Stalls</td>
<td>4.48%</td>
<td>&lt; 20%</td>
<td></td>
</tr>
<tr>
<td>FPU Utilization</td>
<td>0.10%</td>
<td>&gt; 50%</td>
<td></td>
</tr>
<tr>
<td>IOC Bound</td>
<td>0.00%</td>
<td>&lt; 10%</td>
<td></td>
</tr>
</tbody>
</table>
APS HTML REPORT BREAKDOWN – PARALLEL RUNTIMES

- **MPI Time**
  - How much time was spent in MPI calls
  - Averaged by ranks with % of Elapsed time
  - Available for MPICH-based MPIs

- **MPI Imbalance**
  - Unproductive time spent in MPI library waiting for data
  - Available for Intel MPI

- **OpenMP Imbalance**
  - Time spent at OpenMP Synchronization Barriers normalized by number of threads
  - Available for Intel OpenMP

- **Serial time**
  - Time spend outside OpenMP regions
  - Available for Intel OpenMP, shared memory applications only

---

**MPI Time**

- 1.33s
- 10.75% of Elapsed Time

**MPI Imbalance**

- 1.13s
- 9.19% of Elapsed Time

**TOP 5 MPI Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waitall</td>
<td>10.24</td>
</tr>
<tr>
<td>Irecv</td>
<td>0.18</td>
</tr>
<tr>
<td>Isend</td>
<td>0.06</td>
</tr>
<tr>
<td>Barrier</td>
<td>0.03</td>
</tr>
<tr>
<td>Reduce</td>
<td>0.02</td>
</tr>
</tbody>
</table>

**OpenMP Imbalance**

- 3.44s
- 42.25% of Elapsed Time

**Serial Time**

- 4.45s
- 31.11% of Elapsed Time
Memory stalls measurement with breakdown by cache and DRAM

Average DRAM Bandwidth*

NUMA ratio

KNL:
  - back-end stalls with L2-demand access efficiency
  - Average DRAM AND MCDRAM Bandwidth*

*Average DRAM and MCDRAM bandwidth collection is available with Intel driver or perf system wide monitoring enabled on a system
APS HTML REPORT BREAKDOWN – VECTORIZATION

- FPU Utilization based on HW-event statistics with
  - Breakdown by vector/scalar instructions
  - Floating point vs memory instruction ratio

- KNL: SIMD Instr. per Cycle
  - Scalar vs. vectorized instructions
Tip:

```bash
>aps --report=<my_result_dir> | grep -v "\"eliminating verbose descriptions"
```
APS COMMAND LINE REPORTS – ADVANCED MPI STATISTICS

aps-report [keys] [options] <result>

[keys] – what to show
--functions
--mpi_time_per_rank
--collop_time_per_rank
--message_sizes
--transfers_per_communication
--transfers_per_rank
--node_to_node
--transfers_per_function
--communicators_list

[options] – how to show
--rank
--comm_id
--details
--communicators
--volume_threshold
--time_threshold
--number_of_lines
--no_filters
--communicators_list
--format

See descriptions with
>aps-report command
**APS COMMAND LINE REPORTS – ADVANCED MPI STATISTICS (1/3)**

**REPORT EXAMPLES**

- MPI Time per rank
  - `aps-report --mpi_time_per_rank <result>`
### Message Sizes summary for all ranks

<table>
<thead>
<tr>
<th>Message size (B)</th>
<th>Volume (MB)</th>
<th>Volume (%)</th>
<th>Transfers</th>
<th>Time (sec)</th>
<th>Time (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1.49</td>
<td>0.09</td>
<td>195206</td>
<td>27.79</td>
<td>37.93</td>
</tr>
<tr>
<td>176</td>
<td>0.41</td>
<td>0.02</td>
<td>2420</td>
<td>27.67</td>
<td>37.78</td>
</tr>
<tr>
<td>4</td>
<td>0.00</td>
<td>0.00</td>
<td>1150</td>
<td>15.55</td>
<td>21.22</td>
</tr>
<tr>
<td>100264</td>
<td>115.89</td>
<td>6.94</td>
<td>1212</td>
<td>0.27</td>
<td>0.37</td>
</tr>
<tr>
<td>98400</td>
<td>113.74</td>
<td>6.81</td>
<td>1212</td>
<td>0.19</td>
<td>0.26</td>
</tr>
<tr>
<td>66256</td>
<td>38.29</td>
<td>2.29</td>
<td>606</td>
<td>0.17</td>
<td>0.23</td>
</tr>
</tbody>
</table>

[filtered out 57 lines]

| TOTAL           | 1670.60     | 100.00     | 265160    | 73.25      | 100.00   |

**aps-report --message_sizes <result>**
### Data Transfers for Rank-to-Rank Communication

```bash
aps-report --transfers_per_communication <result>
```

<table>
<thead>
<tr>
<th>Rank --&gt; Rank</th>
<th>Volume (MB)</th>
<th>Volume(%)</th>
<th>Transfers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0023 --&gt; 0024</td>
<td>54.35</td>
<td>1.56</td>
<td>13477</td>
</tr>
<tr>
<td>0025 --&gt; 0026</td>
<td>54.35</td>
<td>1.56</td>
<td>13477</td>
</tr>
<tr>
<td>0024 --&gt; 0025</td>
<td>54.15</td>
<td>1.56</td>
<td>13477</td>
</tr>
<tr>
<td>0021 --&gt; 0022</td>
<td>53.04</td>
<td>1.55</td>
<td>13477</td>
</tr>
<tr>
<td>0022 --&gt; 0023</td>
<td>53.40</td>
<td>1.54</td>
<td>13477</td>
</tr>
</tbody>
</table>

(filtered out 16 lines)

<table>
<thead>
<tr>
<th>Rank --&gt; Rank</th>
<th>Volume (MB)</th>
<th>Volume(%)</th>
<th>Transfers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0012 --&gt; 0011</td>
<td>69.60</td>
<td>1.29</td>
<td>13477</td>
</tr>
<tr>
<td>0020 --&gt; 0015</td>
<td>69.29</td>
<td>1.26</td>
<td>13477</td>
</tr>
<tr>
<td>0026 --&gt; 0025</td>
<td>68.78</td>
<td>1.27</td>
<td>13477</td>
</tr>
<tr>
<td>0025 --&gt; 0024</td>
<td>68.38</td>
<td>1.27</td>
<td>13477</td>
</tr>
<tr>
<td>0022 --&gt; 0021</td>
<td>68.38</td>
<td>1.27</td>
<td>13477</td>
</tr>
</tbody>
</table>

(filtered out 17 lines)

<table>
<thead>
<tr>
<th>Rank --&gt; Rank</th>
<th>Volume (MB)</th>
<th>Volume(%)</th>
<th>Transfers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0018 --&gt; 0015</td>
<td>58.81</td>
<td>1.09</td>
<td>13477</td>
</tr>
<tr>
<td>0028 --&gt; 0027</td>
<td>57.69</td>
<td>1.07</td>
<td>13477</td>
</tr>
<tr>
<td>0027 --&gt; 0026</td>
<td>56.99</td>
<td>1.05</td>
<td>13477</td>
</tr>
<tr>
<td>0030 --&gt; 0031</td>
<td>54.74</td>
<td>1.01</td>
<td>13477</td>
</tr>
<tr>
<td>0026 --&gt; 0027</td>
<td>54.44</td>
<td>1.01</td>
<td>13477</td>
</tr>
</tbody>
</table>

(filtered out 1108 lines)

| TOTAL | 5403.22 | 100.00 | 1415619 |
| AVG   | 4.67    | 0.09   | 1224    |
COLLECTION CONTROL API

- To measure a particular application phase or exclude initialization/finalization phases use:
  MPI:
  - Pause: MPI_Pcontrol(0)
  - Resume: MPI_Pcontrol(1)

MPI or Shared memory applications:
- Pause: __itt_pause()
- Resume: __itt_resume()
  - See [how to configure](#) the build of your application to use itt API

Tip: use aps “-start-paused” option allows to start application without profiling and skip initialization phase
SUMMARY

Intel® VTune™ Amplifier’s Application Performance Snapshot is:

- Your entry point for HPC application performance analysis
- A part of Parallel Studio XE or easy to install and free standalone package
- Simple and well-structured command line and HTML reports
- Clear next steps for tuning with connection to detailed performance tools
- Tool-of-choice of MPI efficiency analysis at scale
BACK-UP
**Application Performance Snapshot**

**Application**: hart drove
**Report creation date**: 2017-10-16 18:35:56
**Number of nodes**: 1
**Rank per node**: 1
**Dataset threads per rank**: 4
**MPF gather**: yes
**MPF Batch**: no
**Intel Core**: no
**Collective type**: Event list

---

**129.57s**

**Elapsed Time**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOPS</td>
<td>123.56</td>
</tr>
</tbody>
</table>

---

**1.23**

**CPU**

**MAX**: 1.24 **MIN**: 1.21

---

**MPI Time**

- **CPU**: 97.50%
- **User**: 52.06%
- **Elapsed Time**: 99.06%

**MPI Imbalance**

- 0.56%
- 0.66% of Elapsed Time

**Memory Footprint**

- **Resident**: 5952 MB
- **peak**: 7640 MB
- **AVG**: 647 MB
- **Per node**: 5952 MB
- **Per rank**: 2 MB

**OpenMP Imbalance**

- 0.12%

**Memory Stalls**

- 0.03% of cycles

**FPU Utilization**

- 0.10%

---

**Summary**

- **Elapsed Time**: 129.57s
- **FLOPS**: 47.21 Gflops
- **MPI Imbalance**: 0.12%
- **Memory Stalls**: 0.03% of cycles
- **FPU Utilization**: 0.10%

---

**Your application is MPI bound.**

This may be caused by high busy wait time inside the library (imbalance), non-optimal communication scheme or MPI library settings. Use Intel VTune Amplifier and Collector to explore performance bottlenecks.

---

**Intel® VTune™ Amplifier's Application Performance Snapshot**

**MPI Bound MPI Imbalance**

**CPU Bound**

**Thread-level scalabability issues**

**Intel® Advisor**

**Threading**

**Vectorization**

---

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our **Optimization Notice**.
FULL MPI PROFILING VIA INTEL® TRACE ANALYZER AND COLLECTOR

- Compare 2 communication profiles – focus on bottlenecks
- Shows how MPI processes interact

Summary page
Time interval shown
Aggregation of shown data
Tagging & Filtering
Compare
Idealizer
Perf Assistant
Settings
Imbalance Diagram

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.
IMPROVE PERFORMANCE OF MPI + OPENMP APPLICATIONS

ENHANCED MPI METRICS FOR HPC APPLICATION ANALYSIS

Threading: CPU Utilization
- Serial vs. Parallel time
- Top OpenMP regions by potential gain
- Tip: Use hotspot OpenMP region analysis for more detail

Memory Access Efficiency
- Stalls by memory hierarchy
- Bandwidth utilization
- Tip: Use Memory Access analysis

Vectorization: FPU Utilization
- FLOPS† estimates from sampling
- Tip: Use Intel Advisor for precise metrics and vectorization optimization

MPI Imbalance Metric
- Metric for performance of rank on critical path
- Computational bottlenecks and outlier rank behavior now available in VTune Amplifier
- For communication pattern problems between ranks use Intel® Trace Analyzer and Collector (ITAC)

NEW!

Improve Performance

Enhanced MPI metrics for HPC application analysis

Threading: CPU Utilization
- Serial vs. Parallel time
- Top OpenMP regions by potential gain
- Tip: Use hotspot OpenMP region analysis for more detail

Memory Access Efficiency
- Stalls by memory hierarchy
- Bandwidth utilization
- Tip: Use Memory Access analysis

Vectorization: FPU Utilization
- FLOPS† estimates from sampling
- Tip: Use Intel Advisor for precise metrics and vectorization optimization

MPI Imbalance Metric
- Metric for performance of rank on critical path
- Computational bottlenecks and outlier rank behavior now available in VTune Amplifier
- For communication pattern problems between ranks use Intel® Trace Analyzer and Collector (ITAC)
Find Effective Optimization Strategies

Intel Advisor: Cache-aware roofline analysis

Roofs Show Platform Limits
- Memory, cache & compute limits

Dots Are Loops
- Bigger, red dots take more time so optimization has a bigger impact
- Dots farther from a roof have more room for improvement

Higher Dot = Higher GFLOPs/sec
- Optimization moves dots up
- Algorithmic changes move dots horizontally

Which loops should we optimize?
- A and G are the best candidates
- B has room to improve, but will have less impact
- E, C, D, and H are poor candidates

Roofline tutorial video
Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

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.

Copyright © 2017, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

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.

Notice revision #20110804