OPENMP* ANALYSIS IN INTEL® VTUNE™ AMPLIFIER XE

TALKING TO A USER ABOUT OPENMP* PARALLELIZATION EFFICIENCY IN THE LANGUAGE A PROGRAM WAS WRITTEN IN

DMITRY PROHOROV, VTUNE HPC LEAD
Agenda

• VTune Amplifier XE OpenMP* Analysis: answering on customers’ questions about performance in the same language a program was written in
  • Concepts, metrics and technology inside
  • VTune Amplifier XE OpenMP Analysis Workflow
• OpenMP analysis for hybrid MPI + OpenMP applications
• Summary
Typical customer questions on parallelization efficiency of OpenMP* applications

- “I put pragmas but why my speed up is far from linear?
  - Parallelization inefficiency
- “I ran my app on a system with bigger number of cores but it does not run as efficient as on smaller number”
  - Scalability issues

Decomposing the questions

- Is serial time of my application significant to prevent scaling?
- How efficient is my OpenMP parallelization?
  - If not, how much gain can be achieved if I invest in fighting with the inefficiencies?
- What OpenMP regions/loops/barriers are worth to tune?
  - What are the particular problems with them?
If Performance Information is OpenMP “Unaware”..

The questions are tied to OpenMP program structure – #pragmas
Answers should be given the same way to be understandable and actionable
VTune Amplifier XE OpenMP* Analysis: answering on customers’ questions about performance in the same language a program was written in

Overview on summary pane

Is serial time of my application significant to prevent scaling?

How efficient is my parallelization towards ideal parallel execution?

How much theoretical gain I can get if invest in tuning?

What regions are more perspective to invest?

Links to grid view for more details on inefficiency
Key of OpenMP awareness in VTune – Region based views and metrics

Definition of Region **Potential Gain** (elapsed time metric)

Potential Gain as a sum of inefficiencies normalized by num of threads
Technology under VTune Amplifier XE OpenMP Analysis

**Tracing** of OpenMP constructions to provide region/work sharing context and precise imbalance on barriers

- Provided to VTune by Intel OpenMP Runtime under profiling
  - Fork-Join points of parallel regions with number of working threads (Intel Compiler 14 and later)
  - OpenMP construct barrier points with imbalance info and OpenMP loop metadata
  - -parallel-source-info=2 compiler option to embed source file name to a region name
- Looking at transition to OMPT, working with John M.-C. on interface enrichments for low overhead analysis

**Sampling** to define and classify CPU time - user’s code and OpenMP RTL work

- Classification: Locking, Scheduling, Work Forking
VTune Amplifier XE OpenMP Analysis Workflow

Start with HPC Performance Characterization analysis

Explore CPU Utilization aspect metrics related to OpenMP in summary, grid, source view

CL: `>amplxe-cl –collect hpc-performance <my_app>`
Per Region Details in grid view: inefficiencies in wall time - classification and issue highlighting

- Imbalance on a loop barrier
- Dynamic scheduling overhead on a parallel loop
Details in Grid View: **Serial Time Hotspots**

**CPU Utilization:** 25.8%

*Average CPU Usage:* 22.674 Out of 88 logical CPUs

**Top Serial Hotspots (outside parallel regions):**

- **Serial Time (outside parallel regions):** 4.559% (32.11%)

This section lists the loops and functions executed serially in the master thread outside of any OpenMP region and consuming the most CPU time. Improve overall application performance by optimizing or parallelizing these hotspot functions. Since the Serial Time metric includes the Wait time of the master thread, it may significantly exceed the aggregated CPU time in the table.

**Serial hotspots under Master Thread**

1. **Function:** page fault
   - **Module:** vmlinux
   - **Serial CPU Time:** 0.636s

2. **Function:** [Loop at line 152 in miniFE:ig_solve<miniFE:CSRMatrix:double, int, int>, miniFE:Vector<double, int, int>, miniFE:Vector<double, int, int>, miniFE:Vector<double, int, int>]
   - **Module:** miniFE.x
   - **Serial CPU Time:** 0.53s

3. **Function:** pagetblk.pln_to_page
   - **Module:** vmlinux
   - **Serial CPU Time:** 0.486s

4. **Function:** std::local_fib_tree_decrement
   - **Module:** libstdc++.so.6.0.21
   - **Serial CPU Time:** 0.330s

**Time Filter to exclude initialization phase**

Click on the time filter icon to exclude the initialization phase from the analysis.
Details on Scalable **Timeline**

Super tiny timeline display mode – a bird-eye’s view having all data without scrolling

Intel® Xeon Phi™ profiling result with 288 threads

Region frames on the ruler

More green color – more efficient multithreaded execution
Details for a Region at source file level
Summary

• VTune Amplifier XE OpenMP analysis answers on customer’s questions about performance on the language of OpenMP constructs

• The analysis is well-scalable for many-core systems with good balance of tracing and sampling collection technologies

• The OpenMP analysis is “MPI-aware” that is helpful for inner-node hybrid MPI + OpenMP application tuning

• The full feature set is available in VTune Amplifier XE 2018 with Intel OpenMP and Intel MPI runtimes as a part of Intel® Parallel Studio XE 2018
Back-up
A Use Case: NPB CG imbalance improvement

- Step 1: Profiling original application – NPB CG (Class B)

There is a region with promising potential gain – go to Grid View for more details
A Use Case: NPB CG imbalance improvement

• Step 1: Profiling original application – NPB CG (Class B)
  • There are barriers in region – use experimental “/OpenMP Region/OpenMP Barrier..” grouping

• Imbalance on omp loop in cg.f, lines: 572 - 580, schedule is static
A Use Case: NPB CG imbalance improvement

- Step 2: Trying dynamic scheduling omp do schedule (dynamic)

Elapsed time increased – no improvement
Go to Grid View for details
A Use Case: NPB CG imbalance improvement

- **Step 2: Trying dynamic scheduling** “omp do schedule (dynamic)”

Default chunk size is 1 and it led to scheduling overhead
Let's try bigger chunk size
A Use Case: NPB CG imbalance improvement

- Step 3: Trying dynamic schedule with chunk 20

<table>
<thead>
<tr>
<th>OpenMP Region</th>
<th>Potential Gain (%)</th>
<th>Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>con:grad</td>
<td>2.7%</td>
<td>9.568s</td>
</tr>
<tr>
<td>MAIN:omp:parallel:21@</td>
<td>0.9%</td>
<td>0.201s</td>
</tr>
<tr>
<td>MAIN:omp:parallel:22@</td>
<td>0.0%</td>
<td>0.001s</td>
</tr>
<tr>
<td>MAIN:omp:parallel:23@</td>
<td>0.0%</td>
<td>0.001s</td>
</tr>
<tr>
<td>MAIN:omp:parallel:24@</td>
<td>0.0%</td>
<td>0.000s</td>
</tr>
<tr>
<td>[Others]</td>
<td>0.0%</td>
<td>0.000s</td>
</tr>
</tbody>
</table>

Improved original elapsed time ~15%, eliminated imbalance
Back-up
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