www.locuz.com

Profiling MPI Applications

Presented By: Mandeep Kumar

Converge to the Cloud

Scenario

- This recipe uses Intel® VTune™ Profiler to identify imbalances and communications issues in MPI enabled applications, allowing you to improve the application performance.

Ingredients

This section lists the hardware and software tools used for the performance analysis scenario.

- **Application:** heart_demo sample application, available from GitHub at https://github.com/CardiacDemo/Cardiac_demo.git
- **Tools:**
  - Intel® C++ Compiler
  - Intel® MPI Library 2019
  - Intel VTune Profiler 2019
  - VTune Profiler - Application Performance Snapshot
- **Operating system:** Linux
- **CPU:** Intel® Xeon® Platinum 8168 Processor (formerly code named Skylake)
- **Network Fabric:** Intel® Omni-Path Architecture (Intel® OPA)

Build Application

Build your application with debug symbols so VTune Profiler can correlate performance data with your source code and assembly.

1. Clone the application GitHub repository to your local system:

```
git clone https://github.com/CardiacDemo/Cardiac_demo.git
```

2. Set up the Intel C++ Compiler and Intel MPI Library environment:

```
source <compiler_install_dir>/bin/compilervars.sh intel64
source <mpi_install_dir>/bin/mpivars.sh
```

3. In the root level of the sample package, create a build directory and change to that directory:

```
mkdir build
cd build
```

4. Build the application using the following command:

```
    mpiicpc ../heart_demo.cpp ../luo_rudy_1991.cpp ../rcm.cpp ../mesh.cpp -g -o heart_demo -O3
-std=c++11 -qopenmp -parallel-source-info=2
```

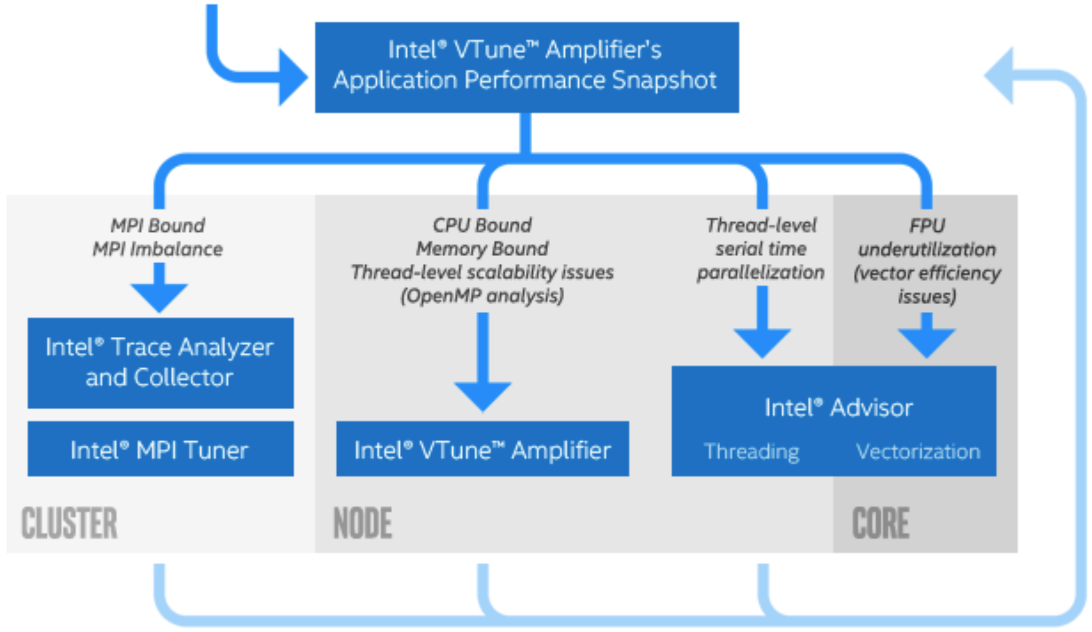
The executable heart_demo should be present in the current directory.

Establish Overall Performance Characteristics

- The recommended application tuning workflow using Intel software tools starts with obtaining a snapshot of your application performance and then focusing on the problematic areas using the most appropriate tool.

- VTune Profiler's Application Performance Snapshot provides overall performance characteristics of an application using a simple interface and a low overhead implementation.

- Use Application Performance Snapshot to gain an understanding of the general properties of your application before moving on to investigate specific issues in detail.


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)
Intel® Trace Analyzer and Collector
Intel® MPI Tuner
Intel® VTune™ Amplifier
Intel® Advisor
Threading
Vectorization
CLUSTER
NODE
CORE

Obtain a Performance Snapshot

Let's obtain a performance snapshot on a set of dual socket nodes using the Intel® Xeon® Scalable processor (code named Skylake). This example uses Intel® Xeon® Platinum 8168 Processor with 24 cores per socket and configures the run to have 4 MPI ranks per node and 12 threads per rank. Modify the specific rank and thread counts in this example to match your own system specification.

- Use the following command line on an interactive session or a batch script to obtain a performance snapshot on four nodes:

```
export OMP_NUM_THREADS=12
mpirun –np 16 –ppn 4 aps ./heart_demo –m ../mesh_mid –s ../setup_mid.txt –t 100
```

Once the analysis is complete, a directory named aps_result_YYYYMMDD is generated with the profiling data, where YYYY the year, MM the month, and DD the day of the collection.

Produce a Single Page HTML Snapshot of the Results

- Run the following command to produce a single page HTML snapshot of the results:

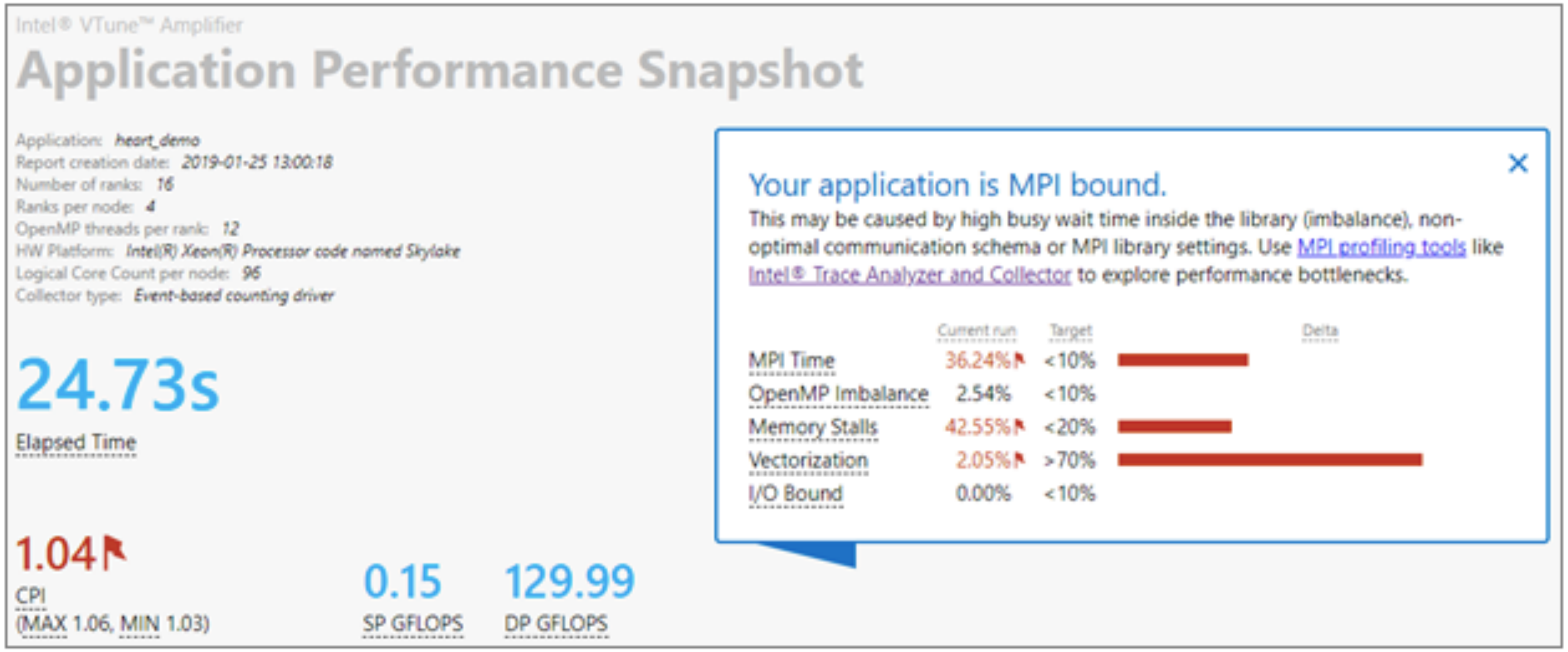
```
aps –report=./aps_result_20210410
```

  The aps_report_YYYYMMDD_<stamp>.html file is created in your working directory, where the <stamp> number is used to prevent overwriting existing reports.

- The report contains information on overall performance, including MPI and OpenMP imbalance, memory footprint and bandwidth utilization, and floating point throughput.

- A note at the top of the report highlights the main areas of concern for the application.

Performance Snapshot



Intel® VTune™ Amplifier

Application Performance Snapshot

Application: *heart_demo*
Report creation date: *2019-01-25 13:00:18*
Number of ranks: *16*
Ranks per node: *4*
OpenMP threads per rank: *12*
HW Platform: *Intel(R) Xeon(R) Processor code named Skylake*
Logical Core Count per node: *96*
Collector type: *Event-based counting driver*

24.73s
Elapsed Time

1.04
CPI
(MAX 1.06, MIN 1.03)

0.15
SP GFLOPS

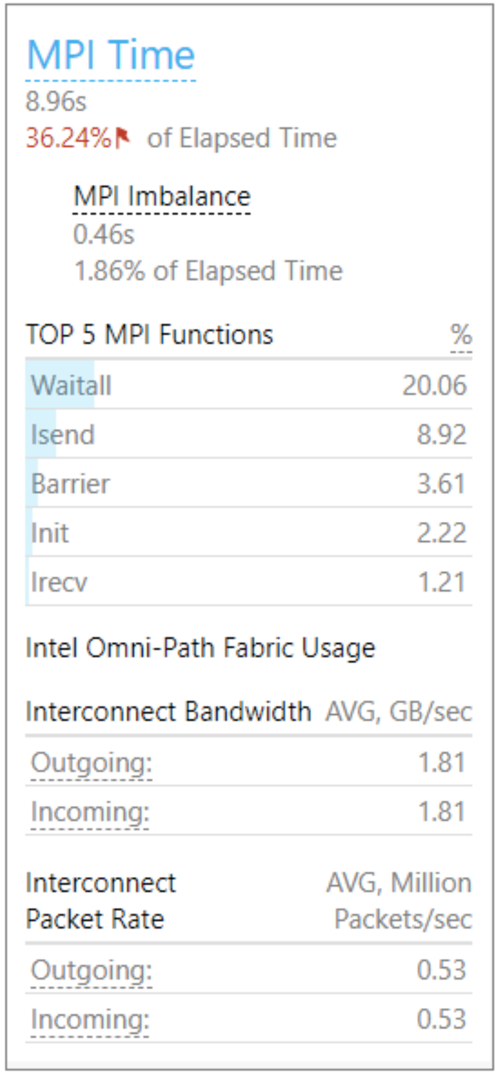
129.99
DP GFLOPS

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.

	Current run	Target	Delta
MPI Time	36.24%	<10%	🟥
OpenMP Imbalance	2.54%	<10%	
Memory Stalls	42.55%	<20%	🟥
Vectorization	2.05%	>70%	🟥
I/O Bound	0.00%	<10%	

The snapshot indicates that overall this application is bound by MPI communication, but that it also suffers from memory and vectorization issues. The **MPI Time** section provides additional details, such as MPI imbalance and the top MPI function calls used. From the additional information it seems that the code uses mainly point to point communication and that the imbalance is moderate.

Performance Snapshot



MPI Time
8.96s
36.24% of Elapsed Time

MPI Imbalance
0.46s
1.86% of Elapsed Time

TOP 5 MPI Functions	%
Waitall	20.06
Isend	8.92
Barrier	3.61
Init	2.22
Irecv	1.21

Intel Omni-Path Fabric Usage

Interconnect Bandwidth	AVG, GB/sec
Outgoing:	1.81
Incoming:	1.81

Interconnect Packet Rate	AVG, Million Packets/sec
Outgoing:	0.53
Incoming:	0.53

This snapshot result points to complex issues in the code. To continue investigating the performance issues and isolate the problems, use the HPC Performance Characterization analysis feature of VTune Profiler.

Configure and Run the HPC Performance Characterization Analysis

Follow these steps to run the HPC Performance Characterization analysis from the command line:

1. Prepare your environment by sourcing the relevant VTune Profiler files. For a default installation using the bash shell, use the following command:

```
source /opt/intel/vtune_Profiler/amplxe-vars.sh
```

2. Collect data for the heart_demo application using the hpc-performance analysis. The application uses both OpenMP and MPI and will be executed using the configuration described earlier, with 16 MPI ranks over a total of 4 compute nodes using the Intel MPI Library. This example is run on Intel® Xeon® Platinum 8168 Processors and uses 12 OpenMP threads per MPI rank:

```
export OMP_NUM_THREADS=12
mpirun –np 16 –ppn 4 amplxe-cl –collect hpc-performance –r vtune_mpi -- ./heart_demo –m ../
mesh_mid –s ../setup_mid.txt –t 100
```

The analysis begins and generates four output directories using the following naming convention: **vtune_mpi.*<node host name>***.

Analyze Results using the VTune Profiler GUI

- The VTune Profiler graphical interface provides a much richer and more interactive experience than the command line for analyzing the collected performance data.

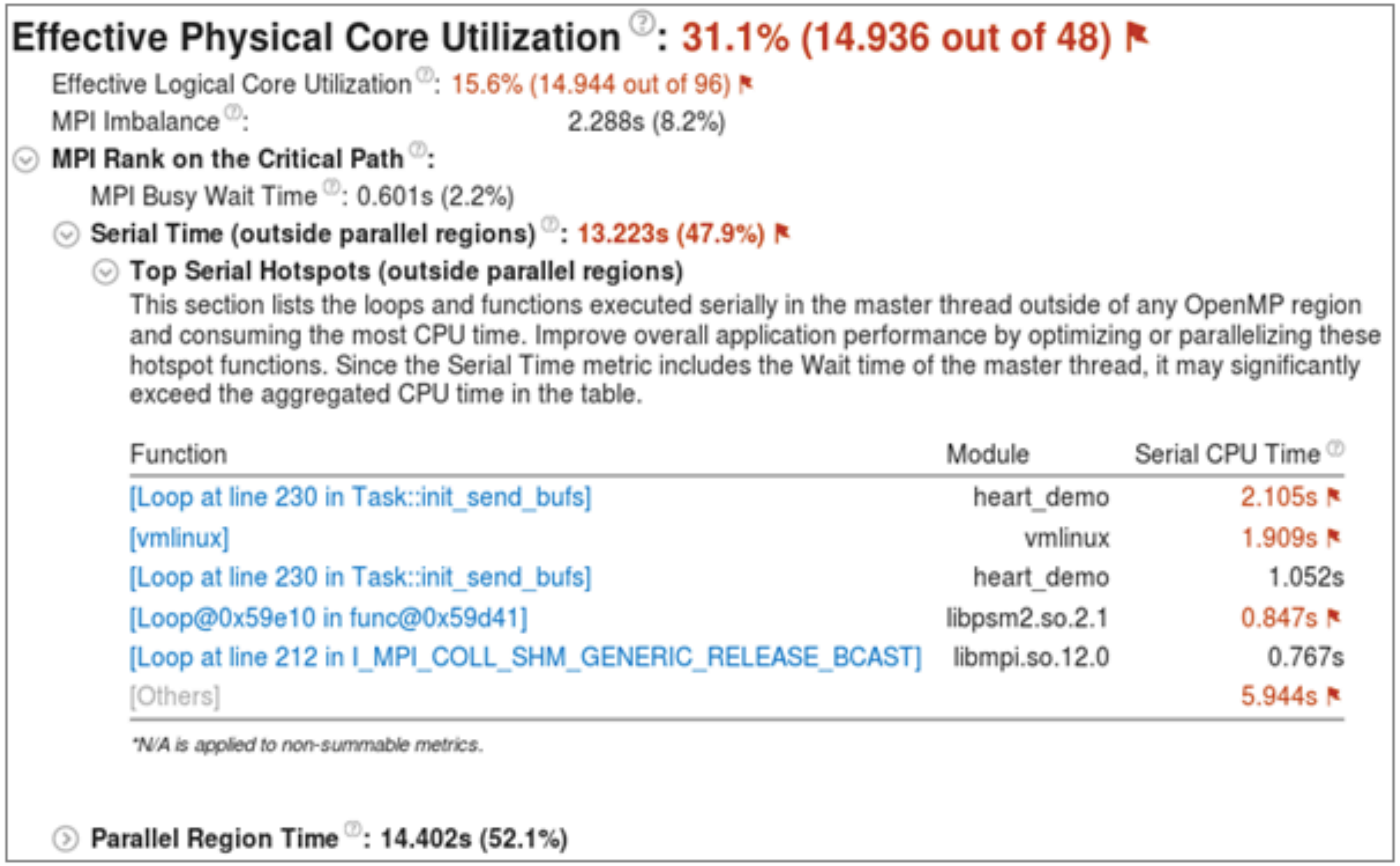
- Start by running the following command to open one of the results in the VTune Profiler user interface:

```
amplxe-gui ./vtune_mpi.node_1
```

Result opens in VTune Amplifer

- The result opens in VTune Amplifer and shows the **Summary** window, which provides an overview of the application performance.
- Because heart_demo is an MPI parallel application, the **Summary** window shows MPI Imbalance information and details regarding the MPI rank in the execution critical path in addition to the usual metrics.
    - **MPI Imbalance** is an average MPI busy wait time by all ranks on the node. The value indicates how much time could be saved if balance was ideal.
    - **MPI Rank on the Critical Path** is the rank with minimal busy wait time.
    - **MPI Busy Wait Time** and **Top Serial Hotspots** are shown for the rank in the critical path. This is a good way to identify severe deficiencies in scalability since they typically correlate with high imbalance or busy wait metrics. Significant **MPI Busy Wait Time** for the rank on the critical path in a multi-node run might mean that the outlier rank is on a different node.

Summary Window



Effective Physical Core Utilization: 31.1% (14.936 out of 48)

Effective Logical Core Utilization: 15.6% (14.944 out of 96)

MPI Imbalance: 2.288s (8.2%)

MPI Rank on the Critical Path:

MPI Busy Wait Time: 0.601s (2.2%)

Serial Time (outside parallel regions): 13.223s (47.9%)

Top Serial Hotspots (outside parallel regions)

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.

Function	Module	Serial CPU Time
[Loop at line 230 in Task::init_send_bufs]	heart_demo	2.105s
[vmlinux]	vmlinux	1.909s
[Loop at line 230 in Task::init_send_bufs]	heart_demo	1.052s
[Loop@0x59e10 in func@0x59d41]	libpsm2.so.2.1	0.847s
[Loop at line 212 in I_MPI_COLL_SHM_GENERIC_RELEASE_BCAST]	libmpi.so.12.0	0.767s
[Others]		5.944s

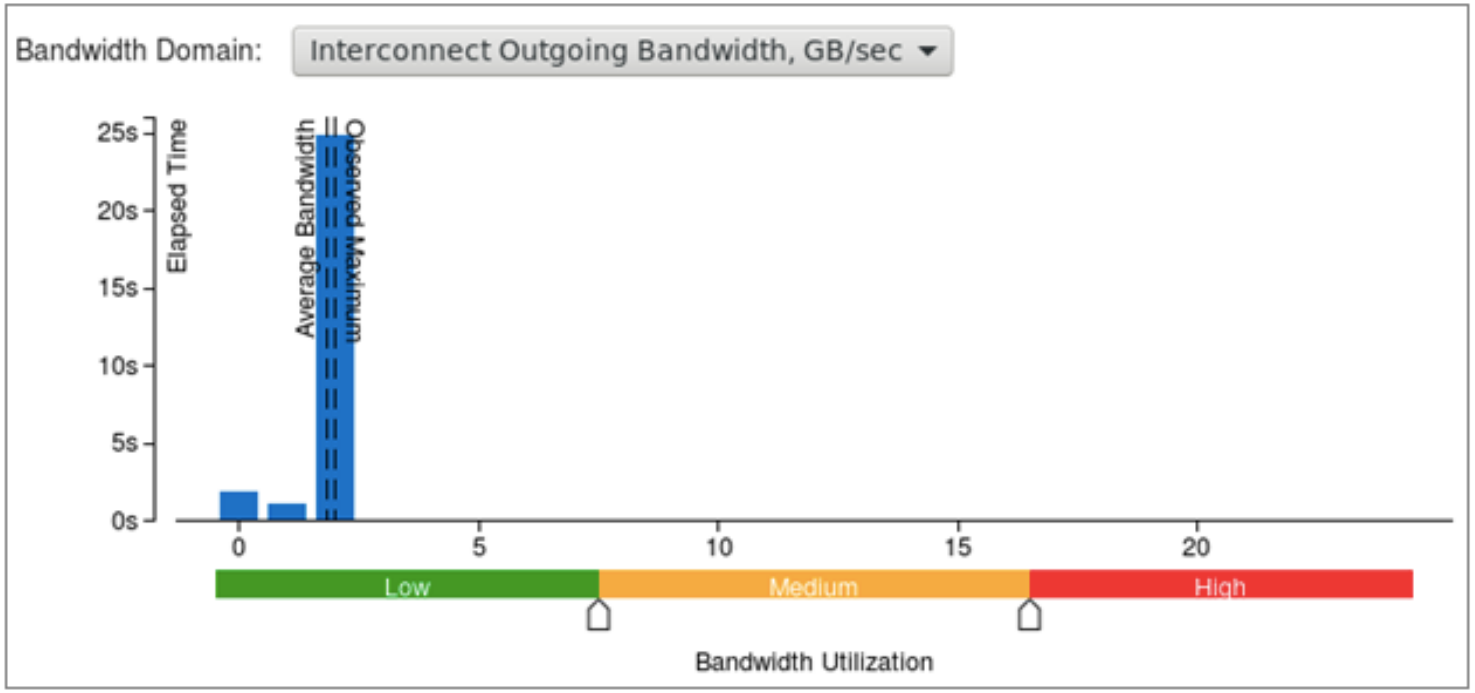
**N/A is applied to non-summable metrics.*

Parallel Region Time: 14.402s (52.1%)

In our example, there is some imbalance and also a significant amount of time spent in serial regions of the code (not shown in the figure).

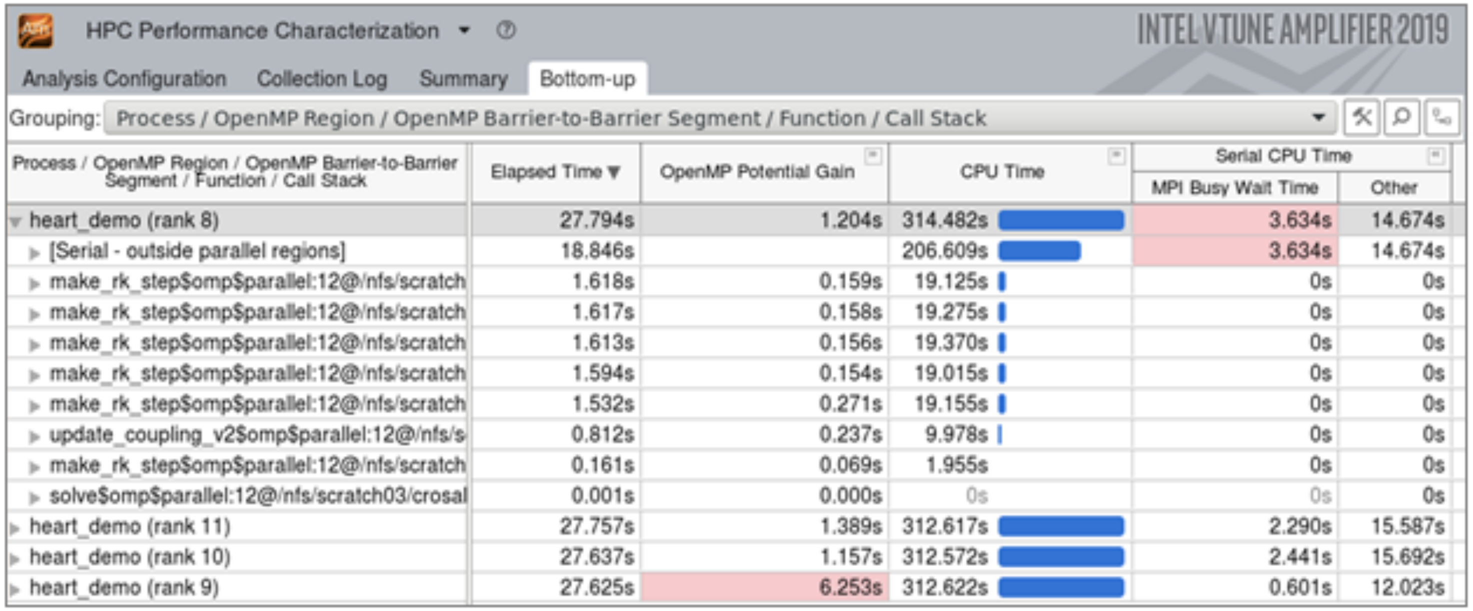
Summary Window

In Intel VTune Amplifer 2019 and higher the **Summary** window contains histograms of Intel® Omni-Path Architecture (Intel® OPA) fabric utilization. The metrics show bandwidth and packet rate and indicate what percentage of the execution time the code was bound by high bandwidth or packet rate utilization. VTune Profiler reports that the heart_demo application spends no time being bandwidth or packet rate bound, but the histogram shows a maximum bandwidth utilization of 1.8 GB/s that is close to the average. This hints to continuous, but inefficient, use of MPI communications.


Bandwidth Domain: Interconnect Outgoing Bandwidth, GB/sec

Summary Window

Switch to the **Bottom-up** tab to get more details. Set the **Grouping** to have Process at the top level to get the following view:



HPC Performance Characterization

INTEL VTUNE AMPLIFIER 2019

Analysis Configuration | Collection Log | Summary | Bottom-up

Grouping: Process / OpenMP Region / OpenMP Barrier-to-Barrier Segment / Function / Call Stack

Process / OpenMP Region / OpenMP Barrier-to-Barrier Segment / Function / Call Stack	Elapsed Time ▼	OpenMP Potential Gain	CPU Time	Serial CPU Time	
				MPI Busy Wait Time	Other
heart_demo (rank 8)	27.794s	1.204s	314.482s	3.634s	14.674s
[Serial - outside parallel regions]	18.846s		206.609s	3.634s	14.674s
make_rk_step$omp$parallel:12@/nfs/scratch	1.618s	0.159s	19.125s	0s	0s
make_rk_step$omp$parallel:12@/nfs/scratch	1.617s	0.158s	19.275s	0s	0s
make_rk_step$omp$parallel:12@/nfs/scratch	1.613s	0.156s	19.370s	0s	0s
make_rk_step$omp$parallel:12@/nfs/scratch	1.594s	0.154s	19.015s	0s	0s
make_rk_step$omp$parallel:12@/nfs/scratch	1.532s	0.271s	19.155s	0s	0s
update_coupling_v2$omp$parallel:12@/nfs/s	0.812s	0.237s	9.978s	0s	0s
make_rk_step$omp$parallel:12@/nfs/scratch	0.161s	0.069s	1.955s	0s	0s
solve$omp$parallel:12@/nfs/scratch03/crosal	0.001s	0.000s	0s	0s	0s
heart_demo (rank 11)	27.757s	1.389s	312.617s	2.290s	15.587s
heart_demo (rank 10)	27.637s	1.157s	312.572s	2.441s	15.692s
heart_demo (rank 9)	27.625s	6.253s	312.622s	0.601s	12.023s

Since this code uses both MPI and OpenMP, the Bottom-up window shows metrics related to both runtimes besides the usual CPU and memory data. In our example, the MPI Busy Wait Time metric is shown in red for the MPI rank furthest from the critical path, which also shows the largest OpenMP potential gain. This hints that threading improvements could help performance.

Analyze Results with a Command Line Report

- VTune Profiler provides informative command line text reports. For example, to obtain a summary report, run the following command:

```
amplxe-cl -report summary -r ./results_dir
```

- A summary of the results prints to the screen. Options to save the output directly to file and in other formats (csv, xml, html) are available, among many others.

```
amplxe-cl -report summary -r ./results_dir -format=html -report-output vtune_summary.html
```

- For details on the full command line options type **amplxe-cl -help**

Thanks!

LOCUZ