

Cross-Architecture Programming for Accelerated Compute; Freedom of Choice for Hardware

Intel® Software Development Tools for HPC: Programming for Distributed HPC Systems using Intel® MPI Library

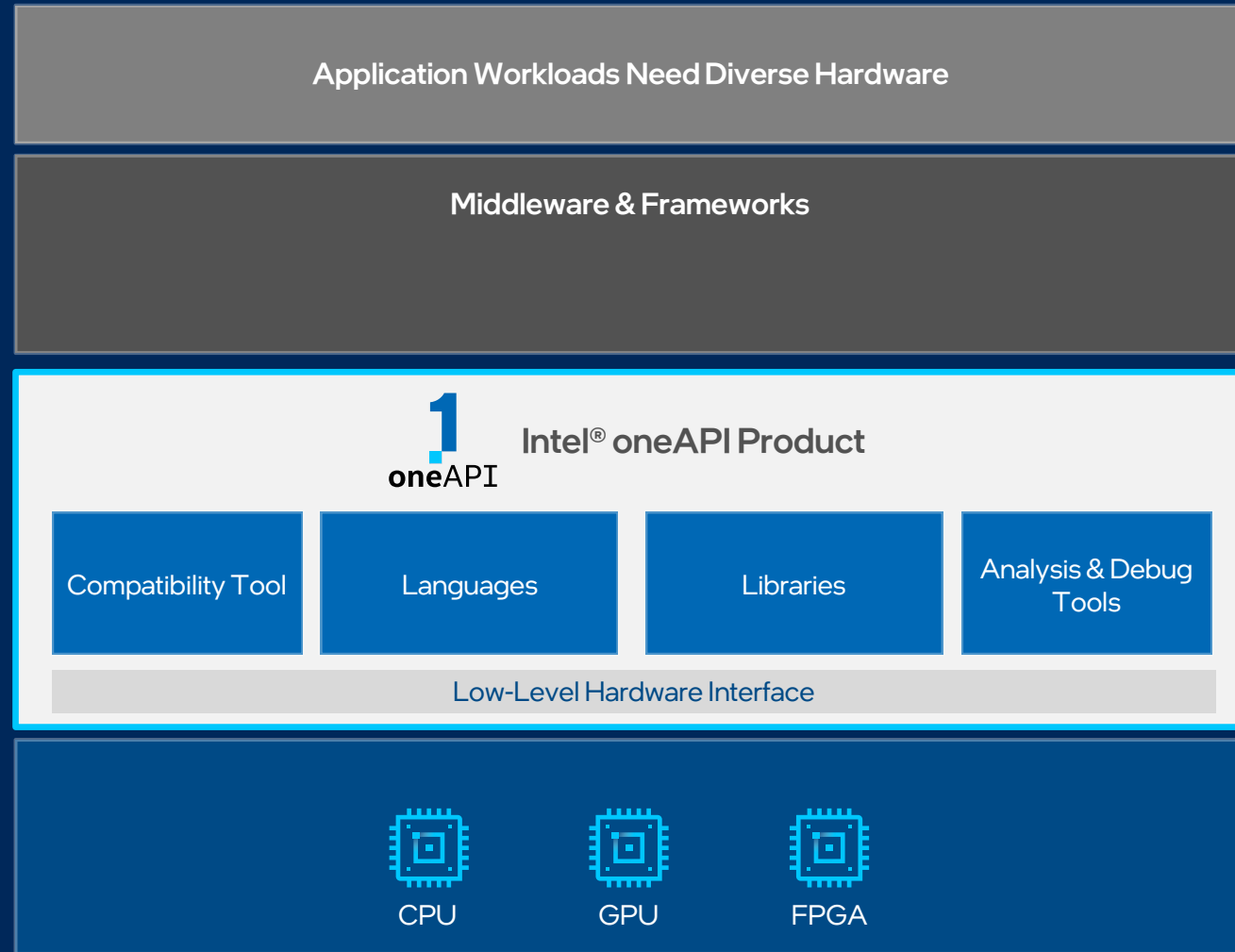


Intel® oneAPI Tools

Built on Intel's Rich Foundation of CPU Tools Expanded to Accelerators

A complete set of advanced compilers, libraries, and porting, analysis and debugger tools

- Accelerates compute by exploiting cutting-edge hardware features
- Interoperable with existing programming models and code bases (C++, Fortran, Python, OpenMP, etc.), developers can be confident that existing applications work seamlessly with oneAPI
- Eases transitions to new systems and accelerators—using a single code base frees developers to invest more time on innovation



[Available Now](#)

Intel® oneAPI Toolkits

A complete set of proven developer tools expanded from CPU to Accelerators



Intel® oneAPI Base Toolkit

A core set of high-performance libraries and tools for building C++, SYCL and Python applications

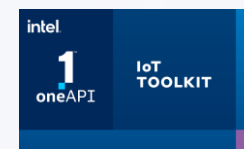


Add-on Domain-specific Toolkits



Intel® oneAPI Tools for HPC

Deliver fast Fortran, OpenMP & MPI applications that scale



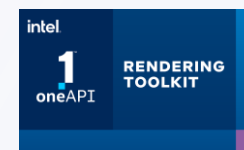
Intel® oneAPI Tools for IoT

Build efficient, reliable solutions that run at network's edge



Intel® oneAPI AI Analytics Toolkit

Accelerate machine learning & data science pipelines with optimized DL frameworks & high-performing Python libraries



Intel® oneAPI Rendering Toolkit

Create performant, high-fidelity visualization applications

Toolkit
powered by oneAPI



Intel® Distribution of OpenVINO™ Toolkit

Deploy high performance inference & applications from edge to cloud

Intel® oneAPI Tools for HPC

Intel® oneAPI HPC Toolkit

Deliver Fast Applications that Scale

What is it?

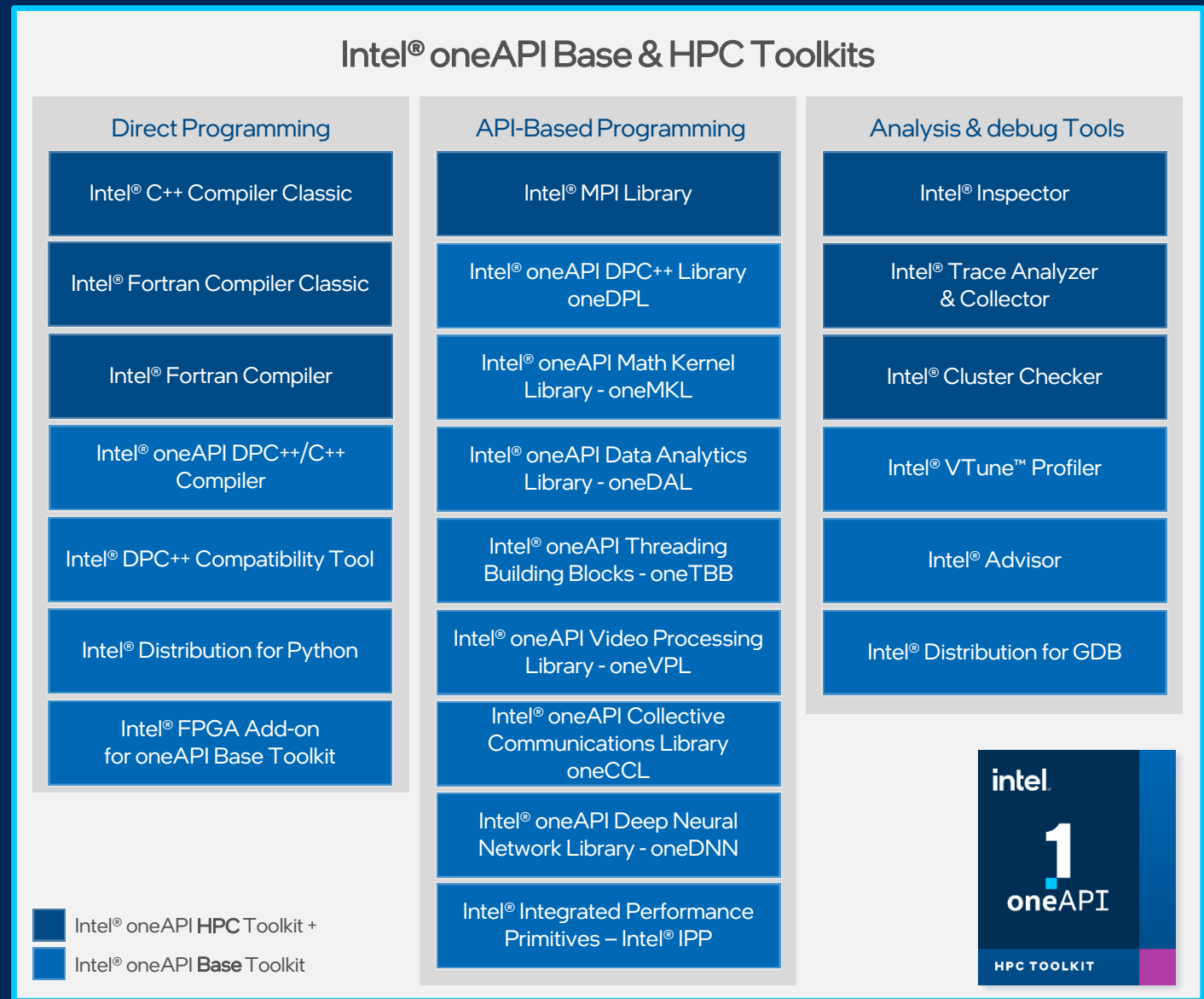
A toolkit that adds to the Intel® oneAPI Base Toolkit for building high-performance, scalable parallel code on C++, SYCL, Fortran, OpenMP & MPI from enterprise to cloud, and HPC to AI applications.

Who needs this product?

- OEMs/ISVs
- C++, Fortran, OpenMP, MPI Developers

Why is this important?

- Accelerate performance on Intel® Xeon® and Core™ Processors and Intel® Accelerators
- Deliver fast, scalable, reliable parallel code with less effort built on industry standards



Intel® MPI Library

Deliver Flexible, efficient, and Scalable Cluster Messaging

Optimized MPI Application Performance

- Application-specific tuning
- Automatic tuning
- Support for latest Intel® Xeon® Scalable Processors

Lower Latency and Multi-vendor Interoperability

- Industry-leading latency
- Performance-optimized support for the fabric capabilities through OpenFabrics Interfaces (OFI)

Faster MPI Communication

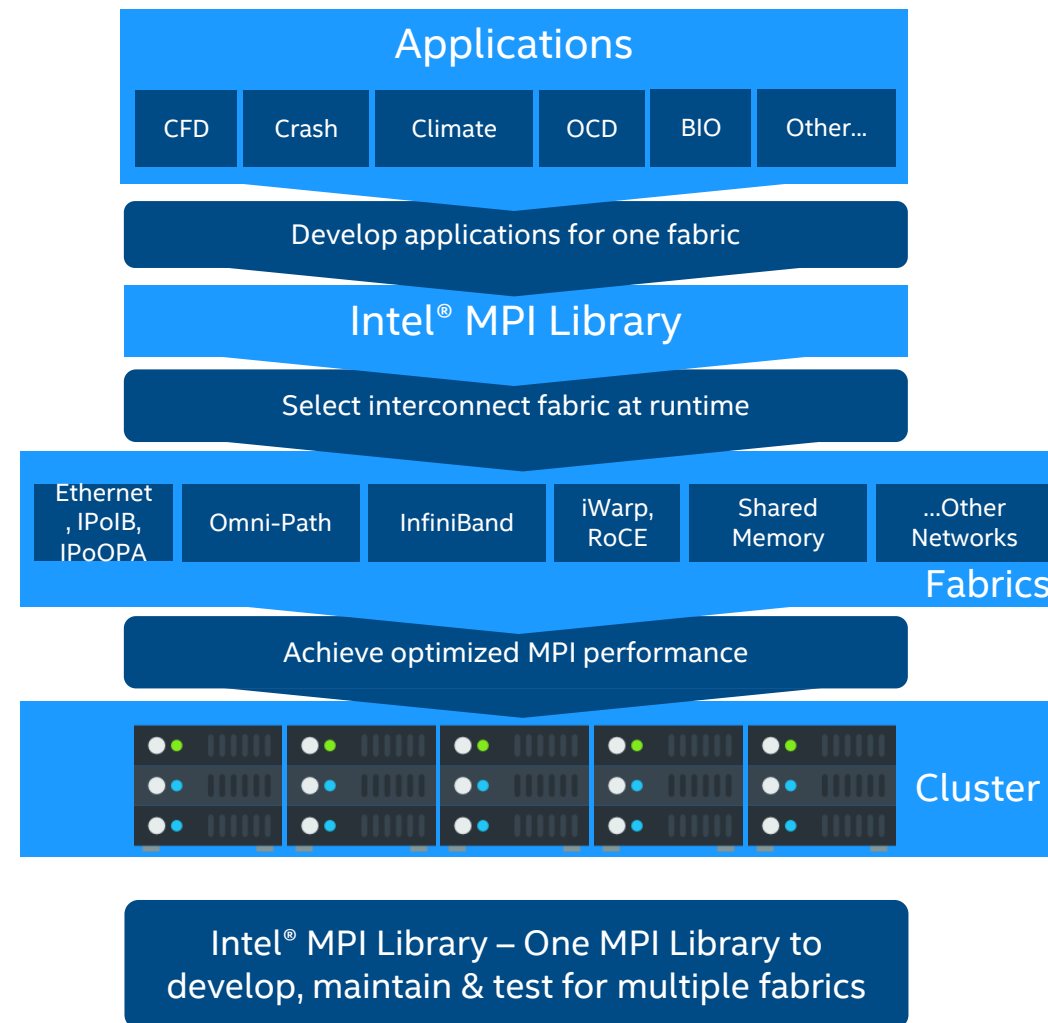
- Optimized collectives

Sustainable scalability

- Native InfiniBand interface support allows for lower latencies, higher bandwidth, and reduced memory requirements

Key Updates

- Intel® GPU pinning support
- Distributed Asynchronous Object Storage (DAOS) support
- Intel® Xeon® Platinum processor 92XX optimizations
- Mellanox ConnectX: 3/4/5/6 (FDR/EDR/HDR) support enhancements



Intel® MPI Library version 2021

Key new features

HPC in cloud

- Google Cloud Platform* (GCP) and Amazon Web Services*(AWS) integrated support

Latest Hardware support

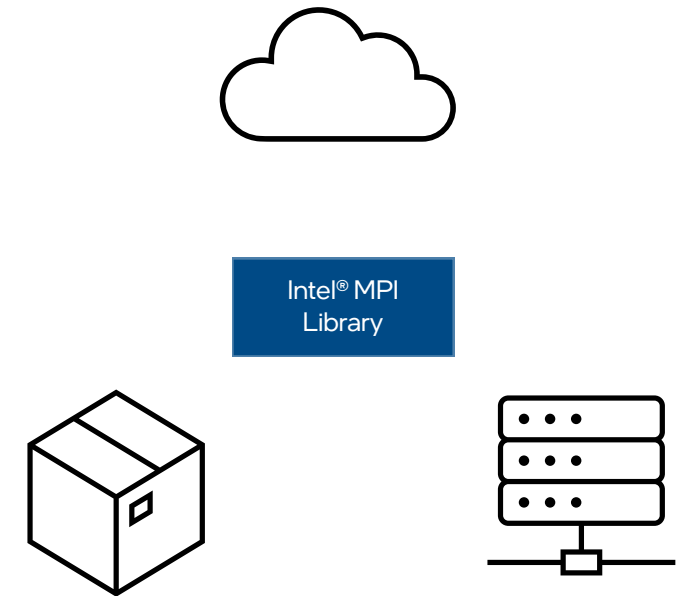
- Intel® Xeon® Scalable 3rd gen processors
- Performance on Intel® Ethernet 800 Series Network Adapters
- Mellanox* ConnectX*: 3/4/5/6 (FDR/EDR/HDR) support enhancements
- Intel GPUs support

Sustainable scalability

- Improved startup time
- Performance and stability improvements for OFI providers
- Spawn improvements

New technology

- Distributed Asynchronous Object Storage (DAOS) support
- Extended Singularity support for IBM* Spectrum* LSF*, SLURM



GPU Programming model

- MPI primitives can be used for communications between GPUs
- Host centric MPI with GPU awareness (current):
 - Host initiated communication:
 - Pipelining or dma_buf (direct GPU memory access)
 - Local GPU to GPU communication
 - GPU kernels for collective operations (MPI_Allreduce, MPI_Reduce)
- GPU centric MPI (in progress):
 - GPU kernel-initiated communication

Intel® MPI Simplifies Programming for GPUs*

- Multiple GPUs with multiple Tiles require manual domain decomposition or implicit scaling mechanism which may not be always desirable
- Intel MPI enables single GPU / Tile programming while leaving distribution to the library
- Intel MPI allows to assign GPUs / Tiles to individual MPI ranks
- Intel MPI allows users to pass GPU memory pointers to the library

Intel® MPI GPU Pinning

Intel GPU pinning support

Intel GPU resources usage with Intel MPI Library

- Automatic Intel GPU resources distribution
- No user code changes required on different GPU configurations
- NUMA aware

Default settings:

I_MPI_OFFLOAD_* family:

TOPOLIB=level_zero

CELL=tile

DOMAIN_SIZE=-1 (auto)

DEVICES=all

E.g: I_MPI_OFFLOAD_TOPOLIB=level_zero

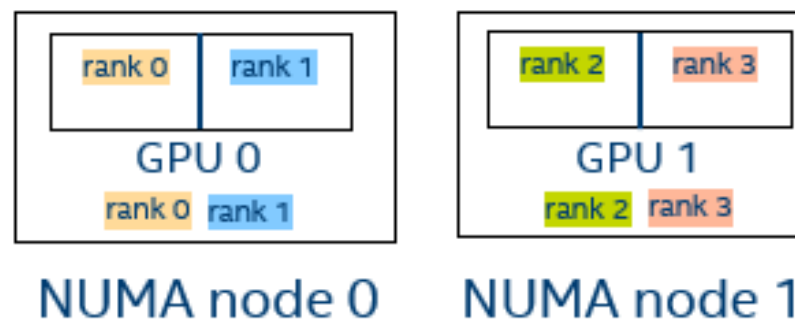
Example – Default: 4 ranks

I_MPI_DEBUG=120

```
[0] MPI startup(): ===== GPU topology on host1 =====  
[0] MPI startup():  NUMA nodes : 2  
[0] MPI startup():  GPUs          : 2  
[0] MPI startup():  Tiles          : 4  
[0] MPI startup():  ===== GPU Placement on packages =====  
[0] MPI startup():  NUMA Id   GPU Id   Tiles      Ranks  
[0] MPI startup():  0         0       (0,1)      0,1  
[0] MPI startup():  1         1       (2,3)      2,3
```

I_MPI_DEBUG=3

```
[0] MPI startup(): ===== GPU pinning on host1 =====  
[0] MPI startup(): Rank Pin tile  
[0] MPI startup():  0     {0}  
[0] MPI startup():  1     {1}  
[0] MPI startup():  2     {2}  
[0] MPI startup():  3     {3}
```



Example – Default: 3 ranks

`I_MPI_DEBUG=3`

`I_MPI_OFFLOAD_DOMAIN_SIZE=-1 (auto) [default]`

[0] MPI startup(): ===== GPU pinning on host1 =====

[0] MPI startup(): Rank Pin tile

[0] MPI startup(): 0 {0}

[0] MPI startup(): 1 {1}

[0] MPI startup(): 2 {2,3}

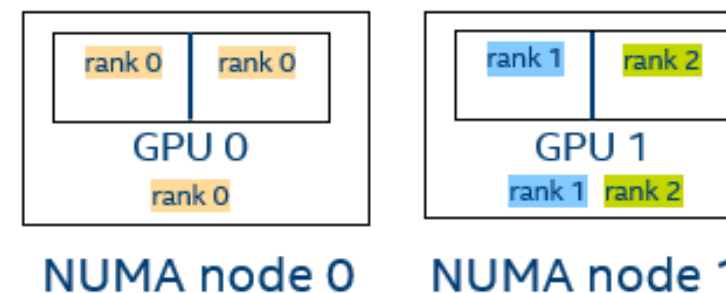
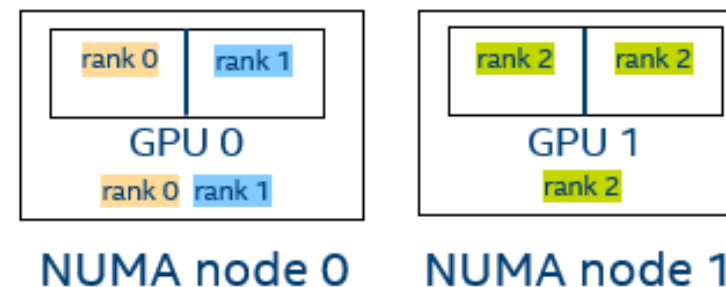
[0] MPI startup(): ===== GPU pinning on host1 =====

[0] MPI startup(): Rank Pin tile

[0] MPI startup(): 0 {0,1}

[0] MPI startup(): 1 {2}

[0] MPI startup(): 2 {3}



Example – I_MPI_OFFLOAD_CELL

I_MPI_OFFLOAD_CELL=device

I_MPI_DEBUG=3

[0] MPI startup(): ===== GPU pinning on host1 =====

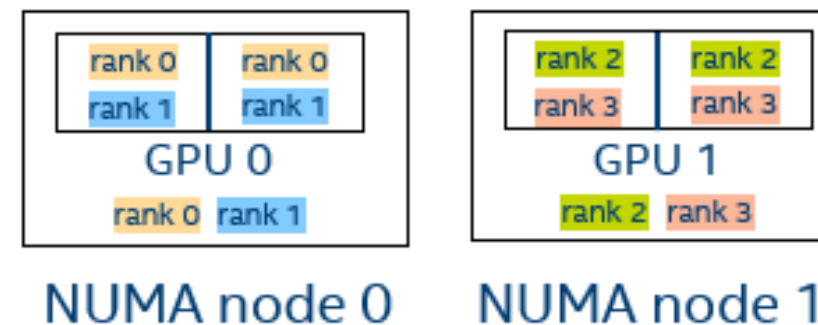
[0] MPI startup(): Rank Pin tile

[0] MPI startup(): 0 {0,1}

[0] MPI startup(): 1 {0,1}

[0] MPI startup(): 2 {2,3}

[0] MPI startup(): 3 {2,3}



Example – I_MPI_OFFLOAD_DOMAIN_SIZE

I_MPI_OFFLOAD_DOMAIN_SIZE=1

I_MPI_DEBUG=3

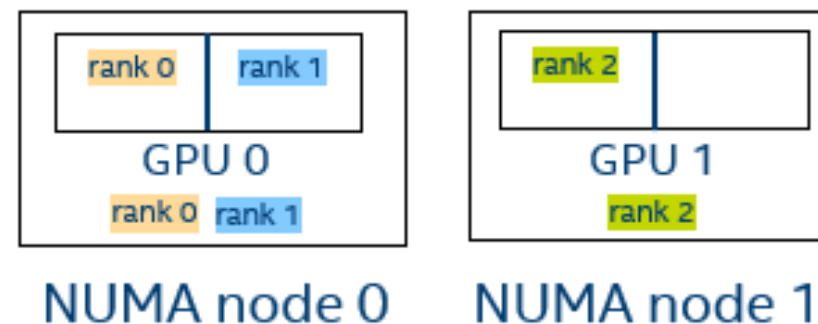
[0] MPI startup(): ===== GPU pinning on host1 =====

[0] MPI startup(): Rank Pin tile

[0] MPI startup(): 0 {0}

[0] MPI startup(): 1 {1}

[0] MPI startup(): 2 {2}



Example – I_MPI_OFFLOAD_DEVICES

I_MPI_OFFLOAD_DEVICES=0

I_MPI_DEBUG=3

[0] MPI startup(): ===== GPU pinning on host1 =====

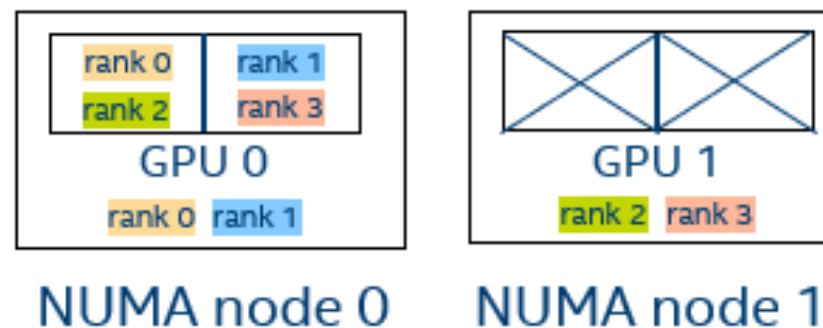
[0] MPI startup(): Rank Pin tile

[0] MPI startup(): 0 {0}

[0] MPI startup(): 1 {1}

[0] MPI startup(): 2 {0}

[0] MPI startup(): 3 {1}



Example – I_MPI_OFFLOAD_DOMAIN

I_MPI_OFFLOAD_DOMAIN=[B,2,5,C]

reverse bit masks:
[1101,0100,1010,0011]

I_MPI_DEBUG=3

[0] MPI startup(): ===== GPU pinning on host1 =====

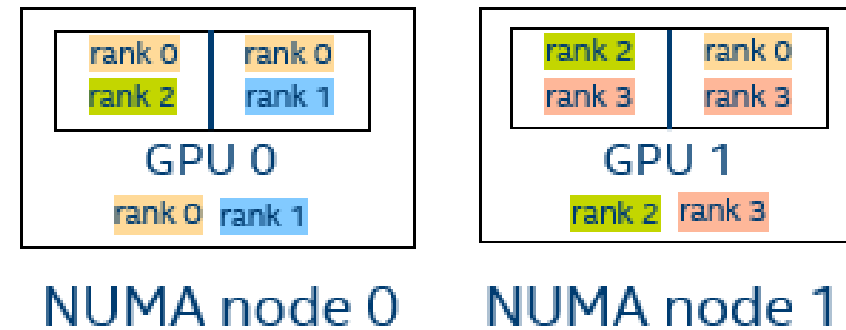
[0] MPI startup(): Rank Pin tile

[0] MPI startup(): 0 {0,1,3}

[0] MPI startup(): 1 {1}

[0] MPI startup(): 2 {0,2}

[0] MPI startup(): 3 {2,3}



Intel® MPI GPU Buffers Support

Explicitly enable Offloading Library

Syntax

`I_MPI_OFFLOAD=<value>`

Arguments

- 0 - Disabled (default value)
- 1 - Auto, expects that libze_loader.so is already loaded
- 2 - Enabled. Intel MPI loads libze_loader.so

Execution models

1. OpenMP using map(tofrom) clause

```
// Copy data from host to device
#pragma omp target data map(to: rank, num_values) map(tofrom: values[0:num_values])
{
    ... // Compute on GPU
    #pragma omp target parallel for
    for (i = 0; i < num_values; i++) {
        values[i] = values[i] + rank + 1;
    }
}

// Communicate on CPU: Send buffer to rank 0 after copy back from GPU to host
MPI_Send(values, num_values, MPI_INT, dest_rank, tag, MPI_COMM_WORLD);
```

Execution models

2. GPU buffer aware (w/ openMP using pointers)

```
// Copy data from host to device
#pragma omp target data map(to: rank, num_values, values[0:num_values]) use_device_ptr(values)
{
    ... // Compute on GPU
    #pragma omp target parallel for is_device_ptr(values)
    for (i = 0; i < num_values; i++) {
        values[i] = values[i] + rank + 1;
    }

    ... // Communicate on CPU: Send buffer to rank 0 without copy back from GPU
    MPI_Send(values, num_values, MPI_INT, dest_rank, tag, MPI_COMM_WORLD);
}
```

Execution models

▪ Build

- C language:

```
$ mpiicc -cc=icx -fiopenmp -fopenmp-targets=spir64 test.c -o test
```

- Fortran language:

```
$ mpiifort -fc=ifx -fiopenmp -fopenmp-targets=spir64 test.f90 -o test
```

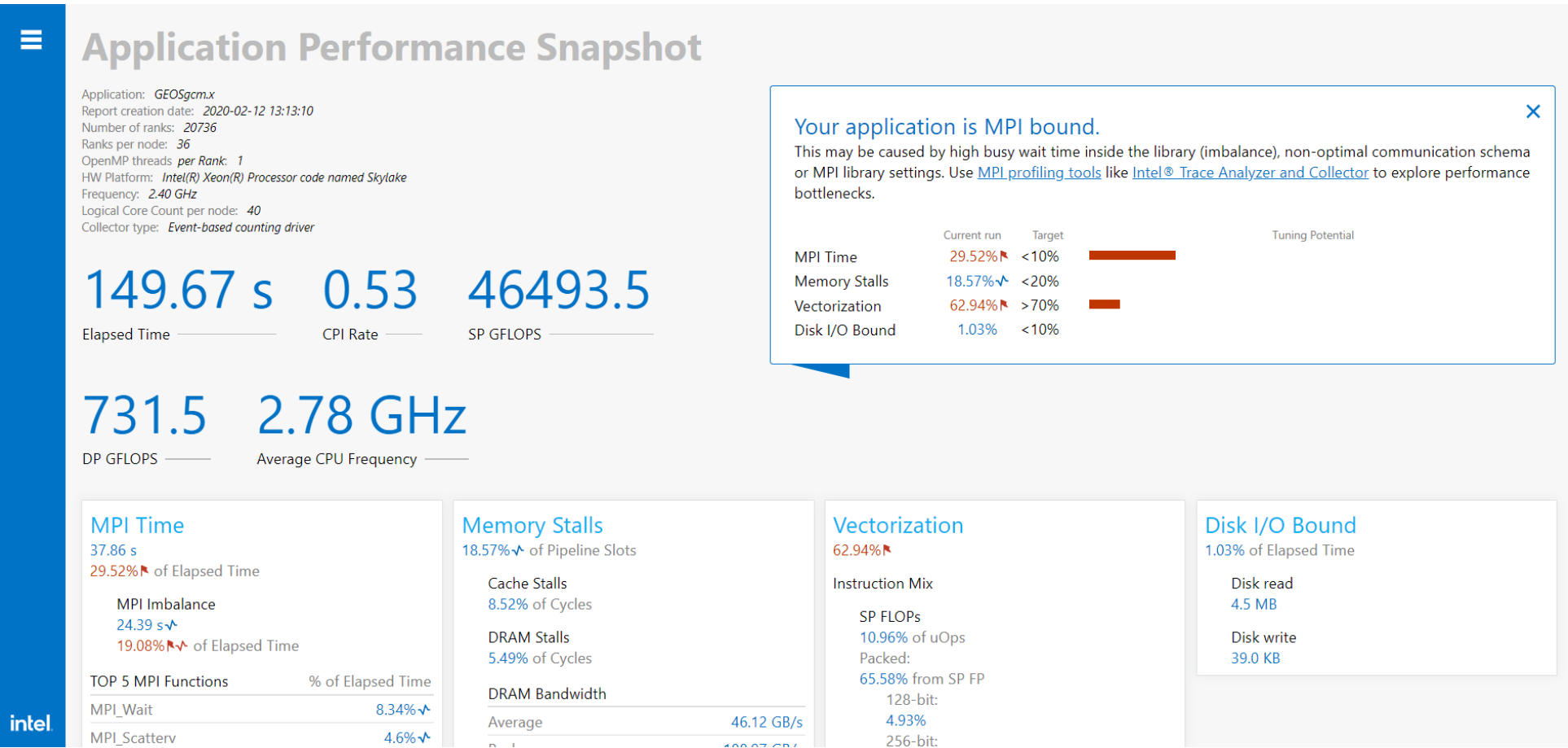
▪ Execute

```
$ I_MPI_OFFLOAD=2 mpirun -n 2 ./test
```

Application Performance Snapshot

Intel® VTune™ Profiler

Application Performance Snapshot



Application Performance Snapshot - APS

Key advantages:

- Wide set of metrics – MPI, OpenMP, CPU, Memory, GPU, PCI, Vectorization, etc.
- Analysis at scale - proven successful collection on 96k ranks scale by a customer.
- Low overhead, small trace size - runtime aggregation of MPI tracing and Hardware counters
- Easy to use – HTML and CL reports

Application Performance Snapshot - APS

- APS is included in VTune profiler package.
- APS provides aggregated metrics for multi-node launch.
- Outlier analysis localizes specific rank/node for detailed analysis with VTune.

Collection

How to collect:

```
mpirun [mpi_options] aps [aps_options] <app> [app_options]
```

Adjustable collection:

- `--collection-mode=[mpi|omp|hwc|all]` – ‘all’ by default
- `--stat-level=[1..5]` – from timing to detailed info about message sizes, communicators, destinations.
- `--mpi-imbalance=[0..2]` – 0 – disabled, 1 – get imbalance from Intel MPI (default), 2 – using inserted barriers
- Collection control through MPI_Pcontrols and ITT API

Low overhead:

- ~ 1-2% in default mode
- < 10% in any other mode

Summary report (aps --report <result>)

Application Performance Snapshot

Application: **GEOSgcm.x**
Report creation date: **2020-02-12 13:13:10**
Number of ranks: **20736**
Ranks per node: **36**
OpenMP threads **per Rank: 1**
HW Platform: **Intel(R) Xeon(R) Processor code named Skylake**
Frequency: **2.40 GHz**
Logical Core Count per node: **40**
Collector type: **Event-based counting driver**

149.67 s Elapsed Time
0.53 CPI Rate
46493.5 SP GFLOPS
731.5 DP GFLOPS

2.78 GHz
Average CPU Frequency

MPI Time

37.86 s
29.52% of Elapsed Time
MPI Imbalance
24.39 s
19.08% of Elapsed Time

TOP 5 MPI Functions	% of Elapsed Time
MPI_Wait	8.34%
MPI_Scatterv	4.6%
MPI_Init_thread	3.82%
MPI_Bcast	3.31%
MPI_Allreduce	3.04%

Intel Omni-Path Fabric Usage

Interconnect Bandwidth	Incoming	Outgoing
Average	0.78 GB/s	0.78 GB/s
Peak	3.32 GB/s	4.03 GB/s
Bound	0%	0%
Interconnect Packet Rate	Incoming	Outgoing
Average	0.02	0.02
Peak	0.88	0.89
Bound	0%	0%

Memory Footprint

Resident
560.47 MB
Resident per Node
20176.91 MB
Virtual
5959.14 MB

Memory Stalls

18.57% of Pipeline Slots

Cache Stalls	8.52% of Cycles
DRAM Stalls	5.49% of Cycles
DRAM Bandwidth	
Average	46.12 GB/s
Peak	188.97 GB/s
Bound	14.51%
NUMA	3.33% of Remote Accesses

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	Tuning Potential
MPI Time	29.52%	<10%	
Memory Stalls	18.57%	<20%	
Vectorization	62.94%	>70%	
Disk I/O Bound	1.03%	<10%	

Vectorization

62.94%

Instruction Mix

SP FLOPs
10.96% of uOps
Packed:
65.58% from SP FP
128-bit:
4.93%
256-bit:
60.66%
512-bit:
0%
Scalar:
34.42% from SP FP

DP FLOPs
0.48% of uOps
Packed:
33.41% from DP FP
128-bit:
2.83%
256-bit:
30.58%
512-bit:
0%
Scalar:
66.59% from DP FP

Non-FP
88.33% of uOps
FP Arith/Mem Rd Instr. Ratio
0.35
FP Arith/Mem Wr Instr. Ratio
0.98

Disk I/O Bound

1.03% of Elapsed Time

Disk read
4.5 MB
Disk write
39.0 KB

Reports

How to generate a report:

```
aps --report [report_options] <result_directory>
```

Detailed reports:

- MPI: functions, message sizes, communication matrix, list of MPI communicators, etc.
- Metrics: OpenMP Imbalance, CPU Utilization, IPC, Memory Bound, GPU Time, etc.
- Node topology

Customizable output:

- Filtering by ranks, nodes, mpi functions, communicators, volume
- Changing the size of communication diagram
- Adjusting level of details
- Different groupings in MPI related reports

Detailed reports

```
[root@nntpat98-144 aps_results]# aps --report -fdf
Loading 100.00%
% - percentage of MPI functions total time
Function summary for all Ranks
```

Function	Time(sec)
MPI_Waitall	972987.32
Min:	0.0000
Avg:	0.0001
Max:	3.9528
MPI_Allreduce	702927.29
Min:	0.0000
Avg:	0.0003
Max:	1.8815
MPI_Alltoallv	351070.07
Min:	0.0000
Avg:	0.0389
Max:	17.3723
MPI_Alltoall	178079.39
Min:	0.0004
Avg:	0.0030
Max:	0.0197
MPI_Barrier	105051.14
Min:	0.0000
Avg:	0.0650
Max:	0.1494
MPI_Isend	37178.85
Min:	0.0000
Avg:	0.0000
Max:	0.2821
MPI_Bcast	19726.74
Min:	0.0000
Avg:	0.0001
Max:	1.4459
MPI_Scatterv	8906.56
Min:	0.0000
Avg:	0.0015
Max:	0.1564

```
[root@nntpat98-144 aps_results]# aps --report --metrics="G
Loading 100.00%
Metric Table
```

Metric Name	Node Name	Metric Value
GPU Inbound PCIe Read, MB/s	s011-n004	207.14
GPU Inbound PCIe Read, MB/s	s011-n005	151.33

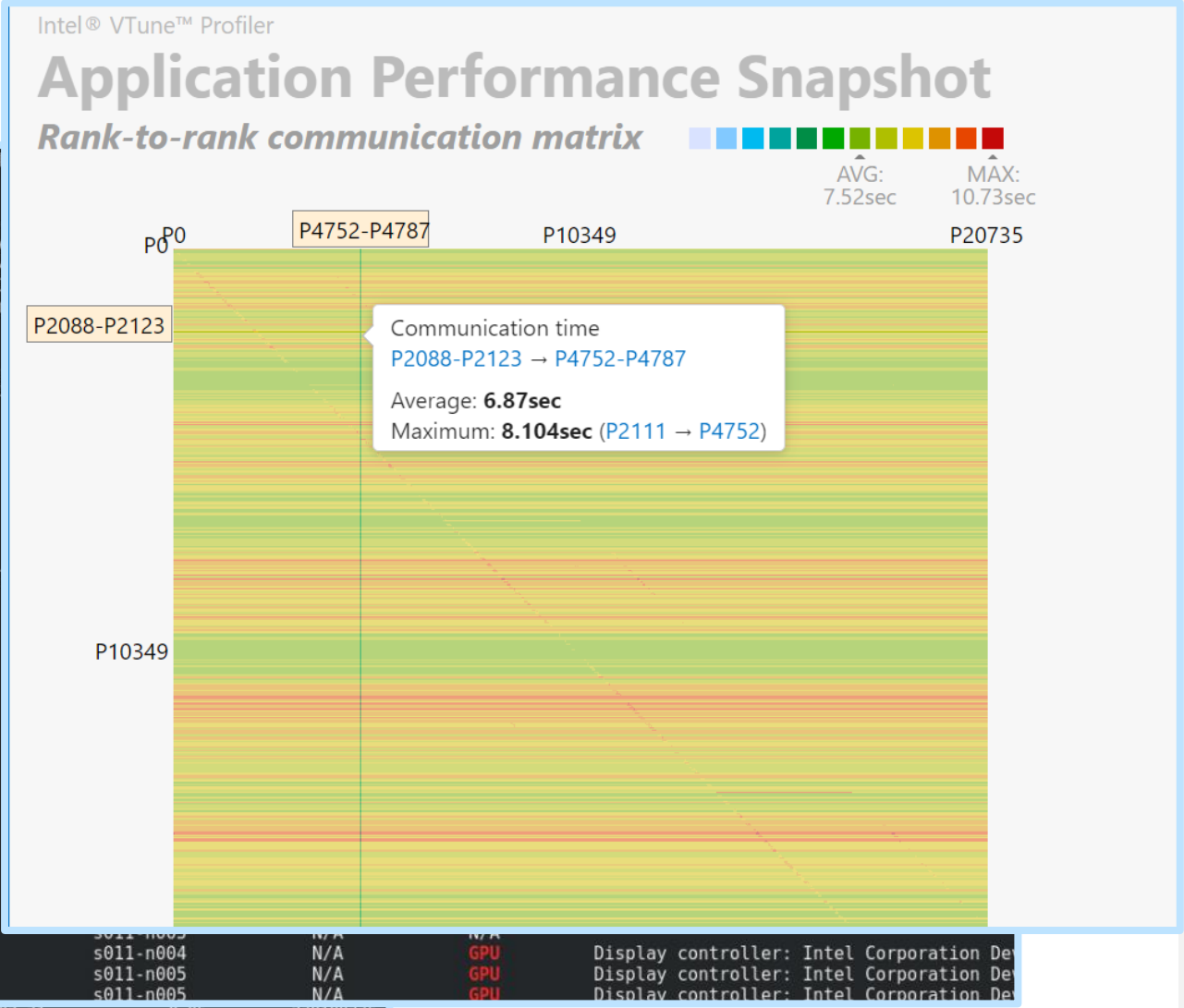
```
[root@nntpat98-144 aps_results]# aps --report --metrics="I
Loading 100.00%
Metric Table
```

Metric Name	Node Name	Dev
Inbound PCIe Read Per Device, MB/s	s011-n004	
Inbound PCIe Read Per Device, MB/s	s011-n005	
Inbound PCIe Read Per Device, MB/s	s011-n005	
Inbound PCIe Read Per Device, MB/s	s011-n004	
Inbound PCIe Read Per Device, MB/s	s011-n005	
Inbound PCIe Read Per Device, MB/s	s011-n004	
Inbound PCIe Read Per Device, MB/s	s011-n005	
Inbound PCIe Read Per Device, MB/s	s011-n004	

```
[root@nntpat98-144 aps_results]# aps --report --metrics="al
Loading 100.00%
```

GPU Time, s					
GPU Time, s					
GPU Time (% of Elapsed Time), % of Elapsed Time					
GPU Time (% of Elapsed Time), % of Elapsed Time					
GPU Utilization when Busy, %					
GPU Utilization when Busy, %					
GPU Occupancy, % of Peak Value					
GPU Occupancy, % of Peak Value					
GPU Inbound PCIe Read, MB/s					
GPU Inbound PCIe Read, MB/s					
GPU Inbound PCIe Write, MB/s					
GPU Inbound PCIe Write, MB/s					
GPU Outbound PCIe Read, MB/s					
GPU Outbound PCIe Read, MB/s					
GPU Outbound PCIe Write, MB/s					
GPU Outbound PCIe Write, MB/s					
Inbound PCIe Read Per Device, MB/s					
Inbound PCIe Read Per Device, MB/s					
Inbound PCIe Read Per Device, MB/s					

0.83	9162.06	0.38	2965603.71	0.05	278189376
0.37	7552.29	0.32	2481687.32	0.04	5789952



s011-n005	N/A	N/A	
s011-n004	N/A	GPU	Display controller: Intel Corporation De
s011-n005	N/A	GPU	Display controller: Intel Corporation De
s011-n005	N/A	GPU	Display controller: Intel Corporation De

GPU metrics


- GPU execution efficiency
 - OA HW counters (per node)
- OpenMP offload efficiency
 - tracing through OMPT (per rank)


```
[root@nntpat98-144 aps_results]# aps --report --metrics="GPU Time" ./aps_result_with_pci/
Loading 100.00%
| Metric Table
|-----|
Metric Name      Node Name      Metric Value
GPU Time, s      s011-n004      1.307
GPU Time, s      s011-n005      0.004
[root@nntpat98-144 aps_results]# aps --report --metrics="GPU Time (% of Elapsed Time)" ./aps_result_with_pci/
Loading 100.00%
| Metric Table
|-----|
Metric Name      Node Name      Metric Value
GPU Time (% of Elapsed Time), % of Elapsed Time  s011-n004      19.5
GPU Time (% of Elapsed Time), % of Elapsed Time  s011-n005      0.1
[root@nntpat98-144 aps_results]# aps --report --metrics="GPU Time (% of Elapsed Time)","GPU Utilization when Bu
Loading 100.00%
| Metric Table
|-----|
Metric Name      Node Name      Metric Value
GPU Time (% of Elapsed Time), % of Elapsed Time  s011-n004      19.5
GPU Time (% of Elapsed Time), % of Elapsed Time  s011-n005      0.1
GPU Utilization when Busy, %                      s011-n004      21.9
GPU Utilization when Busy, %                      s011-n005      0
GPU Occupancy, % of Peak Value                    s011-n004      84.4
GPU Occupancy, % of Peak Value                    s011-n005      0
```

GPU Utilization when Busy

10.95% 

EU State	% of EUs
Active	10.95%
Idle	54.7% 
Stalled	34.4% 

Offload Activity	% of GPU time
Compute	36.31%
Overhead	5.1%
Data Transfer	58.59% 

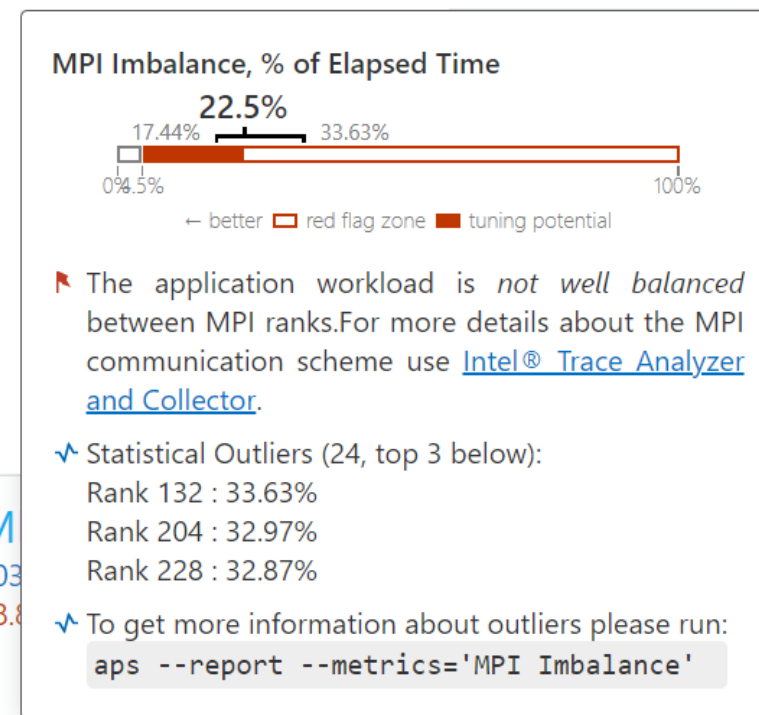
GPU Occupancy
42.2%  of Peak Value

Outliers

Provide Min, Max, Average

Detect statistical and threshold outliers

- Statistical outlier is based on two-sided Grubbs's test with 0.05 significance level
 - Highlighting anomalies and asymmetric distribution of work
 - Show a potential target for detailed analysis
- Threshold outlier – a metric value breaking the threshold.
 - Show an additional tuning potential for a source breaking the threshold.



22.5% of Elapsed Time

TOP 5 MPI Functions	% of Elapsed Time
MPI_Waitall	23.99%
MPI_Allreduce	17.33%
MPI_Alltoallv	8.66%📈
MPI_Alltoall	4.39%📈
MPI_Barrier	2.59%📈

Intel® oneAPI Tools for HPC

Intel® oneAPI HPC Toolkit

Deliver Fast Applications that Scale

What is it?

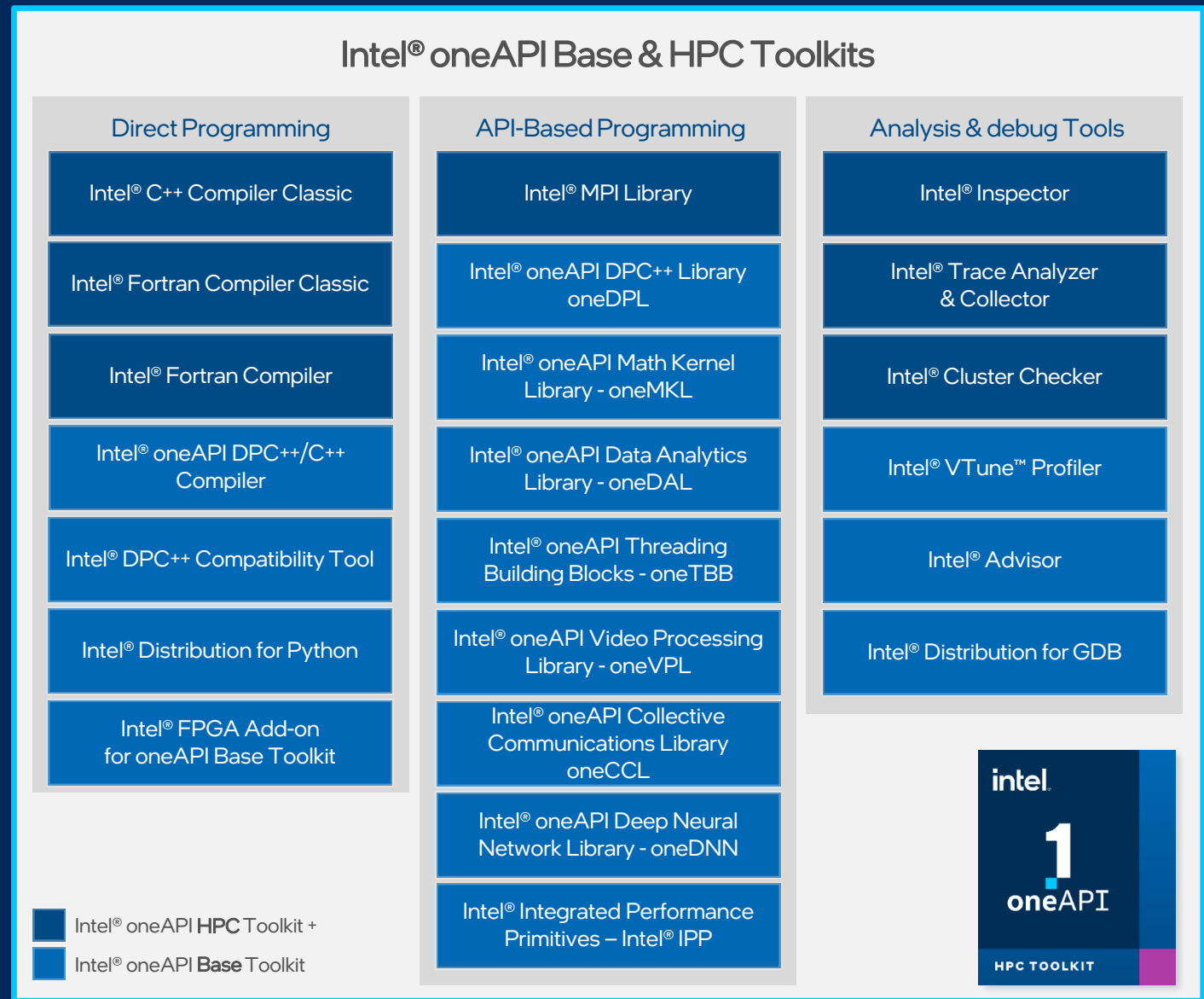
A toolkit that adds to the Intel® oneAPI Base Toolkit for building high-performance, scalable parallel code on C++, SYCL, Fortran, OpenMP & MPI from enterprise to cloud, and HPC to AI applications.

Who needs this product?

- OEMs/ISVs
- C++, Fortran, OpenMP, MPI Developers

Why is this important?

- Accelerate performance on Intel® Xeon® and Core™ Processors and Intel® Accelerators
- Deliver fast, scalable, reliable parallel code with less effort built on industry standards



Notices & Disclaimers

Texas Advanced Computing Center (TACC) Frontera references

Article: [HPCWire: Visualization & Filesystem Use Cases Show Value of Large Memory Fat Notes on Frontera](#).

www.intel.com/content/dam/support/us/en/documents/memory-and-storage/data-center-persistent-mem/Intel-Optane-DC-Persistent-Memory-Quick-Start-Guide.pdf

software.intel.com/content/www/us/en/develop/articles/introduction-to-programming-with-persistent-memory-from-intel.html

wreda.github.io/papers/assise-osdi20.pdf

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

