

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 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.

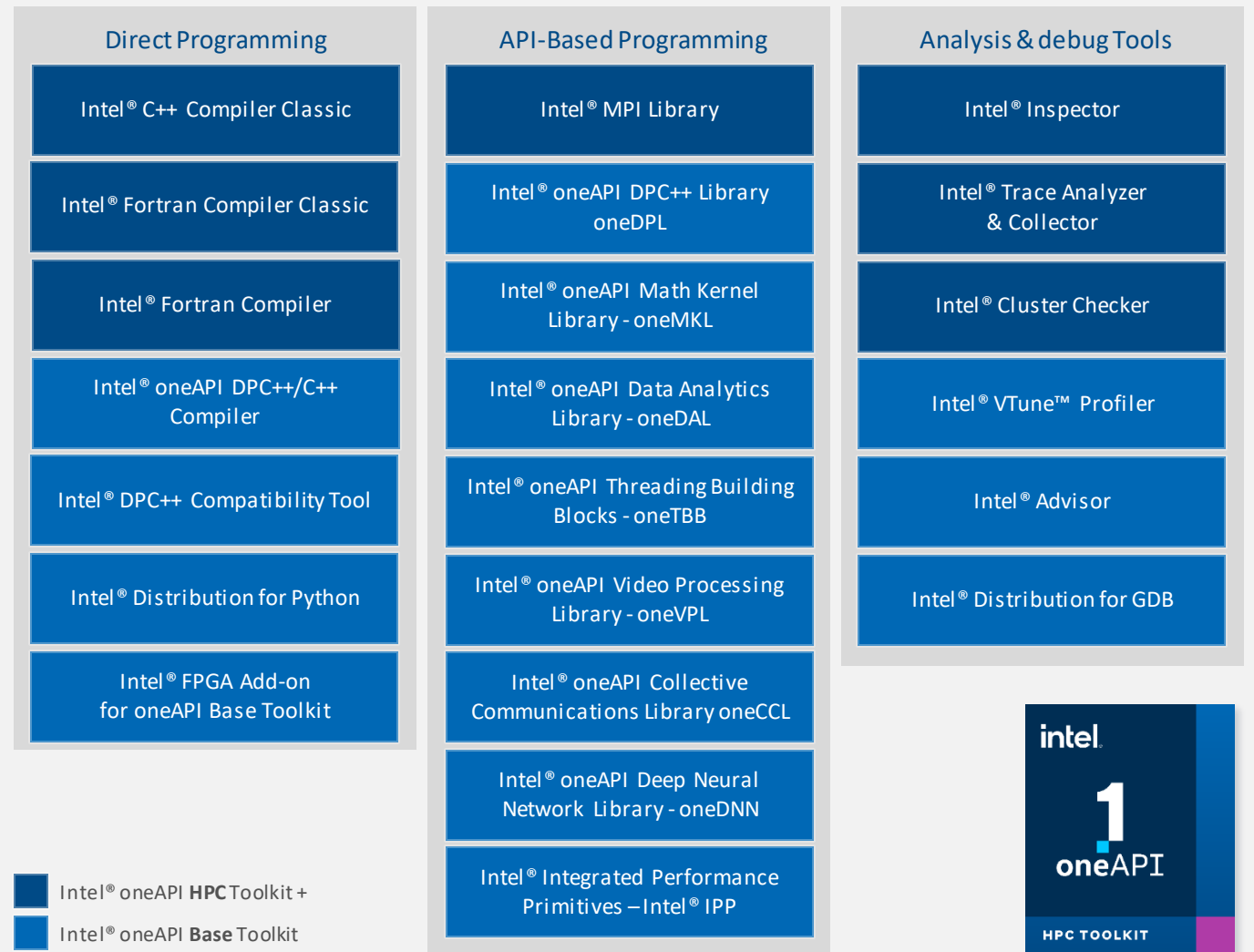
Who needs this product?

- 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® oneAPI Base & HPC Toolkits



Intel® MPI Library

Deliver Flexible, efficient, and Scalable Cluster Messaging

Standard Conformant

- MPI-1, MPI-2.2 and MPI-3.1. MPI-4 is WIP
- C, C++, Fortran 77, Fortran 90, and Fortran 2008 language bindings

Optimized MPI Application Performance

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

Lower Latency and Multi-vendor Interoperability

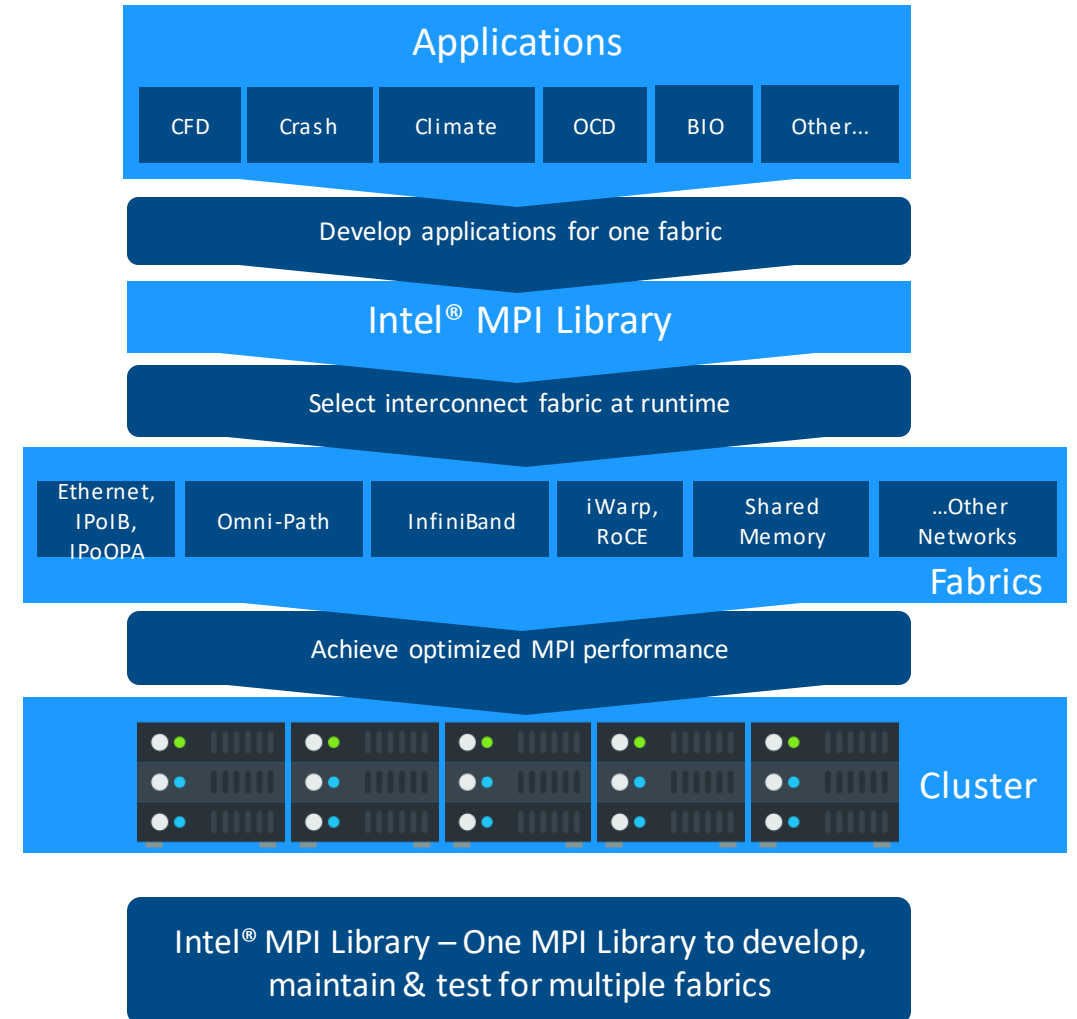
- Industry-leading latency
- Performance-optimized support for the fabric capabilities through OpenFabrics Interfaces (OFI)

Faster MPI Communication

- Optimized collectives

Sustainable scalability

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



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

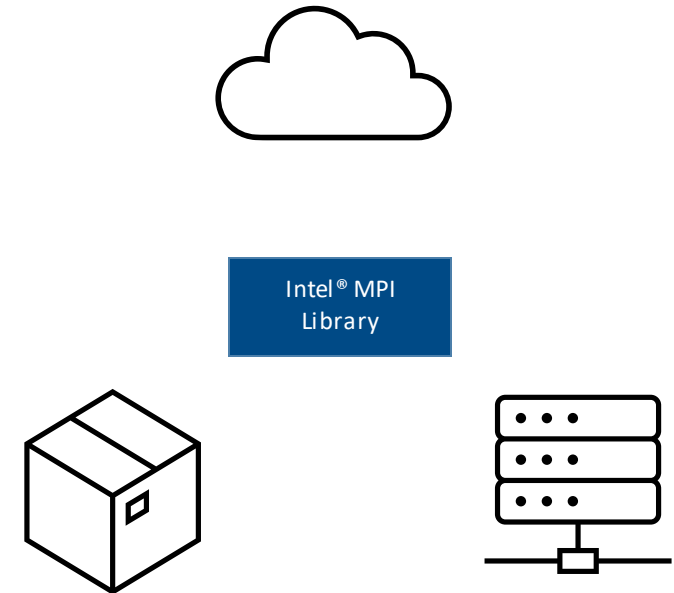
- Intel® Xeon® Scalable 3rd gen processors
- Performance on Intel® Ethernet 800 Series Network Adapters
- Mellanox* ConnectX*: 3/4/5/6 (FDR/EDR/HDR) support enhancements
- **Intel GPUs support**

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

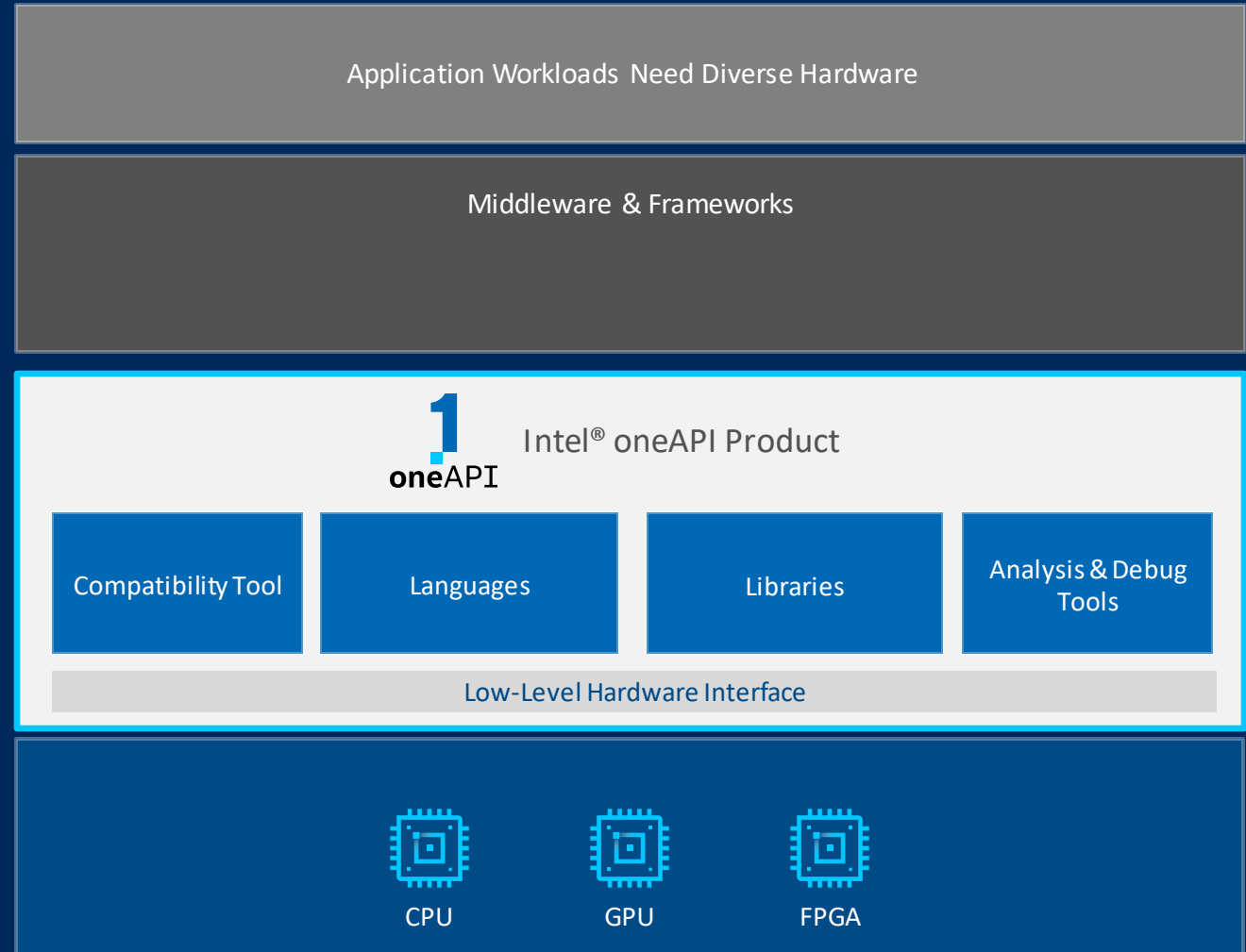


Intel® oneAPI Tools

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](#)

Overview

- Intel® MPI GPU Buffers Support
- I_MPI_OFFLOAD_* Family
- Tips to MPI Applications Developers

Intel® MPI GPU Buffers Support

Execution models

Naïve (OpenMP 4.0): map(tofrom) clause

```
#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;
    }
}

//Send device buffer to rank 0 after copy back from GPU
MPI_Send(values, num_values, MPI_INT, dest_rank, tag, MPI_COMM_WORLD);
```


Execution models

GPU buffer aware (OpenMP 4.5): use_device_ptr

```
#pragma omp target data map(to: rank, values[0:num_values],  
                                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;  
    }  
  
    // Send device buffer to rank 0 without copy back from GPU  
    MPI_Send(values, num_values, MPI_INT, dest_rank, tag, MPI_COMM_WORLD);  
}
```

Compilation

- use **-fc=ifx** or **-cc=icx**, invokes LLVM compilers
- ifx/icc options:
 - use **-fopenmp** instead of **-qopenmp**
 - **-fopenmp-targets=spir64** enables the offloading

Compilation

▪ Build

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

- or

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

▪ Execute

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

I_MPI_OFFLOAD_* Variables Family

Environment Variables

- I_MPI_OFFLOAD
- I_MPI_OFFLOAD_CELL
- I_MPI_OFFLOAD_DOMAIN_SIZE
- I_MPI_OFFLOAD_DEVICES
- I_MPI_OFFLOAD_DEVICE_LIST
- I_MPI_OFFLOAD_DOMAIN
- I_MPI_DEBUG

Environment Variables

- **I_MPI_OFFLOAD**
- I_MPI_OFFLOAD_CELL
- I_MPI_OFFLOAD_DOMAIN_SIZE
- I_MPI_OFFLOAD_DEVICES
- I_MPI_OFFLOAD_DEVICE_LIST
- I_MPI_OFFLOAD_DOMAIN
- I_MPI_DEBUG

Description

Enables handling of device buffers in MPI functions.

Syntax

I_MPI_OFFLOAD=<int>

Arguments

- 0: disabled [default]
- 1: auto
- 2: enabled, loads Level_Zero library

Environment Variables

- I_MPI_OFFLOAD
- I_MPI_OFFLOAD_CELL
- I_MPI_OFFLOAD_DOMAIN_SIZE
- I_MPI_OFFLOAD_DEVICES
- I_MPI_OFFLOAD_DEVICE_LIST
- I_MPI_OFFLOAD_DOMAIN
- I_MPI_DEBUG

Description

Specifies the base unit.

Syntax

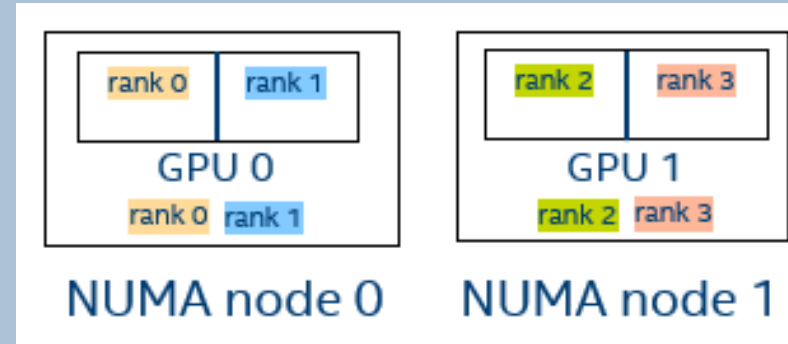
```
I_MPI_OFFLOAD_CELL=<cell>
```

Arguments

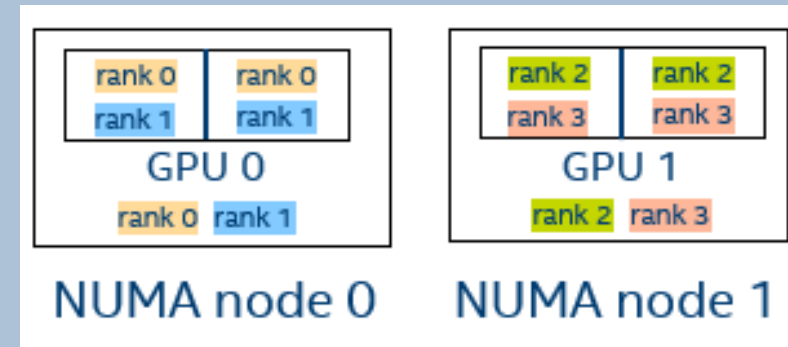
- tile: one tile/subdevice [default]
- device: whole device (GPU) with all subdevices

Environment Variables

- I_MPI_OFFLOAD
- I_MPI_OFFLOAD_CELL
- I_MPI_OFFLOAD_DOMAIN_SIZE
- I_MPI_OFFLOAD_DEVICES
- I_MPI_OFFLOAD_DEVICE_LIST
- I_MPI_OFFLOAD_DOMAIN
- I_MPI_DEBUG



4 MPI ranks, I_MPI_OFFLOAD_CELL=tile



4 MPI ranks, I_MPI_OFFLOAD_CELL=device

Environment Variables

- I_MPI_OFFLOAD
- I_MPI_OFFLOAD_CELL
- I_MPI_OFFLOAD_DOMAIN_SIZE
- I_MPI_OFFLOAD_DEVICES
- I_MPI_OFFLOAD_DEVICE_LIST
- I_MPI_OFFLOAD_DOMAIN
- I_MPI_DEBUG

Description

Defines number of base unit per MPI rank.

Syntax

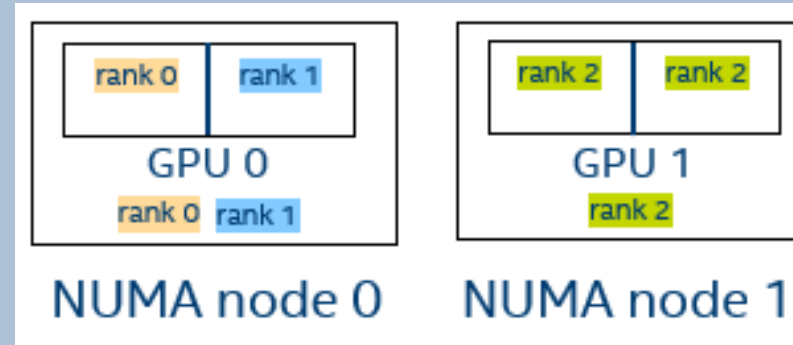
```
I_MPI_OFFLOAD_DOMAIN_SIZE=<int>
```

Arguments

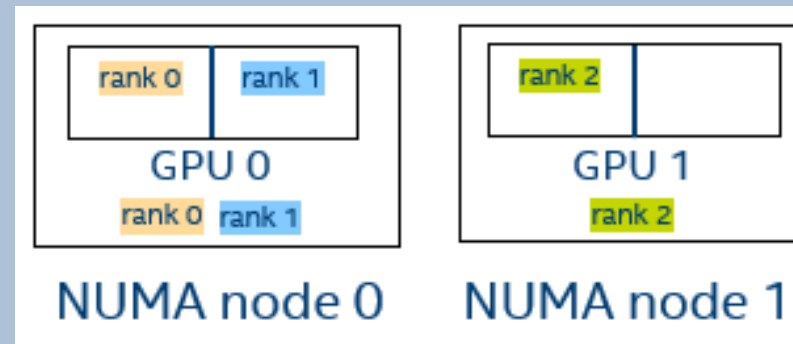
- -1: auto. IMPI will try to use all resources [default]
- >0: fixed size

Environment Variables

- I_MPI_OFFLOAD
- I_MPI_OFFLOAD_CELL
- I_MPI_OFFLOAD_DOMAIN_SIZE
- I_MPI_OFFLOAD_DEVICES
- I_MPI_OFFLOAD_DEVICE_LIST
- I_MPI_OFFLOAD_DOMAIN
- I_MPI_DEBUG



3 MPI ranks, I_MPI_OFFLOAD_DOMAIN_SIZE=-1



3 MPI ranks, I_MPI_OFFLOAD_DOMAIN_SIZE=1

Environment Variables

- I_MPI_OFFLOAD
- I_MPI_OFFLOAD_CELL
- I_MPI_OFFLOAD_DOMAIN_SIZE
- I_MPI_OFFLOAD_DEVICES
- I_MPI_OFFLOAD_DEVICE_LIST
- I_MPI_OFFLOAD_DOMAIN
- I_MPI_DEBUG

Description

Defines available devices

Syntax

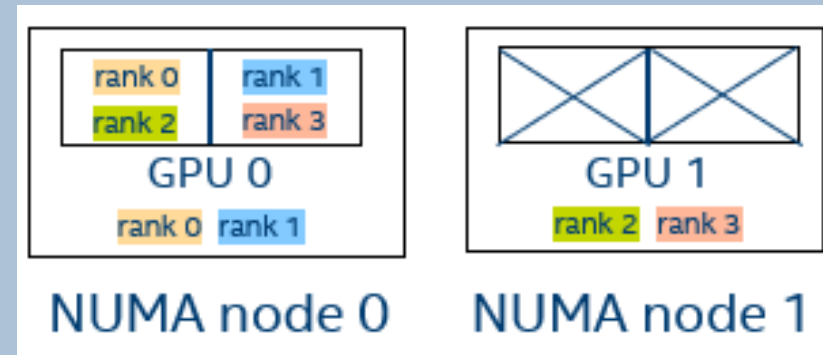
I_MPI_OFFLOAD_DEVICES=<list>

Arguments

- all: uses all visible devices [default]
- i: device with logical number i
- i-j: range, from i to j
- k, i-j: device k and devices from i to j

Environment Variables

- I_MPI_OFFLOAD
- I_MPI_OFFLOAD_CELL
- I_MPI_OFFLOAD_DOMAIN_SIZE
- I_MPI_OFFLOAD_DEVICES
- I_MPI_OFFLOAD_DEVICE_LIST
- I_MPI_OFFLOAD_DOMAIN
- I_MPI_DEBUG



4 MPI ranks, I_MPI_OFFLOAD_DEVICES=0

Environment Variables

- I_MPI_OFFLOAD
- I_MPI_OFFLOAD_CELL
- I_MPI_OFFLOAD_DOMAIN_SIZE
- I_MPI_OFFLOAD_DEVICES
- **I_MPI_OFFLOAD_DEVICE_LIST**
- I_MPI_OFFLOAD_DOMAIN
- I_MPI_DEBUG

Description

Pins MPI ranks to specified base unit (tile or device, depending on I_MPI_OFFLOAD_CELL)

Syntax

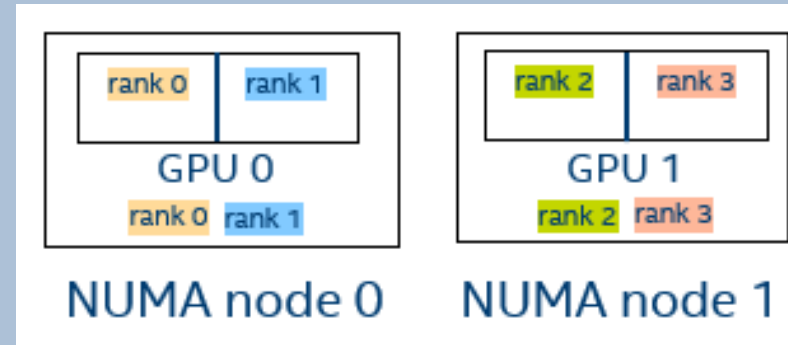
I_MPI_OFFLOAD_DEVICE_LIST=<list>

Arguments

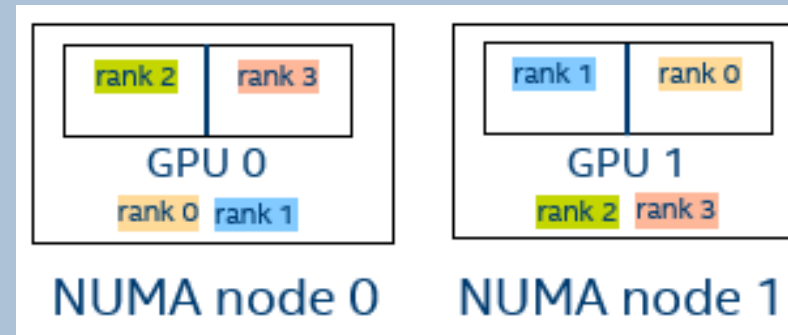
- i: device with logical number i
- i-j: range, from i to j
- k, i-j: device k and devices from i to j

Environment Variables

- I_MPI_OFFLOAD
- I_MPI_OFFLOAD_CELL
- I_MPI_OFFLOAD_DOMAIN_SIZE
- I_MPI_OFFLOAD_DEVICES
- I_MPI_OFFLOAD_DEVICE_LIST
- I_MPI_OFFLOAD_DOMAIN
- I_MPI_DEBUG



unset I_MPI_OFFLOAD_DEVICE_LIST



I_MPI_OFFLOAD_DEVICE_LIST=3,2,0,1

Environment Variables

- I_MPI_OFFLOAD
- I_MPI_OFFLOAD_CELL
- I_MPI_OFFLOAD_DOMAIN_SIZE
- I_MPI_OFFLOAD_DEVICES
- I_MPI_OFFLOAD_DEVICE_LIST
- I_MPI_OFFLOAD_DOMAIN
- I_MPI_DEBUG

Description

Define domains through a mask.

Syntax

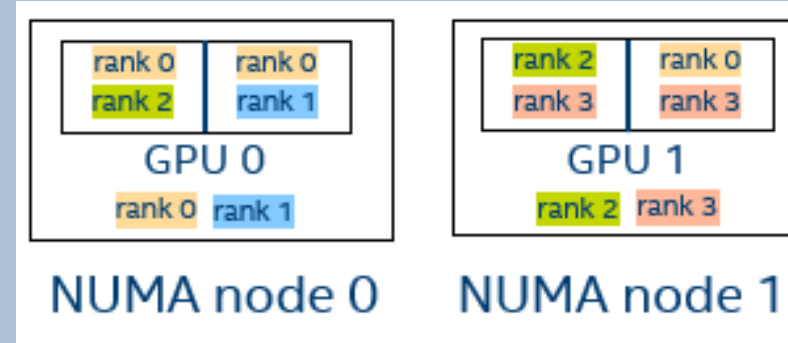
I_MPI_OFFLOAD_DOMAIN=<mask>

Arguments

- comma-separated list (one per MPI rank) of bitmasks translated into hexadecimals, e.g.
 - bitmask: [1101,0100,1010,0011]
 - hexadecimal: [B,2,5,C]

Environment Variables

- I_MPI_OFFLOAD
- I_MPI_OFFLOAD_CELL
- I_MPI_OFFLOAD_DOMAIN_SIZE
- I_MPI_OFFLOAD_DEVICES
- I_MPI_OFFLOAD_DEVICE_LIST
- **I_MPI_OFFLOAD_DOMAIN**
- I_MPI_DEBUG



Four MPI ranks.

I_MPI_OFFLOAD_DOMAIN=[B,2,5,C].

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

Environment Variables

- I_MPI_OFFLOAD
- I_MPI_OFFLOAD_CELL
- I_MPI_OFFLOAD_DOMAIN_SIZE
- I_MPI_OFFLOAD_DEVICES
- I_MPI_OFFLOAD_DEVICE_LIST
- I_MPI_OFFLOAD_DOMAIN
- I_MPI_DEBUG

```
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}
```

Tips to MPI Applications

Intel MPI Tunning

Out-of-the-box tuning is designed to "usual" workloads and topologies. Custom tuning may be profitable for:

- untested number of ranks configurations
- non-standard message sizes (e.g. $512 \text{ KB} < \text{msg_size} < 1024 \text{ KB}$)
- new network topologies
- untested interconnects (e.g. Cray)
- applications with high imbalance
- non-standard/user defined datatypes
- uncommon collectives (e.g. `reduce_scatter`)

Even small performance gains (without code changes/rebuilding) build up over a cluster's service life
=> significant savings!

Autotuner – usage model

Step 1 – Enable autotuner and store results (store is optional):

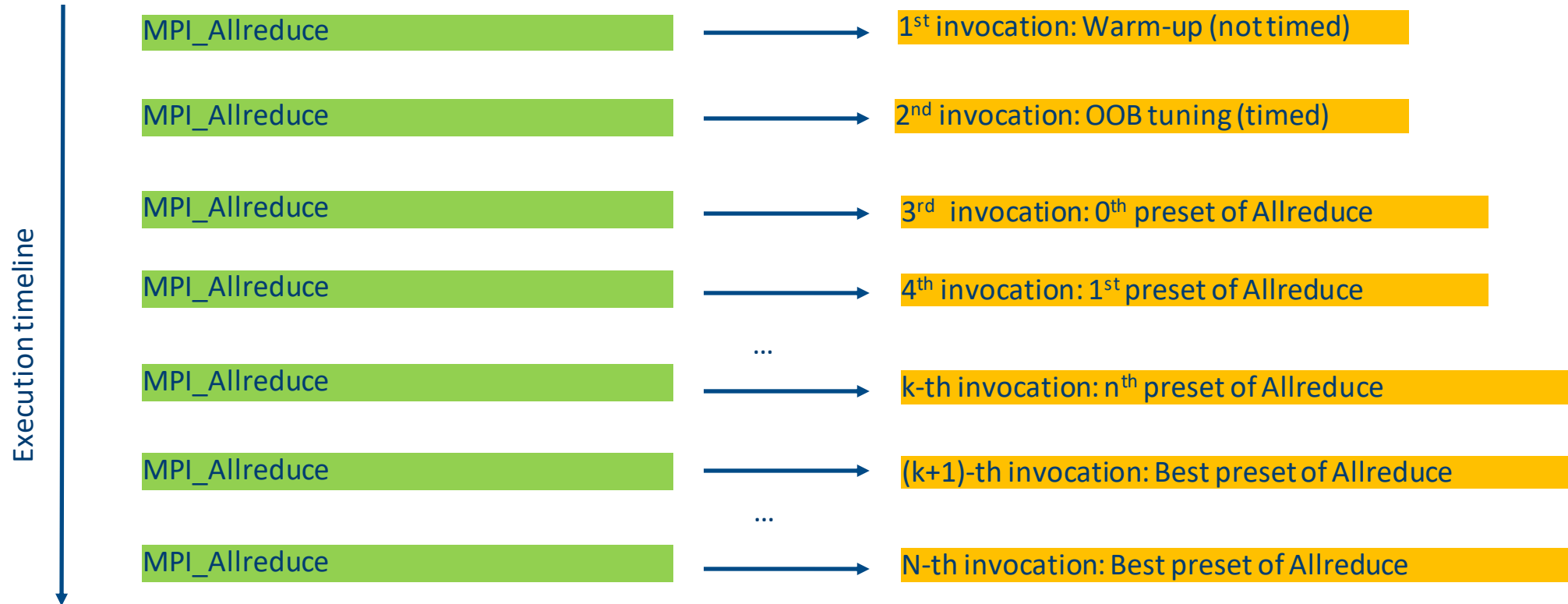
```
$ export I_MPI_TUNING_MODE=auto
$ export I_MPI_TUNING_BIN_DUMP=./tuning_results.dat
$ export I_MPI_TUNING_AUTO_ITER_NUM=1
$ mpirun -n 96 -ppn 48 IMB-MPI1 allreduce <args>
```

(this run may be slower, due to the tuning)

Step 2 – Use the results of autotuner for consecutive launches (optional):

```
$ unset I_MPI_TUNING_MODE
$ export I_MPI_TUNING_BIN=./tuning_results.dat
$ mpirun -n 96 -ppn 48 IMB-MPI1 allreduce <args>
```

Autotuner – dynamic tuning



(performed for each message size/communicator)

Application Performance Snapshot

- Wide set of metrics – **MPI, OpenMP, GPU**, CPU, Memory, 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 countersAPS is included in **VTune** profiler package.
- **Outlier analysis** localizes specific rank/node for detailed analysis with VTune.
- Easy to use – CL and **HTML reports**

Application Performance Snapshot



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

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% | |

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% |

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

Vectorization

62.94%

Instruction Mix
SP FLOPs
10.96% of uOps
Packed:
65.58% from SP FP
128-bit:
4.93%
256-bit:

Disk I/O Bound

1.03% of Elapsed Time

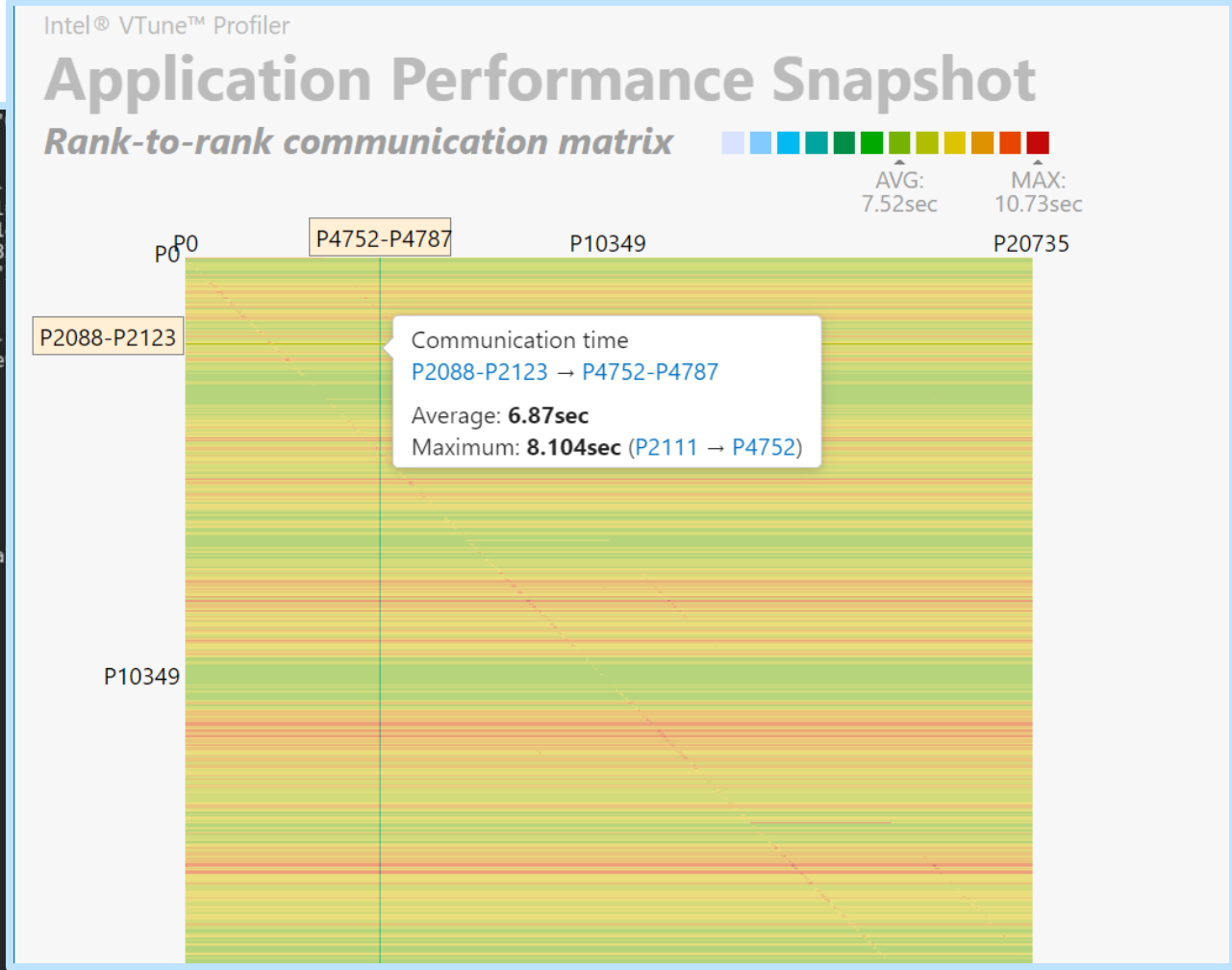
Disk read
4.5 MB

Disk write
39.0 KB

Application Performance Snapshot

```
[root@nntpat98-144 aps_results]# aps --report --metrics="
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="
Loading 100.00%
Metric Table
-----
Metric Name          Node Name  Metric Val
-----
GPU Inbound PCIe Read, MB/s  s011-n004  207.1
GPU Inbound PCIe Read, MB/s  s011-n005  151.3
-----
[root@nntpat98-144 aps_results]# aps --report --metrics="
Loading 100.00%
Metric Table
-----
Metric Name          Node Name  De
-----
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-n005
-----
[red] GPU Time, s
[red] GPU Time, s
[red] GPU Time (% of Elapsed Time), % of Elapsed Time
[red] GPU Time (% of Elapsed Time), % of Elapsed Time
[red] GPU Utilization when Busy, %
[red] GPU Utilization when Busy, %
[red] GPU Occupancy, % of Peak Value
[red] GPU Occupancy, % of Peak Value
[red] GPU Inbound PCIe Read, MB/s
[red] GPU Inbound PCIe Read, MB/s
[red] GPU Inbound PCIe Write, MB/s
[red] GPU Inbound PCIe Write, MB/s
[red] GPU Outbound PCIe Read, MB/s
[red] GPU Outbound PCIe Read, MB/s
[red] GPU Outbound PCIe Write, MB/s
[red] 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
```



Troubleshooting MPI Applications

- Interactive debugging using System's gdb:

```
$ mpirun -n 8 -gdb \
```



```
IMB-MPI1 allreduce
```
- Starts one gdb-server and one gdb-client per rank. User interacts with gdb-server only.

```
[ rafael@icx1 ] $ mpirun -n 4 -gdb ./a.out
mpigdb: attaching to 50265 ./a.out icx1
mpigdb: attaching to 50266 ./a.out icx1
mpigdb: attaching to 50267 ./a.out icx1
mpigdb: attaching to 50268 ./a.out icx1
[0-3] (mpigdb) b test.c:37
[0-3] Breakpoint 1 at 0x400912: file ./test.c, line 37.
[0-3] (mpigdb) r
[0-3] Continuing.
[1-3]
[0]
[1] Breakpoint 1, printHello (rank=1, size=4) at ./test.c:37
[2] Breakpoint 1, printHello (rank=2, size=4) at ./test.c:37
[3] Breakpoint 1, printHello (rank=3, size=4) at ./test.c:37
[0] Breakpoint 1, printHello (rank=0, size=4) at ./test.c:37
[1-3] 37 MPI_Get_processor_name(name, &namelen);
[0] 37 MPI_Get_processor_name(name, &namelen);
[0-3] (mpigdb) r
[0-3] Continuing.
Hello world: rank 0 of 4 running on icx1
Hello world: rank 1 of 4 running on icx1
Hello world: rank 2 of 4 running on icx1
Hello world: rank 3 of 4 running on icx1
[0] [Inferior 1 (process 50265) exited normally]
[1] [Inferior 1 (process 50266) exited normally]
[2] [Inferior 1 (process 50267) exited normally]
[3] [Inferior 1 (process 50268) exited normally]
```

Troubleshooting MPI Applications

- Intel Trace Analyser and Collector – Correctness Check:

```
$ mpirun -n 8 -check_mpi IMB-MPI1 allreduce
```

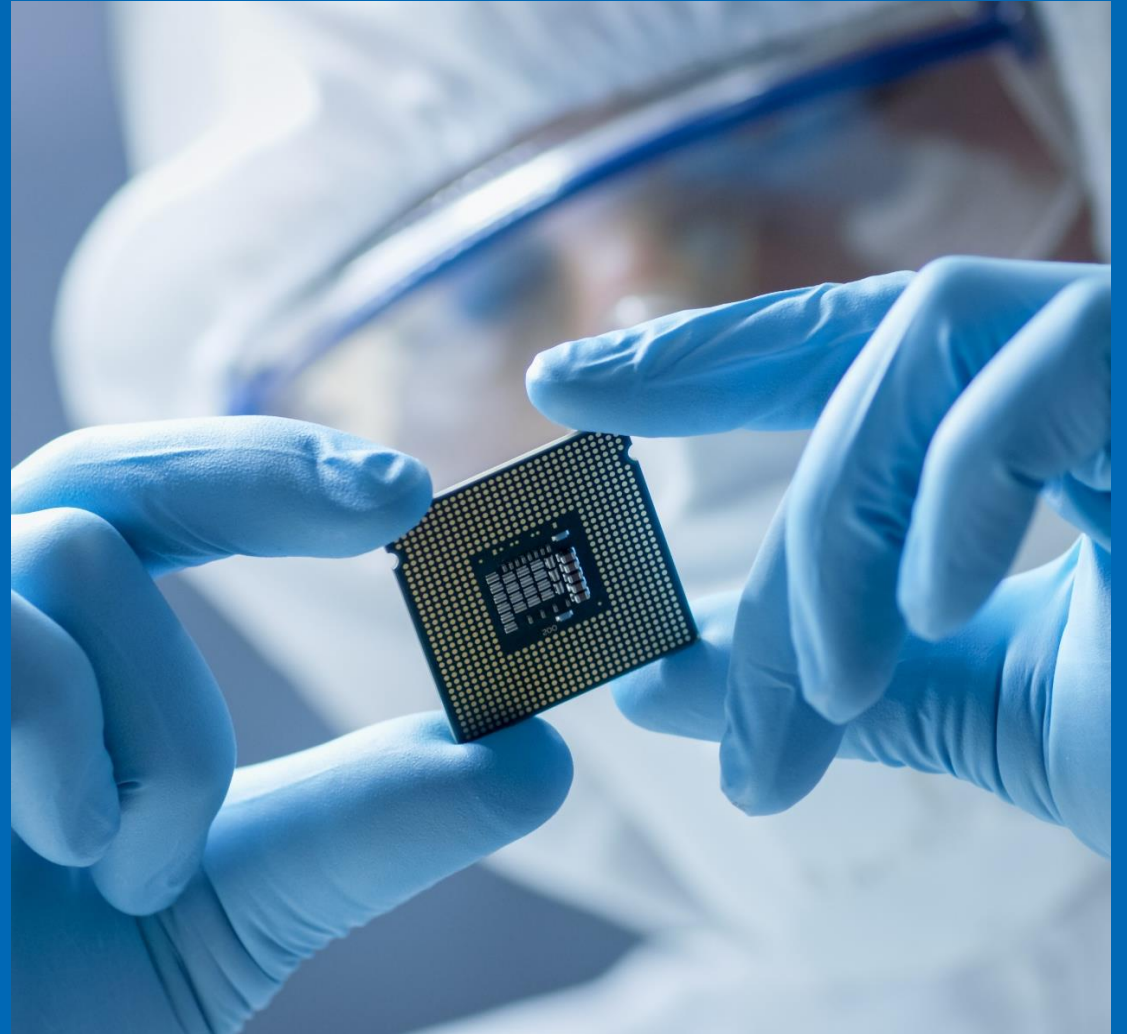
(Attention: it can
be quite verbose!)

```
[ raphael@icx1 ] $ mpirun -n 4 -check_mpi ./a.out
[0] INFO: CHECK LOCAL:EXIT:SIGNAL ON
[0] INFO: CHECK LOCAL:EXIT:BEFORE_MPI_FINALIZE ON
[0] INFO: CHECK LOCAL:MPI:CALL_FAILED_ON
[0] INFO: CHECK LOCAL:MEMORY:OVERLAP ON
(...)
[0] INFO: CHECK GLOBAL:COLLECTIVE:INVALID_PARAMETER ON
[0] INFO: CHECK GLOBAL:COLLECTIVE:COMM_FREE_MISMATCH ON
[0] INFO: maximum number of errors before aborting: CHECK-MAX-ERRORS 1
[0] INFO: maximum number of reports before aborting: CHECK-MAX-REPORTS 0 (= unlimited)
[0] INFO: maximum number of times each error is reported: CHECK-SUPPRESSION-LIMIT 10
[0] INFO: timeout for deadlock detection: DEADLOCK-TIMEOUT 60s
[0] INFO: timeout for deadlock warning: DEADLOCK-WARNING 300s
[0] INFO: maximum number of reported pending messages: CHECK-MAX-PENDING 20
Hello world: rank 0 of 4 running on icx1
[1] ERROR: LOCAL:MPI:CALL_FAILED: error
[1] ERROR: Invalid rank has value 100 but must be nonnegative and less than 4.
[1] ERROR: Error occurred at:
[1] ERROR: MPI_Send(*buf=0x7ffdcc877c38, count=1, datatype=MPI_INT, dest=100, tag=1, comm=MPI_COMM_WORLD)
[1] ERROR: printHello (/home/rafael/area51/support/ICC_PLAYGROUND/./test.c:49)
[1] ERROR: main (/home/rafael/area51/support/ICC_PLAYGROUND/./test.c:26)
[1] ERROR: __libc_start_main (/lib64/libc-2.31.so)
[1] ERROR: _start (/home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S:120)
[1] INFO: 1 error, limit CHECK-MAX-ERRORS reached => aborting
```

Take-away Message

- «GPU Pinning» and «GPU Buffers Support» features in Intel® MPI can significantly simplify programming for GPUs
- Reference:
<https://software.intel.com/content/www/us/en/develop/documentation/mpi-developer-reference-linux/top/environment-variable-reference/gpu-support.html>

Questions?



The Intel logo is centered on a solid blue background. It features the word "intel" in a white, lowercase, sans-serif font. A small blue square is positioned above the letter "i". To the right of the word "intel" is a registered trademark symbol (®) enclosed in a white circle.

intel®

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

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.

| Metric | Current run | Target | Tuning Potential |
|----------------|-------------|--------|------------------|
| MPI Time | 29.52% | <10% | High |
| Memory Stalls | 18.57% | <20% | High |
| Vectorization | 62.94% | >70% | Low |
| Disk I/O Bound | 1.03% | <10% | High |

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 | | |
|--------------------------|----------|----------|
| Average | Incoming | Outgoing |
| Average | 0.02 | 0.02 |
| Peak | 0.88 | 0.89 |
| Bound | 0% | 0% |

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

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

Memory Footprint

Resident: 560.47 MB
 Resident per Node: 20176.91 MB
 Virtual: 5959.14 MB

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

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 Busy"
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%  |

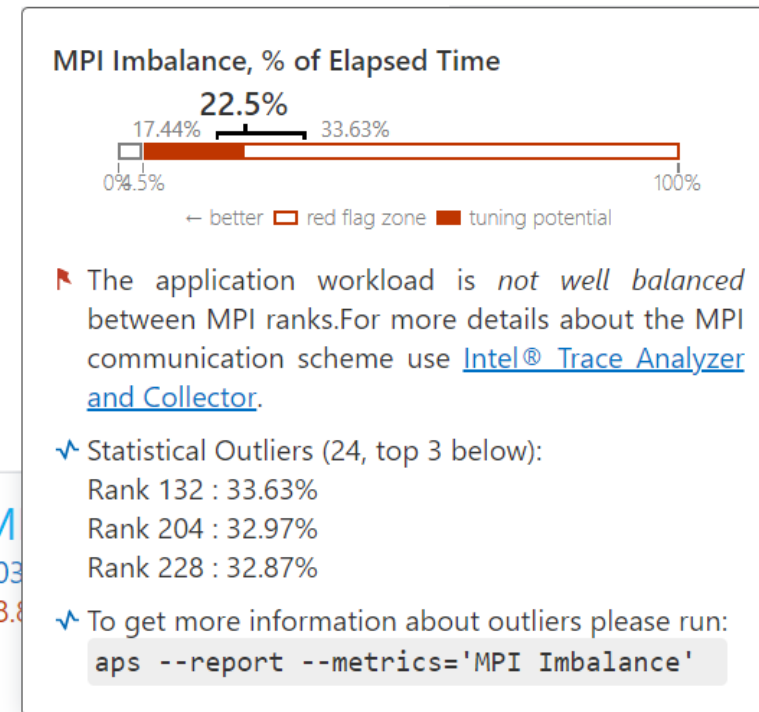
GPU Occupancy
42.2%  of Peak Value

Outliers

Provide Min, Max, Average

Detect statistical and threshold outliers

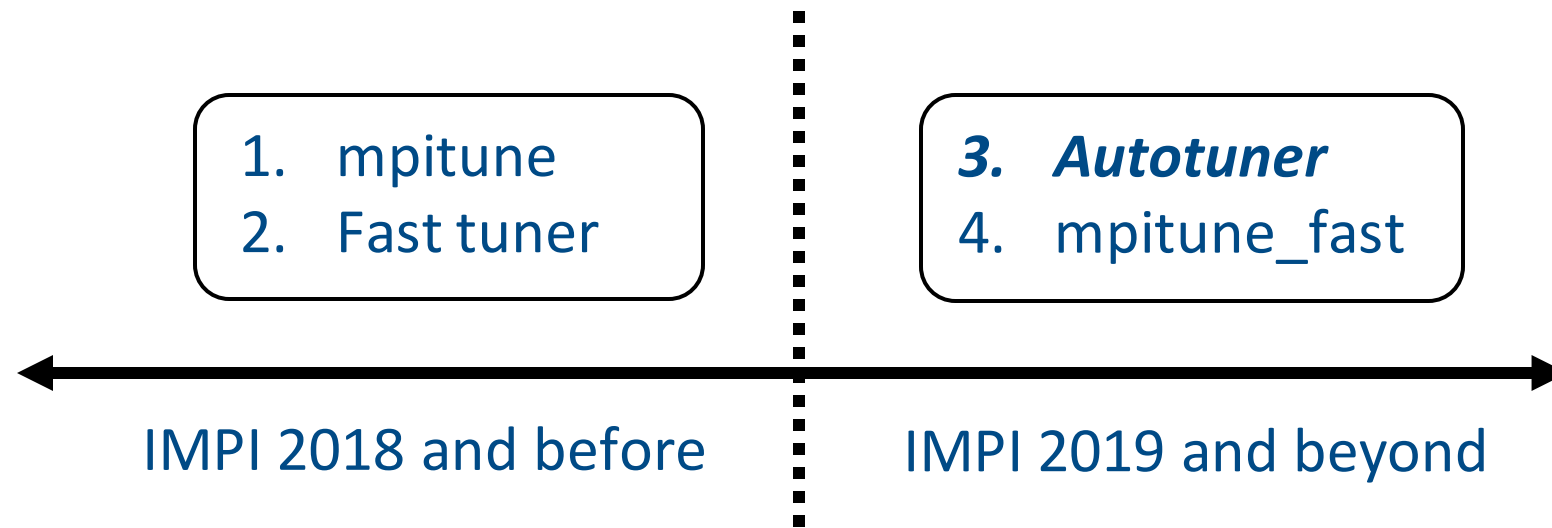
- Statistical outlier is based on two-sided Grubbs's test with 0.05 significance level
 - Highlighting anomalies and asymmetric distribution of work
 - Show a potential target for detailed analysis
- Threshold outlier – a metric value breaking the threshold.
 - Show an additional tuning potential for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



Introduction

Good 
Ok 
Bad 

| Tuning utility | MPItune | Fast Tuner | Autotuner | mpitune_fast |
|-------------------------------------|----------------|-------------------|------------------|---------------------|
| Parameter | | | | |
| Tuning overhead | Bad | Ok | Good | Good |
| Ease of use | Bad | Bad | Good | Good |
| Application tuning | Bad | Ok | Good | Bad |
| Microbenchmark tuning | Good | Good | Good | Good |
| Adoption in production environments | Bad | Bad | Good | Good |

Environment variables – Main flow control

`I_MPI_TUNING_MODE=<auto|auto:application|auto:cluster>` (**disabled** by default)

`I_MPI_TUNING_AUTO_ITER_NUM=<number>` Tuning iterations number (**1** by default).

`I_MPI_TUNING_AUTO_SYNC=<0|1>` Call internal barrier on every tuning iteration (**disabled** by default)

Guidance on I_MPI_TUNING_AUTO_ITER_NUM

Min invocations required for a certain collective call for a certain message size in a certain

communicator = `I_MPI_TUNING_AUTO_WARMUP_ITER_NUM` + [(range+1)*
`I_MPI_TUNING_AUTO_ITER_NUM`]

Autotuner Example

Configuration possibly slowing down tuning run in favour of results.:

- `I_MPI_TUNING_MODE=auto`
- `I_MPI_TUNING_AUTO_WARMUP_ITER_NUM=1`
- `I_MPI_TUNING_AUTO_ITER_NUM=128`
- `I_MPI_TUNING_AUTO_SYNC=1`
- `I_MPI_TUNING_AUTO_ITER_POLICY_THRESHOLD=4194304`
- `I_MPI_TUNING_AUTO_STORAGE_SIZE=4194304`
- `I_MPI_TUNING_BIN_DUMP=./my_tuning_file.dat`

Apply tuning results via

- `I_MPI_TUNING_BIN=./my_tuning_file.dat`

Restricting the scope of implementations

Remove failed implementation/s and switch back to the release version of Intel MPI Library and rerun autotuner. E.g. removing 11th implementation.:

```
$ export I_MPI_ADJUST_ALLREDUCE_LIST=0-10,12-25
```

This technique can also be used outside of tuning scenarios to find failed implementations in Intel MPI Library.

mpitune_fast

| | | |
|--------------|-----------------------------|-----------------------|
| | Autotuner | mpitune_fast |
| Scope | Application specific tuning | Cluster wide tuning |
| Intended for | Regular users | System administrators |

- tunes the Intel® MPI Library to the cluster configuration using autotuner functionality.
- iteratively launches the Intel® MPI Benchmarks with the proper autotuner environment and generates a tuning file.
- supports Slurm and LSF job managers. mpitune_fast automatically finds job allocated hosts and performs launches.
- **Example**
`$ mpitune_fast -f ./hostfile -c alltoall,allreduce,barrier`