Platform Efficiency for large-scale workloads

Martin Hilgeman
HPC Performance Lead
Distinguished Member of Technical Staff

martin.hilgeman@dell.com

@MartinHilgeman

DELL Technologies

Dell Technologies HPC market leadership

11 generations of technologies in HPC clusters

1999 — Cornell University Center for Advanced Computing — 1st HPC cluster

2001 — HPC solutions program officially launches

Industry's 1st HPC Solution Bundle

2004 — NCSA Tungsten #4

2005 — Sandia National Laboratories Thunderbird #6

DCS formed

2008 — C-series joins PowerEdge

2012 — TACC Texas Advanced Computing Center Stampede

2015 — Zenith System launched at HPC & AI Innovation Lab

2016 — Dell EMC merger — Isilon joins HPC portfolio

2017 — Dell EMC AI solutions announced

2018 — University of Michigan Great Lakes

2019 — TACC Texas Advanced Computing Center Frontera

2020 — First systems with DCLC (OSC) and HDR — Ohio Supercomputer Center An OH·TECH Consortium Member Pitzer

2021 — eni

DELLTechnologies

Dell Technologies HPC Strategy

DEMOCRATIZING

OPTIMIZING

ADVANCING

Dell Technologies HPC Strategy

DEMOCRATIZING

Enable everyone to accelerate science, engineering and analytics

OPTIMIZING

ADVANCING

DELL Technologies

Dell Technologies HPC Strategy

DEMOCRATIZING

OPTIMIZING

Offering a robust portfolio, optimized for HPC, providing performance and efficiency

ADVANCING

Dell Technologies HPC strategy

DEMOCRATIZING

OPTIMIZING

ADVANCING

Providing expertise, innovation and partnerships — from workgroup to the TOP500

Dell Technologies HPC and Deep Learning Team Charter

Design, develop and integrate HPC and DL systems
- Flexible reference architectures
- Systems tuned for research computing, manufacturing, life sciences, oil and gas, etc.

Investment - People and innovation ecosystem

Conduct application performance studies and develop best practices
- White papers, blogs, presentations
- www.hpcatdell.com

HPC and AI Innovation Lab

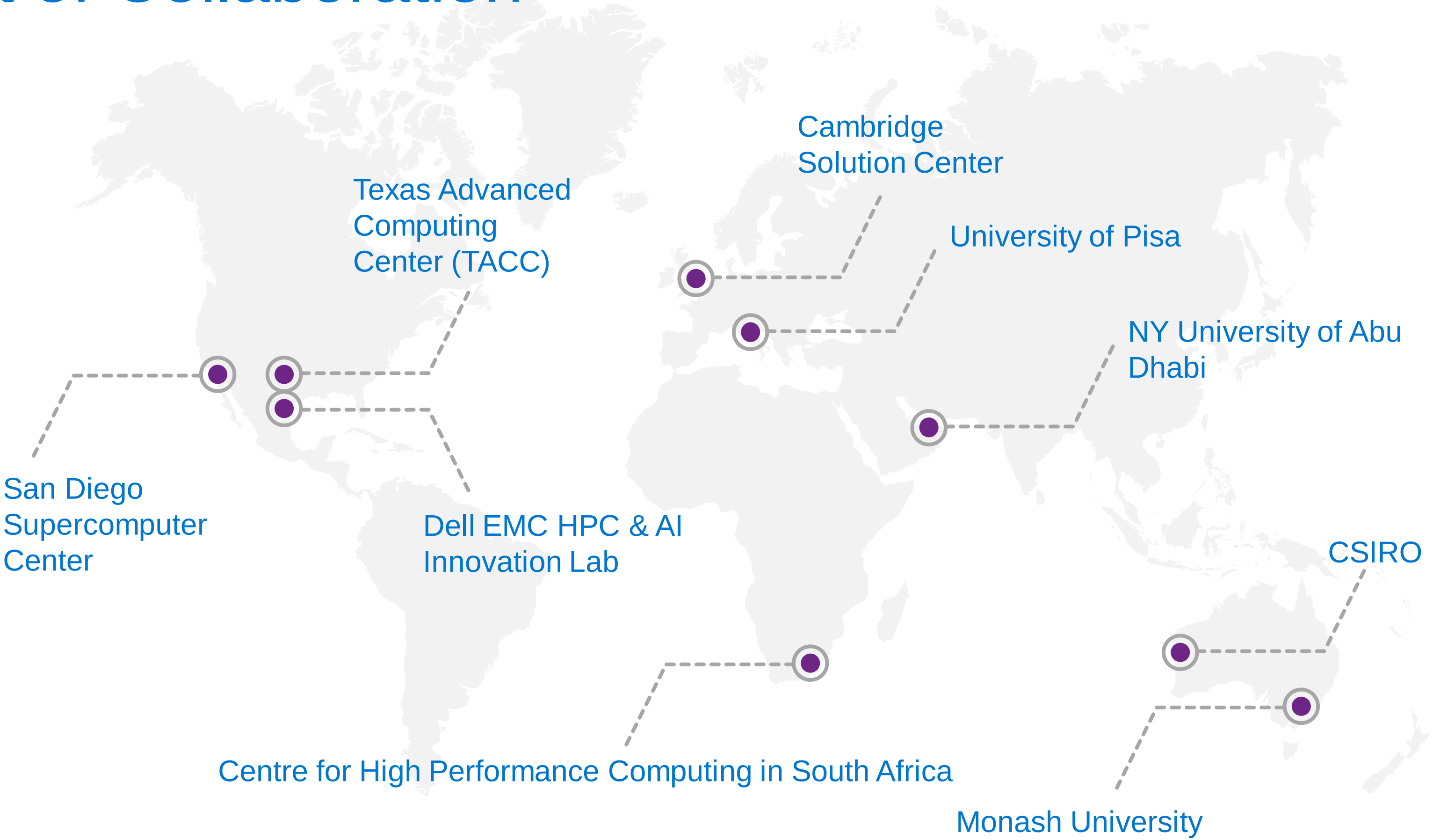
Act as the focal point for joint R&D activities
- Technology collaboration with partners for joint innovation
- Research coordination with Dell Solution Centers, Centers For Innovation and customers

Technical briefings, tours, remote access

Prototype and evaluate advanced technologies
- NVMe, FPGAs, Containers, DL workloads, new memory technologies, HPC+Cloud, etc.

www.dellemc.com/innovationlab

The Spirit of Collaboration



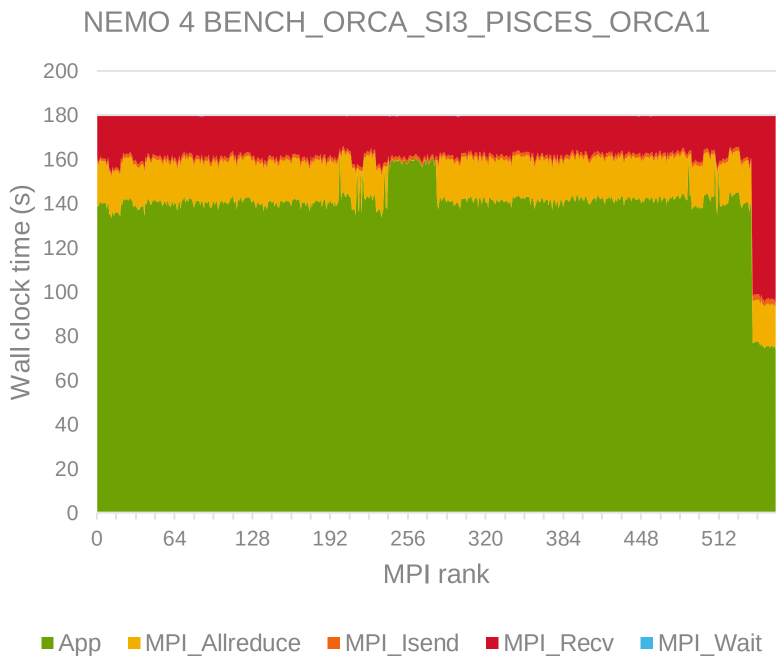
DELL Technologies

Examples of MPI studies

Collective MPI function rationale

- Collective functions are called by all ranks participating in the communicator used in the function
- A collective function implies a barrier
- Most MPI libraries have implemented various kernels that are tuned for a certain topology or message size:
  - Intel MPI: `I_MPI_ADJUST_ALLREDUCE=<val>` (1-Recursive doubling, 2-Rabenseifner's, 3-Reduce + Bcast, 4-Topology aware Reduce + Bcast, 5-Binomial gather + scatter, 6-Topology aware binominal gather + scatter, 7-Shumilin's ring, 8-Ring, 9-Knomial, 10-Topology aware SHM-based flat, 11-Topology aware SHM-based Knomial, 12-Topology aware SHM-based Knary)
  - Open MPI: `mpirun --mca coll_tuned_allreduce_algorithm <val>` (0-ignore, 1-basic_linear, 2-nonoverlapping, 3-recursive_doubling, 4-ring, 5-segmented_ring, 6-rabenseifner)
  - MVAPICH2: algorithms are used based on thresholds set by environment variables like `MV2_SHMEM_ALLREDUCE_MSG` and `MV2_ALLREDUCE_2LEVEL_MSG`

Collective MPI function rationale



- Collective functions spend a lot of time “fixing“ load imbalance
- MPI ranks spend different amounts of time in MPI_Allreduce

MPI-3 feature: non-blocking collectives

- MPI-3 has added support for non-blocking collectives
  - They combine the potential benefits of nonblocking point-to-point operations, to exploit overlap and to avoid synchronization, with the optimized implementation and message scheduling provided by collective operations
  - The call initiates the operation, which indicates that the system may start to copy data out of the send buffer and into the receive buffer
  - Workings are similar to non-blocking point to point calls (Isend/Irecv)

- Blocking:

```
int MPI_Allreduce(  const void *sendbuf, void *recvbuf,
                    int count, MPI_Datatype datatype, MPI_Op op,
                    MPI_Comm comm)
```

- Non-blocking:

```
int MPI_Iallreduce(const void *sendbuf, void *recvbuf,
                   int count, MPI_Datatype datatype, MPI_Op op,
                   MPI_Comm comm, MPI_Request *request)
```

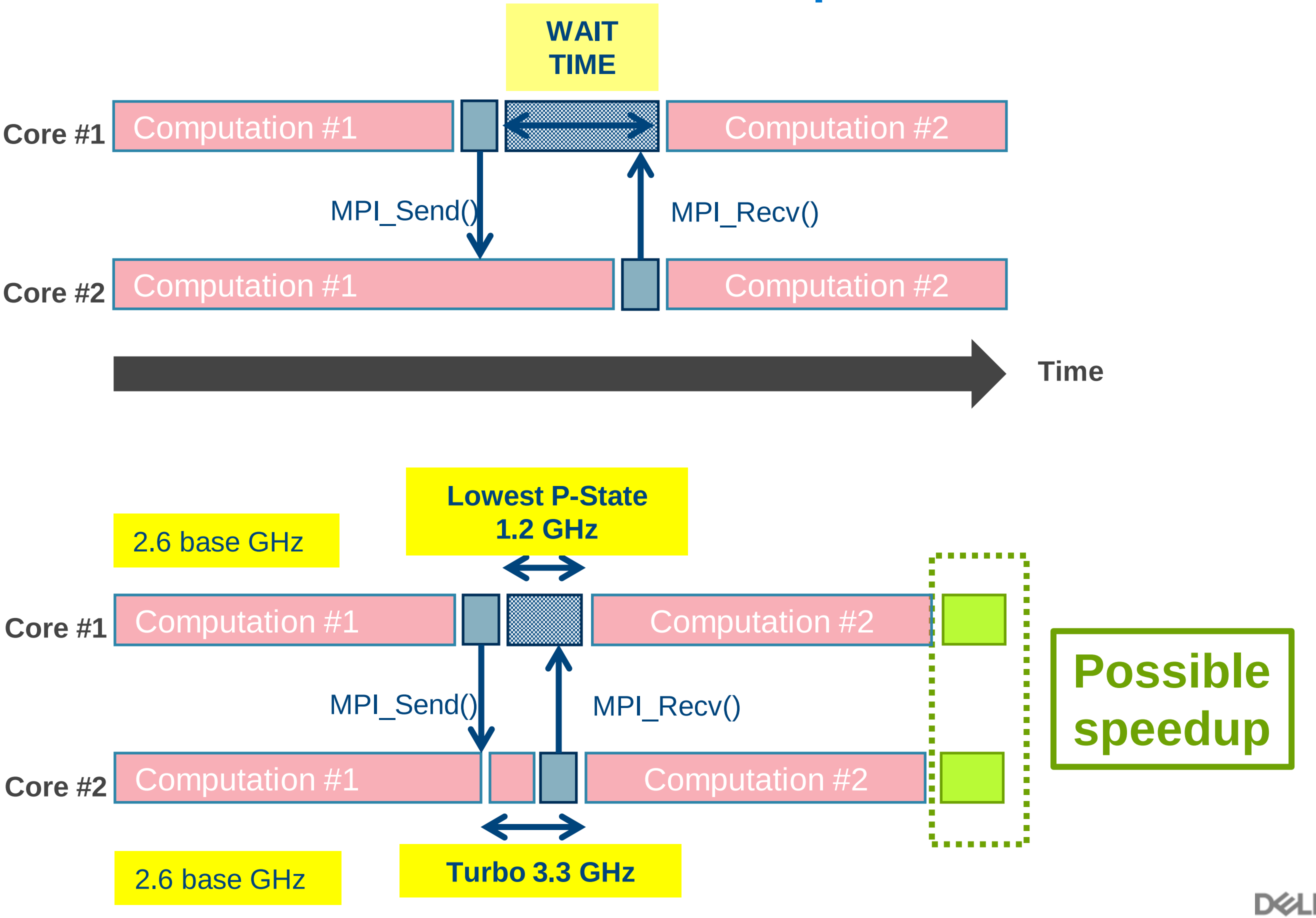
The simple collective example – non blocking

```c
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>

int main(int argc, char* argv[])
{
    int size, my_rank, result = 0;
    MPI_Request request;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    MPI_Iallreduce(&my_rank, &result, 1, MPI_INT,
MPI_SUM, MPI_COMM_WORLD, &request);

    printf("[MPI Process %d] Doing something else
here.\n", my_rank);

    MPI_Wait(&request, MPI_STATUS_IGNORE);

    printf("[MPI Process %d] The sum of all ranks is
%d.\n", my_rank, result);

    MPI_Finalize();

    return EXIT_SUCCESS;
}
```

- Similar calling sequence as MPI_Allreduce
- MPI_Request parameter is extra, just like in Isend/Irecv

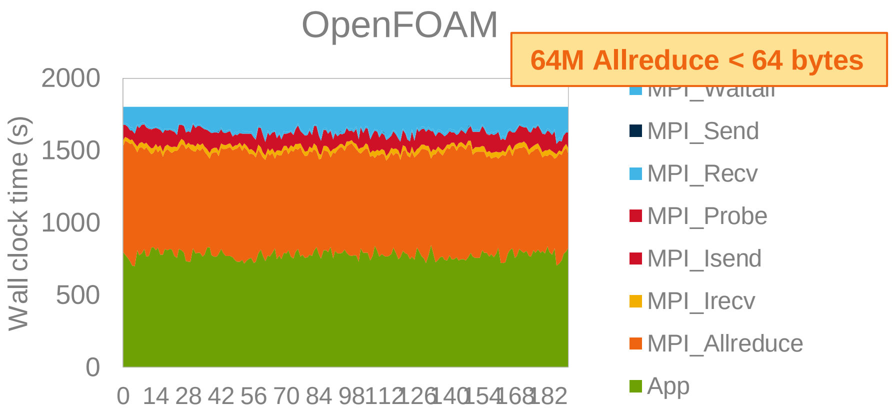
```
% mpirun -np 4 iallreduce.exe
[MPI Process 3] Doing something else here.
[MPI Process 0] Doing something else here.
[MPI Process 1] Doing something else here.
[MPI Process 1] The sum of all ranks is 6.
[MPI Process 2] Doing something else here.
[MPI Process 2] The sum of all ranks is 6.
[MPI Process 3] The sum of all ranks is 6.
[MPI Process 0] The sum of all ranks is 6.
```

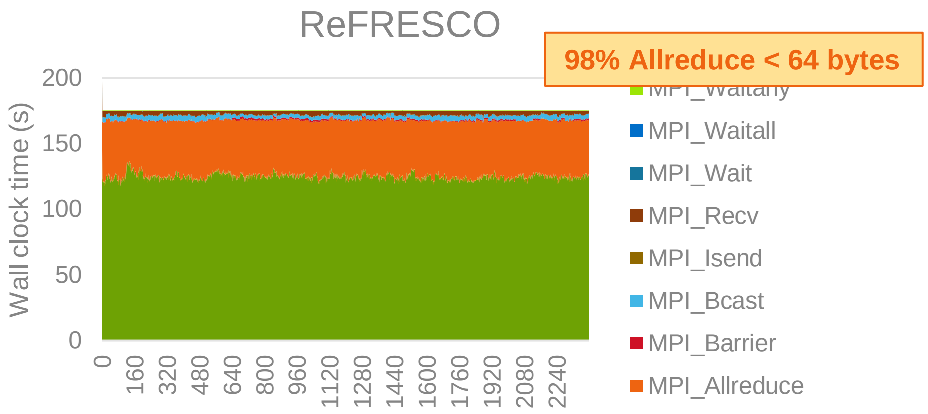
The effect of turbo mode could help load imbalance

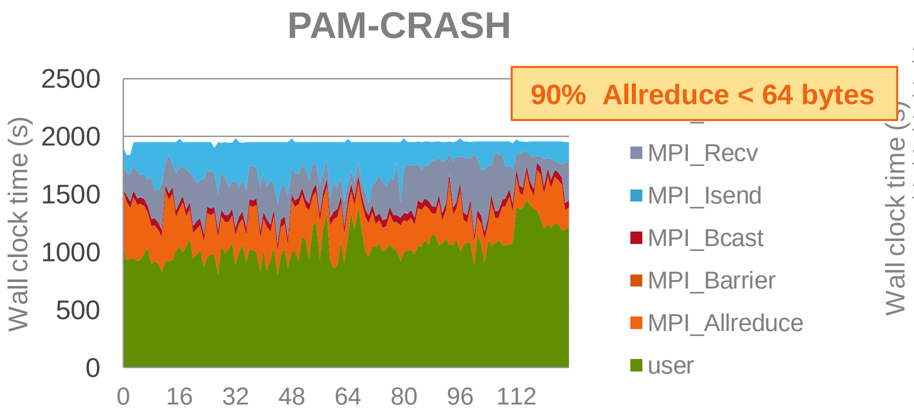


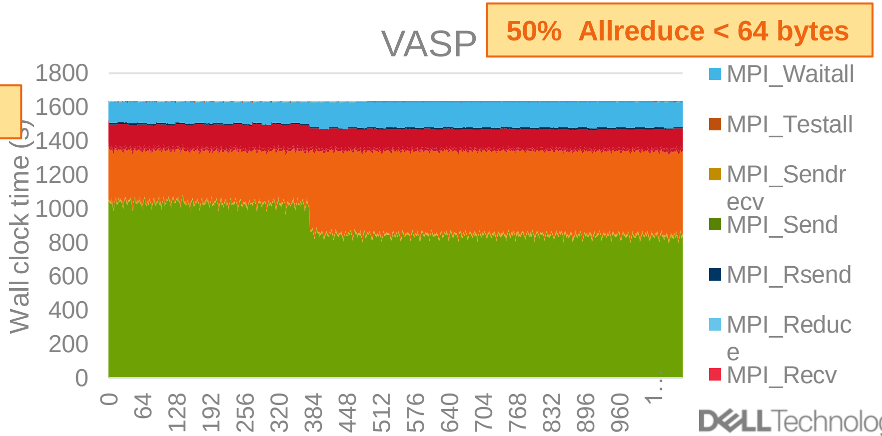
DELL Technologies

Small MPI_Allreduce is killing MPI performance

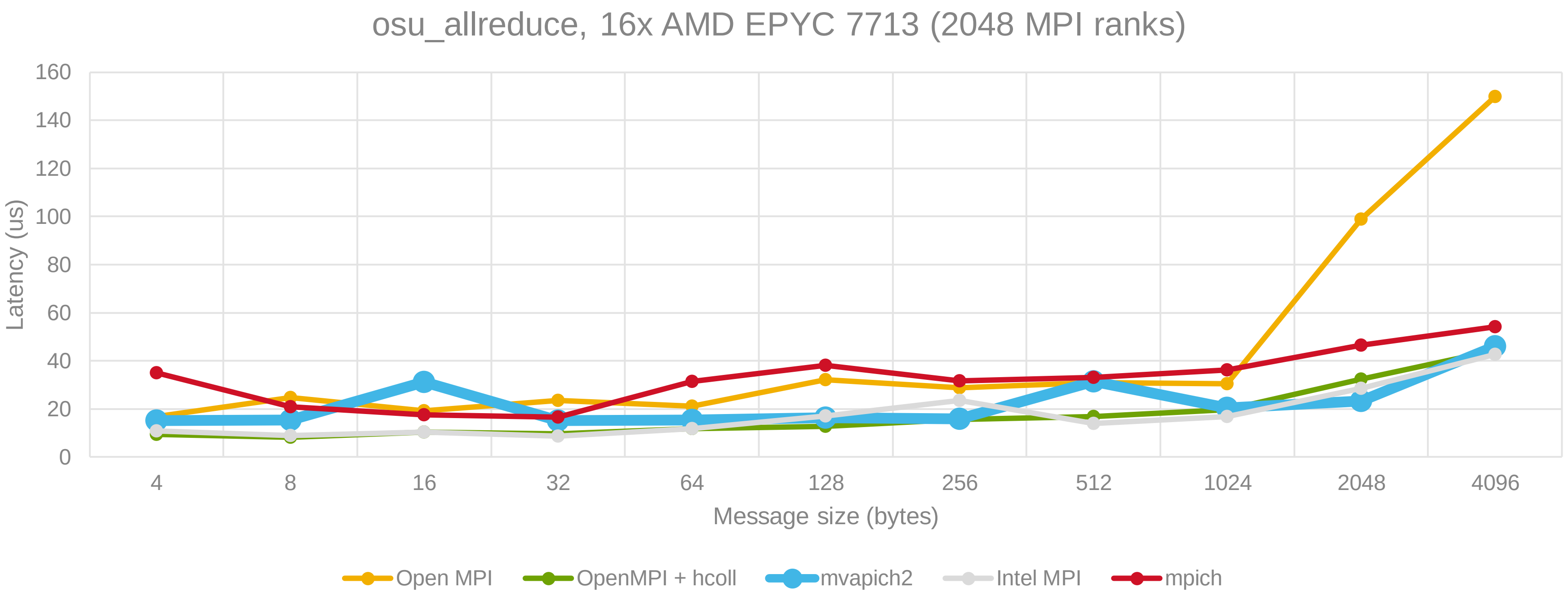




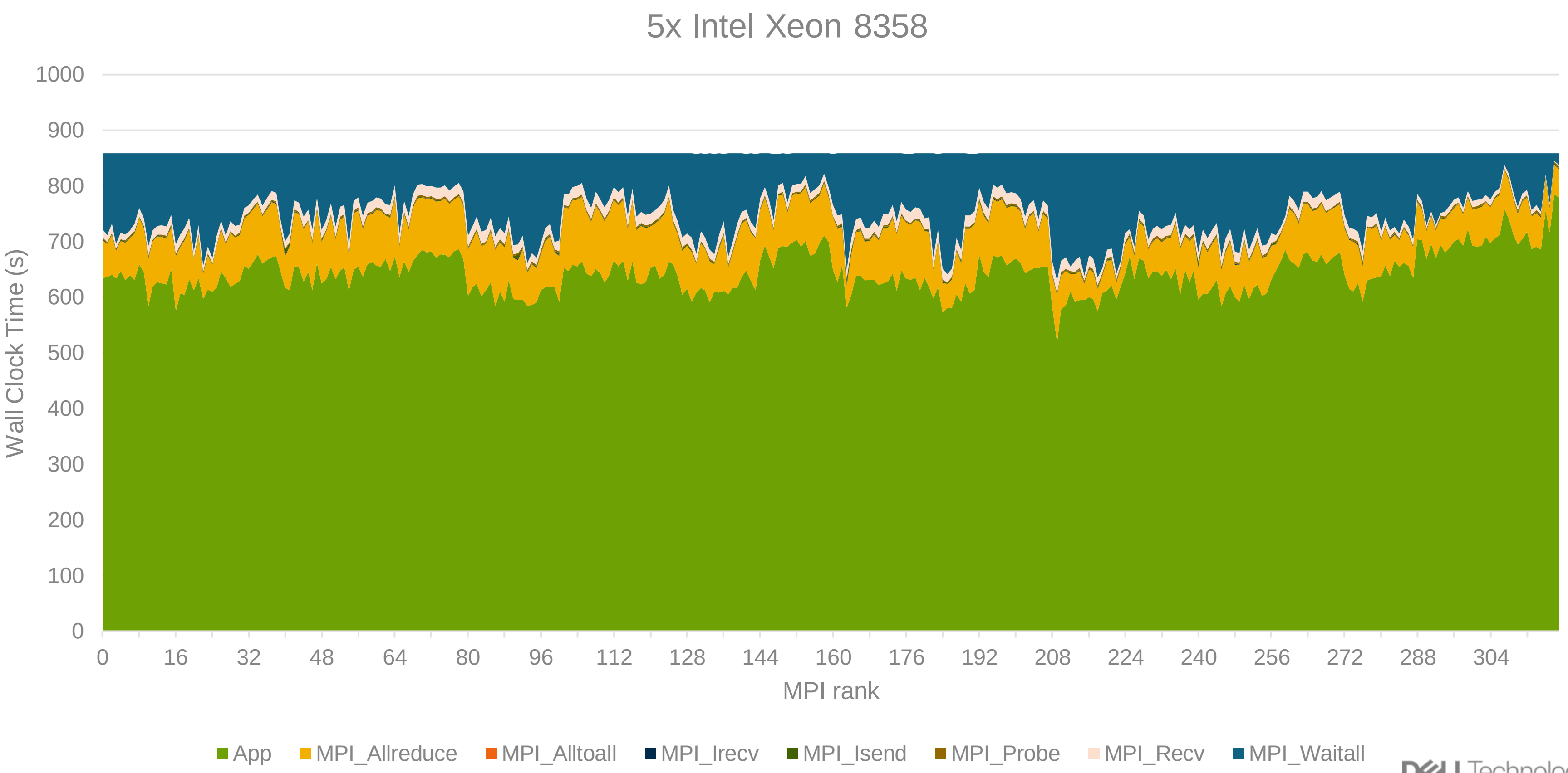




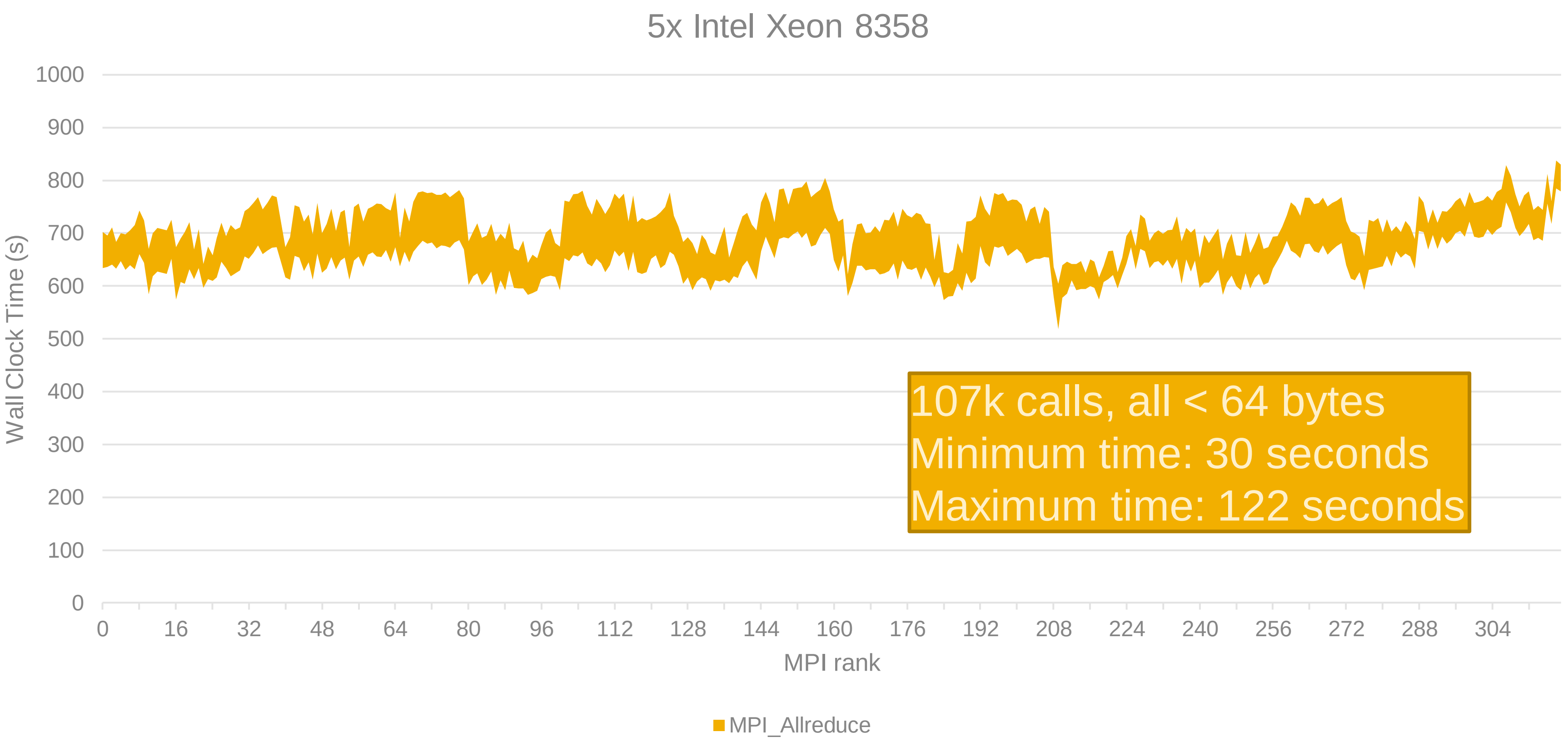
Allreduce for small message sizes



OpenFOAM v2106 simpleBenchmark2006



OpenFOAM v2106 simpleBenchmark2006

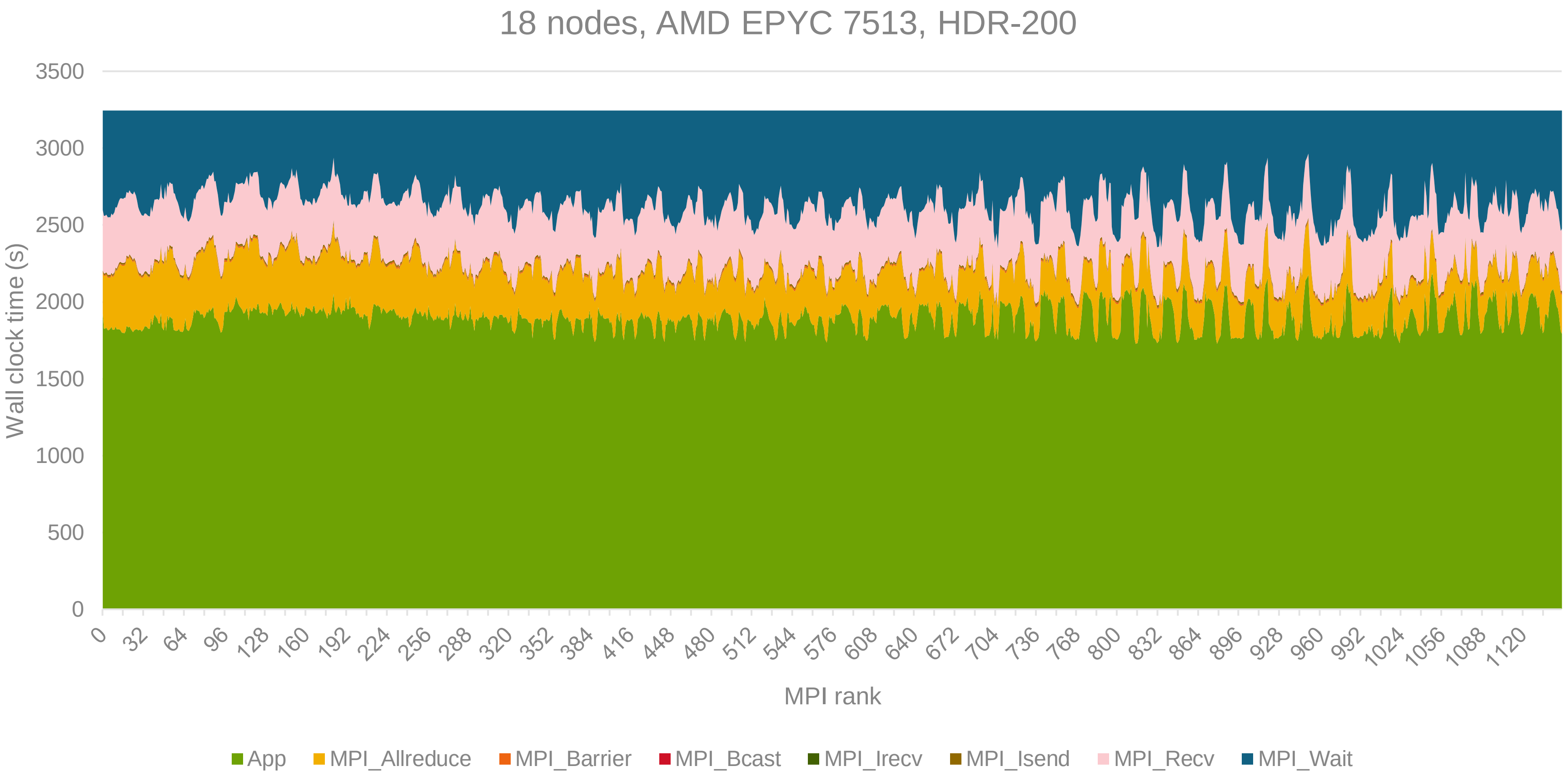


Looking for real communication overhead

- We have seen a load imbalance in the OpenFOAM benchmark, what about real communication overhead?
- Profiling library simulates collective operations as follows:
  - `PMPI_Barrier(comm)` **is doing an explicit synchronization to rule out load imbalance**
  - **The "real" MPI_Allreduce call is now on perfectly synchronized data**

```
int MPI_Allreduce(const void *sendbuf, void *recvbuf, int count,
MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)
{
    int ret, len = 0, msgsize = 0;
    double t0, t1, t2;

    t0 = my_time();
    if ( datatype != MPI_DATATYPE_NULL ) {
        PMPI_Type_size(datatype, &msgsize);
        len = count * msgsize;
    }

    if (libtoolbox.coll_stats) {
        PMPI_Barrier(comm);
        t2 = my_time() - t0;
        mpi.coll_time[ALLREDUCE] += t2;
    }

    ret = PMPI_Allreduce(sendbuf, recvbuf, count, datatype, op,
comm);

    if (libtoolbox.stats) {
        t1 = my_time() - t0;
        mpi.time[MPI_ALLREDUCE] += t1;
        mpi.len[MPI_ALLREDUCE] += len;
        update_bin(MPI_ALLREDUCE, len);
    }

    return ret;
}
```

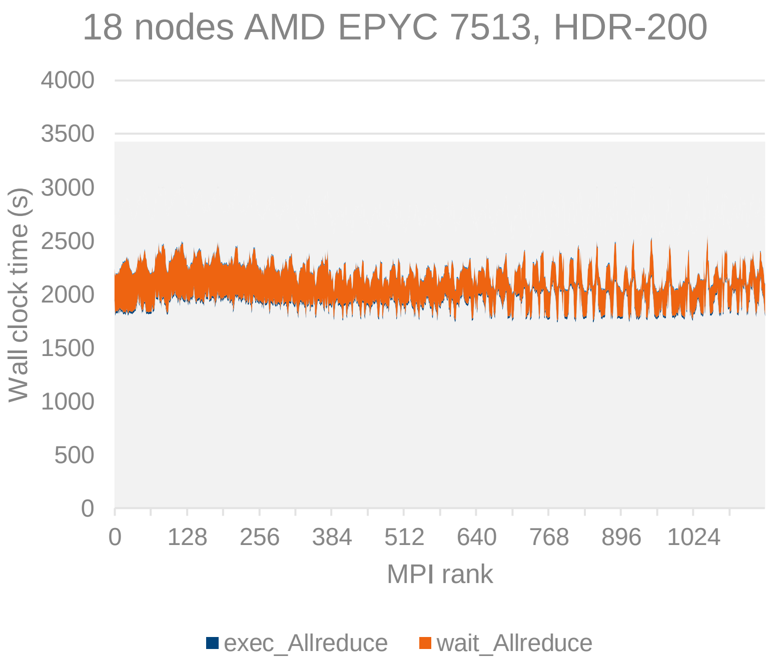
Allreduce example: MOM5 benchmark



Result of perfect interconnect simulation (zoomed in)

- Wait_*: load imbalance
- Exec_*: actual communication
- Communication overhead is negligible
- Load imbalance is the cause of parallel overhead and potential scaling issue

123k calls, 119k < 64 bytes
Minimum time: 191 seconds
Maximum time: 463 seconds
Transfer time: 18 seconds

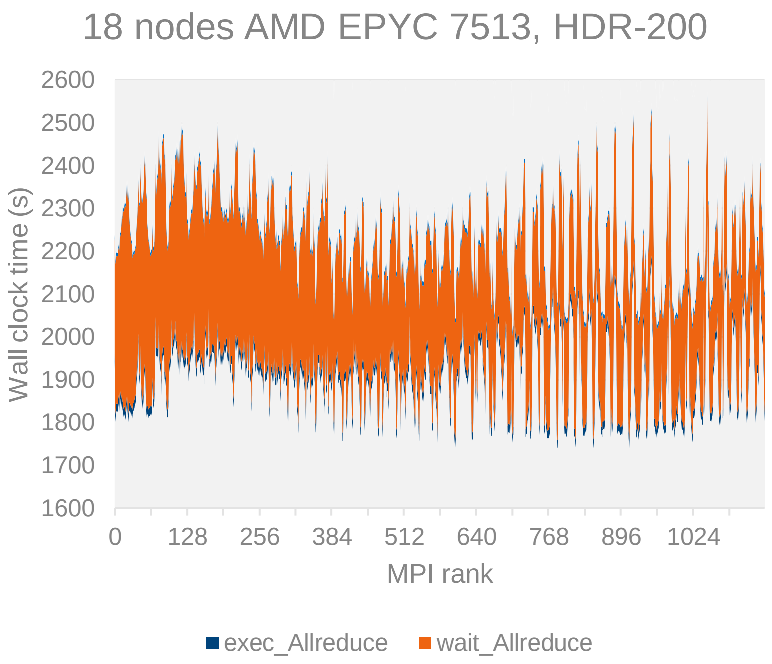


DELL Technologies

Result of perfect interconnect simulation (zoomed in)

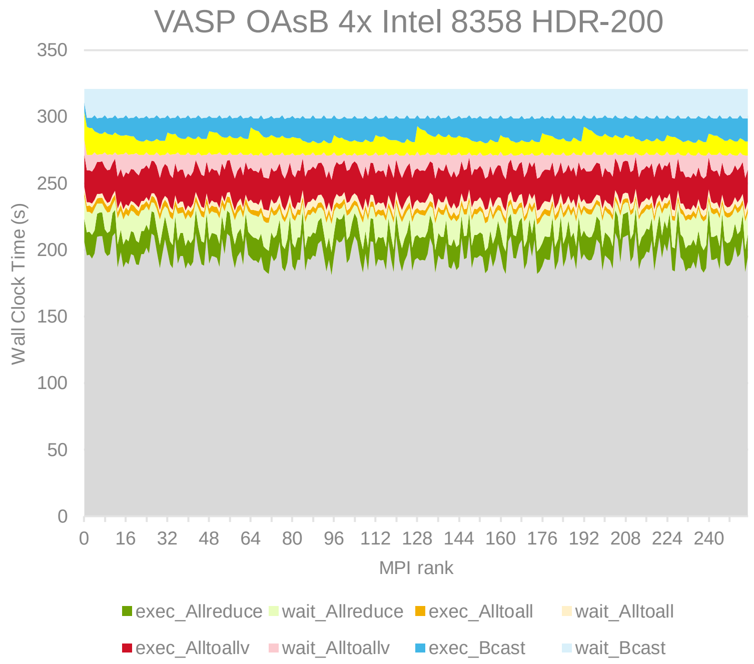
- Wait_*: load imbalance
- Exec_*: actual communication
- Communication overhead is negligible
- Load imbalance is the cause of parallel overhead and potential scaling issue

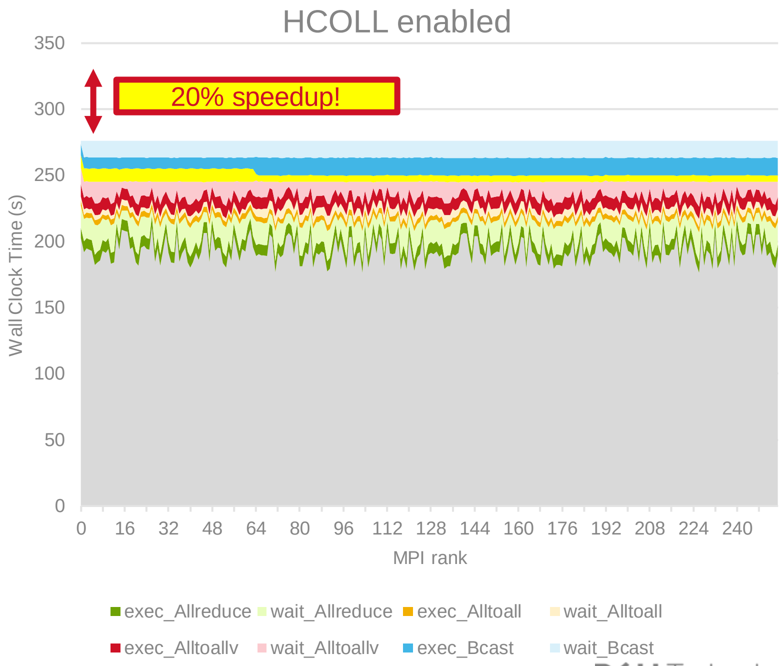
123k calls, 119k < 64 bytes	
Minimum time:	191 seconds
Maximum time:	463 seconds
Transfer time:	18 seconds



Use hardware offload of the IB card for collectives

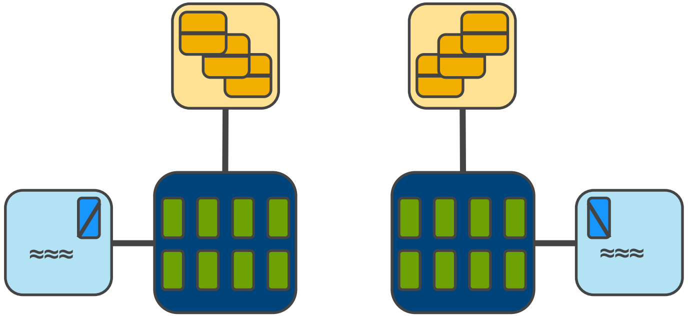
```
% mpirun -genv HCOLL_ENABLE 1             -np 512 <app>   # MVPAPICH2
% mpirun --mca coll_hcoll_enable 1        -np 512 <app>   # Open MPI
% mpirun -genv I_MPI_COLL_EXTERNAL hcoll -np 512 <app>   # Intel MPI
```





DELL Technologies

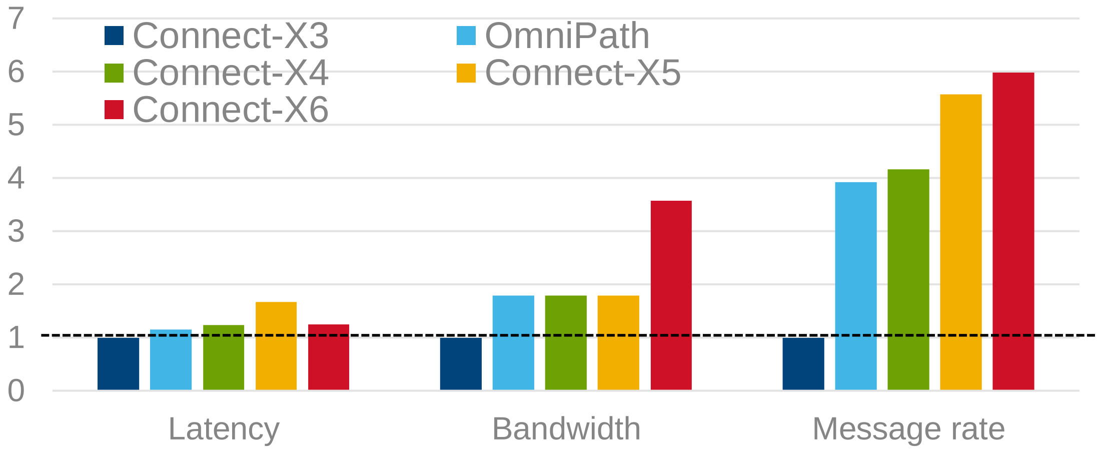
Message rate matters!



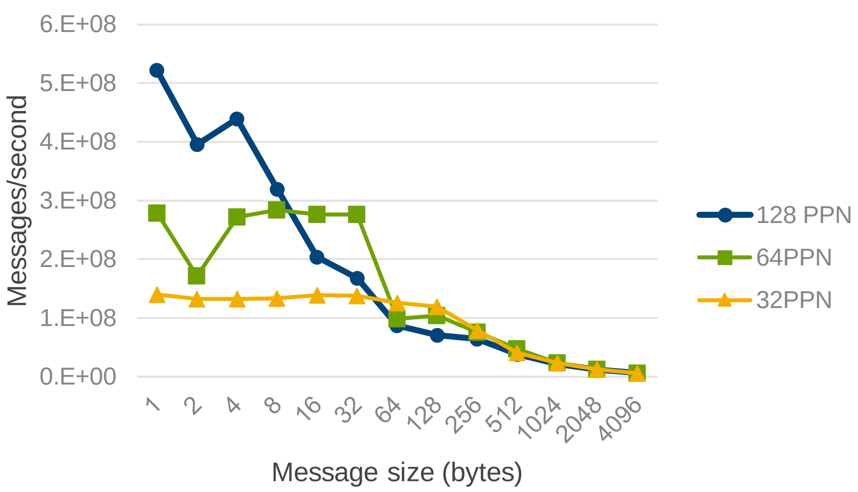
MPI_Allreduce() between two nodes

2012: 16 cores per node, one FDR IB HCA: 2 x 16 = 32 transfers (messages)

2022: 128 cores per node, one HDR IB HCA: 2 x 128 = 256 transfers (messages)

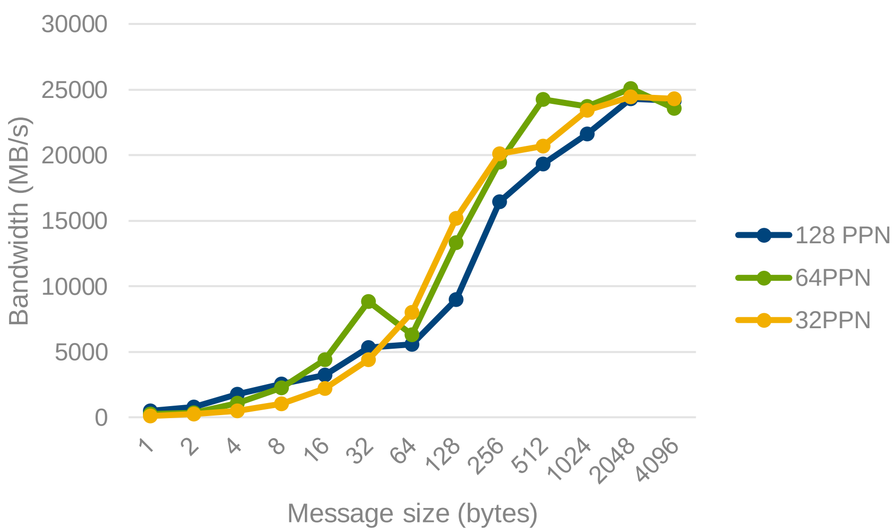


AMD EPYC 7713 message rate

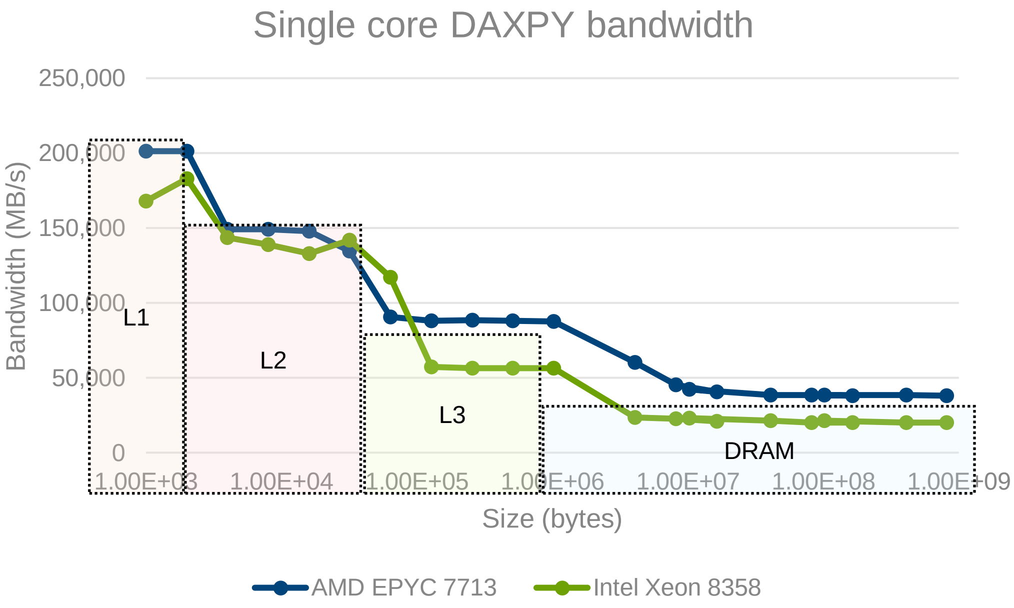


- Adding more cores helps message rate for small messages

- But it doesn't add bandwidth



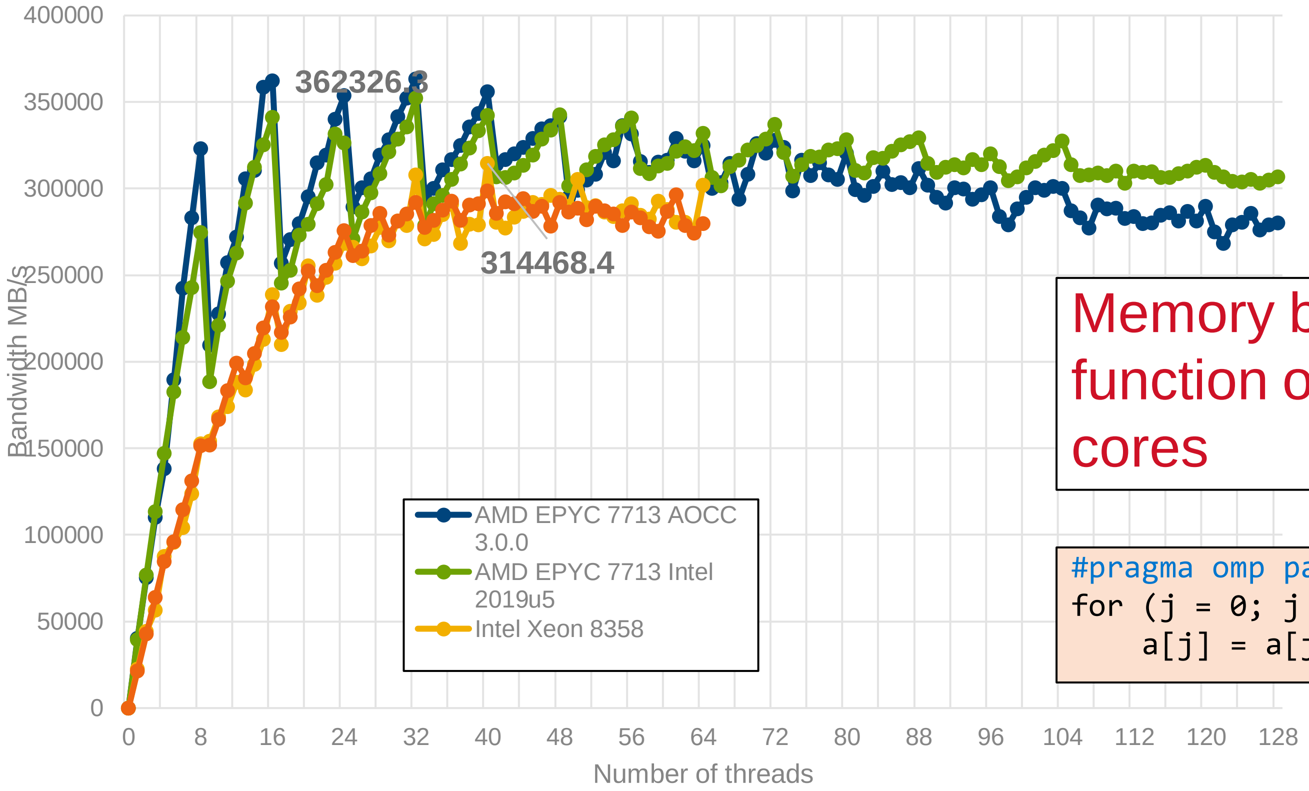
Measuring single core memory performance



DAXPY kernel

```
#pragma ivdep
for (j = 0; j < SIZE; j++)
    a[j] = a[j] + scalar * c[j];
```

Measuring node memory performance

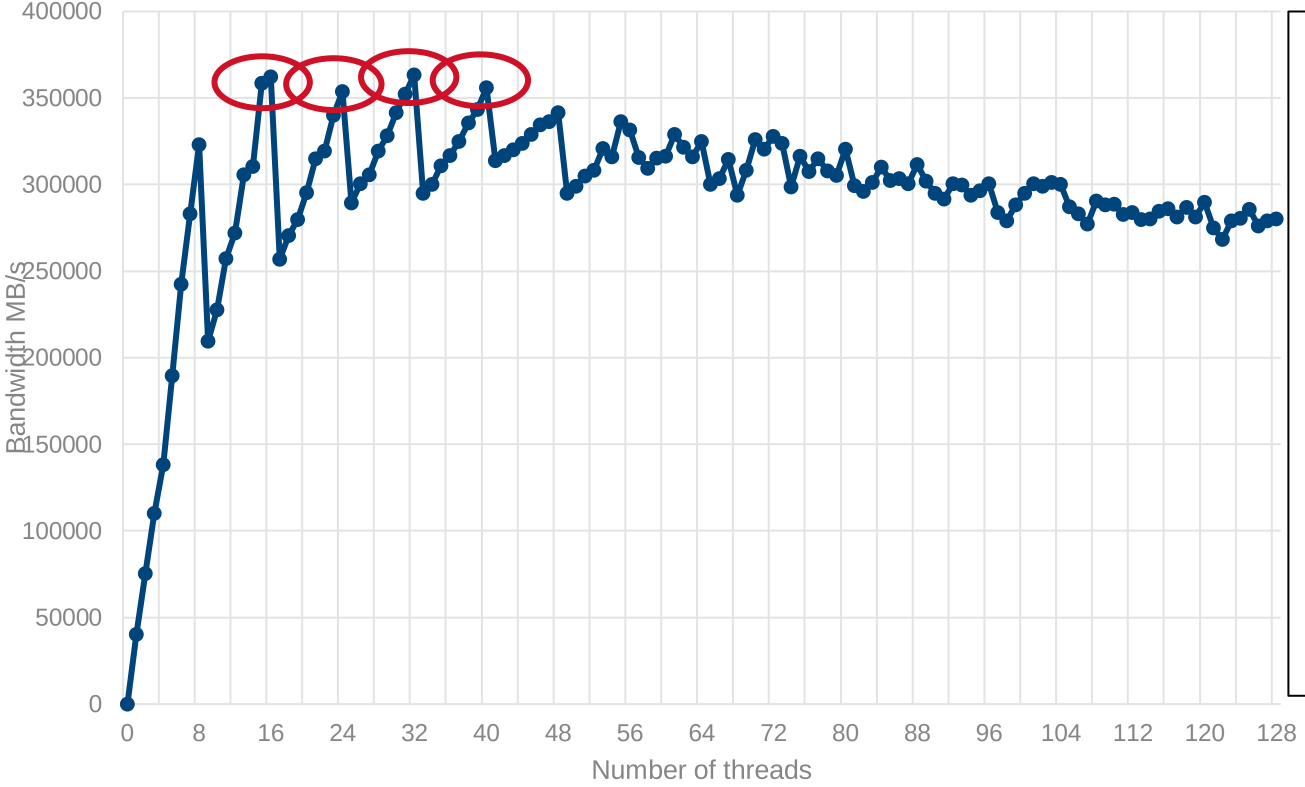


Memory bandwidth as a function of the number of cores

```
#pragma omp parallel for
for (j = 0; j < STREAM_ARRAY_SIZE; j++)
    a[j] = a[j] + scalar * c[j];
```

DELLTechnologies

Why is this important?



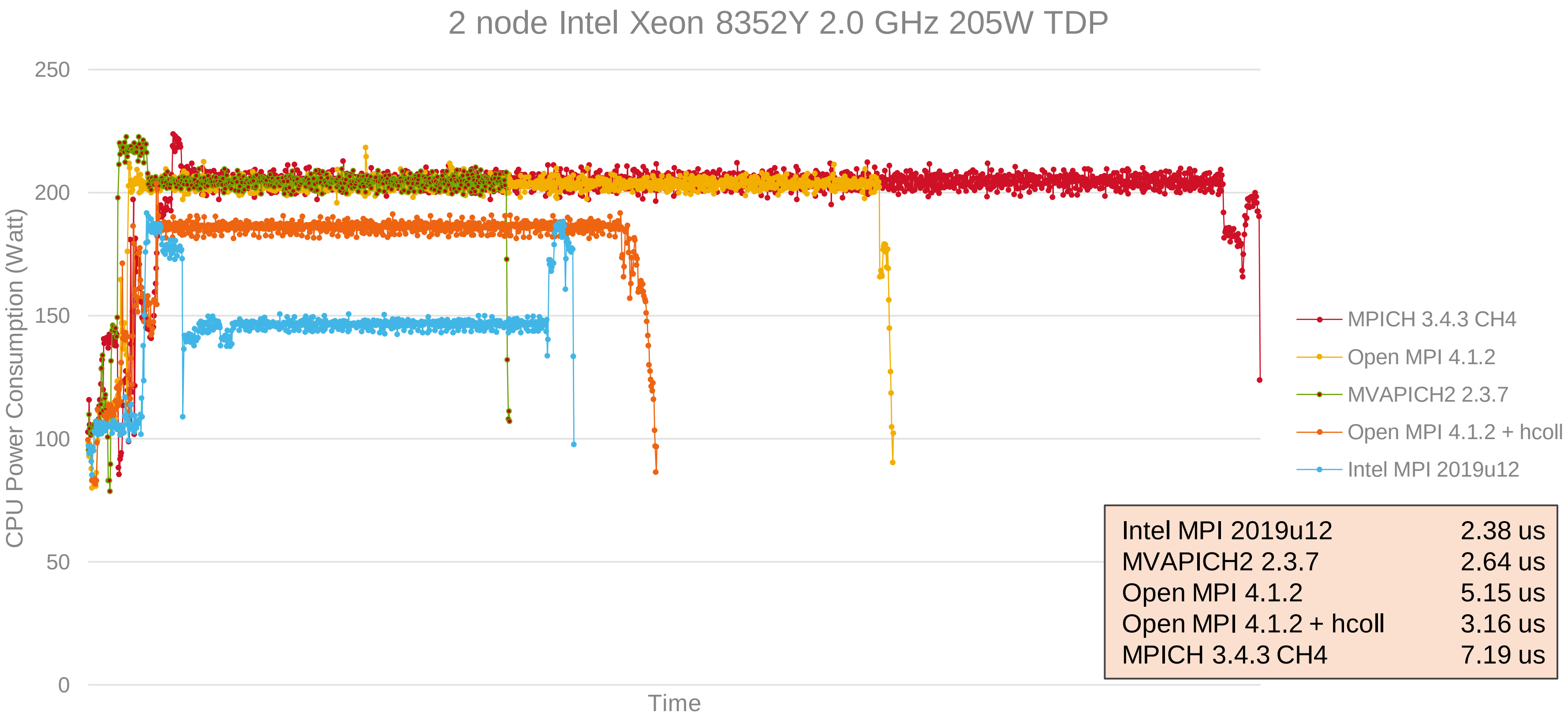
Maximum memory bandwidth at

- 16 cores
- 24 cores
- 32 cores
- 40 cores

30% better than on 128 cores

DELL Technologies

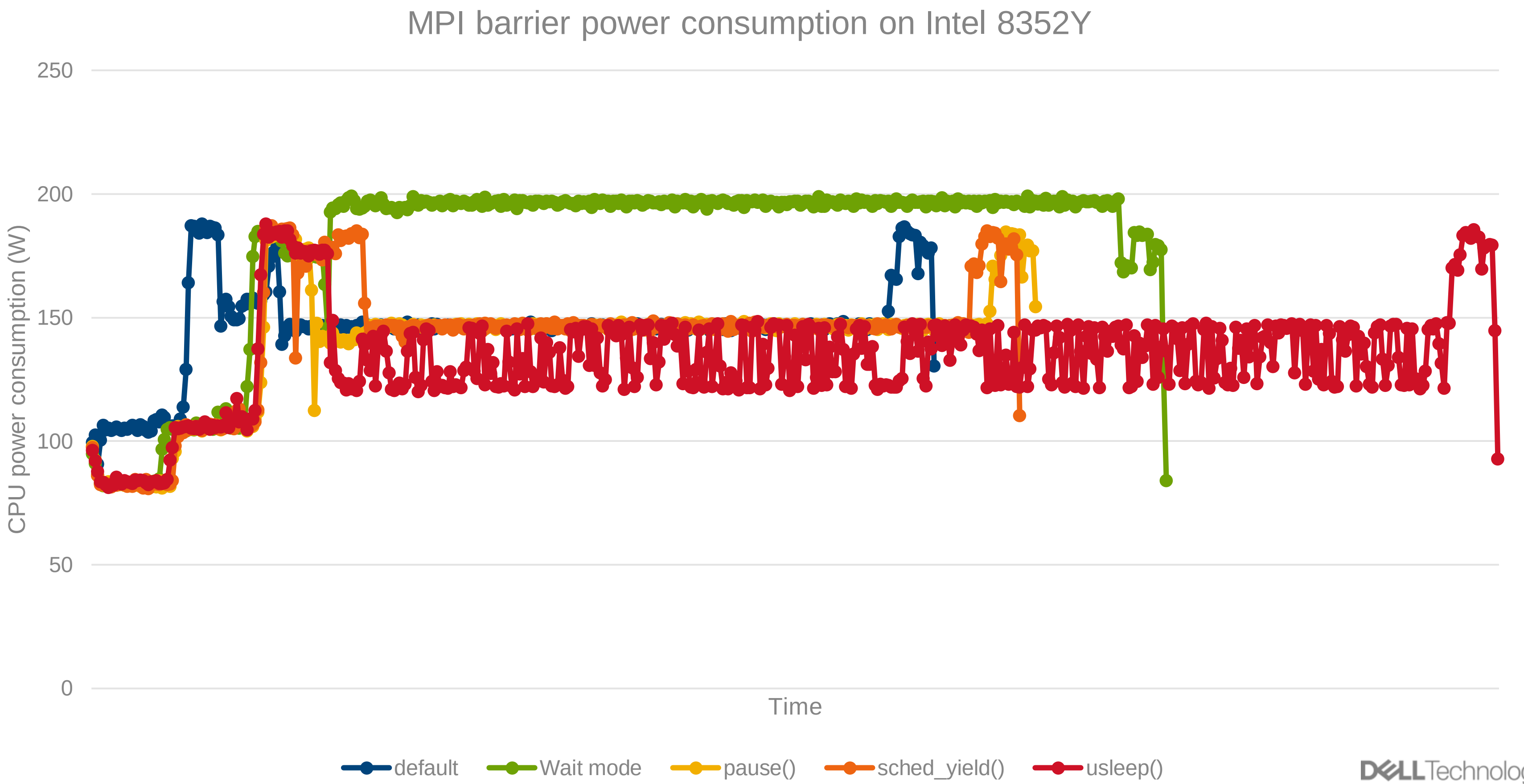
MPI_Barrier() power consumption



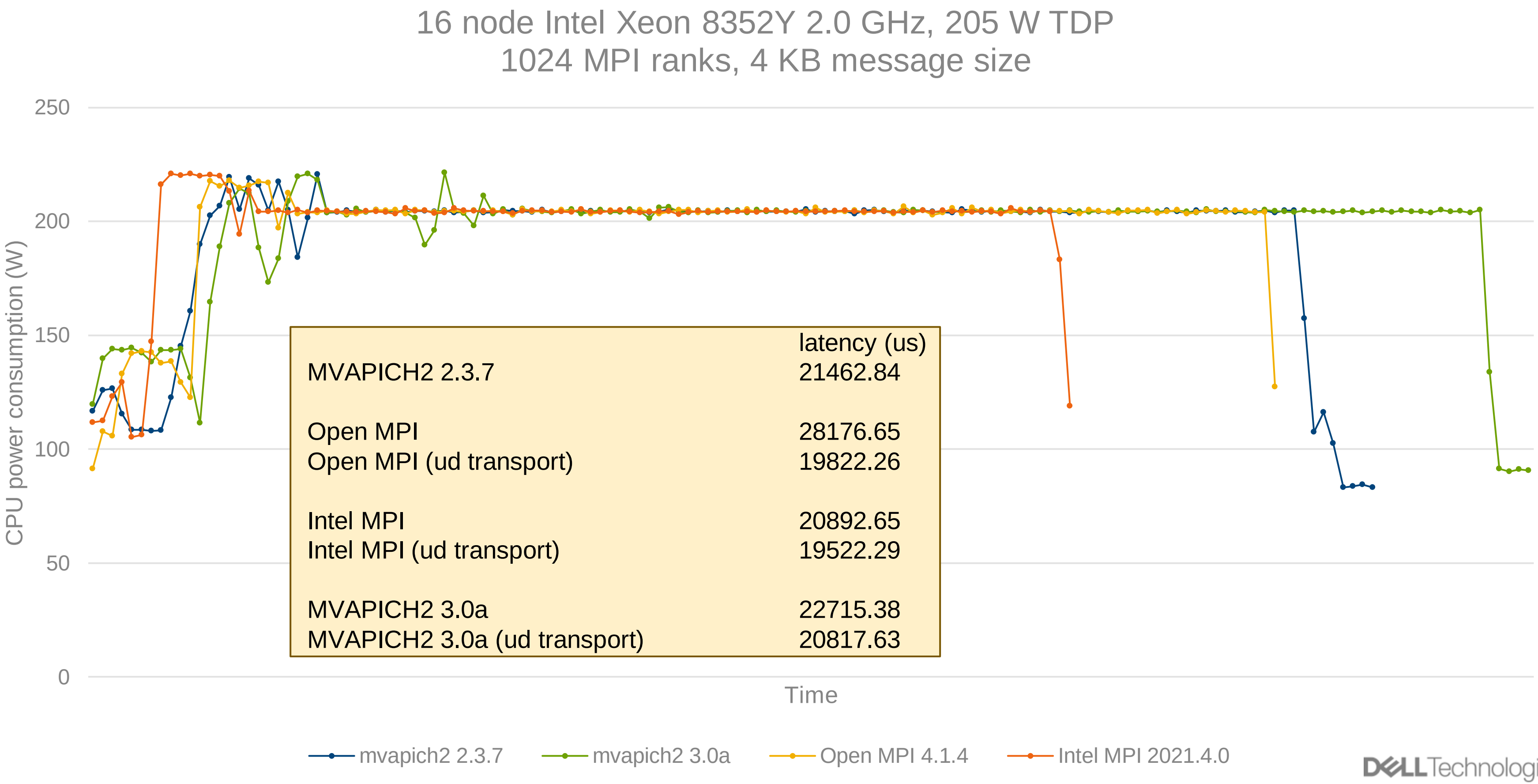
Intel MPI 2019u12	2.38 us
MVAPICH2 2.3.7	2.64 us
Open MPI 4.1.2	5.15 us
Open MPI 4.1.2 + hcoll	3.16 us
MPICH 3.4.3 CH4	7.19 us

DELL Technologies

MPI_Barrier() power consumption



MPI_Alltoall() power consumption



	latency (us)
MVAPICH2 2.3.7	21462.84
Open MPI	28176.65
Open MPI (ud transport)	19822.26
Intel MPI	20892.65
Intel MPI (ud transport)	19522.29
MVAPICH2 3.0a	22715.38
MVAPICH2 3.0a (ud transport)	20817.63

DELL Technologies

THANK YOU FOR BEING PART OF THE COMMUNITY

- Dell Technologies HPC Community: dellhpc.org
- HPC & AI Engineering: hpcatdell.com and https://infohub.delltechnologies.com/t/high-performance-computing/
- HPC & AI Innovation Lab: delltechnologies.com/innovationlab
- HPC/AI Centers of Excellence: delltechnologies.com/coe
- delltechnologies.com/hpc
- Stay tuned to https://insidehpc.com/dell