MPI Overlap: Benchmark and Analysis

Alexandre Denis
Inria Bordeaux, France
Alexandre.Denis@inria.fr

François Trahay
CNRS Samovar, Télécom SudParis, Université Paris-Saclay, France
Francois.Trahay@telecom-sudparis.eu

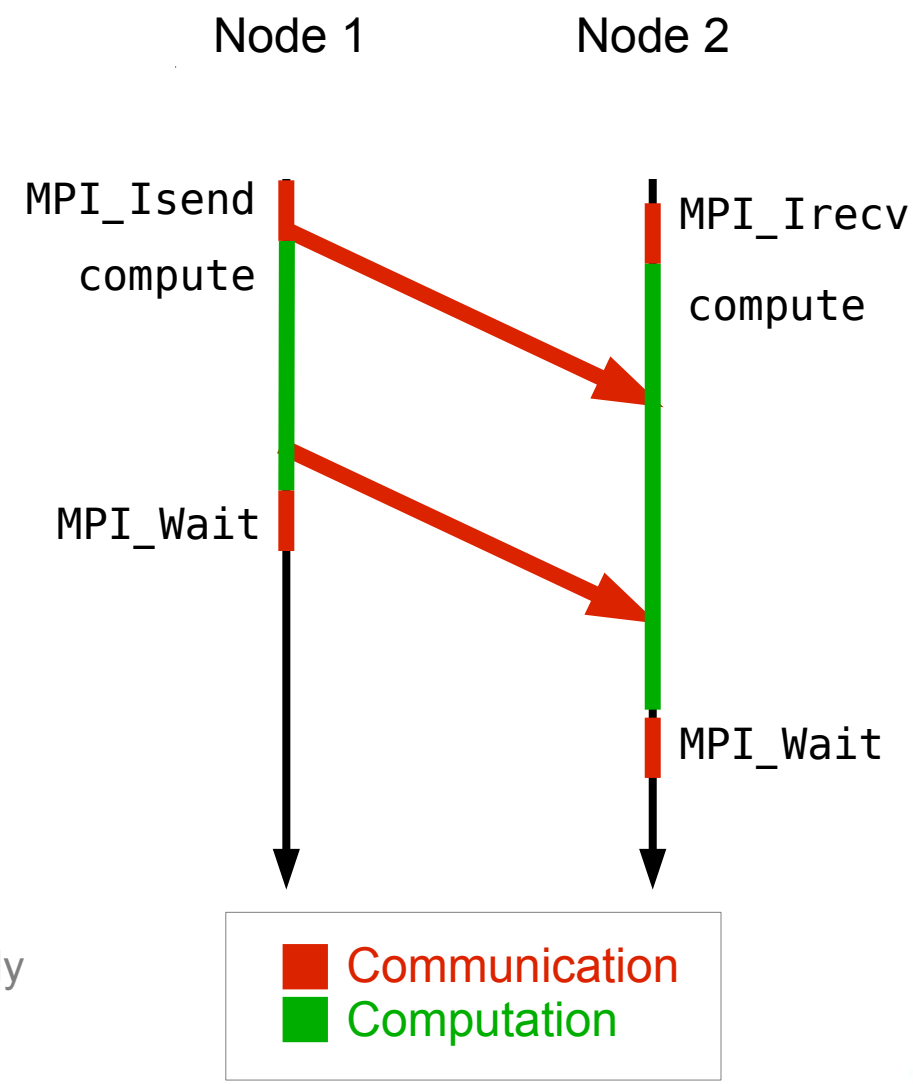
ICPP - Philadelphia - August 2016

1

MPI Overlap Benchmark

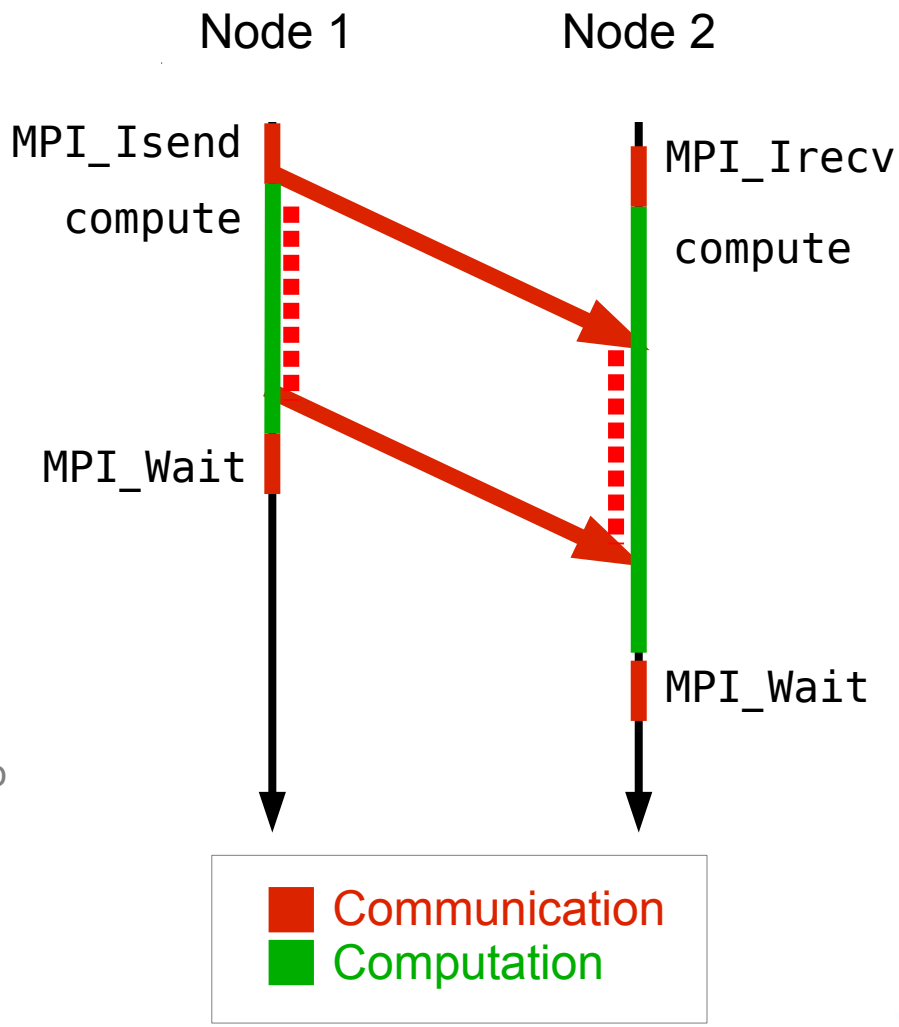
MPI Overlap

- **Amortize communication cost** through overlap
- Non-blocking MPI communications
  - **Post** a non-blocking communication request
  - Do some **computation**
  - **Wait** for request completion
  - Communication progressed in background
- **Application programmers assume communications progress** automagically
  - But **not guaranted** by MPI spec!



Communication Progression

- Communication may progress:
  - Only **inside MPI_* function** calls i.e. implementation is synchronous
    - Permitted by the MPI spec
    - No background progression
  - Using **NIC offloading**
    - Setup DMA, then let the hardware run the transfer
    - Limited to what the hardware can do
  - Using **threads**
    - pthreads, kernel tasklets, dedicated core, etc.

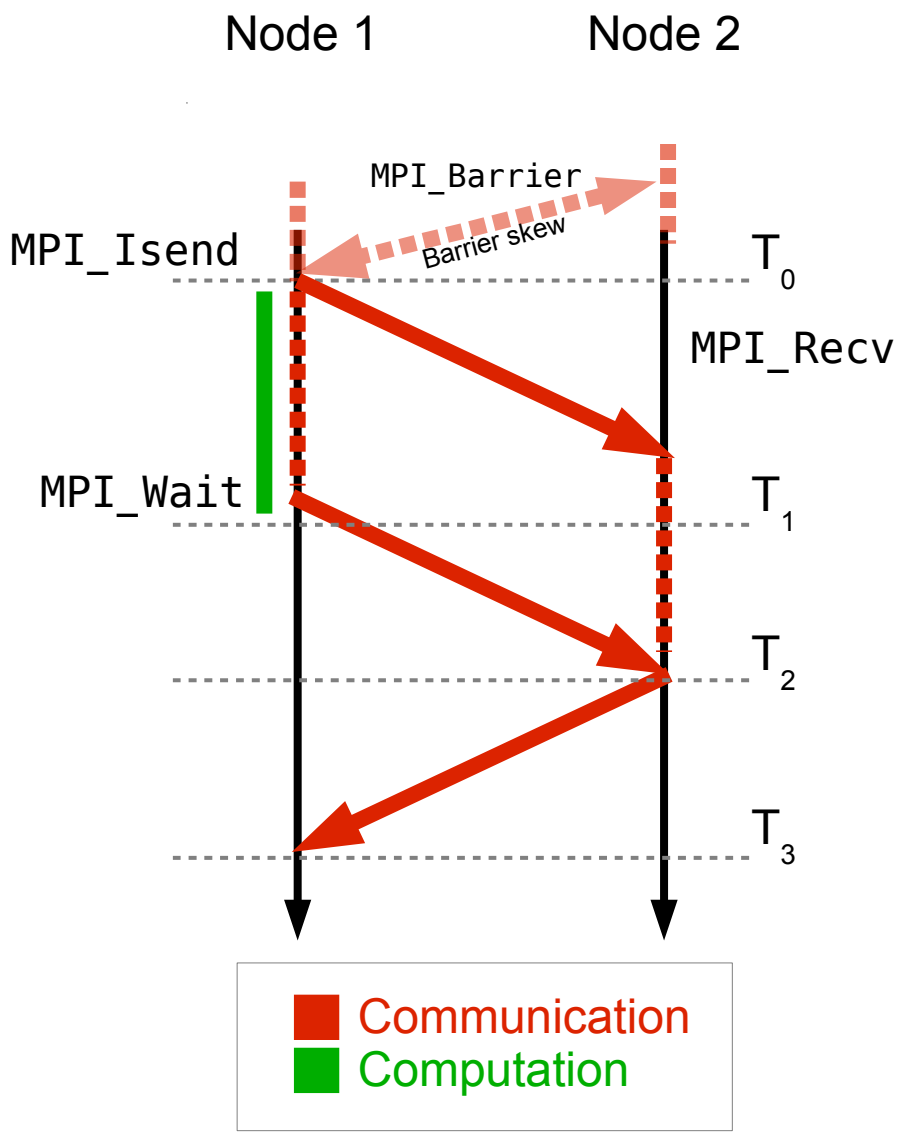


Features to benchmark

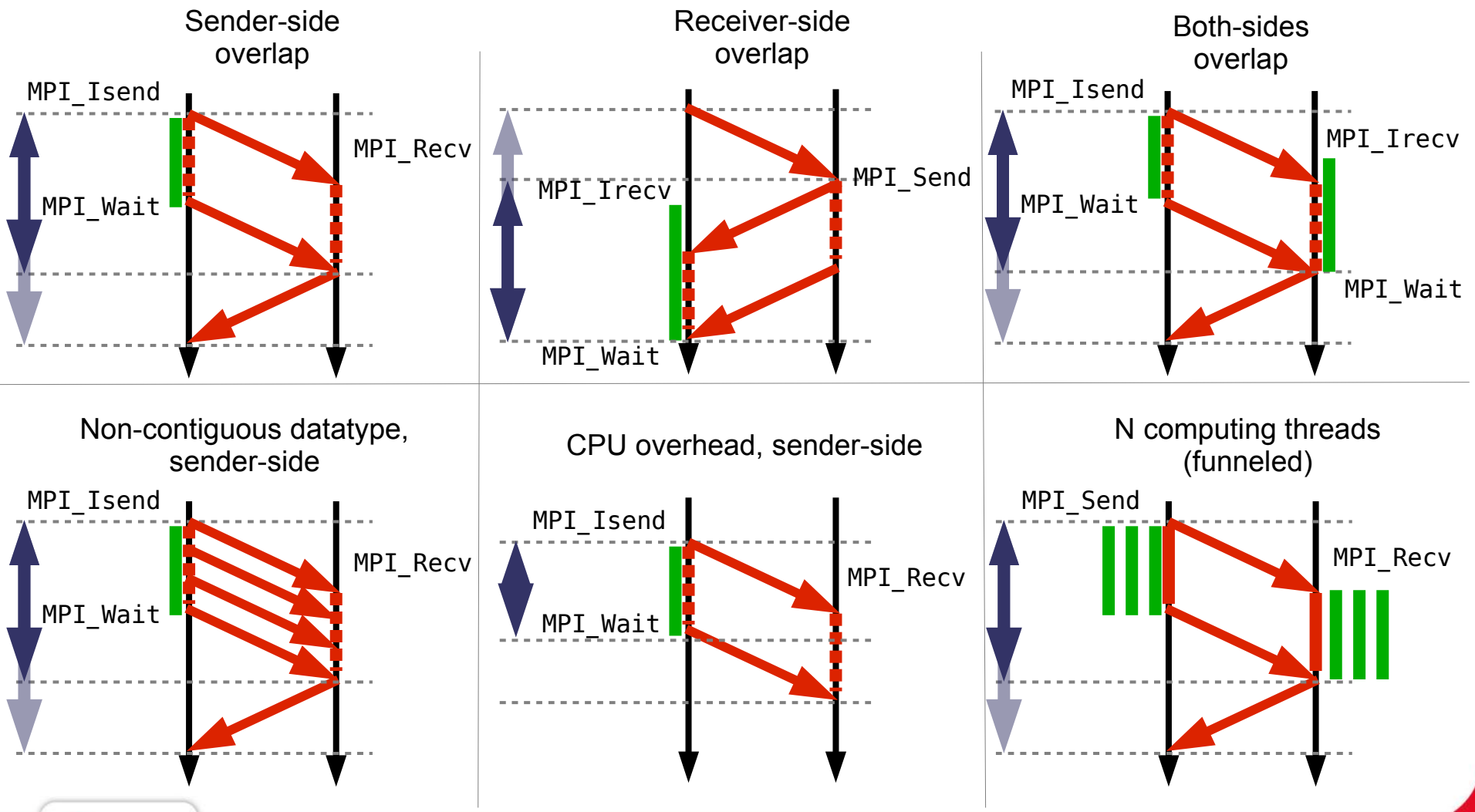
- Exploration space : **2D space** (message size x computation time)
  - Local behavior may not represent global behavior
    - e.g. thresholds on message size (PIO/DMA/rendez-vous, cache effects), thresholds on time (fixed cost overhead, TCP timers, etc.)
- Sender side, **receiver side**, both sides
  - Internal protocol (rendez-vous, RDMA write, etc.) is likely to exhibit non-symmetrical properties
- Contiguous, **non-contiguous datatype**
  - NIC offloading is ok for contiguous data blocks, but what about non-contiguous derived datatypes?
- **CPU overhead**
  - Measure CPU consumption used for communication progression
- **Multi-threaded** computation
  - Blocking communication in one thread, computation in other threads

Measure overlap

- Clock start/stop options:
  - $T_2 - T_0$: transfer time with overlap
    - Need global clock
  - Round-trip with overlap on both ways
    - Only for overlap on sender side
    - No way to synchronize computation start and receive
  - **$T_1 - T_0$**: send time
    - No guarantee data arrived – may have been buffered
    - Only sender side
  - **$T_3 - T_0$**: uncertainty on T_3-T_2



Benchmarks


Sender-side overlap
MPI_Isend
MPI_Recv
MPI_Wait
Receiver-side overlap
MPI_Irecv
MPI_Send
MPI_Wait
Both-sides overlap
MPI_Isend
MPI_Irecv
MPI_Wait
MPI_Wait
Non-contiguous datatype, sender-side
MPI_Isend
MPI_Recv
MPI_Wait
CPU overhead, sender-side
MPI_Isend
MPI_Recv
MPI_Wait
N computing threads (funneled)
MPI_Send
MPI_Recv

Metrics and visualization

- **2D parameters** space
  - X axis: data size
  - Y axis: computation time
- **Overhead** ratio relative to perfect overlap

  0 : perfect overlap

  1 : no overlap, computation & communication serialized

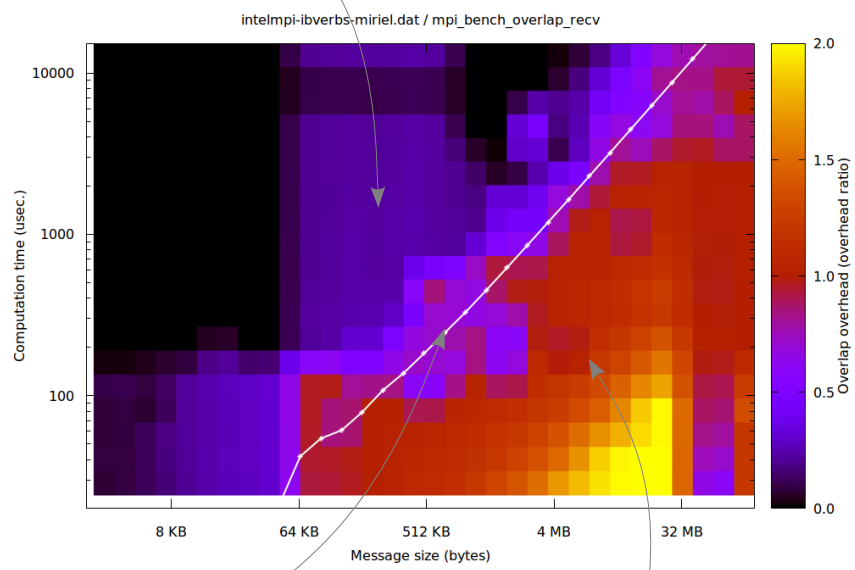
  >1 : slower than serialized

Expected time if perfect overlap

$$ratio = \frac{measured - max(computation, comm)}{min(computation, comm)}$$

Normalizing such as ratio=1 for no overlap

Computation longer than communication



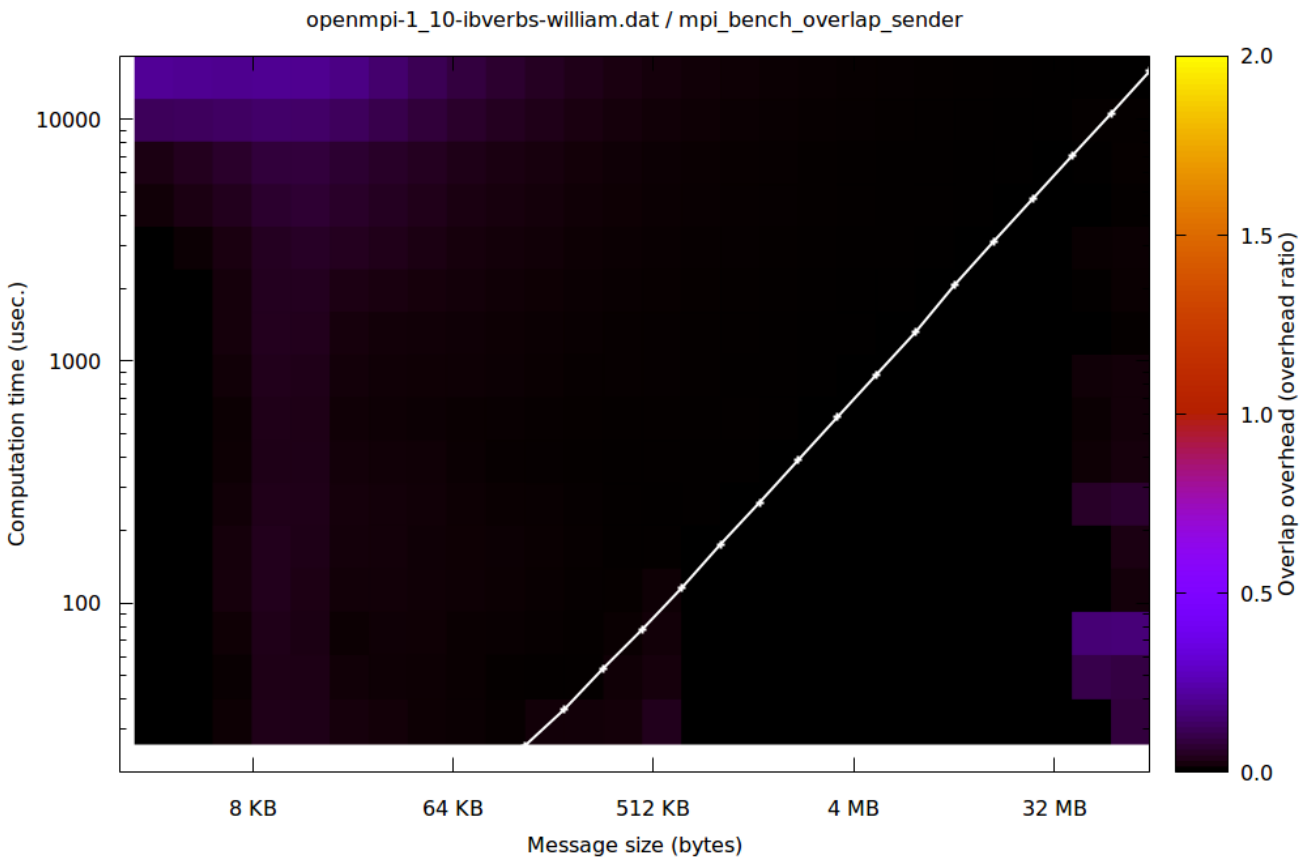
Communication time = computation time

Communication longer than computation

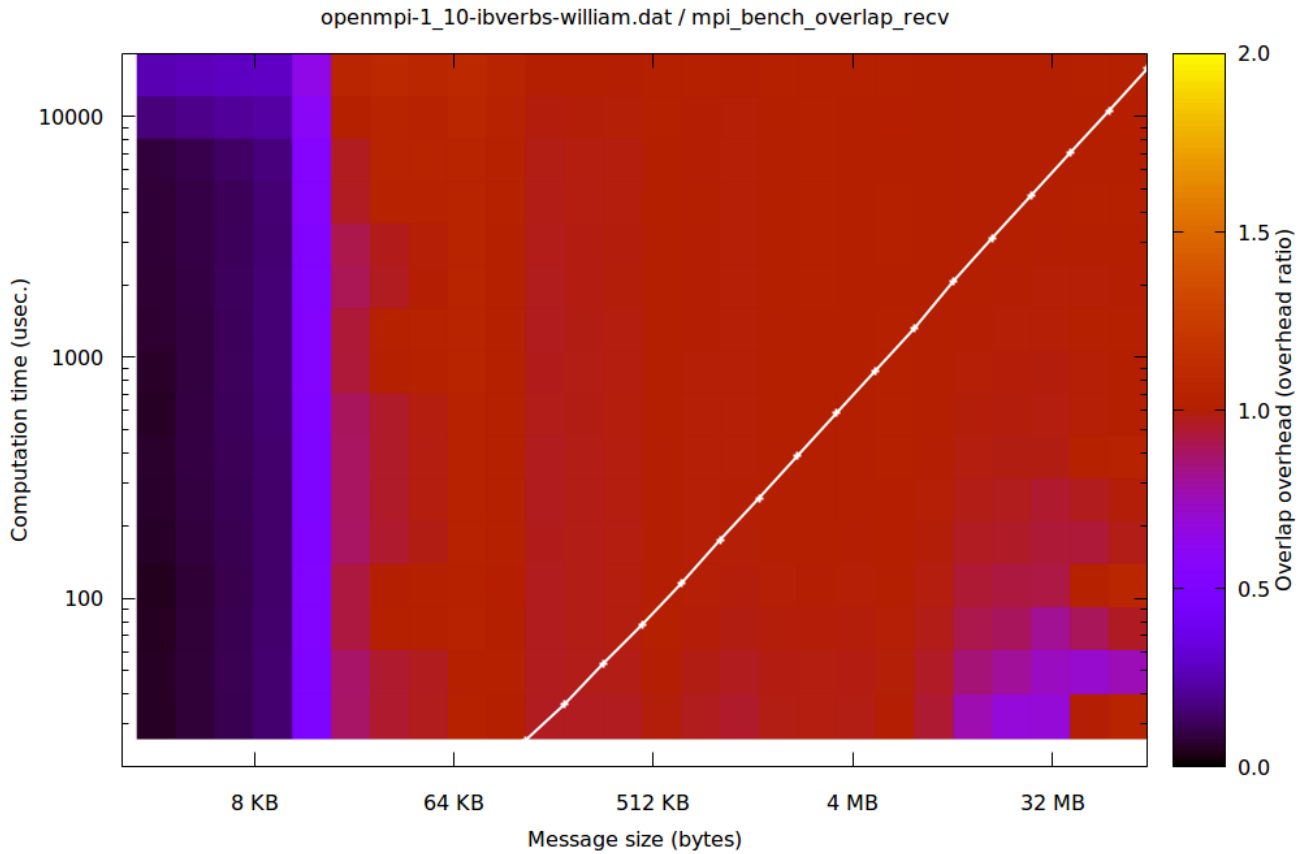
2

Benchmark results

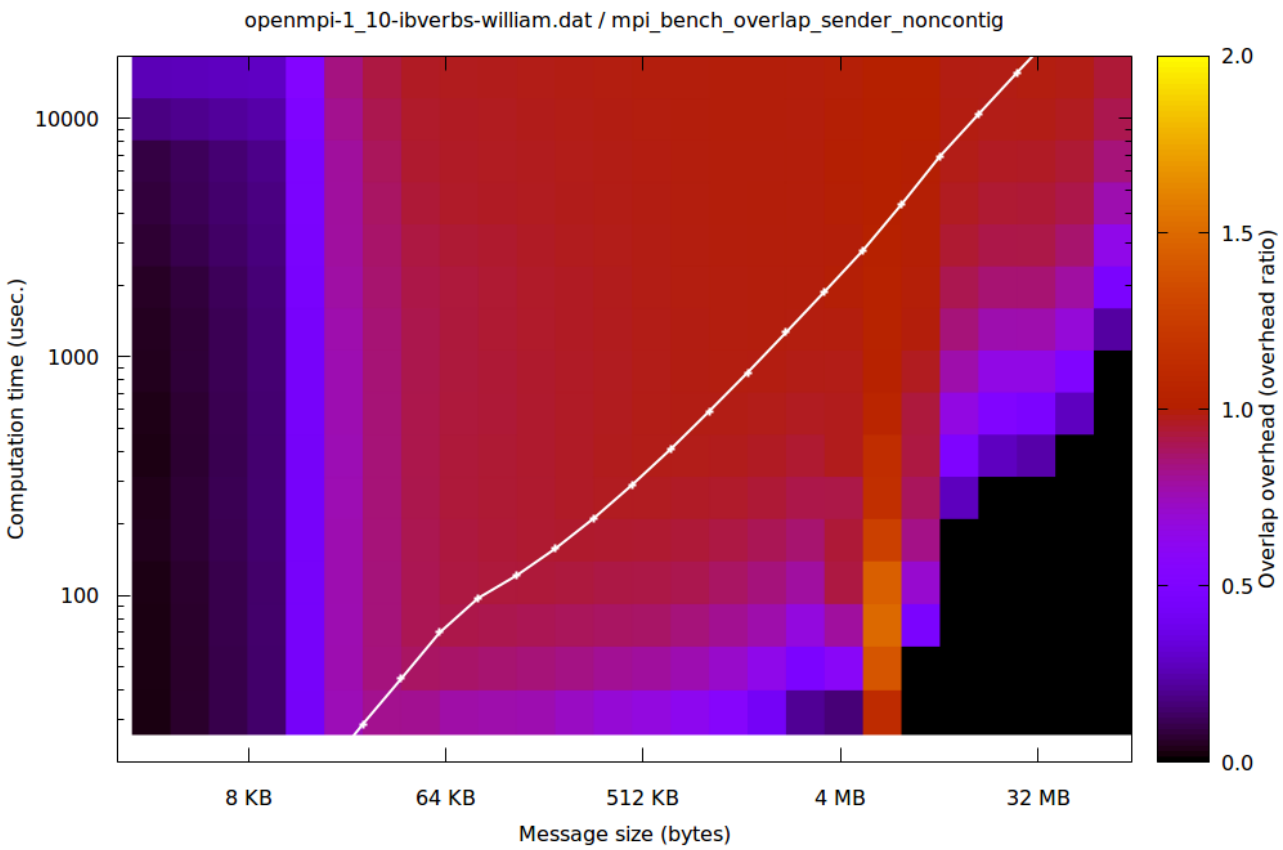
OpenMPI 1.10, ibverbs – sender-side overlap



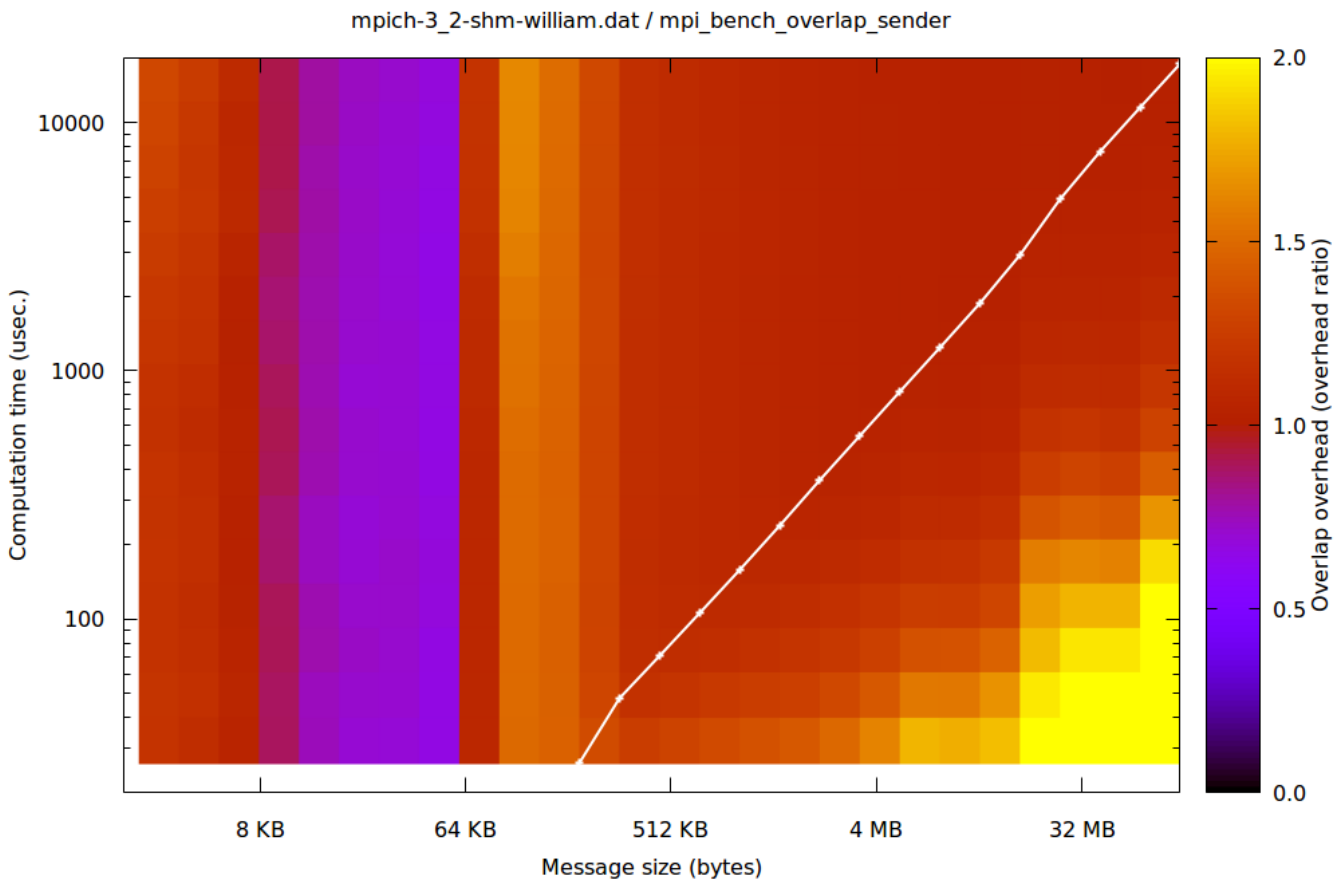
OpenMPI 1.10, ibverbs – receiver-side overlap



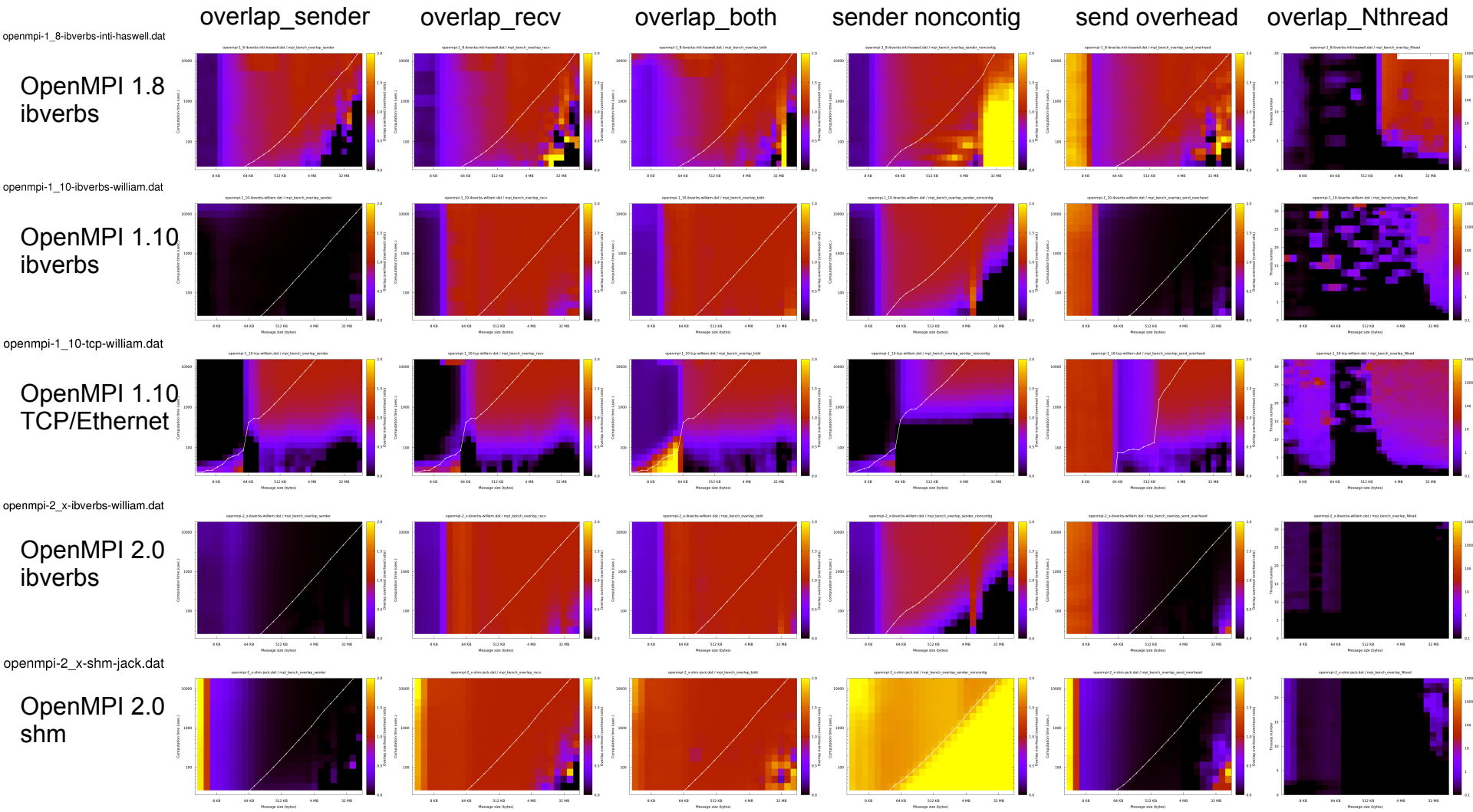
OpenMPI 1.10, ibverbs – sender-side, non-contig overlap



MPICH 3.2 – send overlap, shm



OpenMPI Mosaic

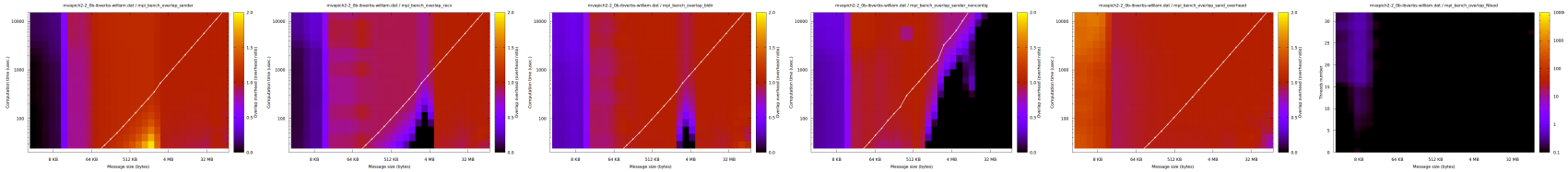


MPICH & MVAPICH

overlap_sender | overlap_recv | overlap_both | sender noncontig | send overhead | overlap_Nthread

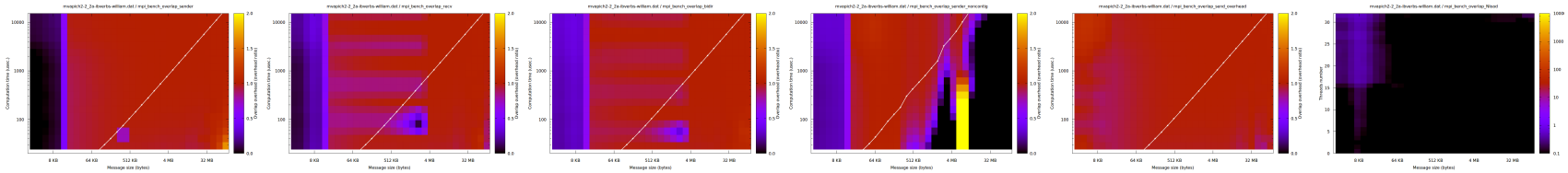
mvapich2-2_0b-ibverbs-william.dat

MVAPICH 2.0 ibverbs



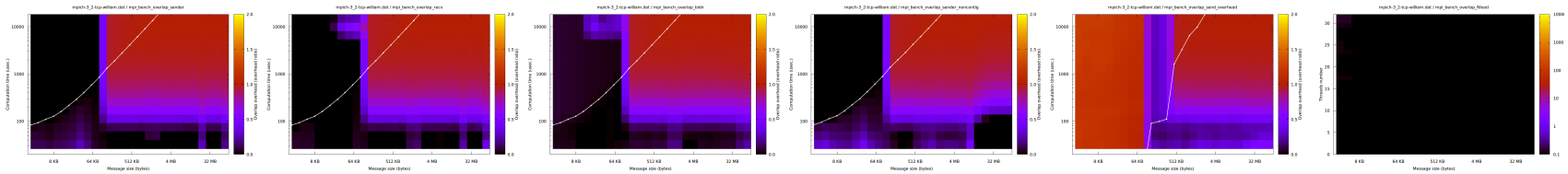
mvapich2-2_2a-ibverbs-william.dat

MVAPICH 2.2 ibverbs



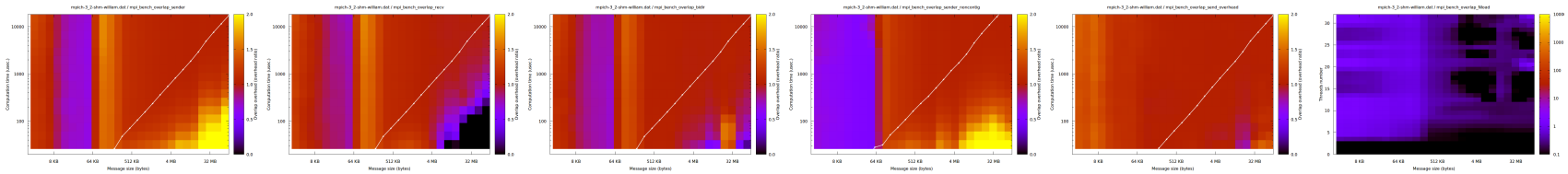
mpich-3_2-tcp-william.dat

MPICH 3.2 TCP/Ethernet



mpich-3_2-shm-william.dat

MPICH 3.2 shm



Supercomputers

overlap_sender | overlap_recv | overlap_both | sender noncontig | send overhead | overlap_Nthread

bluegene-juqueen-default.dat

IBM BluGene/Q (default)

bluegene-juqueen-threadmultiple.dat

IBM BlueGene/Q (thread multiple)

bluewaters-gemini.dat

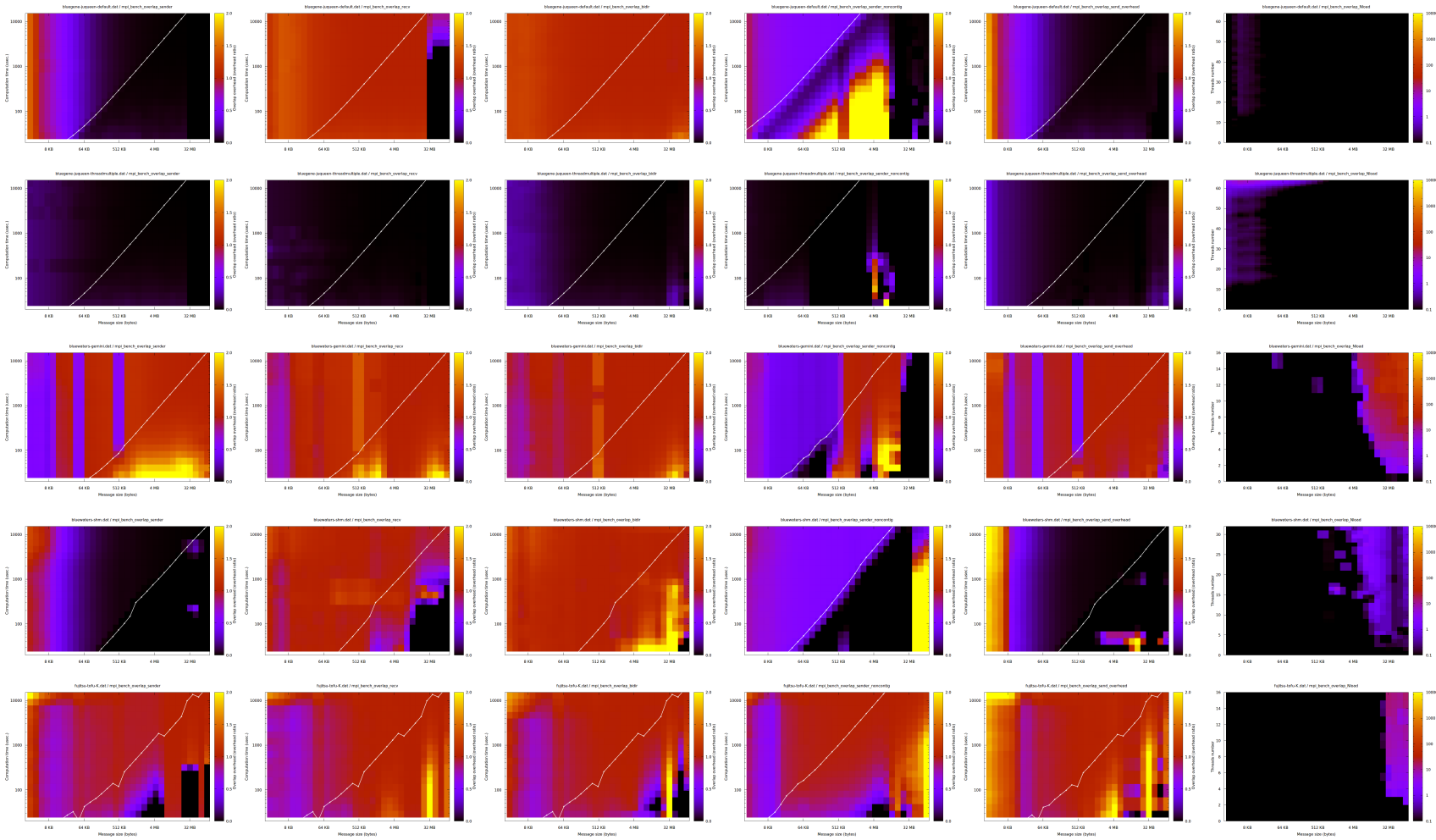
Cray Gemini Bluewaters

bluewaters-shm.dat

Cray shm Bluewaters

fujitsu-tofu-K.dat

Fujitsu Tofu K Computer

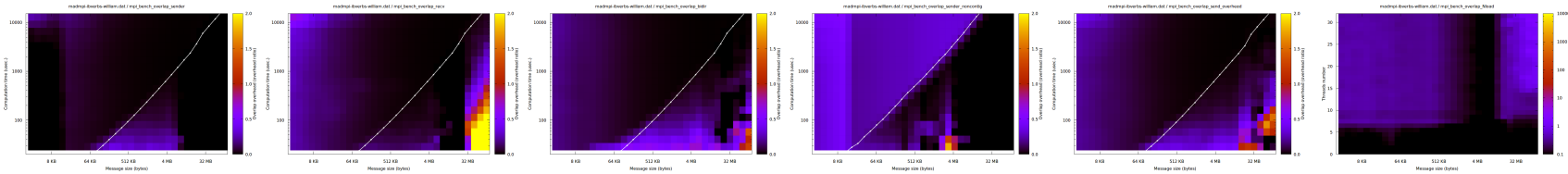


Misc

overlap_sender | overlap_recv | overlap_both | sender noncontig | send overhead | overlap_Nthread

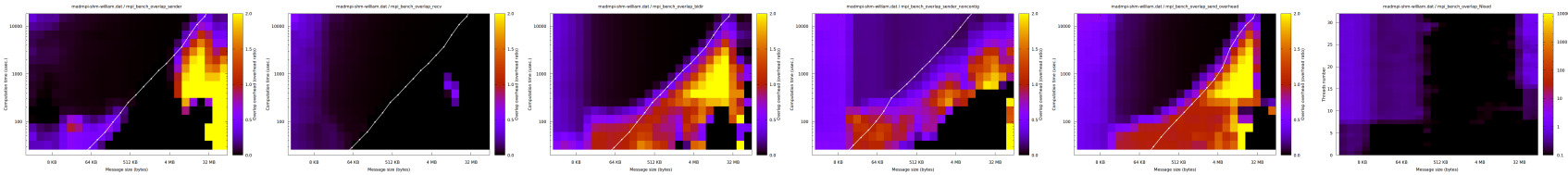
madmpi-ibverbs-william.dat

MadMPI ibverbs



madmpi-shm-william.dat

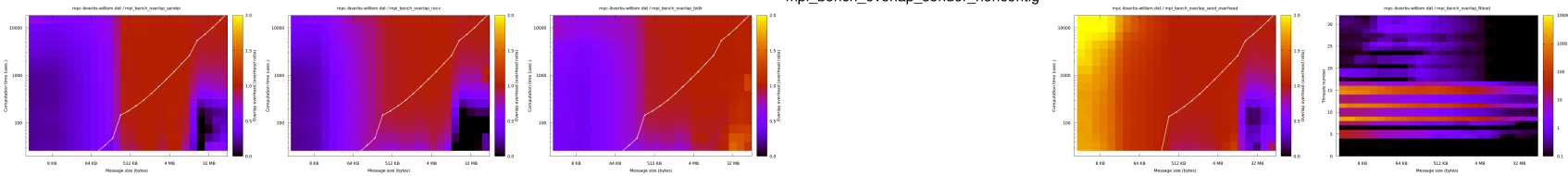
MadMPI shm



mpc-ibverbs-william.dat

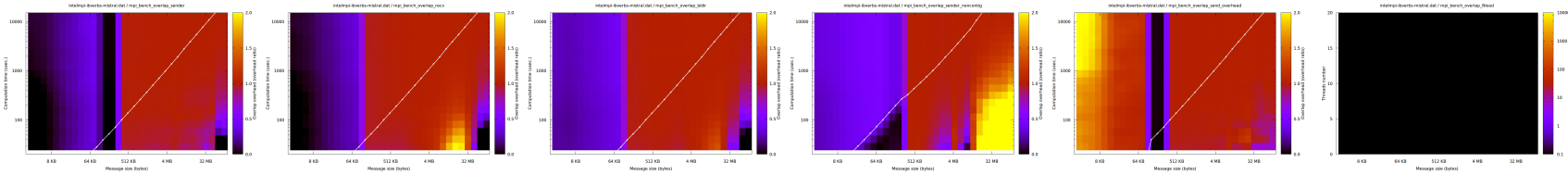
MPC ibverbs

mpi_bench_overlap_sender_noncontig



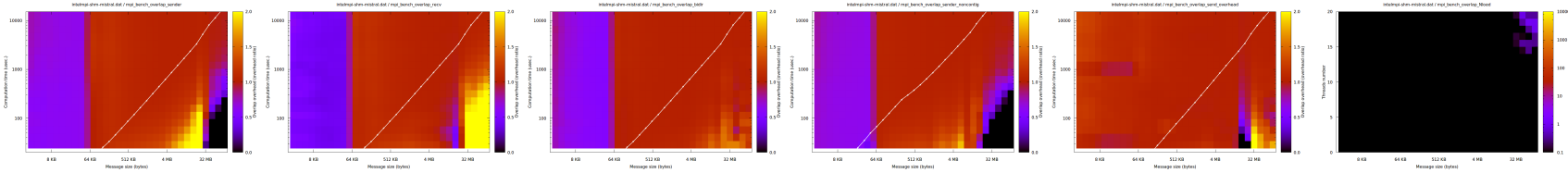
intelmpi-ibverbs-mistral.dat

Intel MPI 5.1 ibverbs



intelmpi-shm-mistral.dat

Intel MPI 5.1 shm

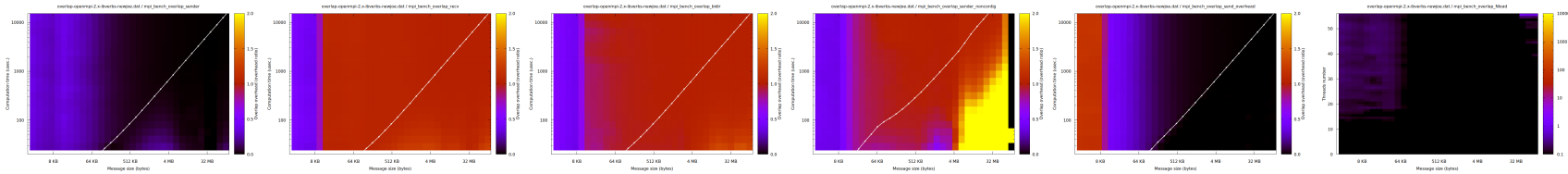


Latest additions (not in the paper)

overlap_sender | overlap_recv | overlap_both | sender noncontig | send overhead | overlap_Nthread

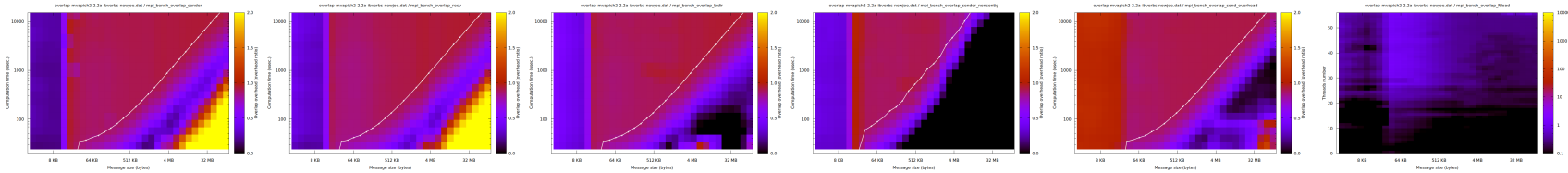
overlap-openmpi-2.x-ibverbs-newjoe.dat

OpenMPI 2.0
IB EDR
ibverbs



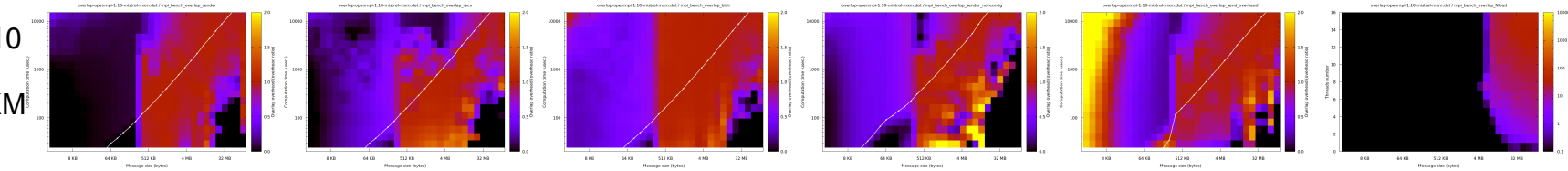
overlap-mvapich2-2.2a-ibverbs-newjoe.dat

MVAPICH 2.2
IB EDR
ibverbs

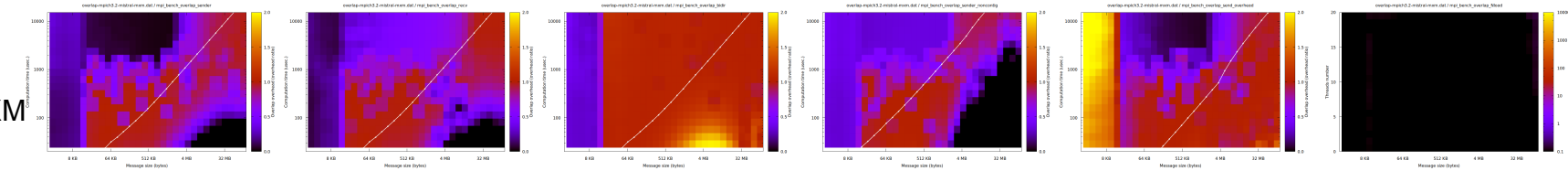


overlap-openmpi-1.10-mistral-mxm.dat

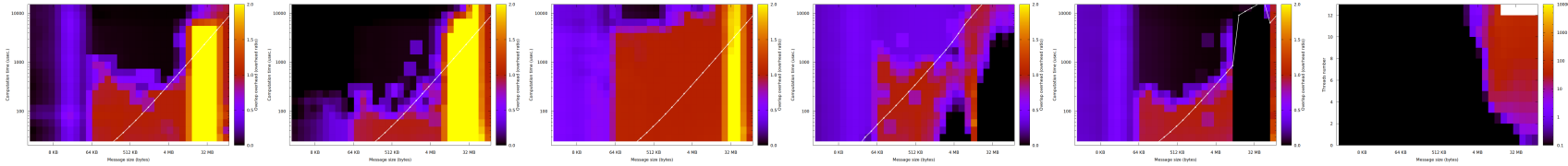
OpenMPI 1.10
IB FDR
Mellanox MXM



MPICH 3.2
IB FDR
Mellanox MXM



Intel MPI 5.1
Omni-Path



3

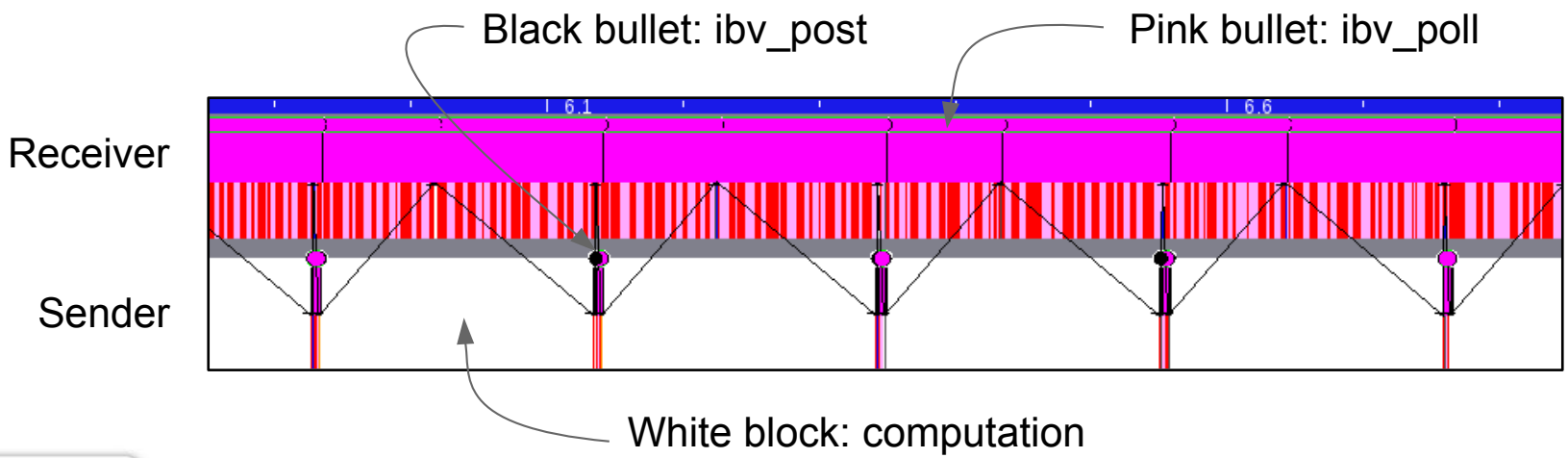
Results analysis

Classification of results

- Threaded/non-threaded library
  - **Threaded libraries** (MadMPI, IBM MPI BluGene/Q): good overlap
  - Threaded computation + blocking communication: (mostly) good overlap
- Rendez-vous threshold
  - **Large messages** need CPU to process rendez-vous request; hard to overlap
- TCP
  - Kernel threads makes communication progress
- **Shared memory**
  - CPU-hungry, hard to overlap, often slower than serialized
- **Non-contiguous datatypes**
  - Needs CPU to gather/scatter data; hard to overlap

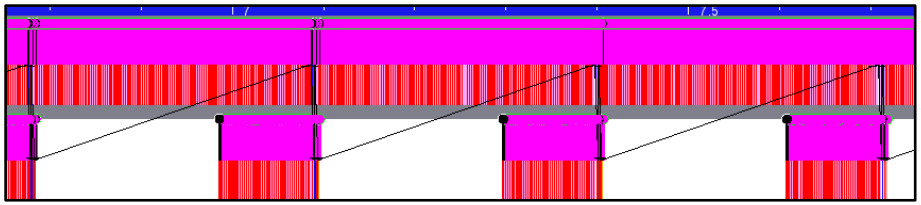
Interleaved MPI and InfiniBand traces

- Trace functions calls to MPI and ibverbs (low-level network driver)
  - Observe how the MPI library makes communication progress
- **EZTrace** trace generator & **ViTE** trace visualizer
  - Use builtin `mpi` plugin and specially developped `ibverbs` plugin
  - **Interactive exploration** needed to get details (zoom, event properties)
    - Hard to read on screenshots, and not enough time for a demo

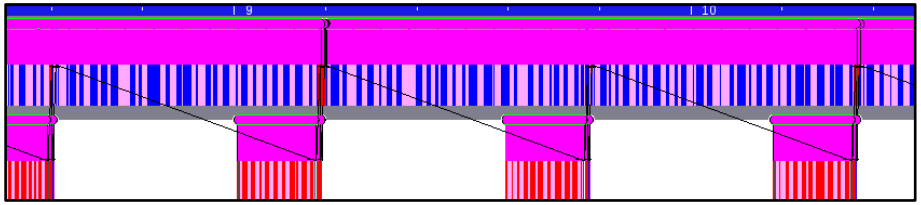


Pathological case traces

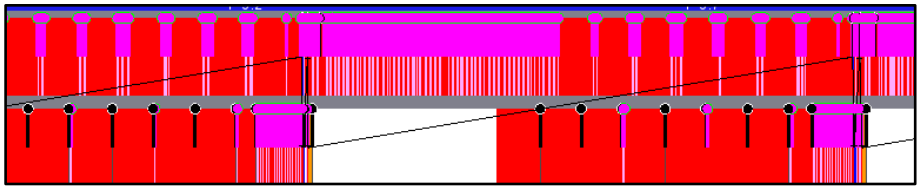
- Sender-side overlap: sender notices **RTR** (rendez-vous reply) after computation



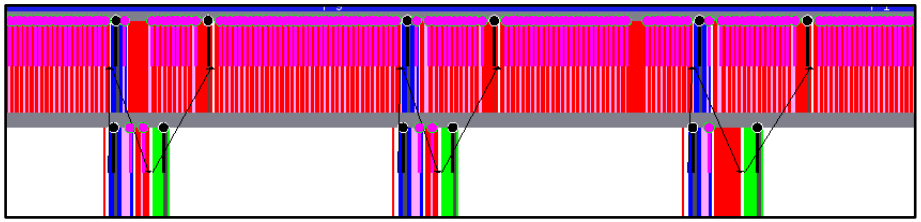
- Receiver-side overlap: receiver notices **rendez-vous request** after computation



- Non-contiguous, sender-side overlap: data sent in 64KB chunks, **only first chunk is overlaped**



- Sender-side CPU overhead: MPI_Isend has a **fixed**, non-overlaped **cost** (2 usec.)



Conclusion

- Amortize communication cost through MPI overlap
- Contributions:
  - **Benchmark suite** to assess actual MPI overlap
  - **Results** on large panel of MPI libraries & machines
  - Trace framework to **understand pathological cases**
- Future works: RMA, multi-threaded/OpenMP benchmark, collectives
- **MPI application programmers assume overlap happens**
  - In real life: it's more complicated!
- Release and results:
  - **`http://pm2.gforge.inria.fr/mpibenchmark/`**

Thank you

Get the overlap benchmark from:

`http://pm2.gforge.inria.fr/mpibenchmark/`