Message Passing Interface
Lecture 5

Manfred Liebmann
Technische Universität München
Chair of Optimal Control
Center for Mathematical Sciences, M17
manfred.liebmann@tum.de

December 1, 2015

Network Technology: InfiniBand

InfiniBand (IB), a computer-networking communications standard used in high-performance computing, features *very high throughput* and *very low latency*. InfiniBand is also utilized as either a direct, or switched interconnect between servers and storage systems, as well as an interconnect between storage systems.

- InfiniBand
  - De facto standard software stack developed by the OpenFabrics Alliance: Verbs API
  - InfiniBand host bus adapters (HBA) and network switches: Mellanox, Intel
  - InfiniBand provides remote direct memory access (RDMA) capabilities
  - Most commonly used interconnect in supercomputers
  - Copper and optical fiber cables: QSFP connectors
  - Speeds: 10Gb/s SDR, 20Gb/s DDR, 40Gb/s QDR, 56Gb/s FDR, 100Gb/s EDR
  - Latency: 5μs SDR, 2.5μs DDR, 1.3μs QDR, 0.7μs FDR, 0.5μs EDR

Network Technology: InfiniBand

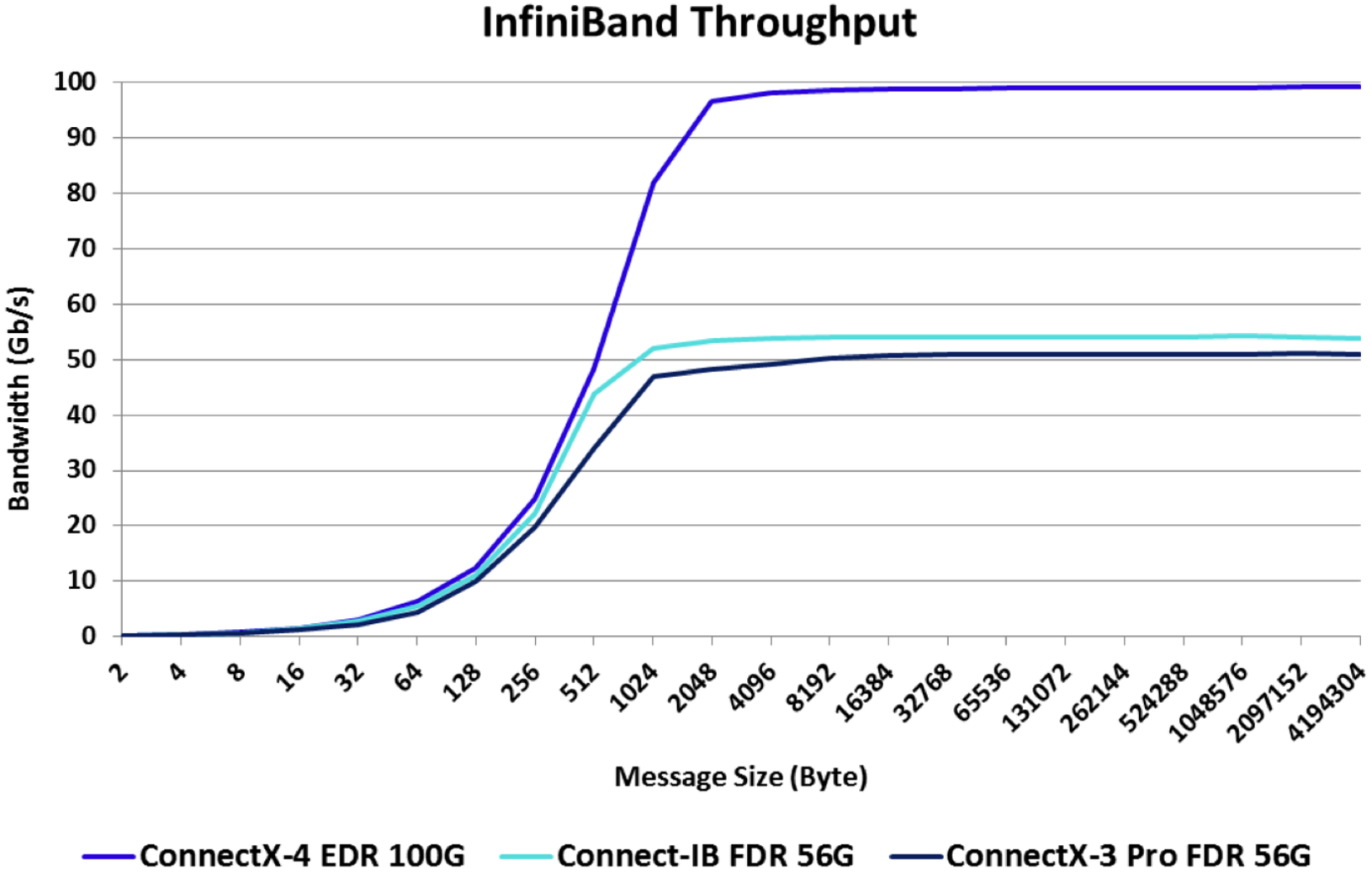

Figure 1: InfiniBand host bus adapter comparison

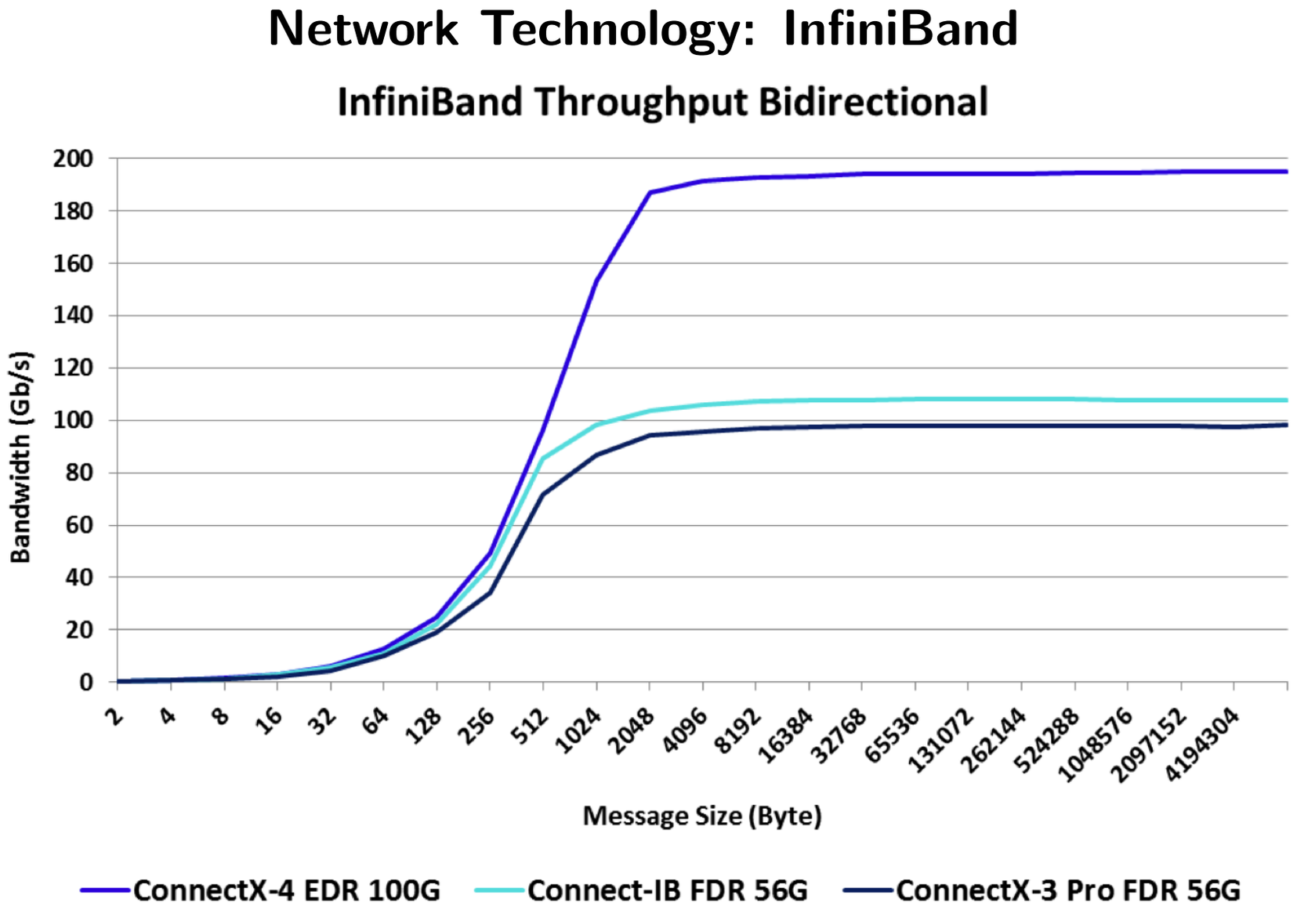

Figure 2: InfiniBand host bus adapter comparison

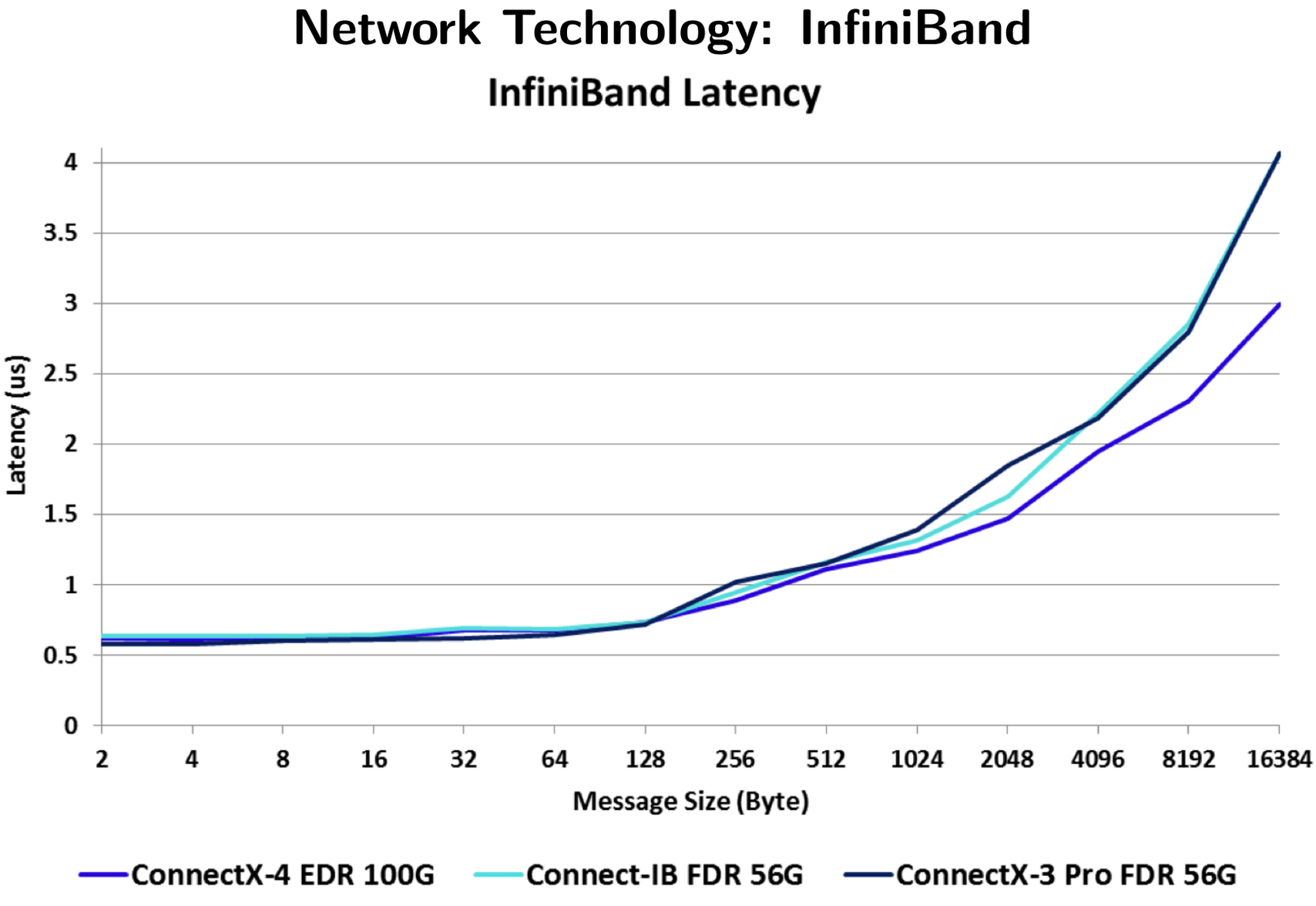

Figure 3: InfiniBand host bus adapter comparison

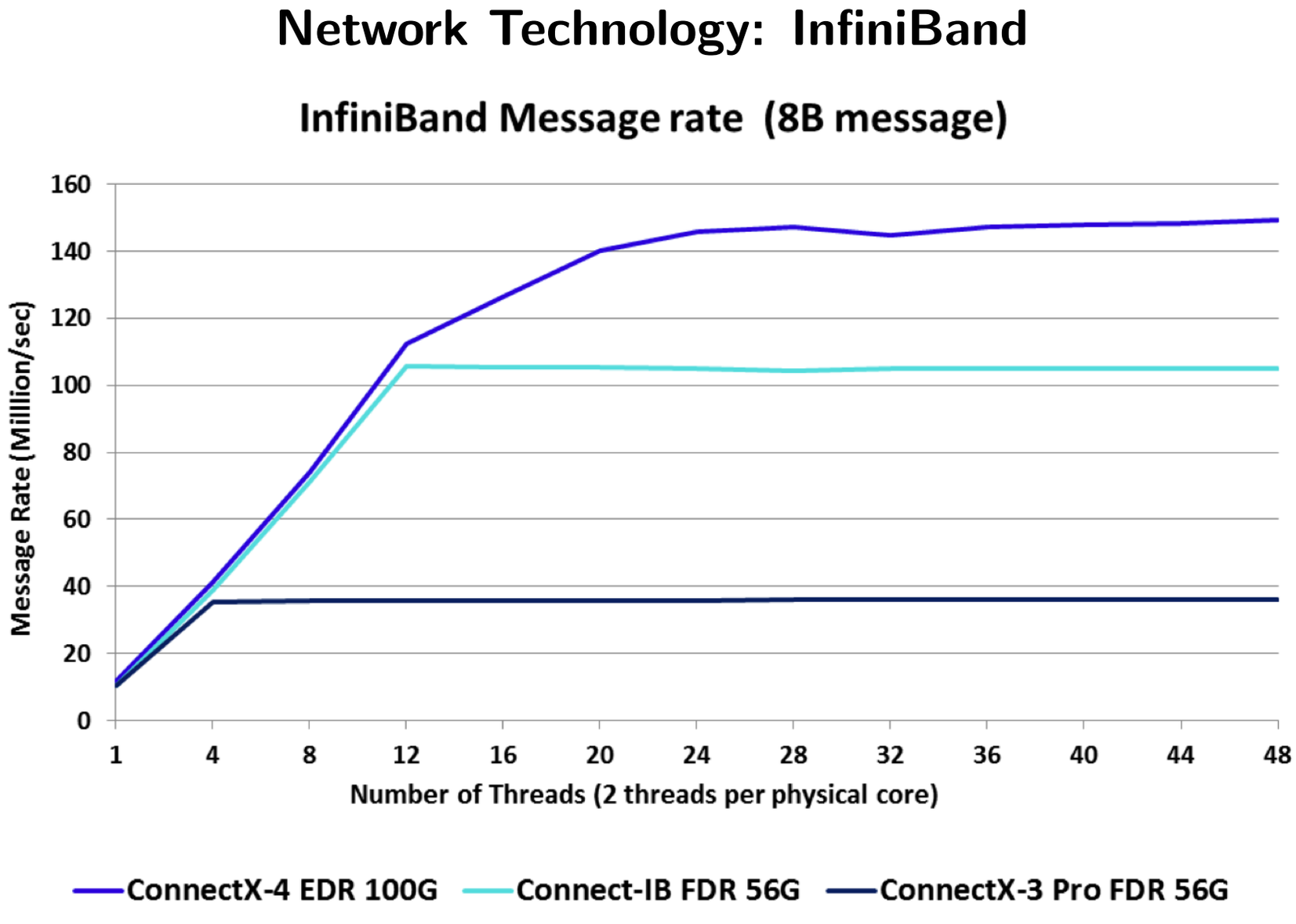

Figure 4: InfiniBand host bus adapter comparison

Network Technology: GPUDirect RDMA

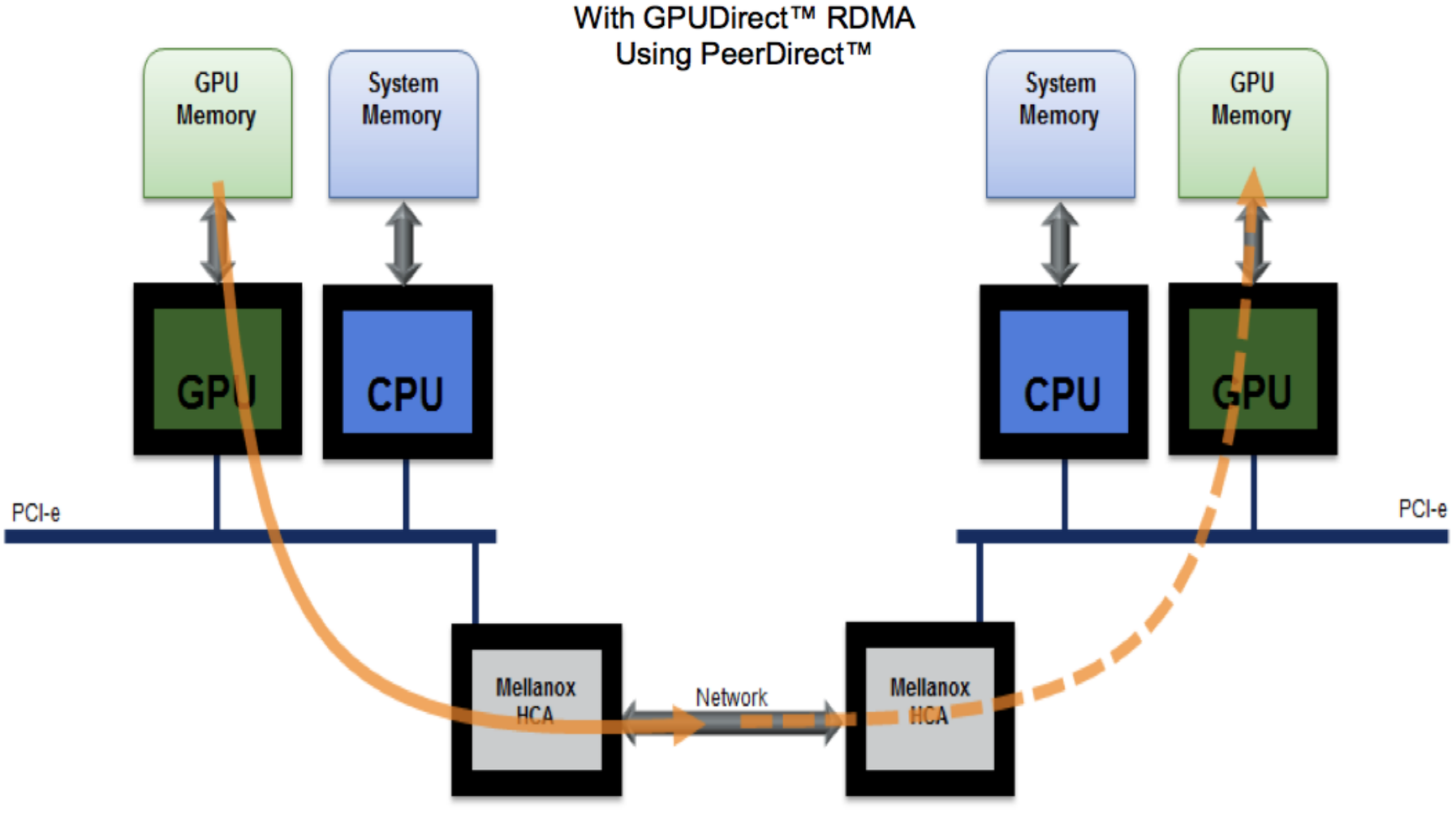

Figure 5: Remote direct memory access (RDMA) data transfer between GPUs

Speedup and Efficiency

Speedup: Let T_1 be the execution time of an application measured for a single process and T_n be the execution time measured for n processes, then the speedup is defined as

$$S := \frac{T_1}{T_n} \, . \tag{1}$$

Efficiency: Let T_1 and T_n be the execution times of an application measured for a single and n processes, then the efficiency is defined as

$$E := \frac{S}{n} = \frac{T_1}{nT_n} \, . \tag{2}$$

Super-linear speedup: $S > n, \; E > 1$.

Amdahl's Law

Let T_1 be the total execution time of the program for one process and T_{seq} the execution time of the sequential part and T_{par} the execution time of the parallel part, i.e. $T_1 = T_{seq} + T_{par}$, then the total runtime of the program for n processes is

$$T_n = T_{seq} + \frac{T_{par}}{n} \, . \tag{3}$$

The maximum achievable speedup of a program with $p = T_{par}/T$ percent parallel code and $(1-p) = T_{seq}/T$ percent sequential code running on n processes is given by Amdahl's law.

$$S_n := \frac{1}{1 - p + \frac{p}{n}} = \frac{T_1}{T_{seq} + \frac{T_{par}}{n}} = \frac{T_1}{T_n} \tag{4}$$

The limit case is given by

$$S^* := \lim_{n \to \infty} S_n = \frac{1}{1-p} = \frac{T_1}{T_{seq}} \tag{5}$$

5% of sequential code limits the possible speedup to a factor of 20!

Amdahl's Law

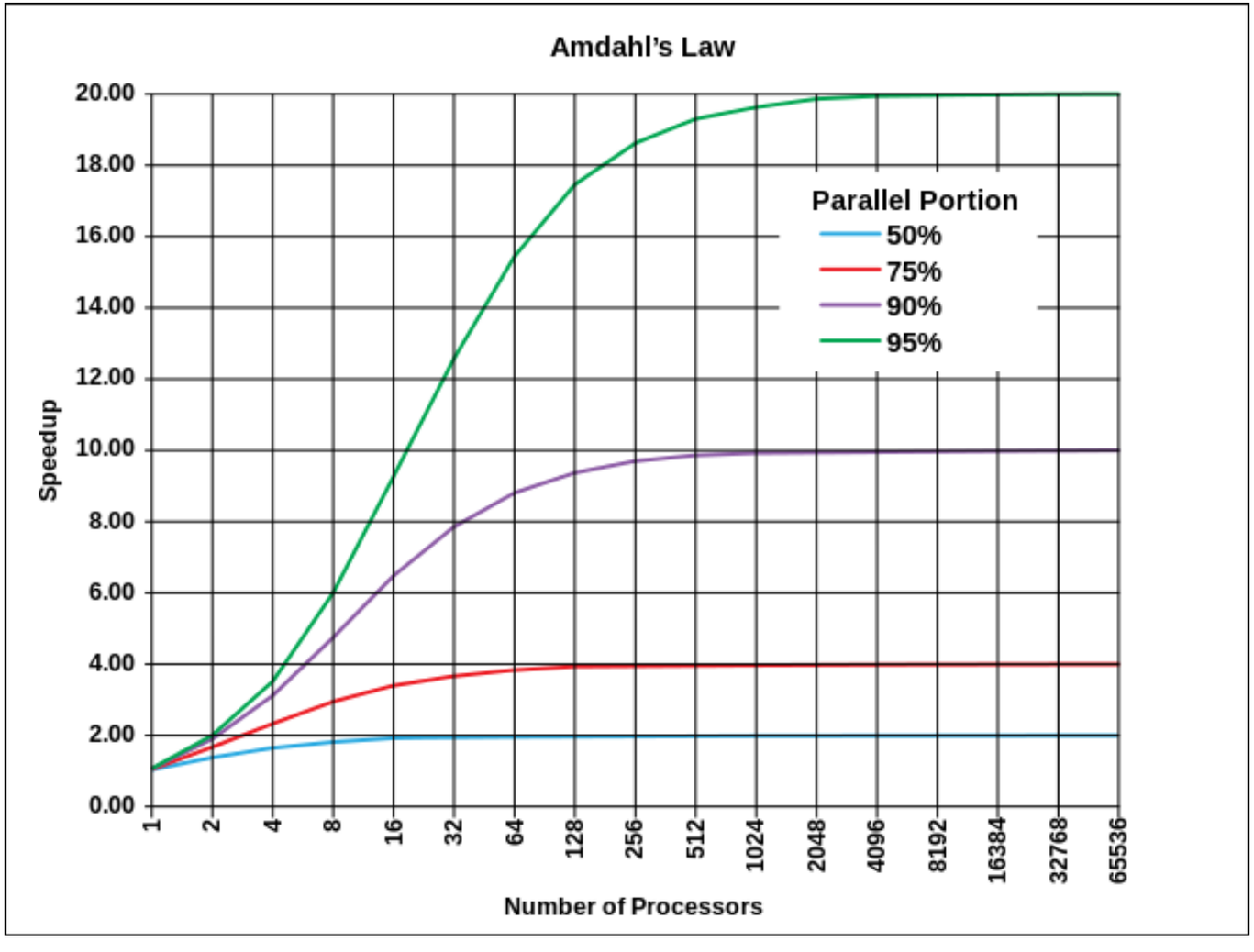

Figure 6: Amdahl's law for different degrees of parallelization

Job Submission on the Mephisto Cluster

Batch-queuing system installed on `mephisto.uni-graz.at`: Sun grid engine (SGE)

The following job submission scripts are available in the `/share/apps/` directory: `simple.sh`, `cpu.sh`, `gpu.sh`, and `phi.sh`. The `simple.sh` script is the most basic submission script for parallel CPU jobs.

```
[liebmann@mephisto ~]$ cat /share/apps/simple.sh
#!/bin/sh -f
#$ -V
#$ -cwd
#$ -j y
#$ -pe mpi 4

while read line; do
    echo $line "slots=12"
done < $TMPDIR/machines > $TMPDIR/hostfile
cat $TMPDIR/hostfile

mpirun --np 48 --hostfile $TMPDIR/hostfile ./armocpu armo16.inp
```

Job submission to the SGE queue: `qsub simple.sh`

Job statistics of the SGE queue: `qstat`

Job Submission on the Mephisto Cluster

The `cpu.sh` requests 4 compute nodes with 12 cores each, `#$ -pe mpi 4`, using the parallel environment `mpi`. The `mpirun` command starts then the program with 48 processes and distributes the processes according to the automatically generated host and rank files.

```
[liebmann@mephisto ~]$ cat /share/apps/cpu.sh
#!/bin/sh -f
#$ -V
#$ -cwd
#$ -j y
#$ -pe mpi 4

rank=0;
while read line; do
    for i in 0 1 2 3 4 5 6 7 8 9 10 11; do
    echo "rank" $rank"="$line "slot="$i
    let "rank += 1"
    done
done < $TMPDIR/machines > $TMPDIR/rankfile
cat $TMPDIR/rankfile

mpirun --np 48 --hostfile $TMPDIR/machines --rankfile $TMPDIR/rankfile ./armocpu armo16.inp
```

Job Submission on the Mephisto Cluster

The `gpu.sh` submission script allocates 2 nodes with 4 processes each. Every process is bound to a socket and handles a single GPU giving a quad-GPU configuration on every node.

```
[liebmann@mephisto ~]$ cat /share/apps/gpu.sh
#!/bin/sh -f
#$ -V
#$ -cwd
#$ -j y
#$ -pe mpi 2

rank=0;
while read line; do
    for i in 0 3 6 9; do
    echo "rank" $rank"="$line "slot="$i
    let "rank += 1"
    done
done < $TMPDIR/machines > $TMPDIR/rankfile
cat $TMPDIR/rankfile

mpirun --np 8 --hostfile $TMPDIR/machines --rankfile $TMPDIR/rankfile ./armogpu armo16.inp
```

Job Submission on the Mephisto Cluster

The `phi.sh` submission script allocates a single node with a single Intel Xeon Phi accelerator board. The parallel environment `#$ -pe phi 1` is configured for the accelerator.

```
[liebmann@mephisto ~]$ cat /share/apps/phi.sh
#!/bin/sh -f
#$ -S /bin/bash
#$ -V
#$ -cwd
#$ -j y
#$ -pe phi 1

./exe-offload.x
```