Parallel Deep Learning

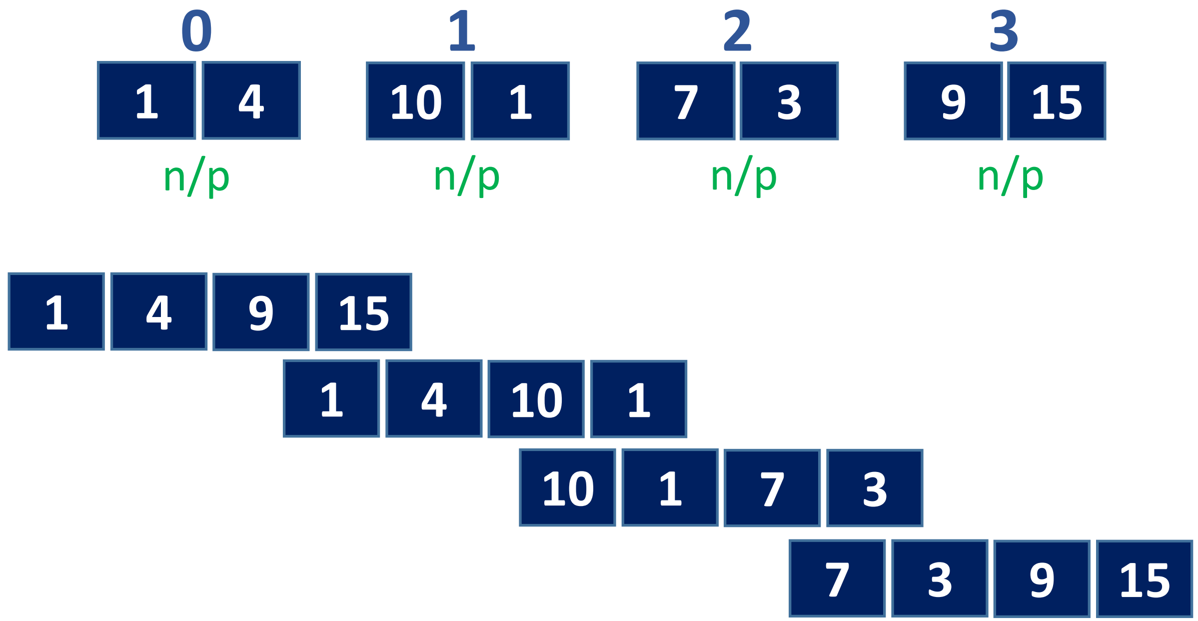
Lecture 21

April 23, 2025

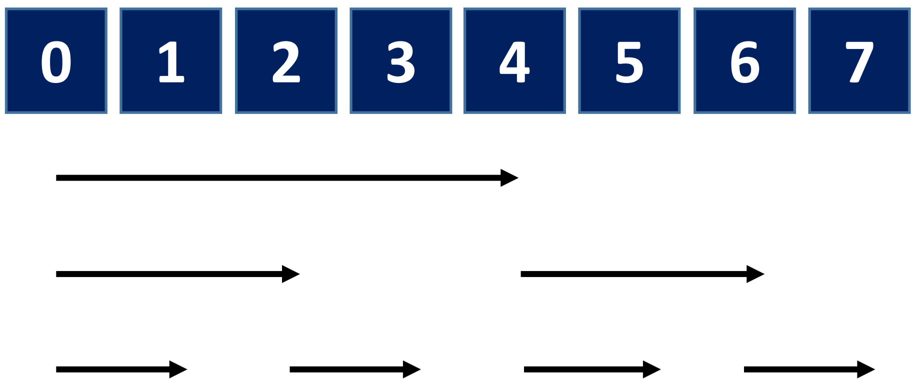
Recap – Algorithms for Collectives

Allgather – Ring Algorithm

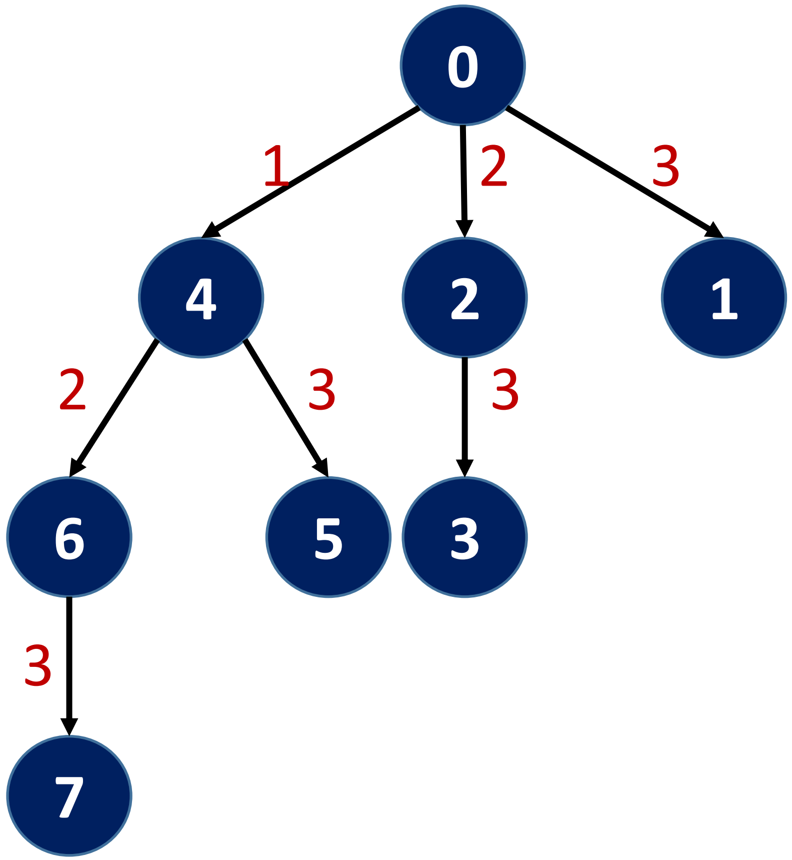
- Every process sends to and receives from everyone else
- Assume p processes and total n bytes
- Every process sends and receives n/p bytes
- Time
  - (p – 1) * (L + n/p*(1/B))
- How can we improve?



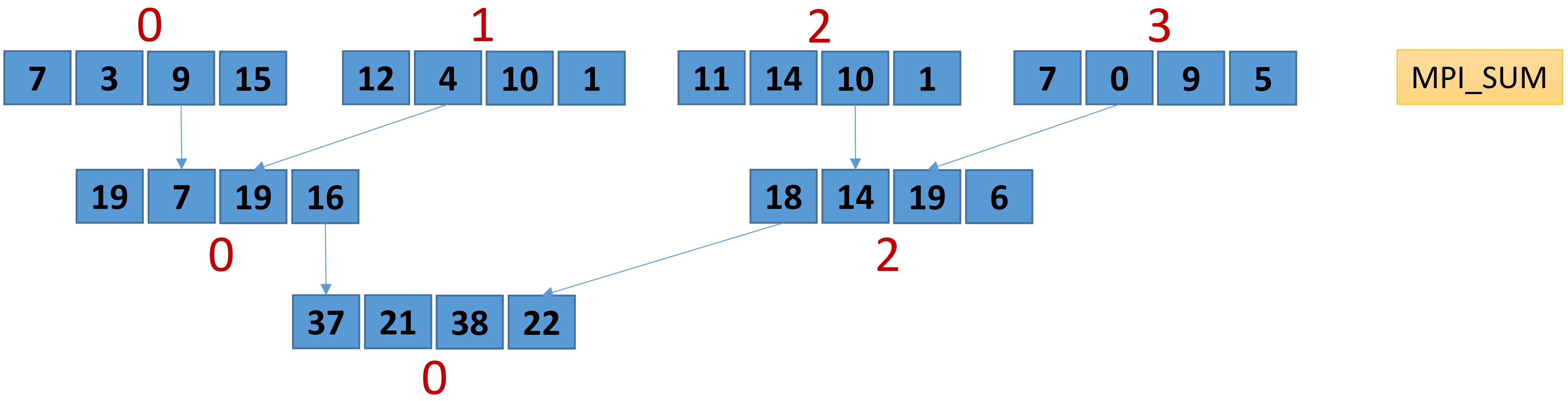
Broadcast – Binomial Tree



- #Steps for p ($=2^d$) processes?
  - log p
- Transfer time for n bytes
  - $T(p) = \log p * (L + n/B)$

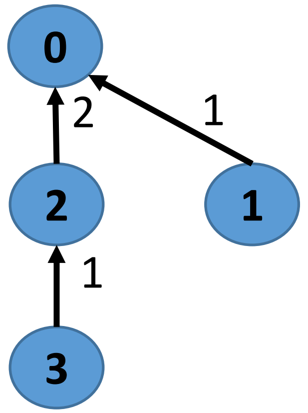


Reduce Algorithm – Recursive doubling

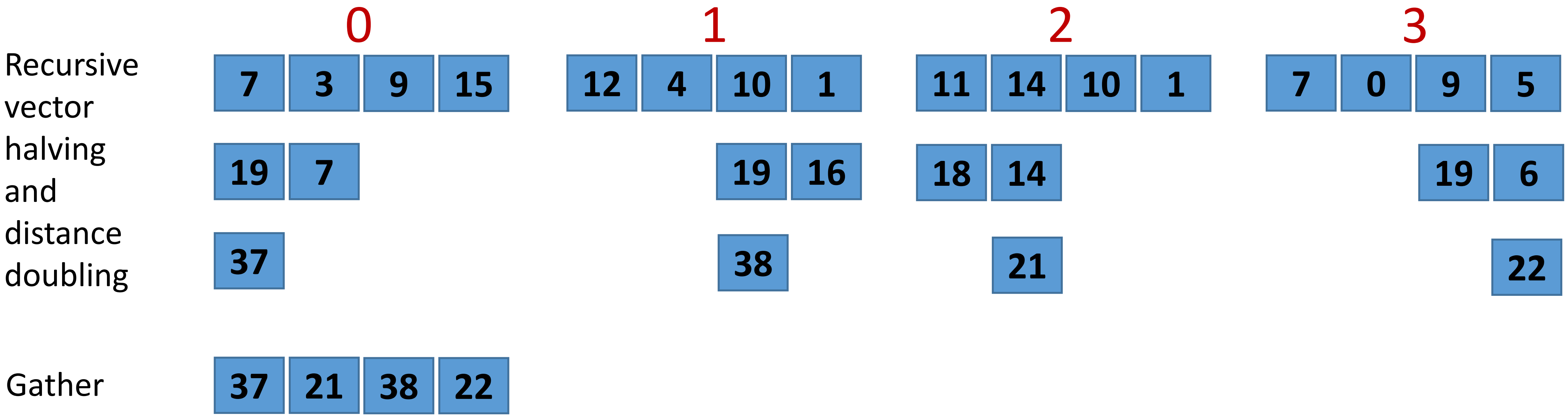


Time: log p (L + n*(1/B) + n*c)
L = latency, B = bandwidth
c = compute cost per byte

Used for short messages



Reduce – Rabenseifner's Algorithm



Time:

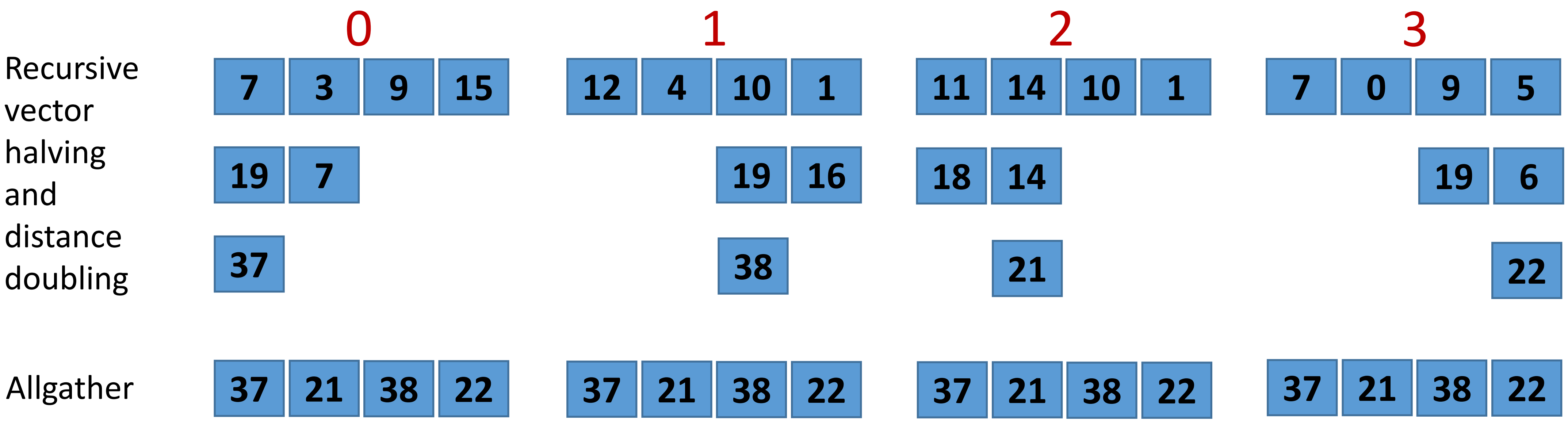
log p * L + (p-1)/p*(n/B) + (p-1)/p*n*c (reduce-scatter) +
log p * L + (p-1)/p*(n/B) (gather using binomial)

n = data size L = latency

p = #processes B = bandwidth

c = compute cost per byte

Allreduce – Rabenseifner's Algorithm



Time:

log p * L + (p-1)/p*(n/B) + (p-1)/p*n*c (reduce-scatter) +

log p * L + (p-1)/p*(n/B) (allgather using recursive vector doubling and distance halving)

n = data size L = latency

p = #processes B = bandwidth

c = compute cost per byte

Reduce/Allreduce (Ring)

0				1				2				3			
7	3	9	15	12	4	10	1	11	14	10	1	7	0	9	5

i^{th} segment is reduced by i^{th} process following a ring algorithm (i.e. rank r sends to rank (r+1) mod P)

Gather/
Allgather

37	21	38	22	37	21	38	22	37	21	38	22	37	21	38	22

$$2(p - 1)\alpha + 2\ p{-}1/p\ {*}N/\beta + p{-}1/p\ {*}N\gamma$$

Efficient MPI-AllReduce for large-scale deep learning on GPU-clusters, Nyugen et al., CCPE, 2019

Illustration of MPI_Allreduce (Ring)

a0	b0	c0	d0
a1	b1	c1	d1
a2	b2	c2	d2
a3	b3	c3	d3

a0	b0	c0+c3	d0
a1	b1	c1	d0+d1
a1+a2	b2	c2	d2
a3	b2+b3	c3	d3

Step 1

a0	b0+b2+b3	c0+c3	d0
a1	b1	c0+c1+c3	d0+d1
a1+a2	b2	c2	d0+d1+d2
a1+a2+a3	b2+b3	c3	d3

Step 2

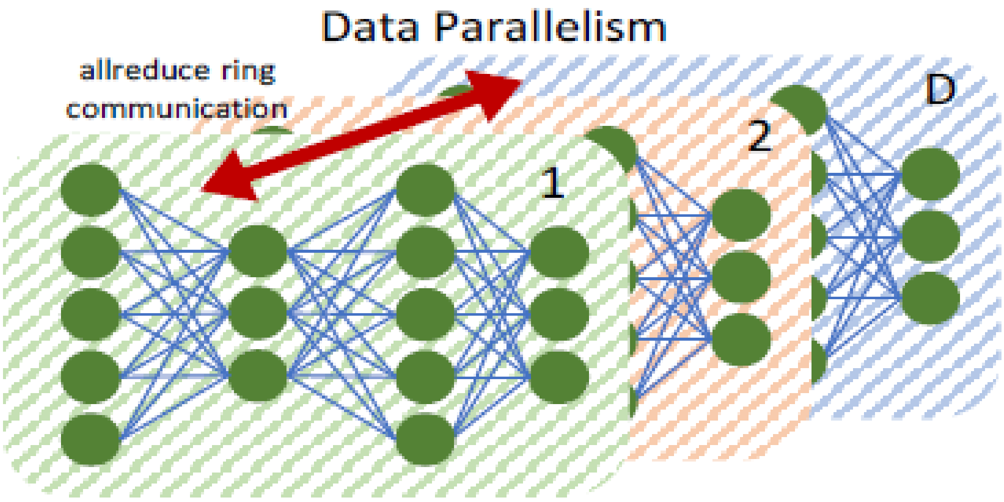
Illustration (contd.) of MPI_Allreduce (Ring)

a0+a1+a2+a3	b0+b2+b3	c0+c3	d0
a1	b0+b1+b2+b3	c0+c1+c3	d0+d1
a1+a2	b2	c0+c1+c2+c3	d0+d1+d2
a1+a2+a3	b2+b3	c3	d0+d1+d2+d3

Step 3

Distributed Deep Learning

- Data Parallelism
- …



HammingMesh: A Network Topology for Large-Scale Deep Learning, Hoefler et al., SC22

Distributed DL Frameworks

- Horovod
- Tensorflow + Horovod
- PyTorch + Horovod
- …

Accelerating distributed deep neural network training with pipelined MPI_Allreduce, Castello et al., Cluster Computing, 2021

Introduction

MPI (Message Passing Interface) [28] is the *de facto* standard for distributed high performance computing (HPC) applications. Therefore, it has been naturally adopted as the communication layer for distributed training frameworks such as Google's TensorFlow (TF) [1], TF+Horovod (HVD) [25], and PyTorch [24]. The MPI application programming interface (API) comprises a large variety of peer-to-peer and collective communication primitives. Among these, the DP scheme for distributed DNN training basically relies on the blocking `MPI_Allreduce` primitive, which internally reduces a collection of local values broadcasting the global result to all processes participating in the communication.

Quite a few papers address this limitation in the context of ML/DL

Pipelined MPI_Allreduce

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

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

Segmented MPI_Iallreduce

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

```
for (….)
        MPI_Iallreduce (…count/x…)
```

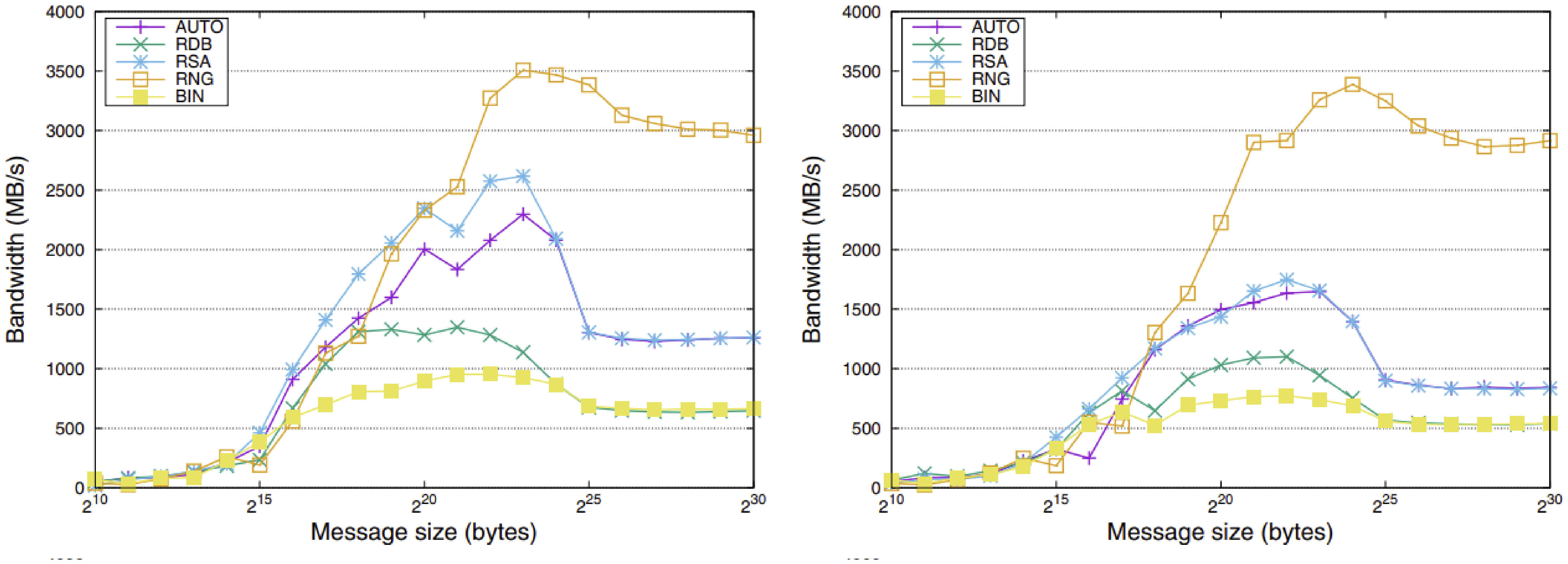
How do we segment?

- Segment of fixed size
- Fixed number of segments

Algorithms for MPI_Allreduce

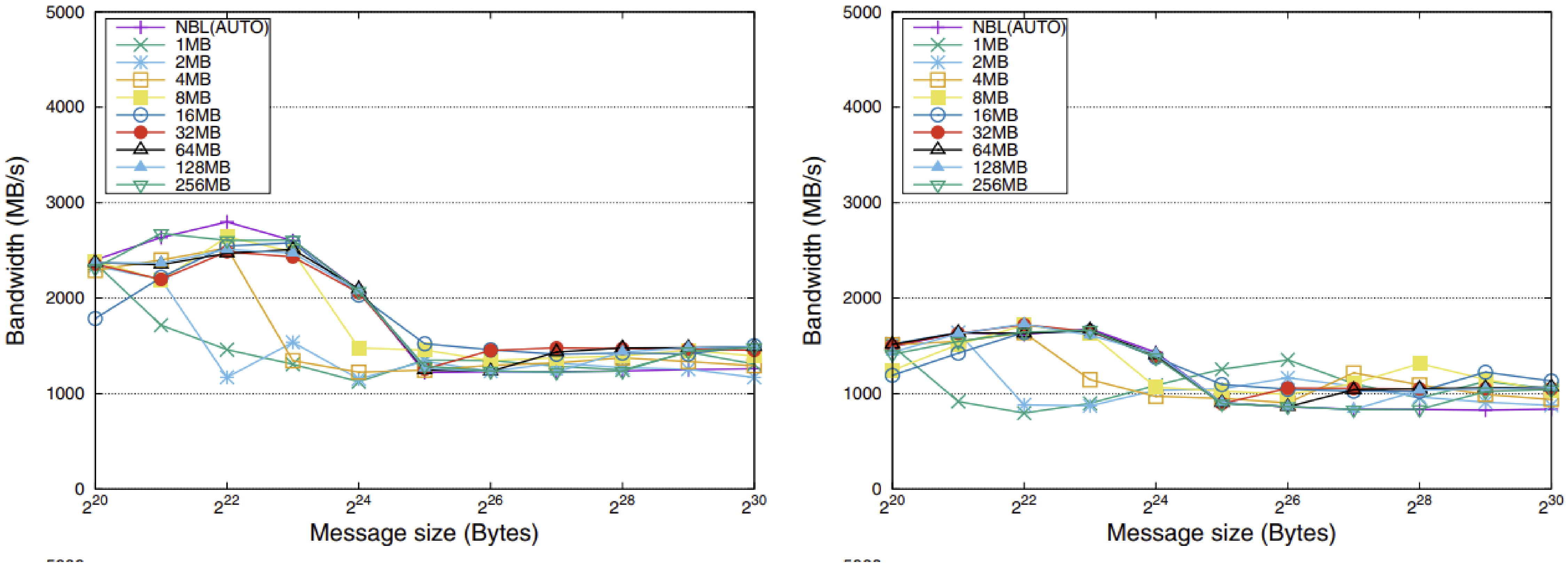
Algorithm	Latency factor	Bandwidth factor	Computation factor
Binomial tree	$\log(p)\alpha$	$\log(p)N\beta$	$\log(p)N\gamma$
Recursive-doubling	$\log(p)\alpha$	$\log(p)N\beta$	$\log(p)N\gamma$
Rabenseifner[32]	$2\log(p)\alpha$	$2\frac{p-1}{p}N\beta$	$\frac{p-1}{p}N\gamma$
Logical ring[33]	$2(p-1)\alpha$	$2\frac{p-1}{p}N\beta$	$\frac{p-1}{p}N\gamma$

Performance of MPI_Allreduce using OpenMPI implementation (8 and 9 processes)



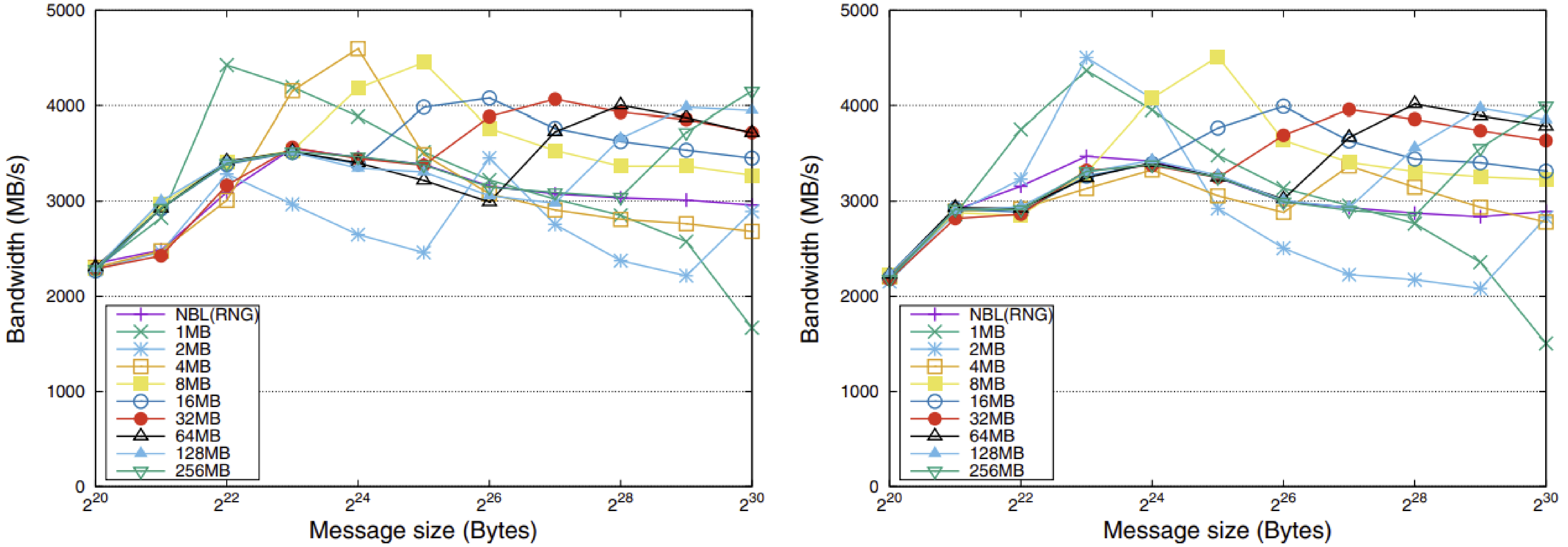
Performance with Fixed Segment Size (AUTO)

8 and 9 processes



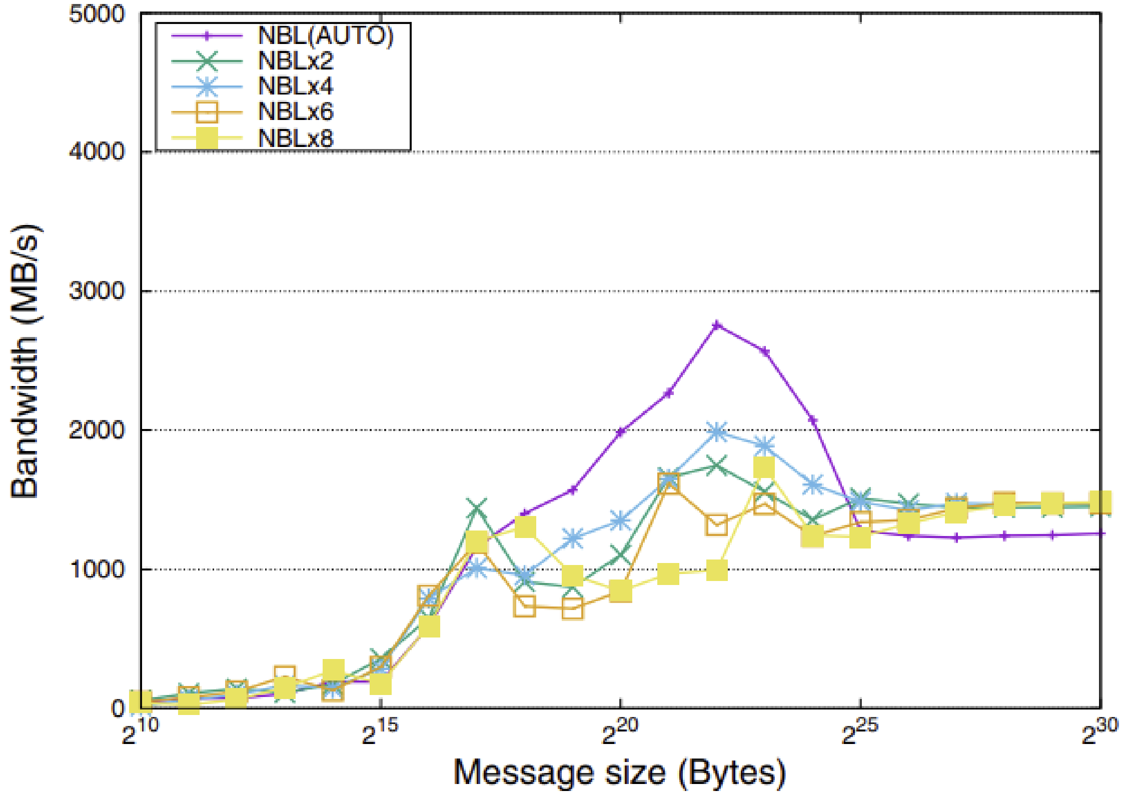
Performance with Fixed Segment Size (RING)

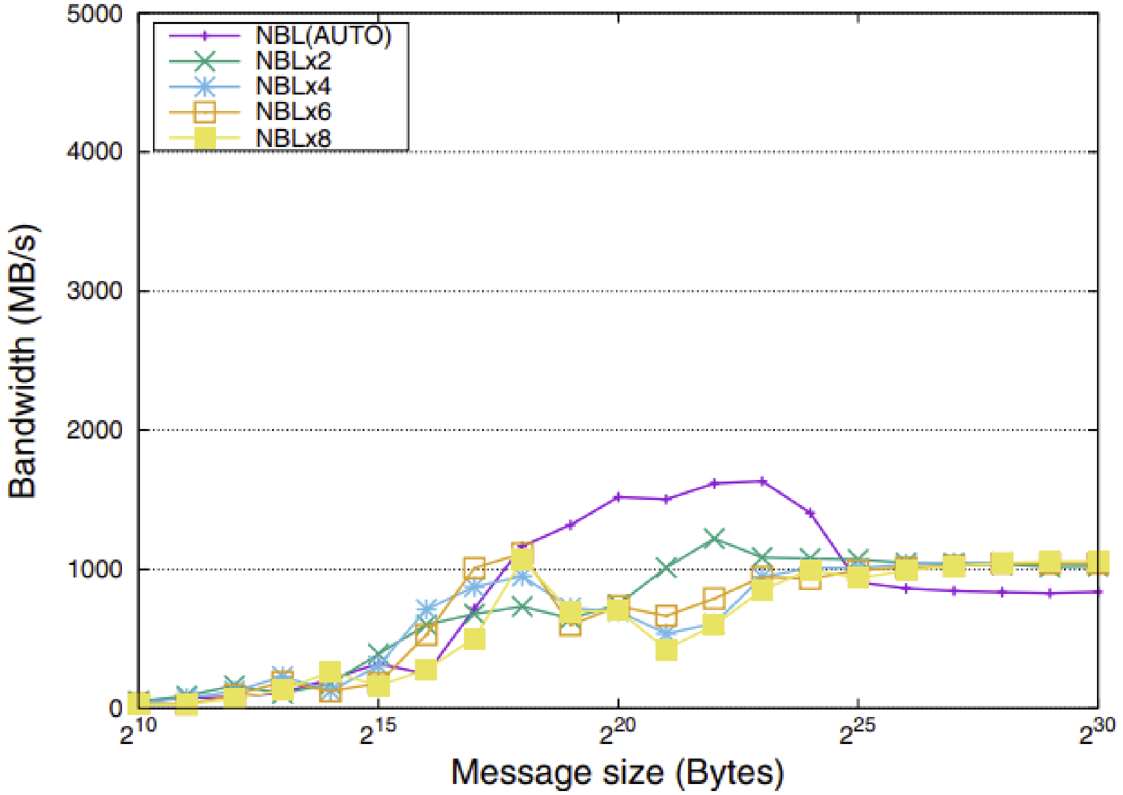
8 and 9 processes



Performance with Fixed #Segments (AUTO)

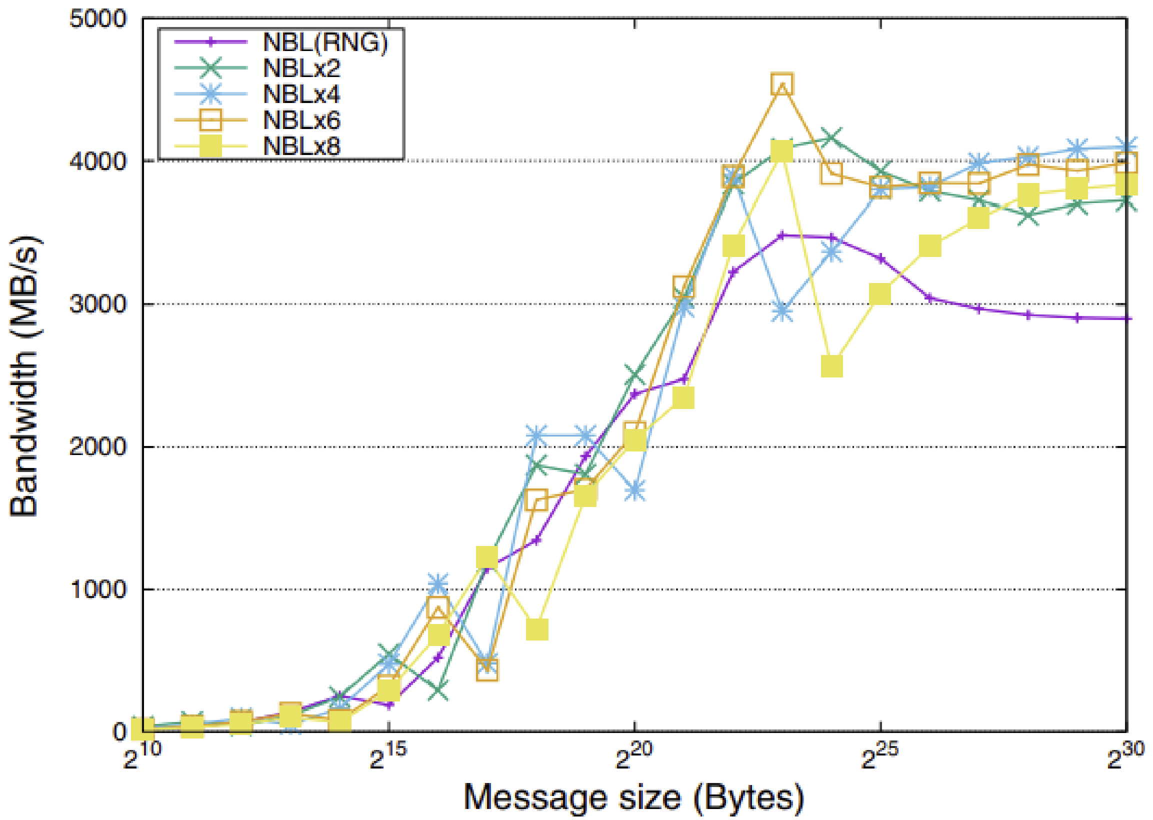
8 and 9 processes

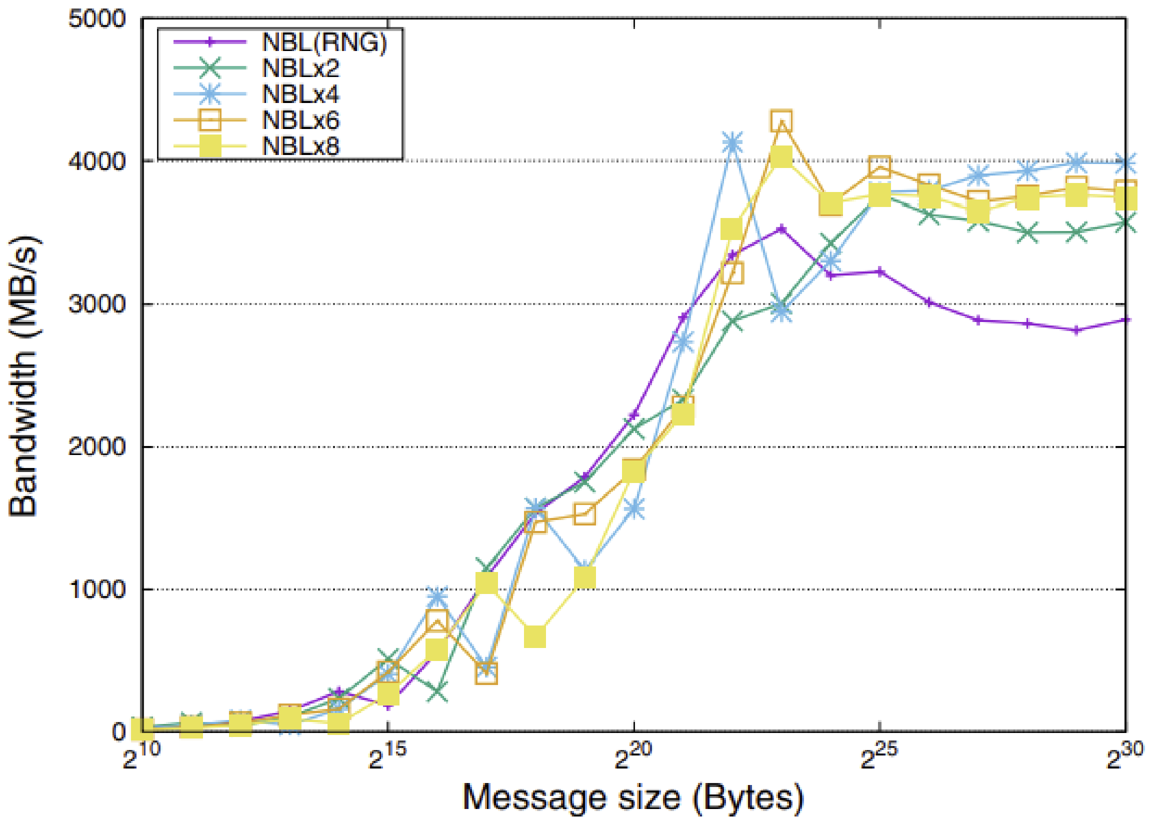




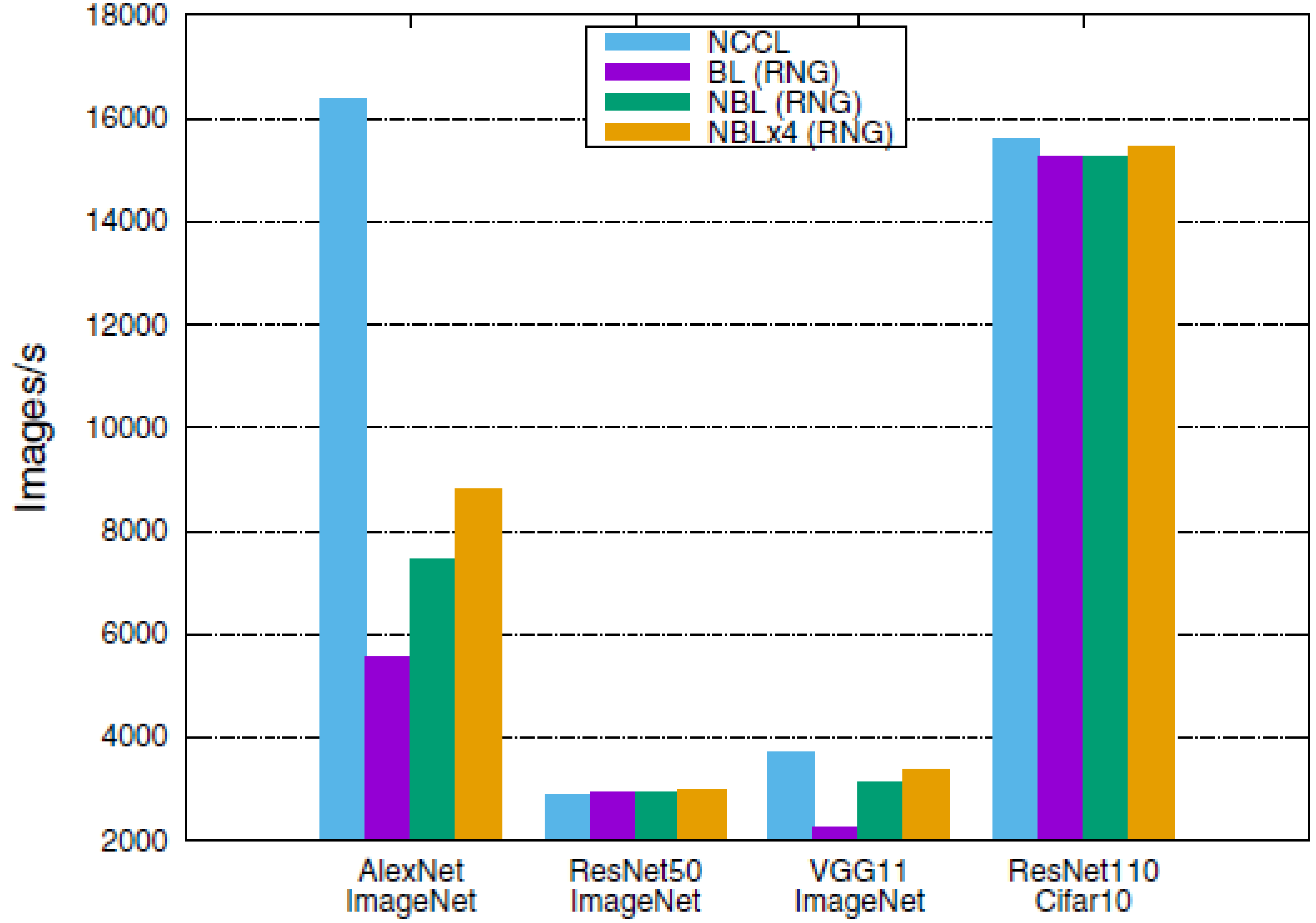
Performance with Fixed #Segments (RING)

8 and 9 processes

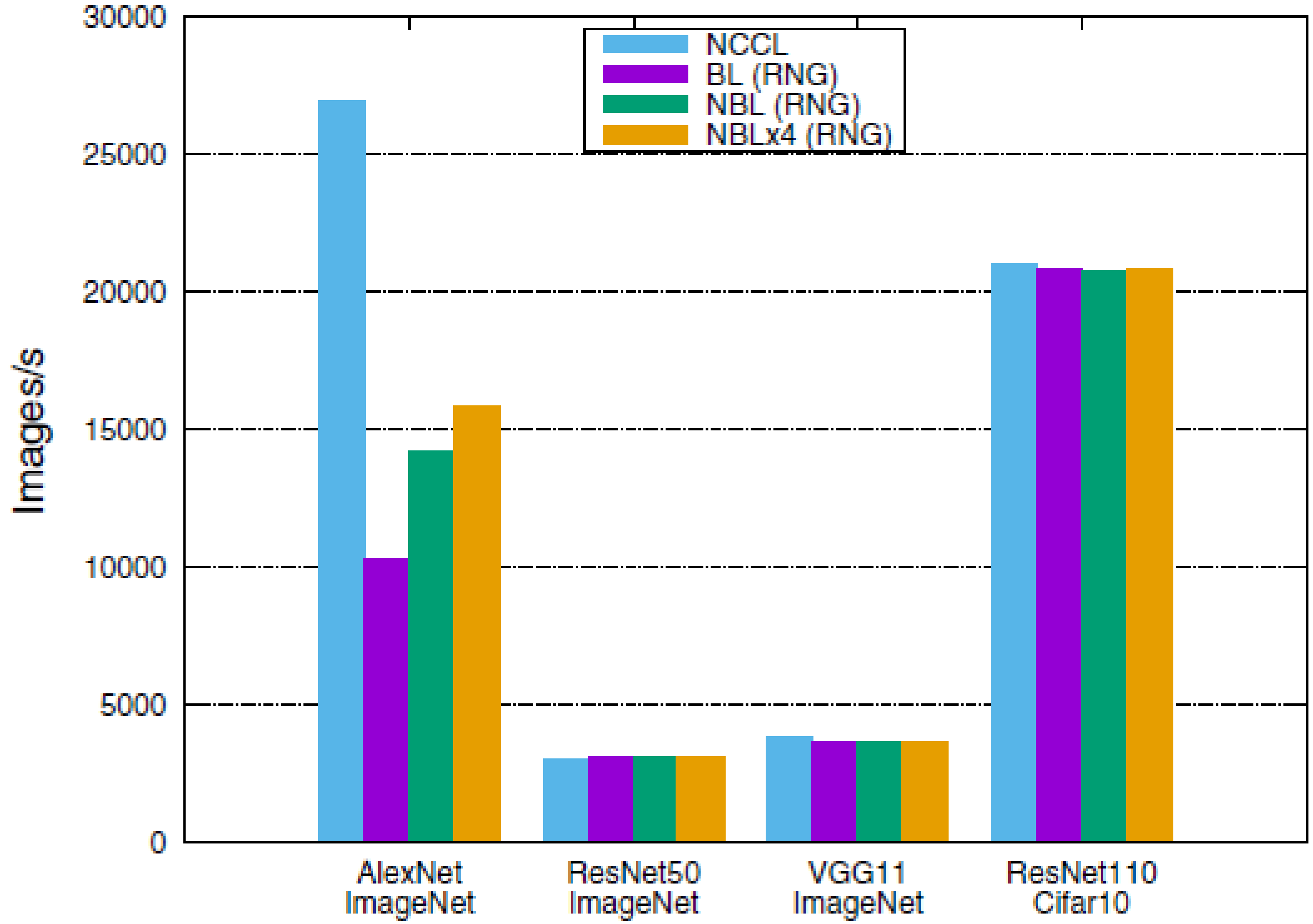




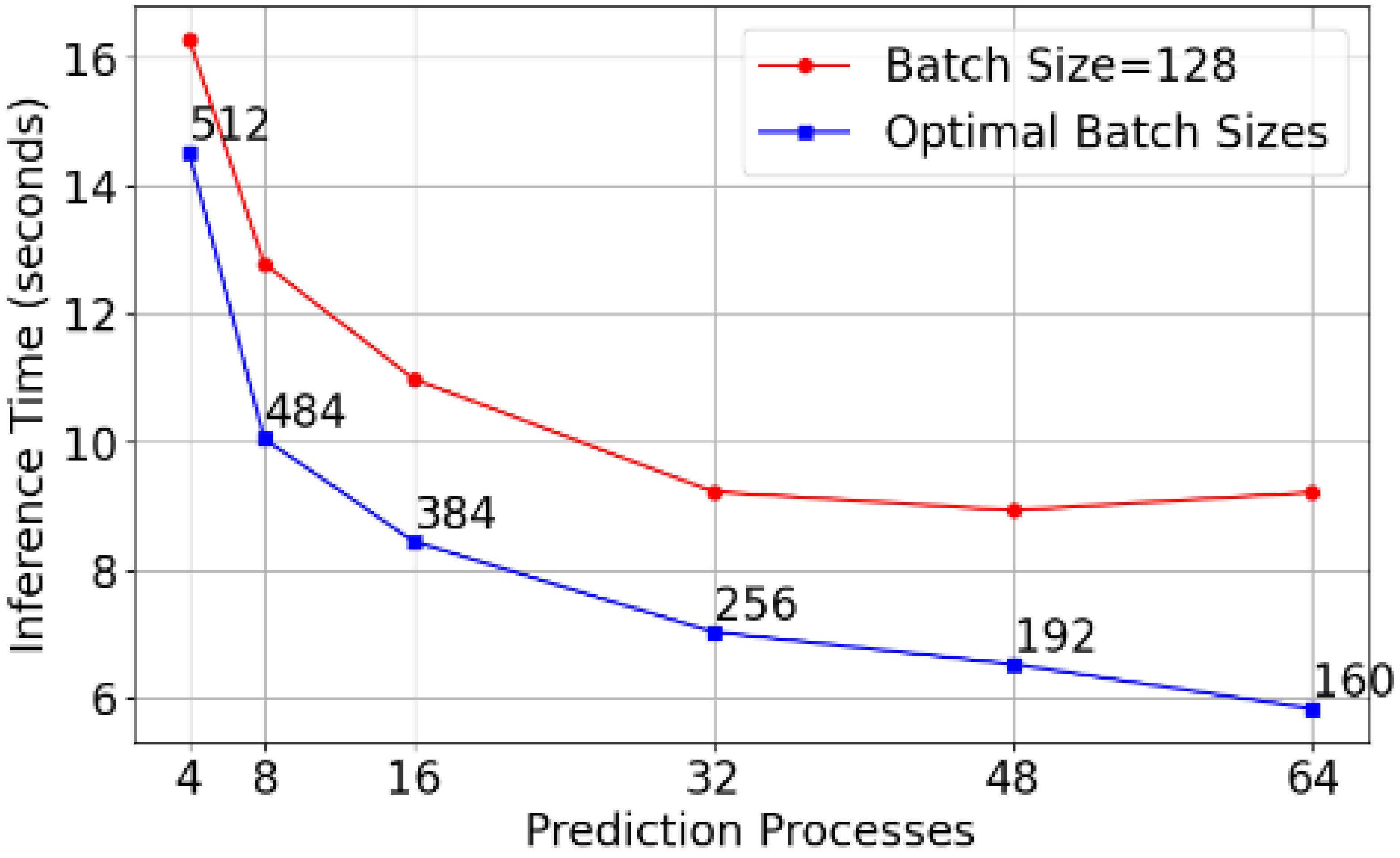
Performance of TF+HVD (Batch size = 128)



Performance of TF+HVD (Batch size = 256)



Optimal Batch Sizes Vary



Credit: M.A. Wani

Thank you for your attention!