ADAPT: An Event-Based Adaptive Collective Communication Framework

Xi Luo[1], Wei Wu[2], George Bosilca[1], Thananon Patinyasakdikul[1], Linnan Wang[3], Jack Dongarra[14]

1. University of Tennessee
2. Los Alamos National Laboratory
3. Brown University
4. Oak Ridge National Laboratory

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

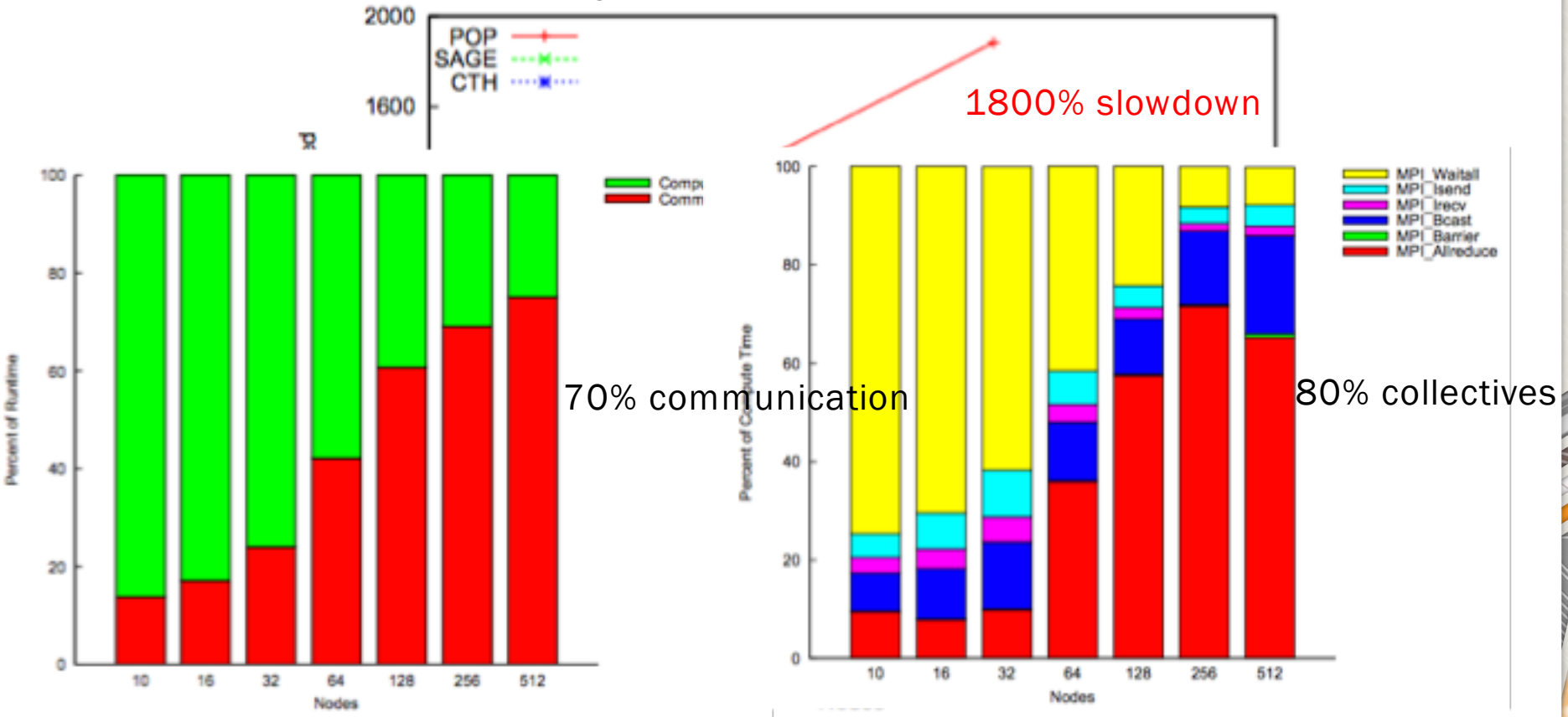
Los Alamos NATIONAL LABORATORY EST. 1943

The performance of collective operations

- Application performance is directly linked with communication performance
  - Point-to-point (two-sided and one-sided)
  - Collective operations
- What impacts the performance of collectives
  - Underlying algorithms
  - P2P performance
  - Noise (any thing can delay a function call)
  - Network topologies (and mapping strategies)
  - ……

Propagation of noise

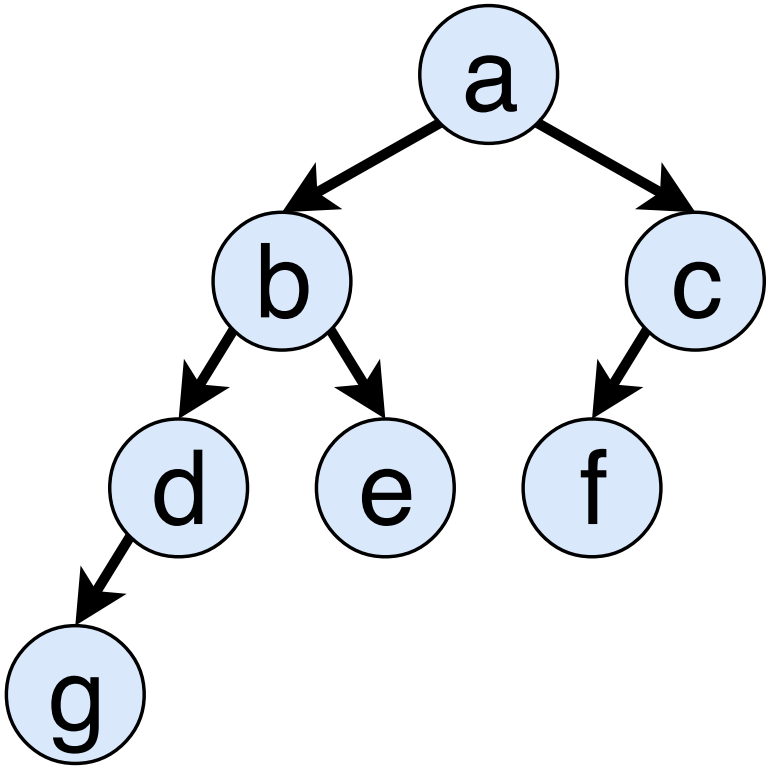
Application Performance with 2.5% noise (10 Hz frequency and 2.5 ms detour)



K. B. Ferreira, P. Bridges, and R. Brightwell. 2008. Characterizing application sensitivity to OS interference using kernel-level noise injection.

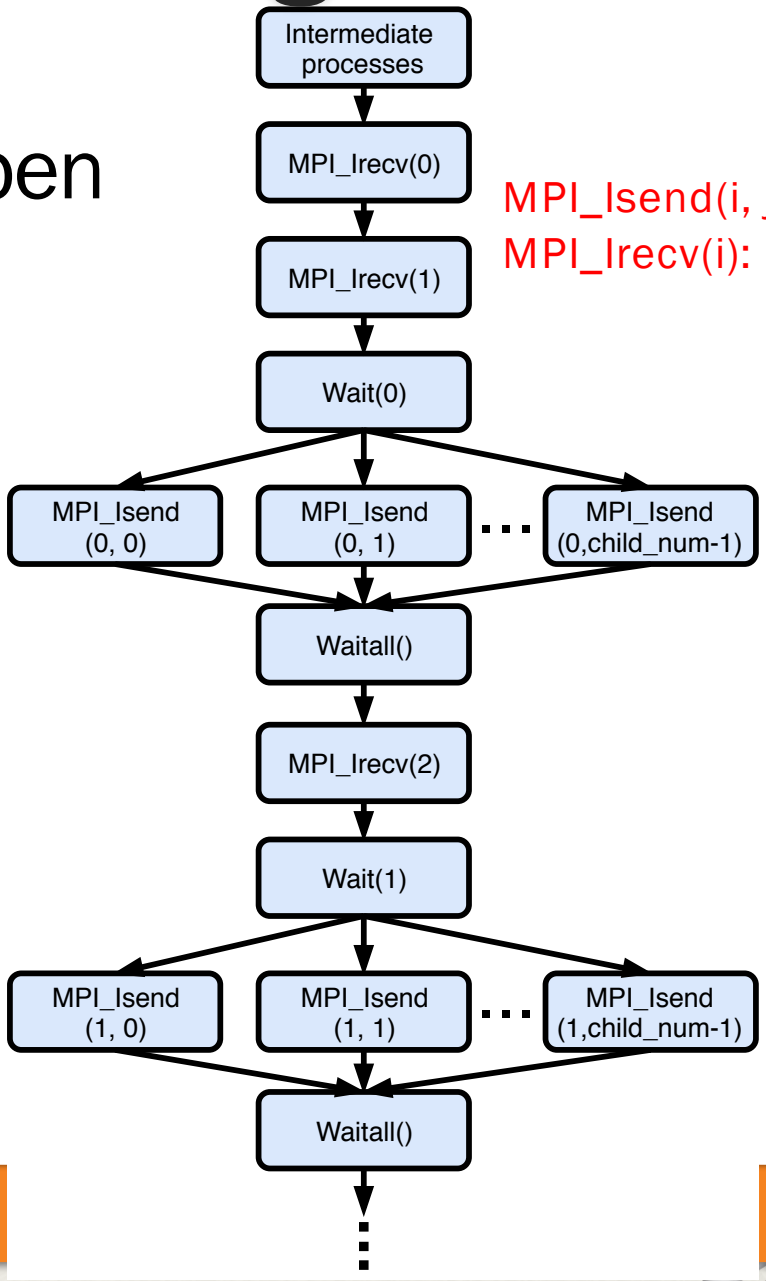
MPI_Bcast

- Collective is based on Point-to-point(P2P).
- Propagate messages from root to all the other processes.
- Apply pipelining for big messages.
  - Divided messages into small segments.
  - Segments are propagated one by one.
- Focus on intermediate.



MPI_Bcast using non-blocking P2P

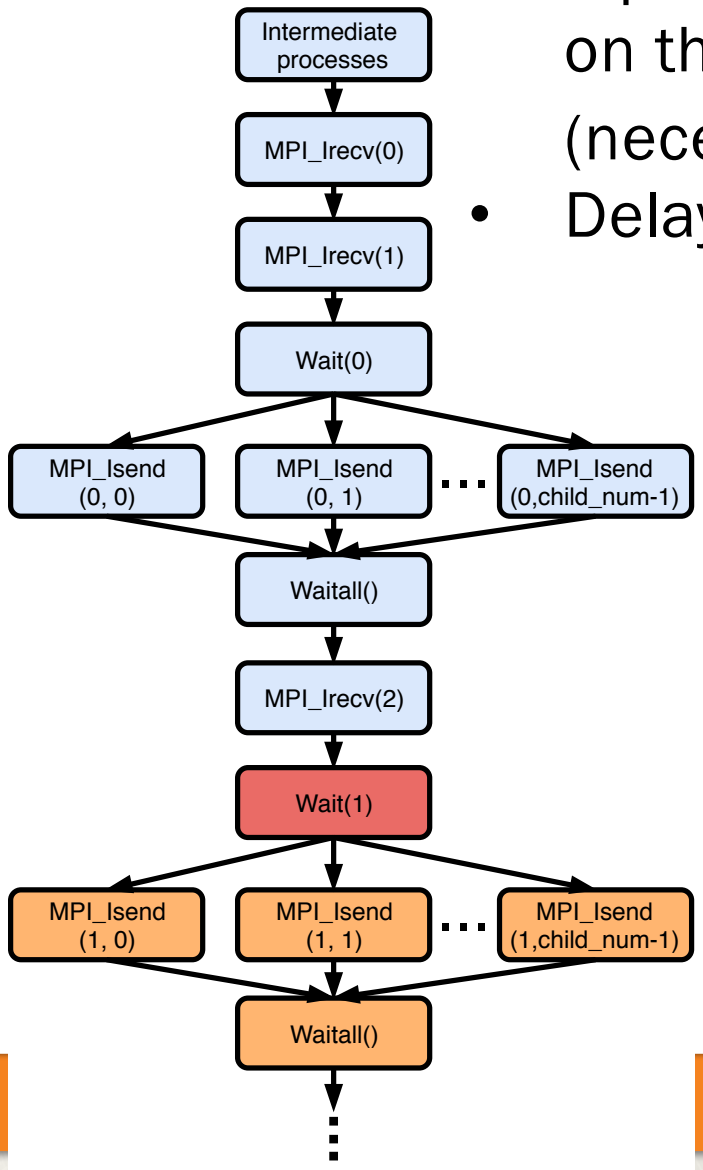
- Default in Open MPI

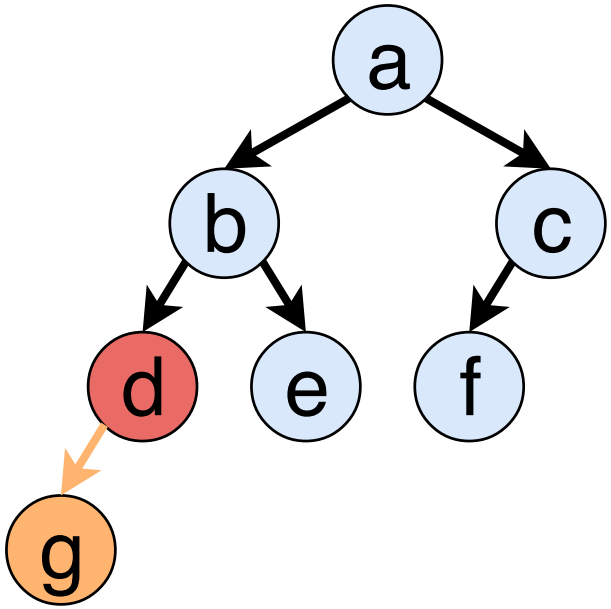


MPI_Isend(i, j): Send segment i to child j
MPI_Irecv(i): Recv segment i

Data dependency

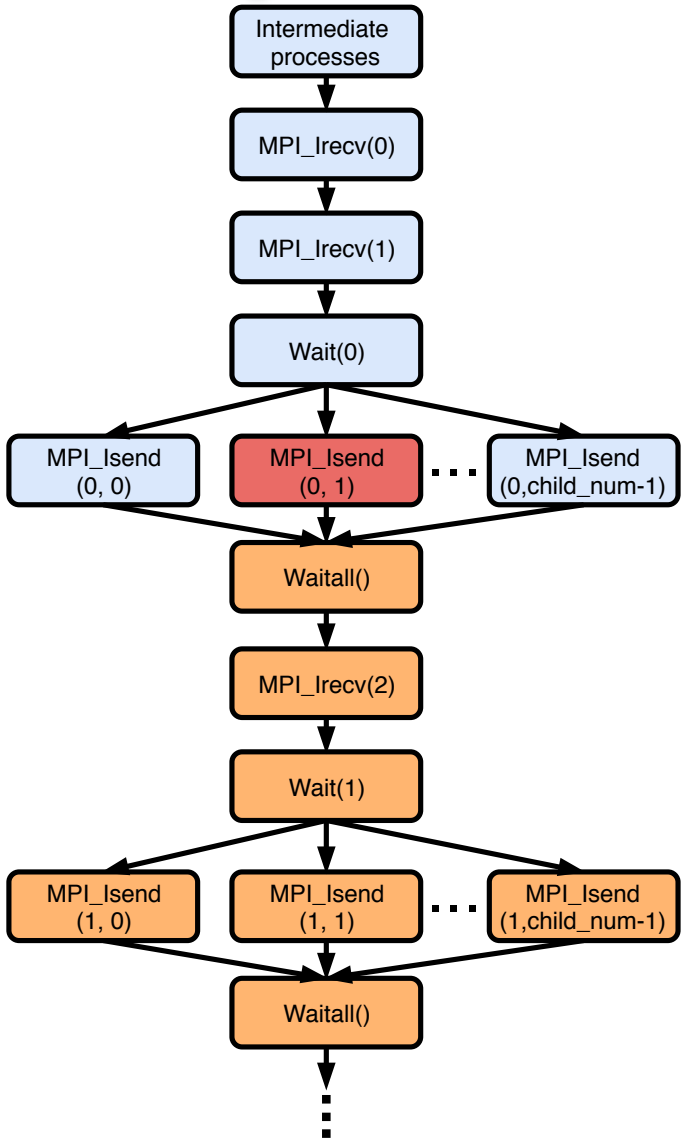
- Input data of some P2P routines depends on the output data of other P2P routines. (necessary)
- Delay





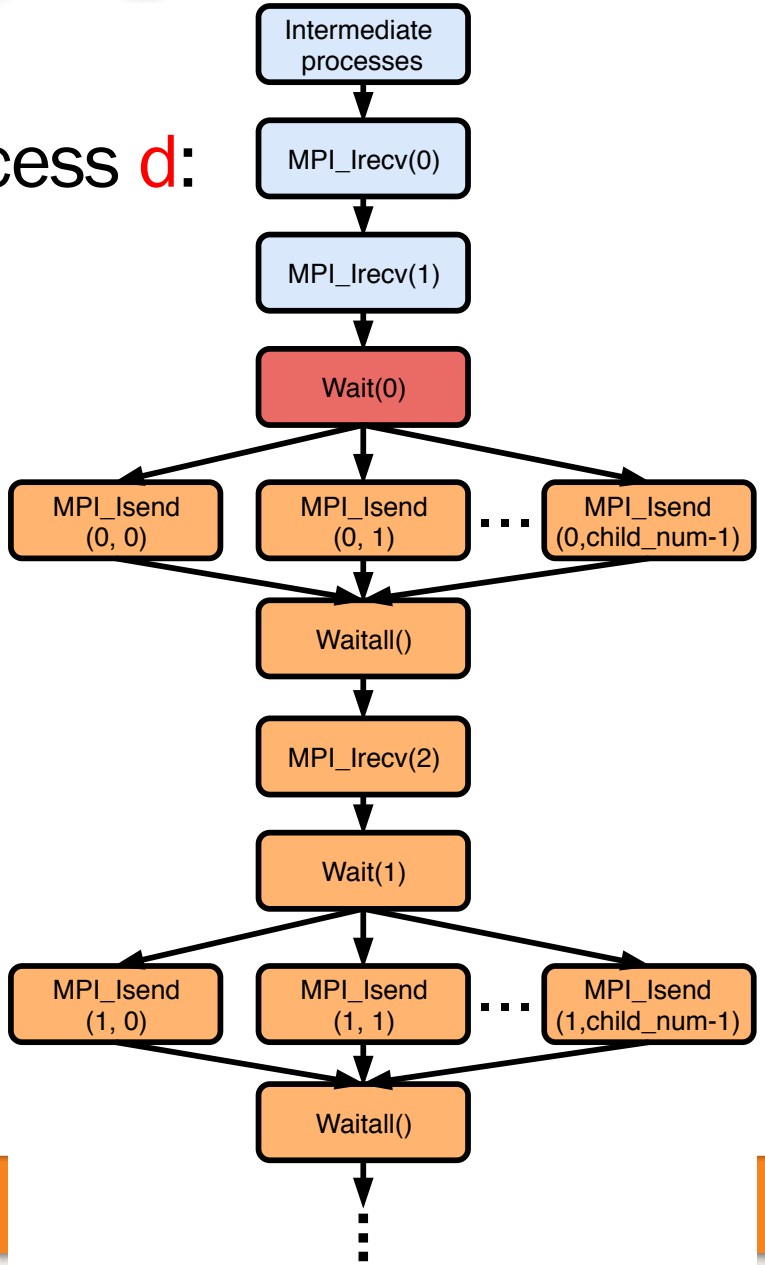
Synchronization dependency

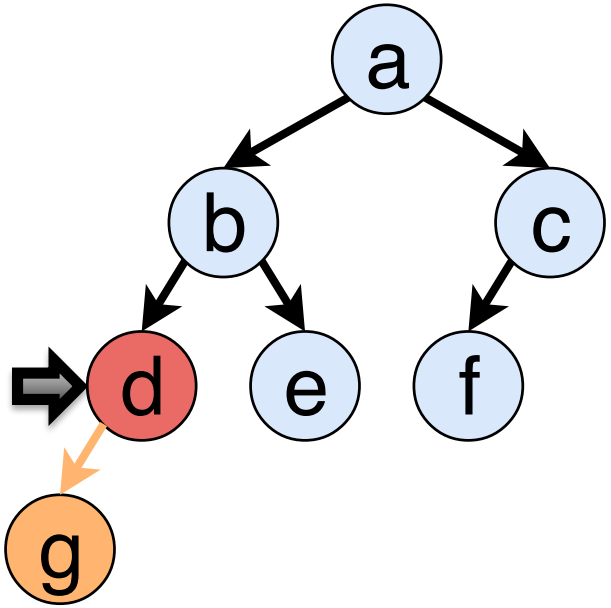
- Caused by synchronizations between P2P routines. (unnecessary)
- Waitall and Wait act as synchronizations that order P2P routines between them.
  - Segment (in order)
- Delay



Propagation of noise

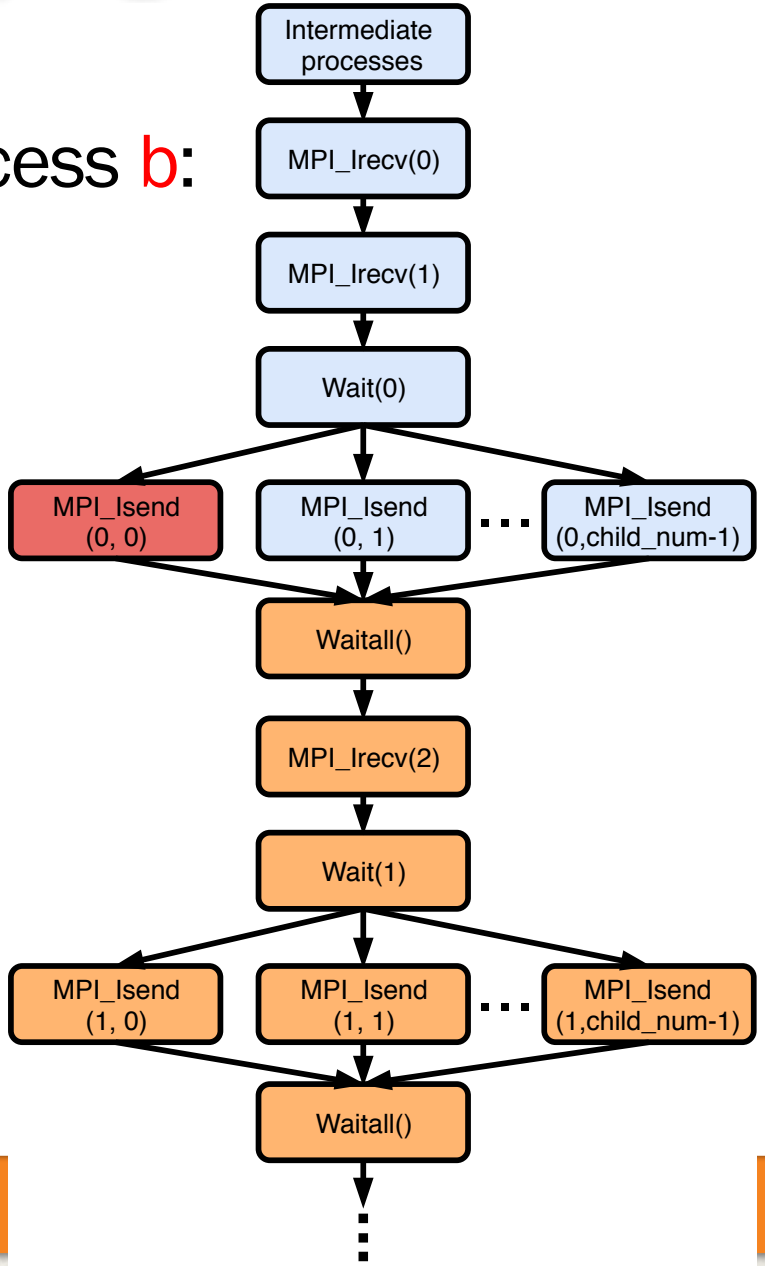
On process d:

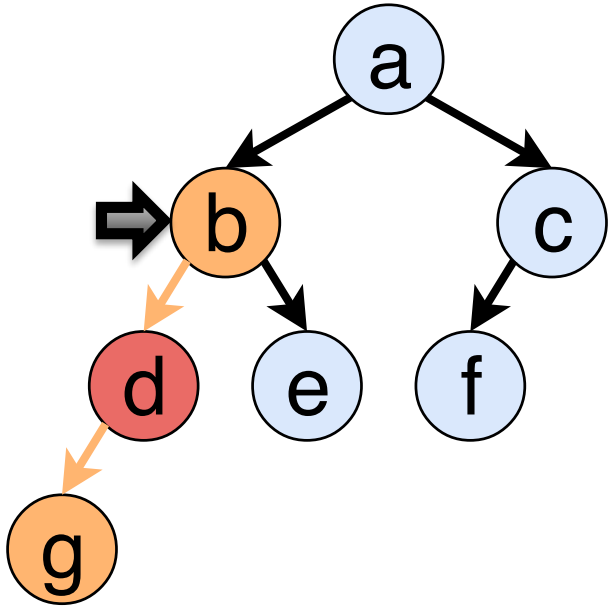




Propagation of noise

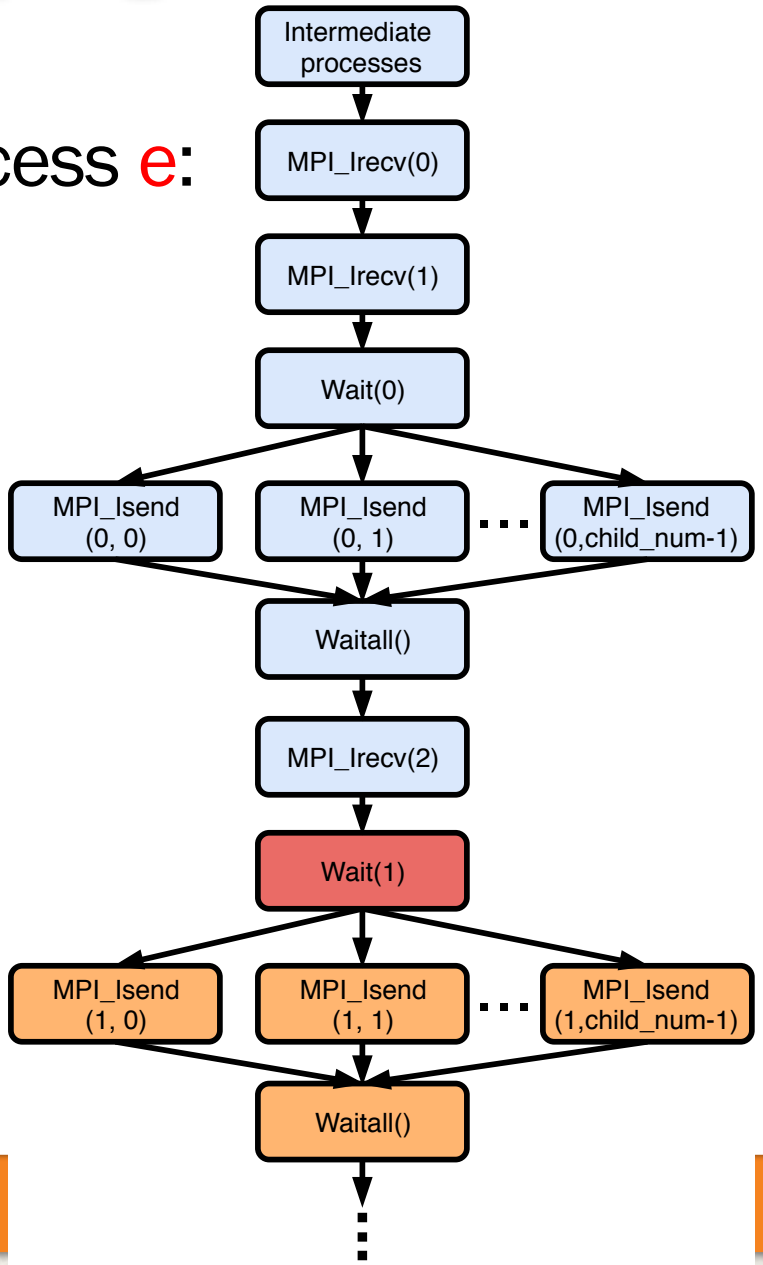
On process b:

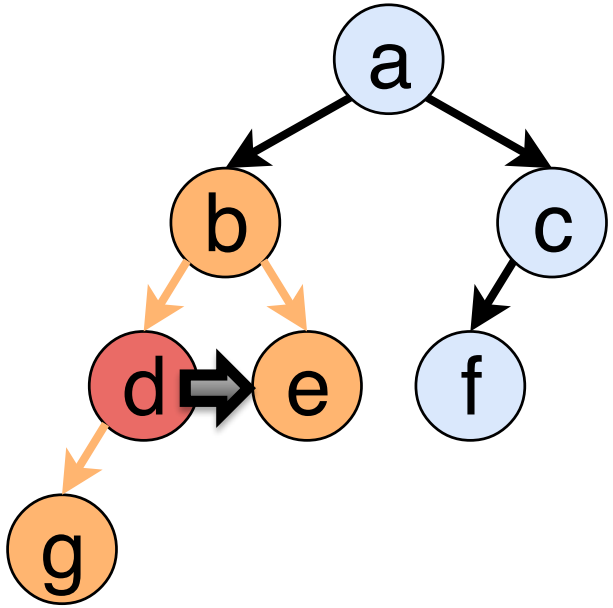




Propagation of noise

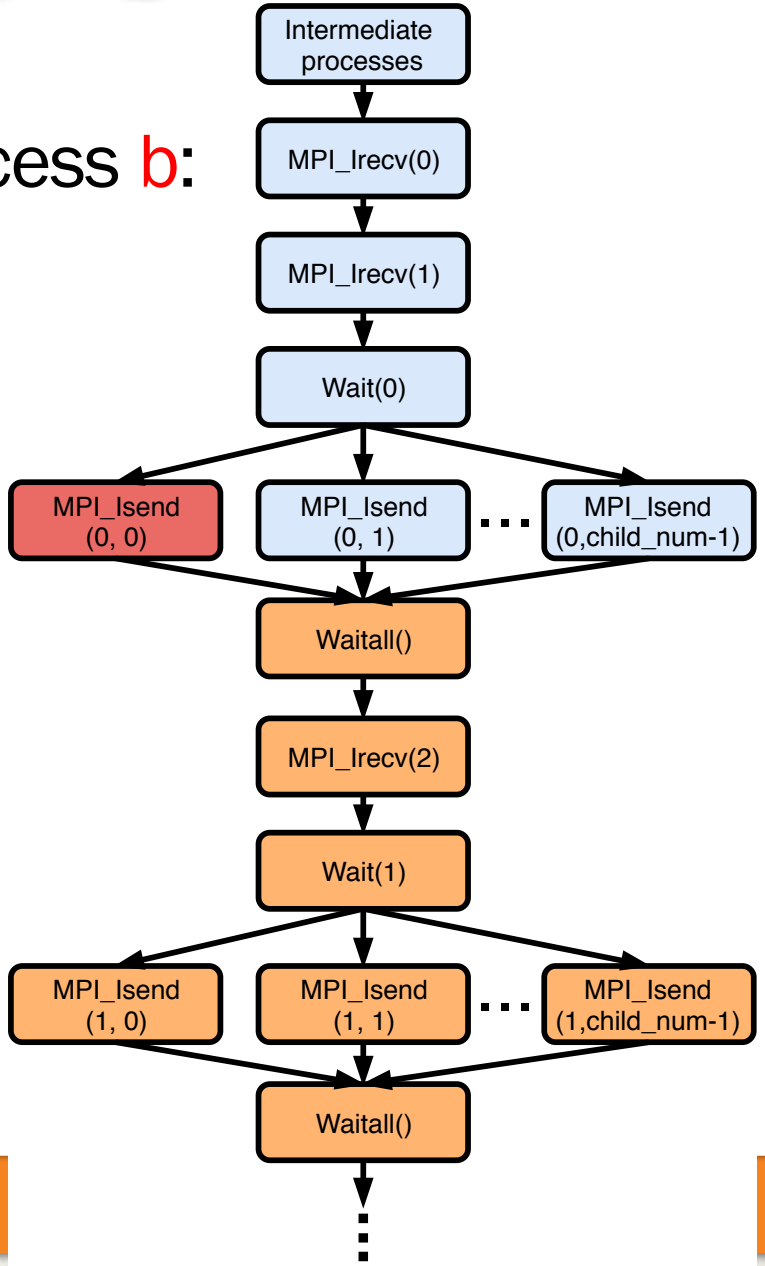
On process e:

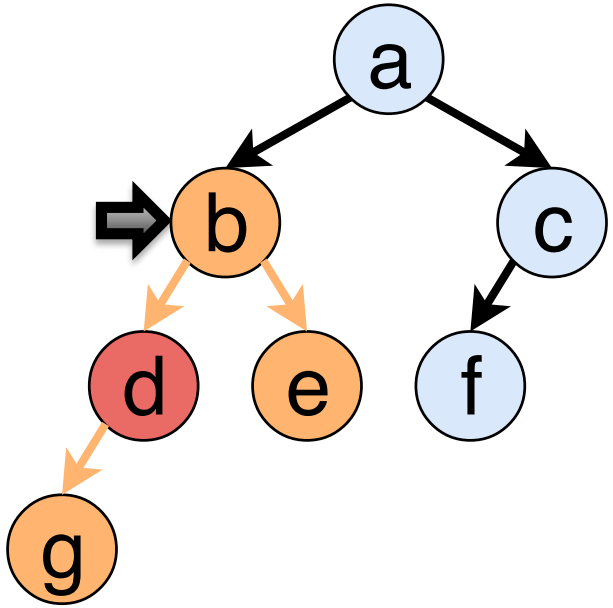




Propagation of noise

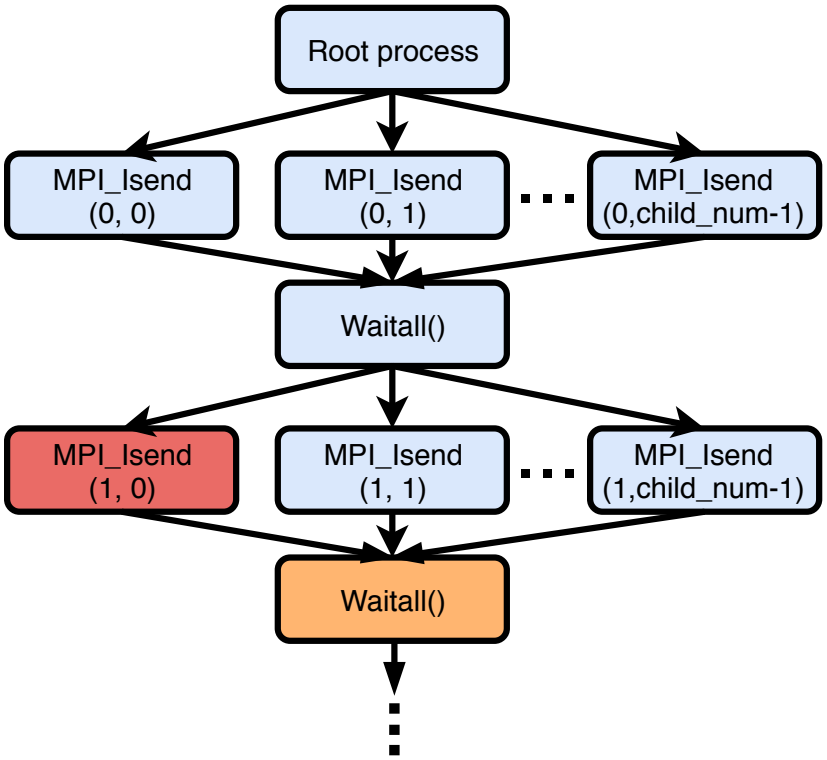
On process b:

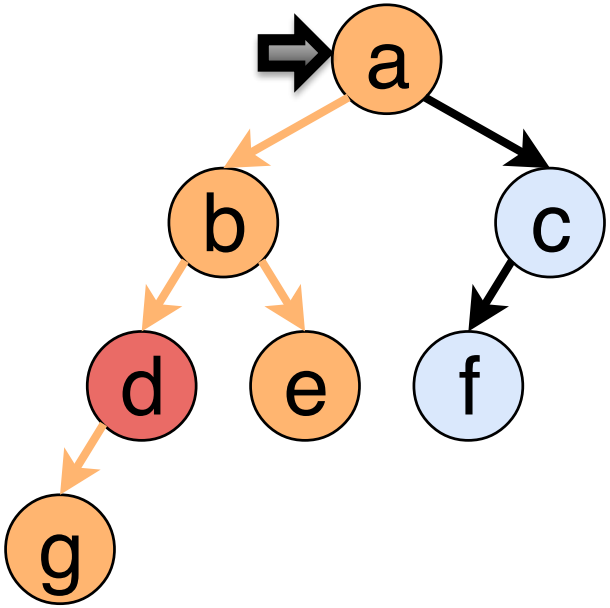




Propagation of noise

On process a:

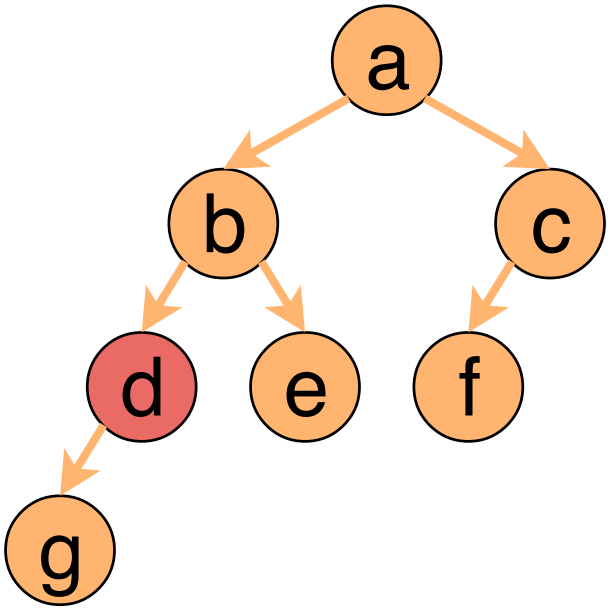




Propagation of noise

Summary:

- Segment (in order)
- One process will delay all the processes



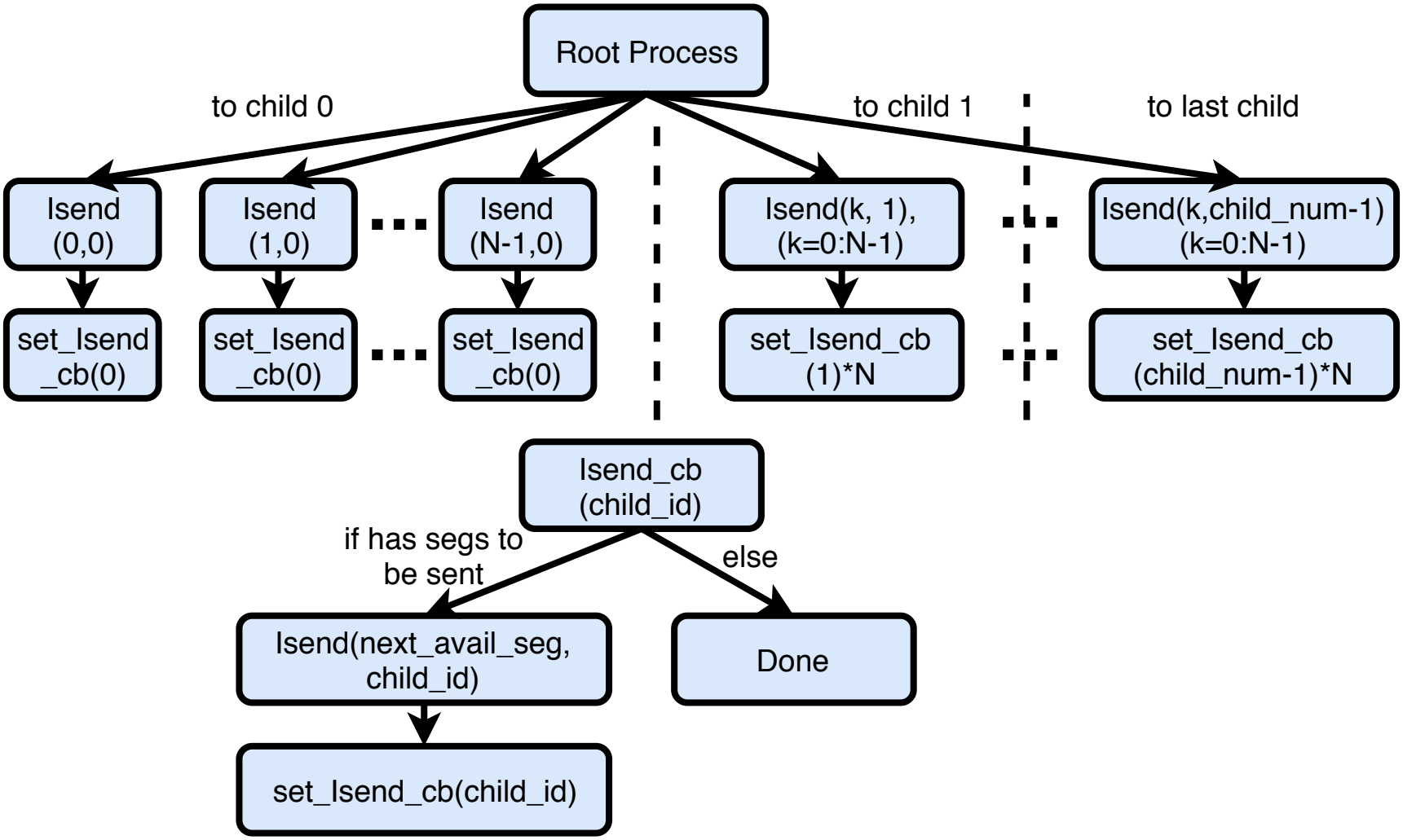
Event-driven Design

- Event-driven programming is a long existing programming model.
  - Event driven I/O.
  - Embedded systems.
  - Mobile networks.
  - Server applications.
- An event loop to detect events.
- When an event occurs, the corresponding callback is triggered.

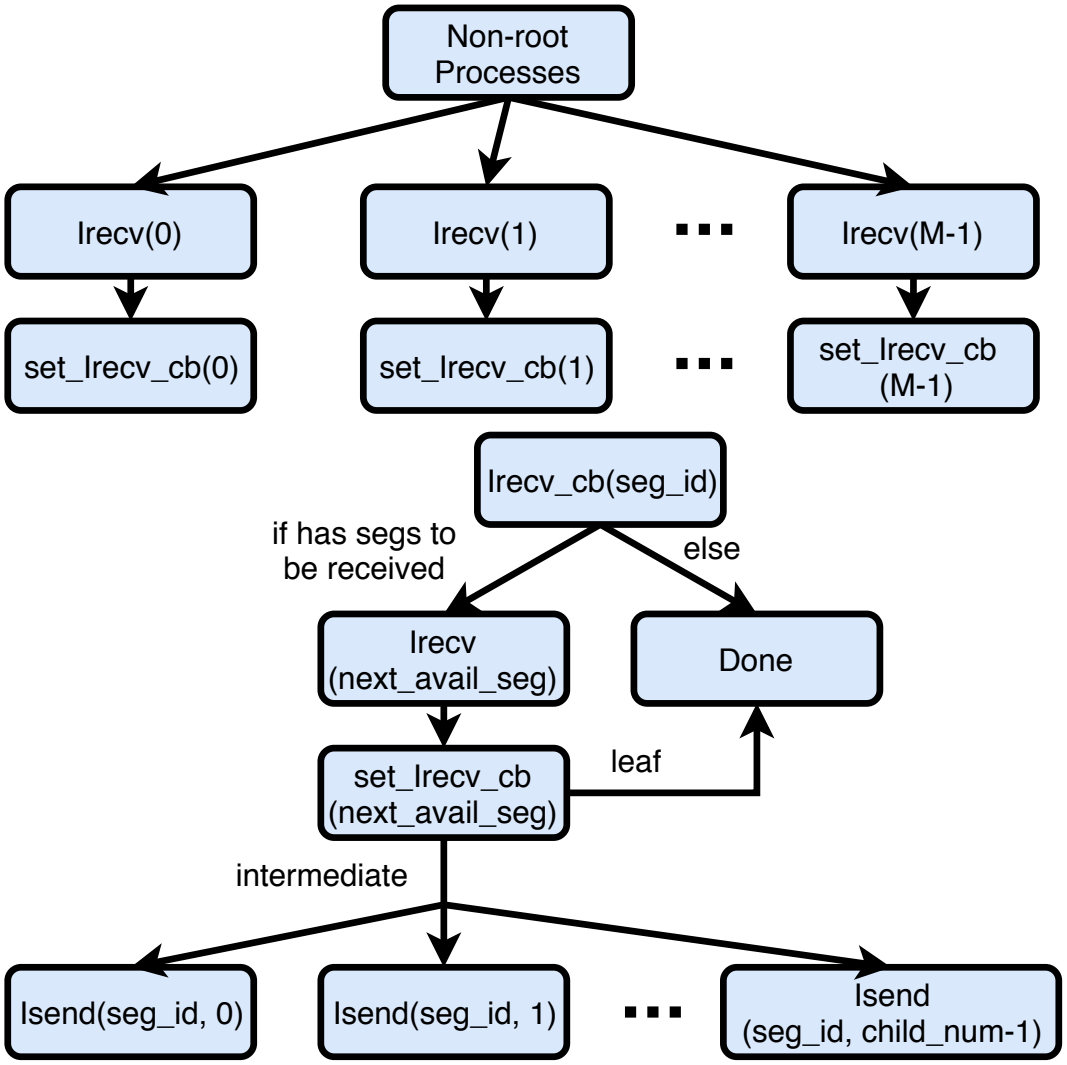
ADAPT

- Design collective operations with events and callbacks.
  - Eliminate the need to wait for separate P2P to complete.
  - Relax the synchronizations.
- Event loop: Open MPI progress engine.
- Events: the completion of non-blocking P2P routines.
- Callbacks: analyze the state of the collective algorithms, and post new non-blocking P2P if necessary.
- Using lower level non-blocking P2P.

MPI_Bcast in ADAPT (root)

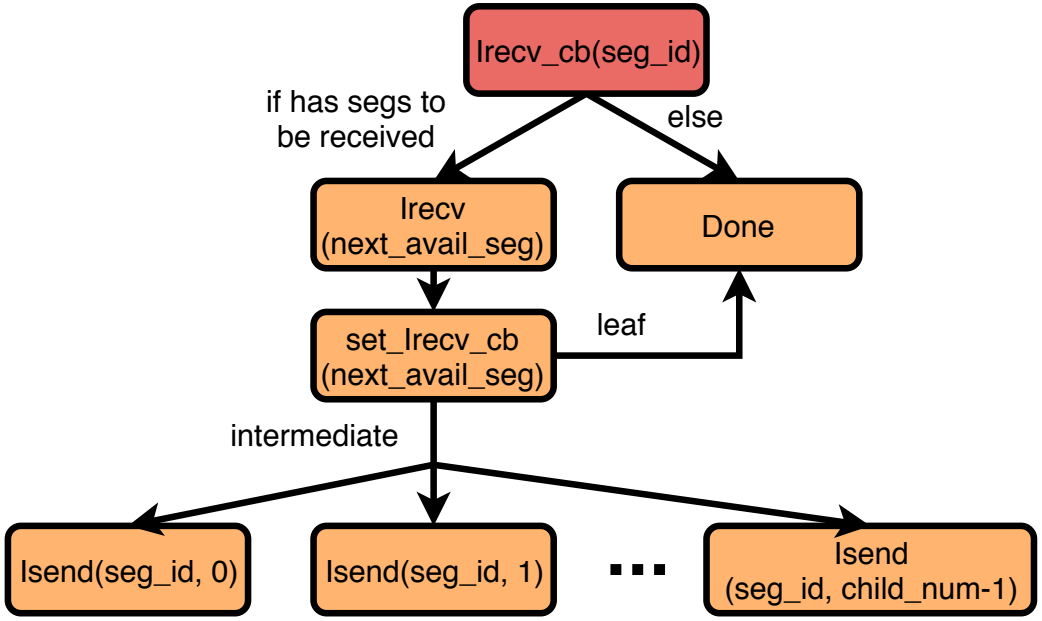


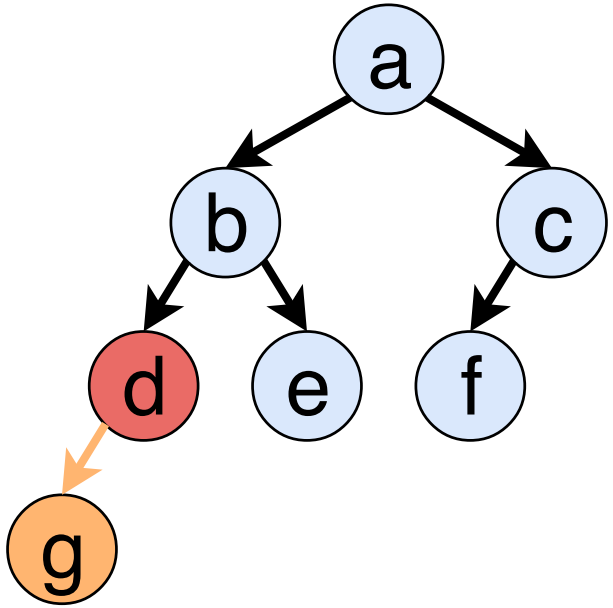
MPI_Bcast in ADAPT (non-root)



Data dependency

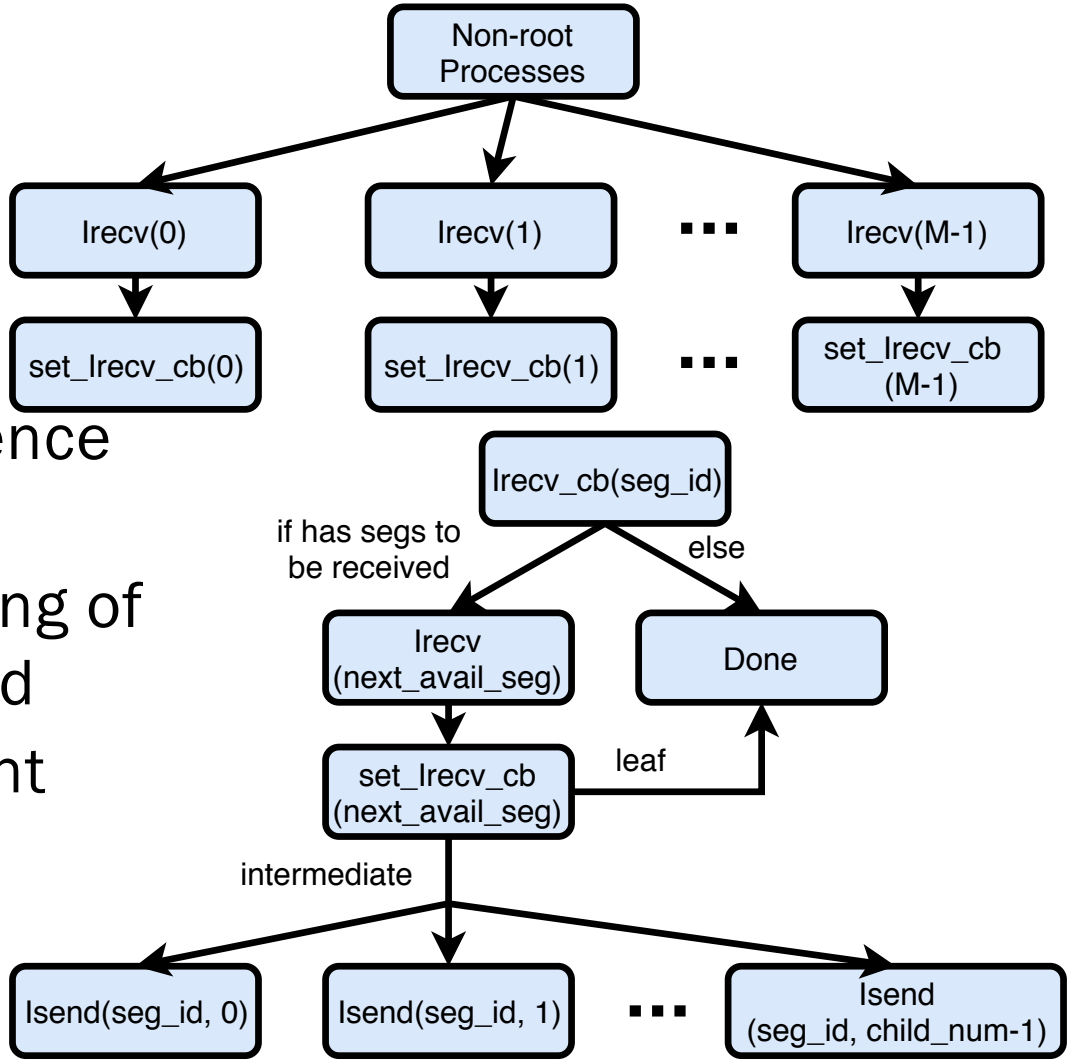
- same as previous implementations.





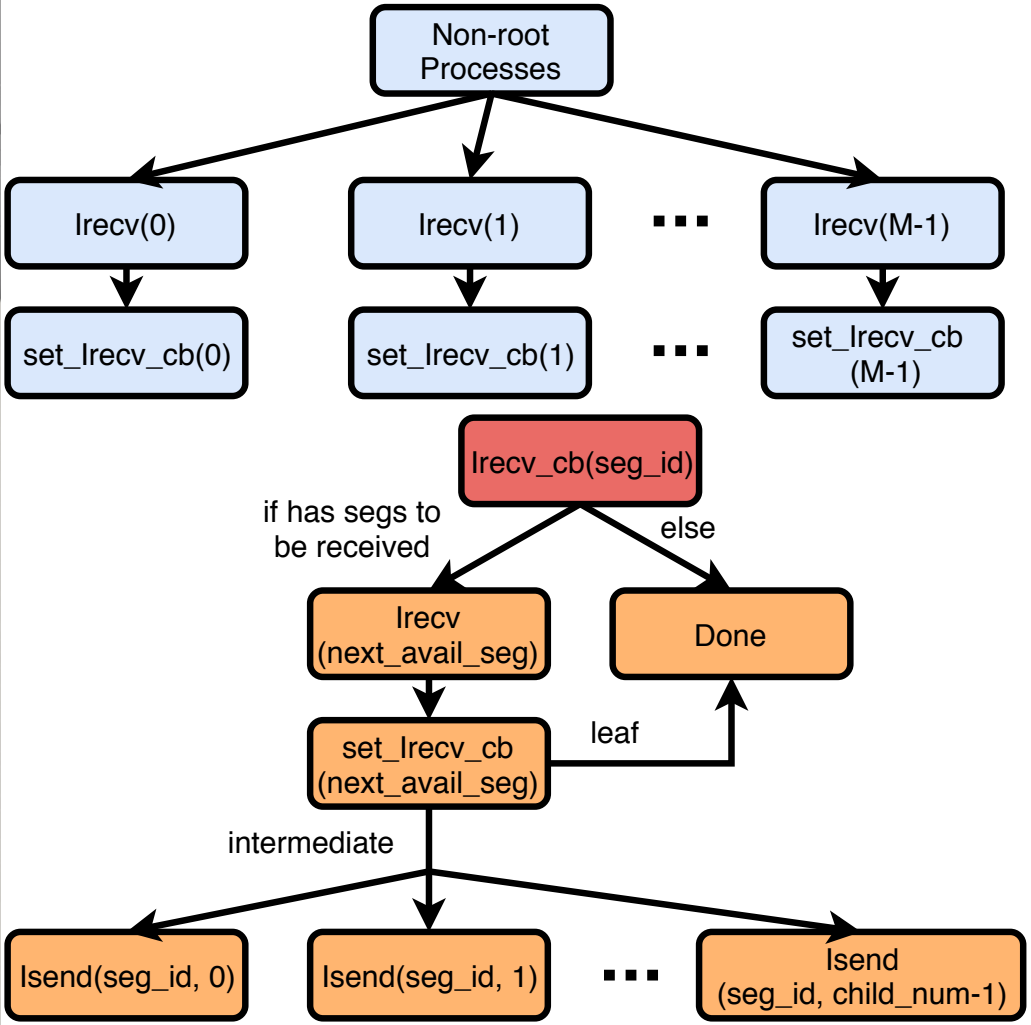
Synchronization dependency

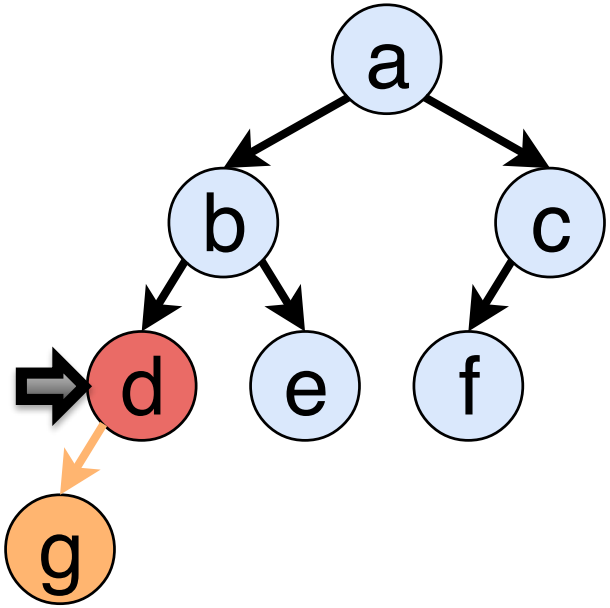
- Segment independence
  - Rebalance
  - Decouple receiving of next segment and sending of current segment
- Child independence



Propagation of noise

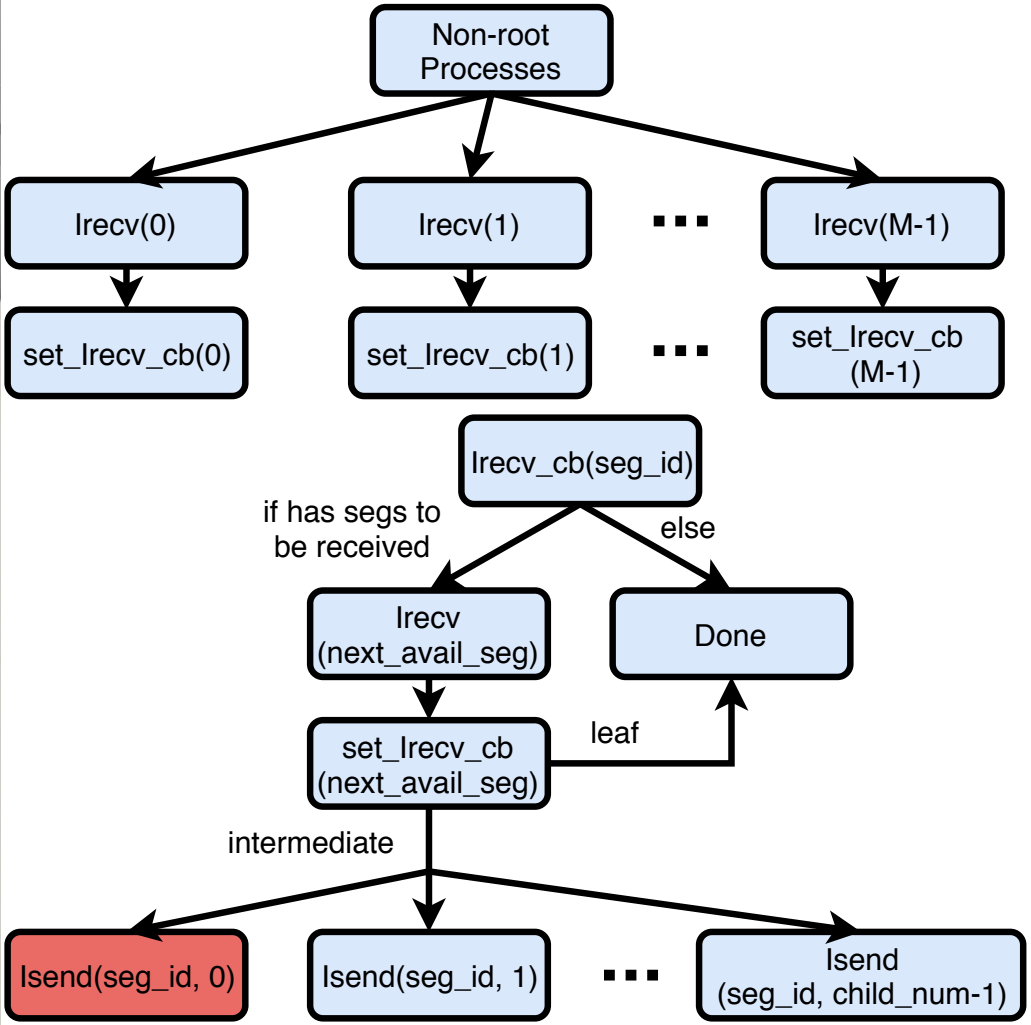
On process d:

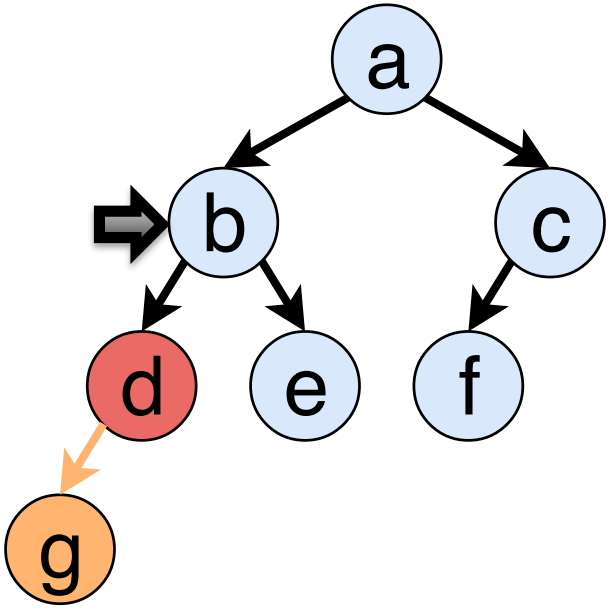




Propagation of noise

On process b:

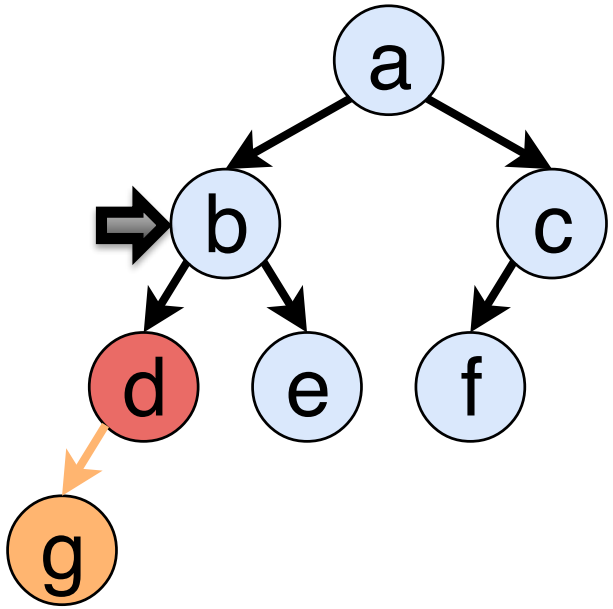




Propagation of noise

Summary:

- Segment independence
- Child independence
- One process will only delay its descendants



Extend to other collective operations

General Problem: Issue multiple send/recv

Algorithm 1: Blocking P2P Implementation

```
1 for i ← 0 to k do
2     MPI_Send(i)/MPI_Recv(i);
```

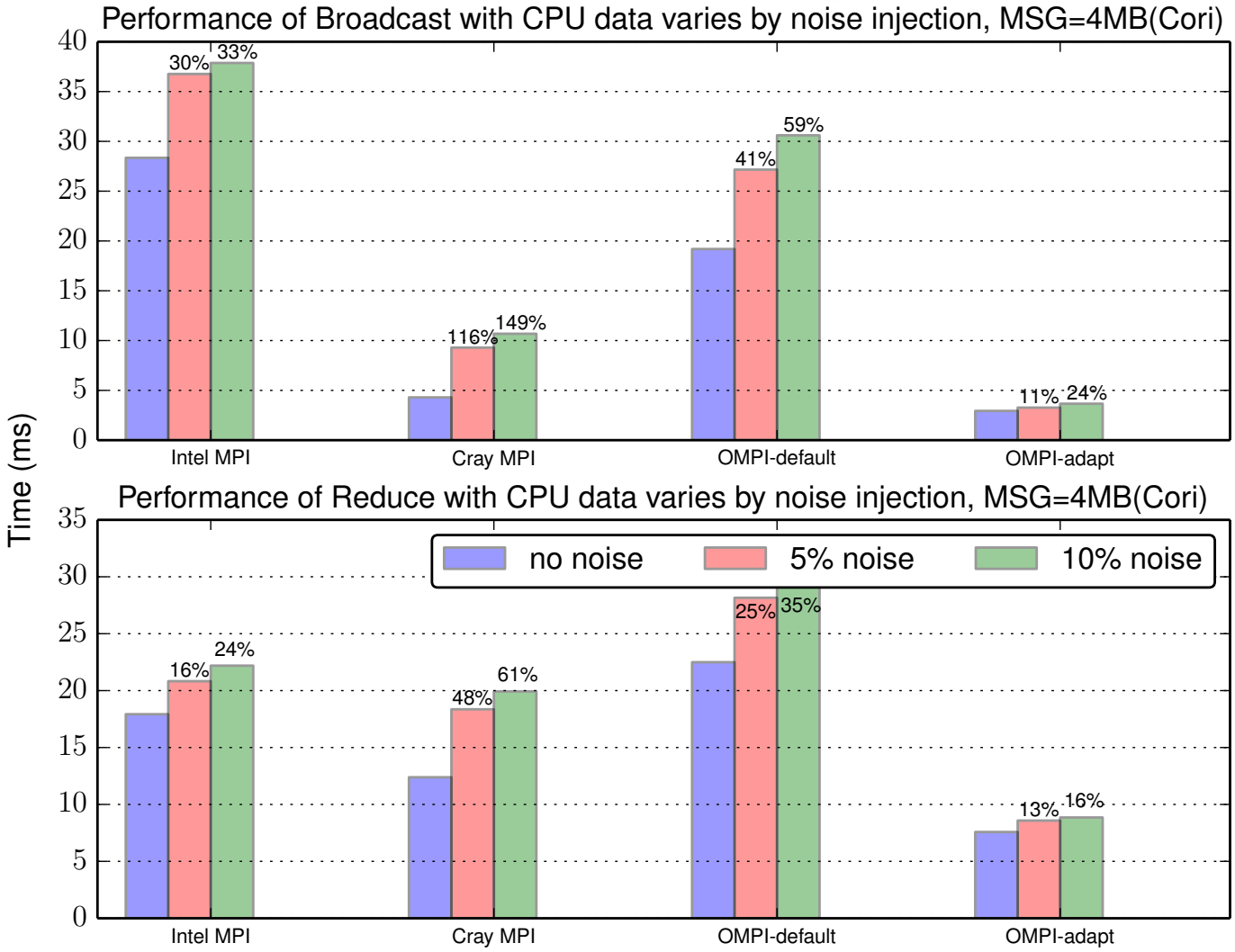
Algorithm 2: Nonblocking P2P Implementation

```
1 for i ← 0 to k do
2     MPI_Isend(i)/MPI_Irecv(i);
3 Waitall()
```

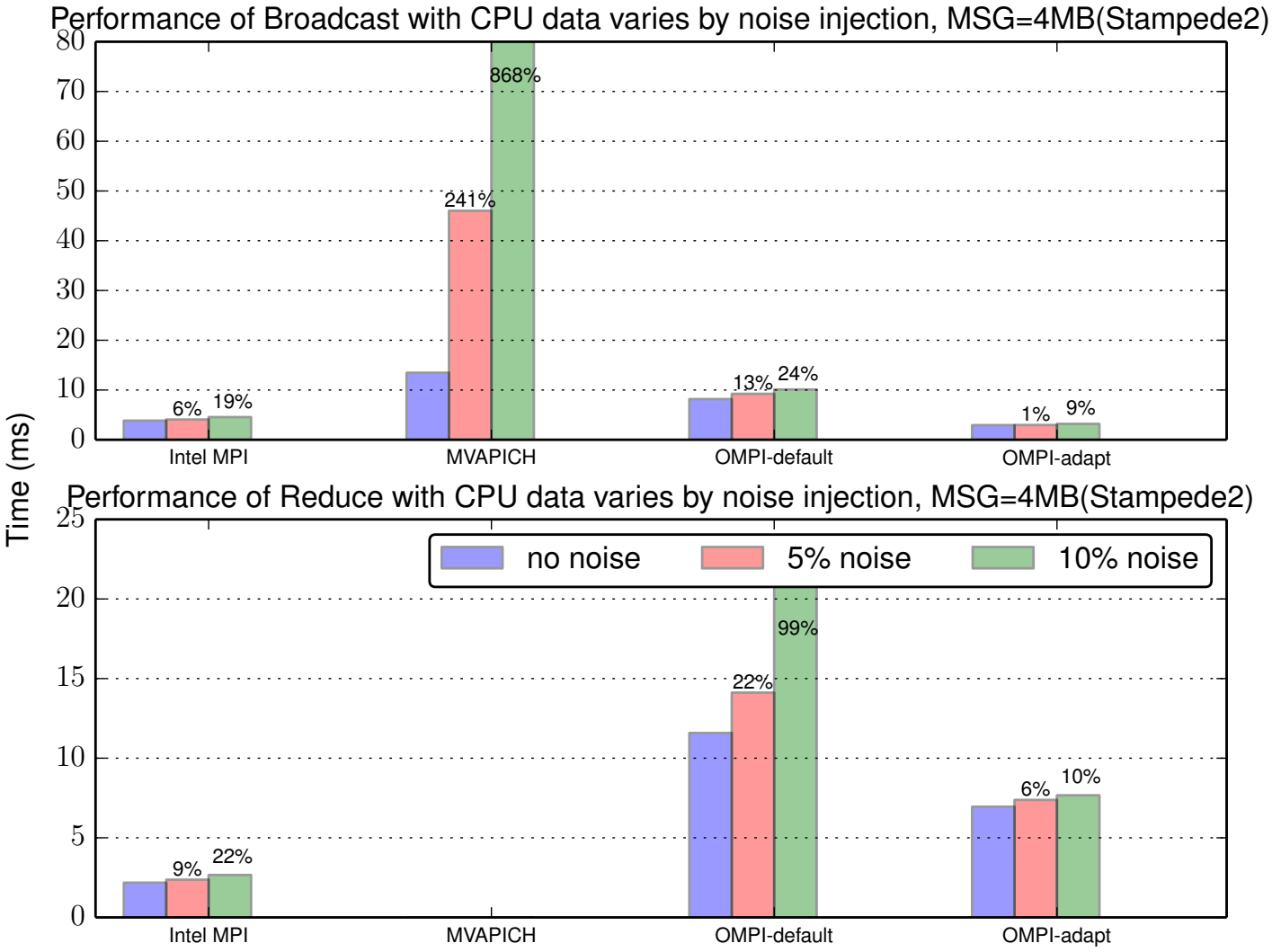
Algorithm 3: Adapt Implementation

```
1 for i ← 0 to k do
2     Isend(i)/Irecv(i);
3     set_Isend_cb(i)/set_Irecv_cb(i);
```

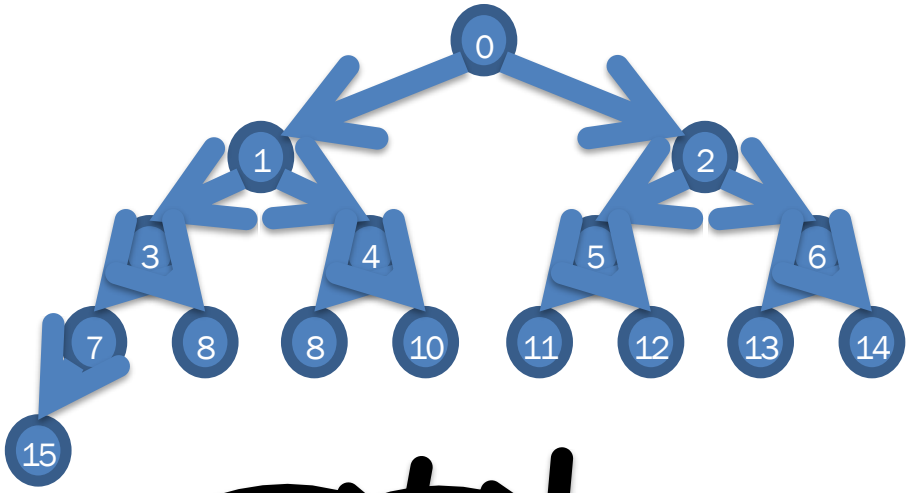
Experiments (noise injection)

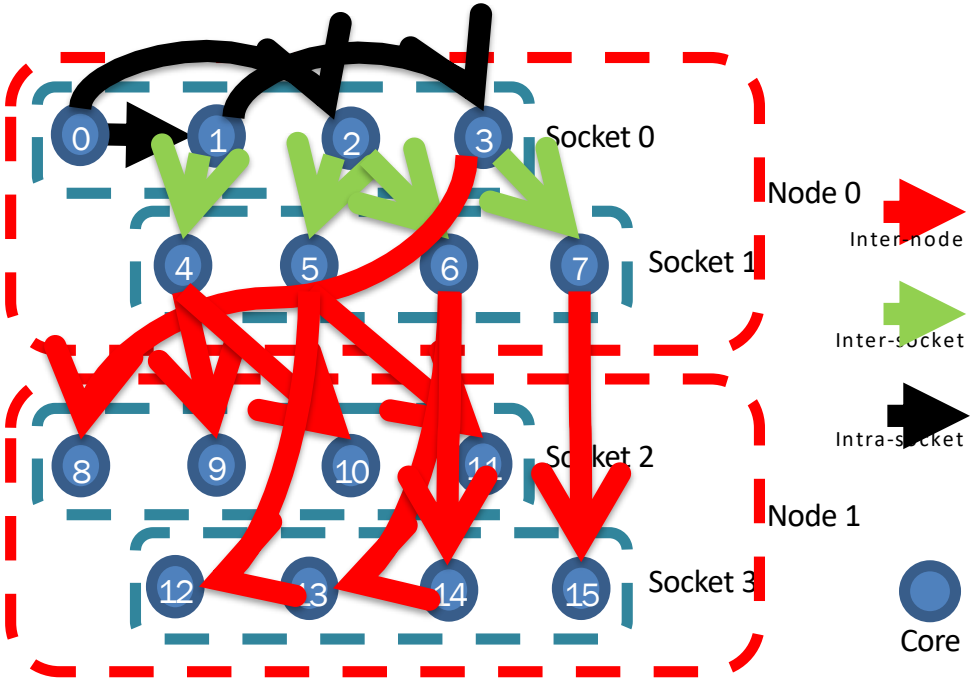


Experiments (noise injection)



Network Topology





15 P2P communication

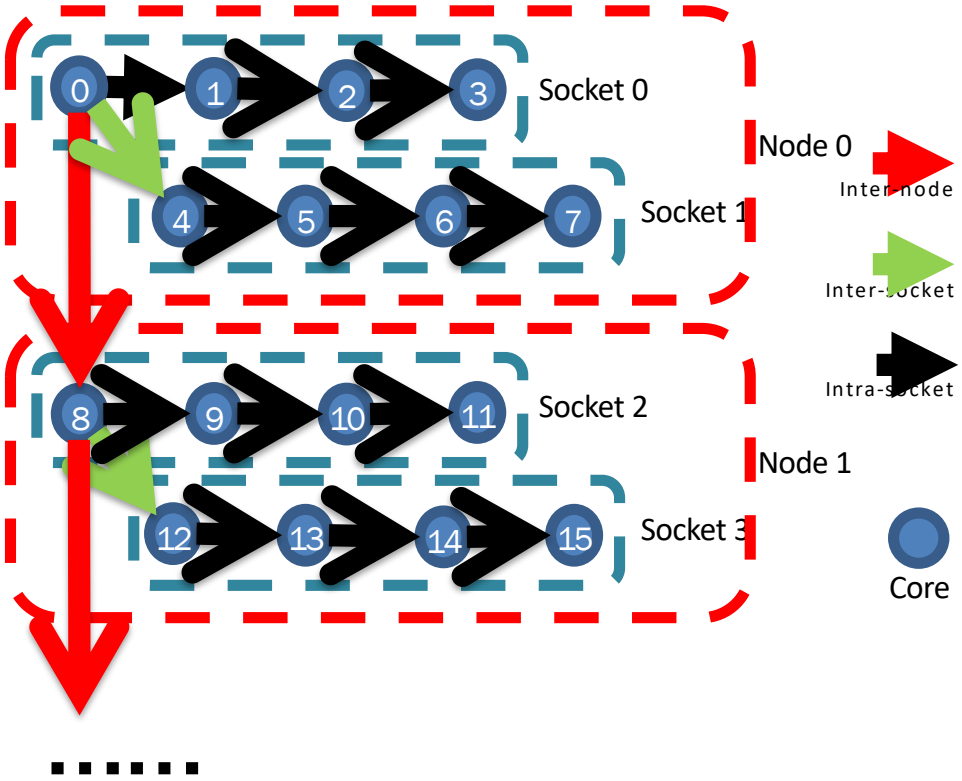
8 inter-node communication

4 inter-socket communication

3 intra-socket communication

Topology-aware tree

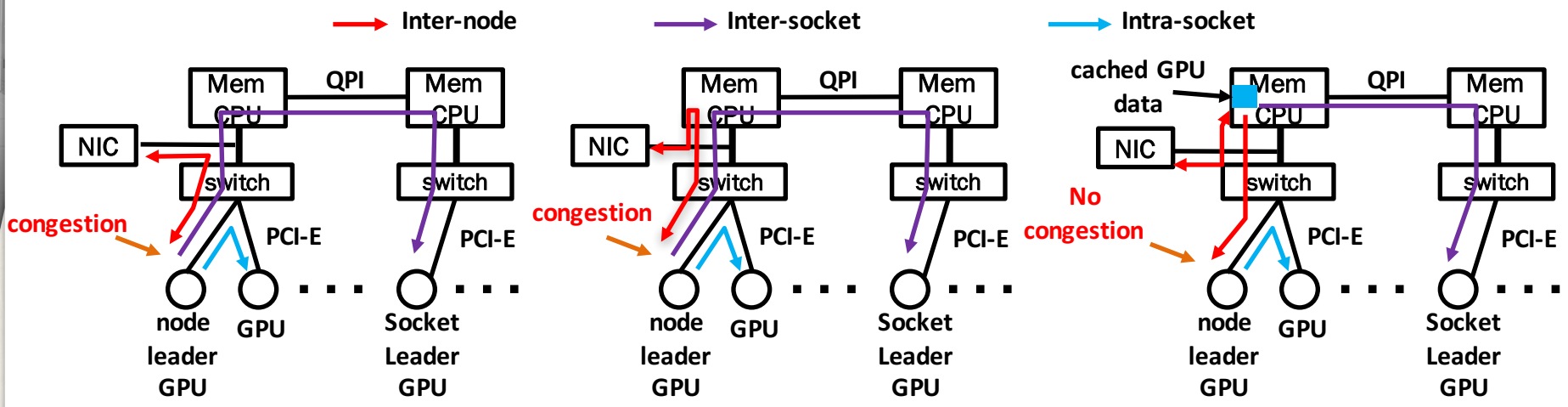
1. Non-blocking P2P
2. ADAPT



1 inter-node communication
2 inter-socket communication
12 intra-socket communication

GPU Optimizations

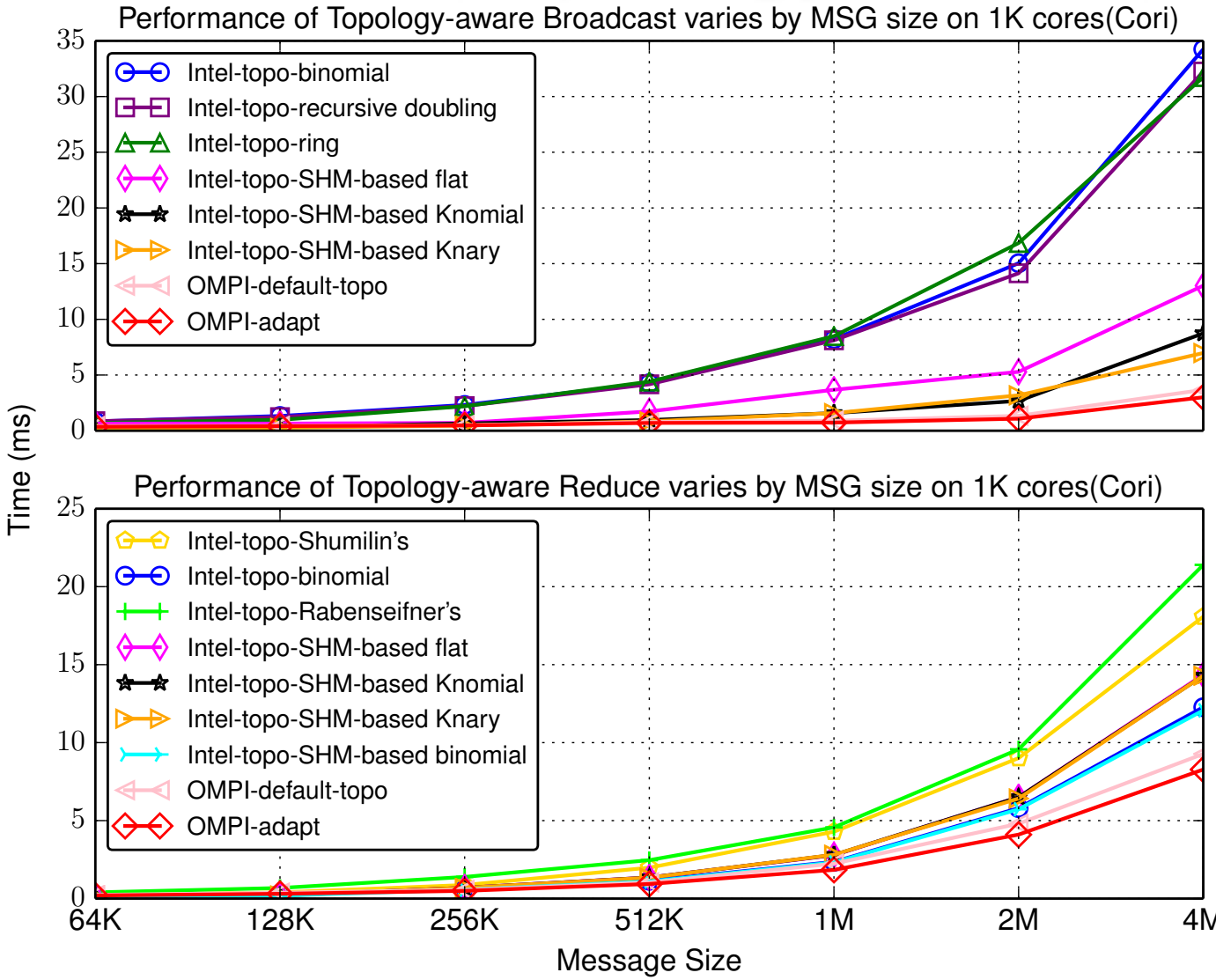
1. Minimize Communications over PCI-Express



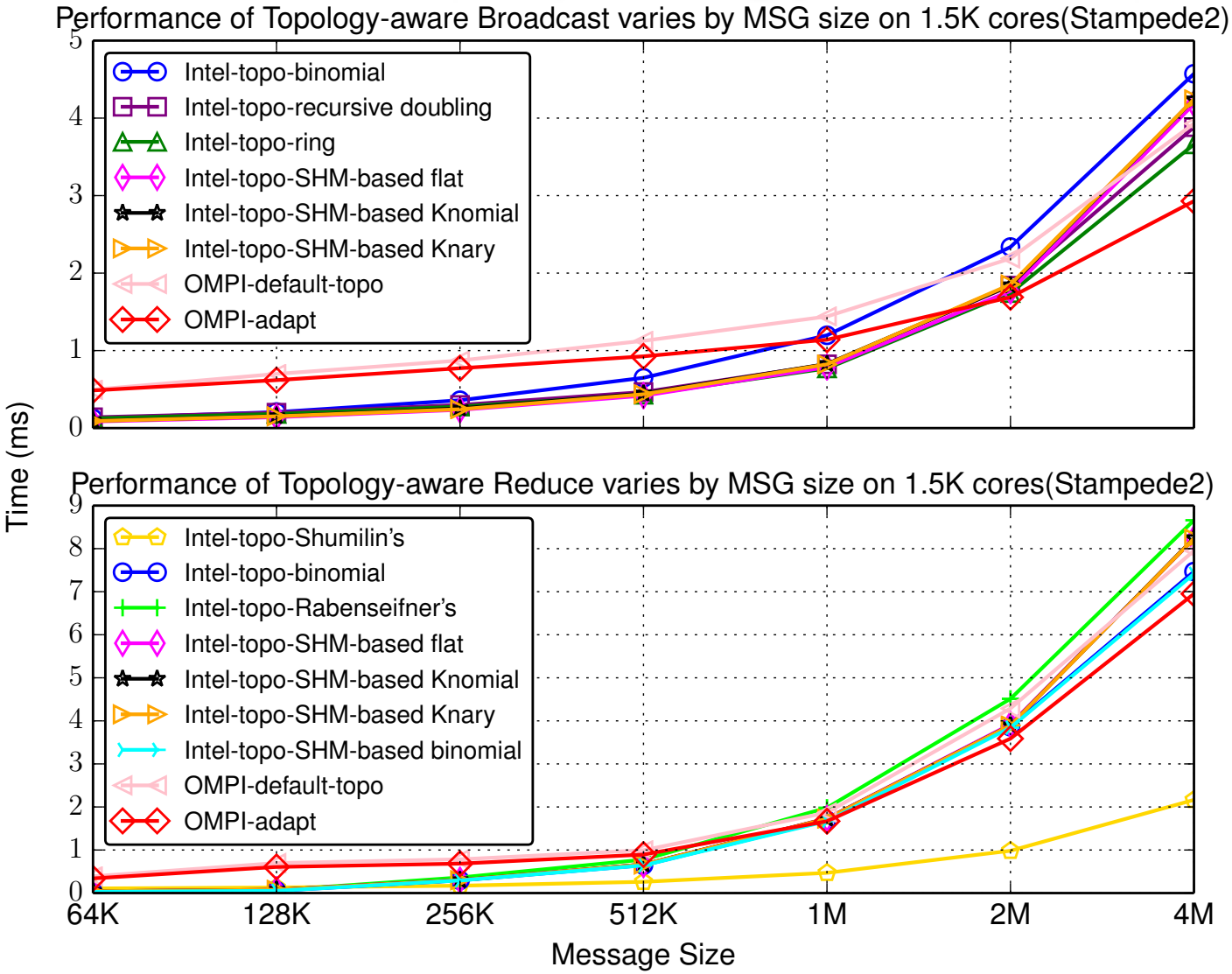
(a) With GPUdirect (b) Without GPUdirect (c)

2. Offload Reduction Operation asynchronously on GPU

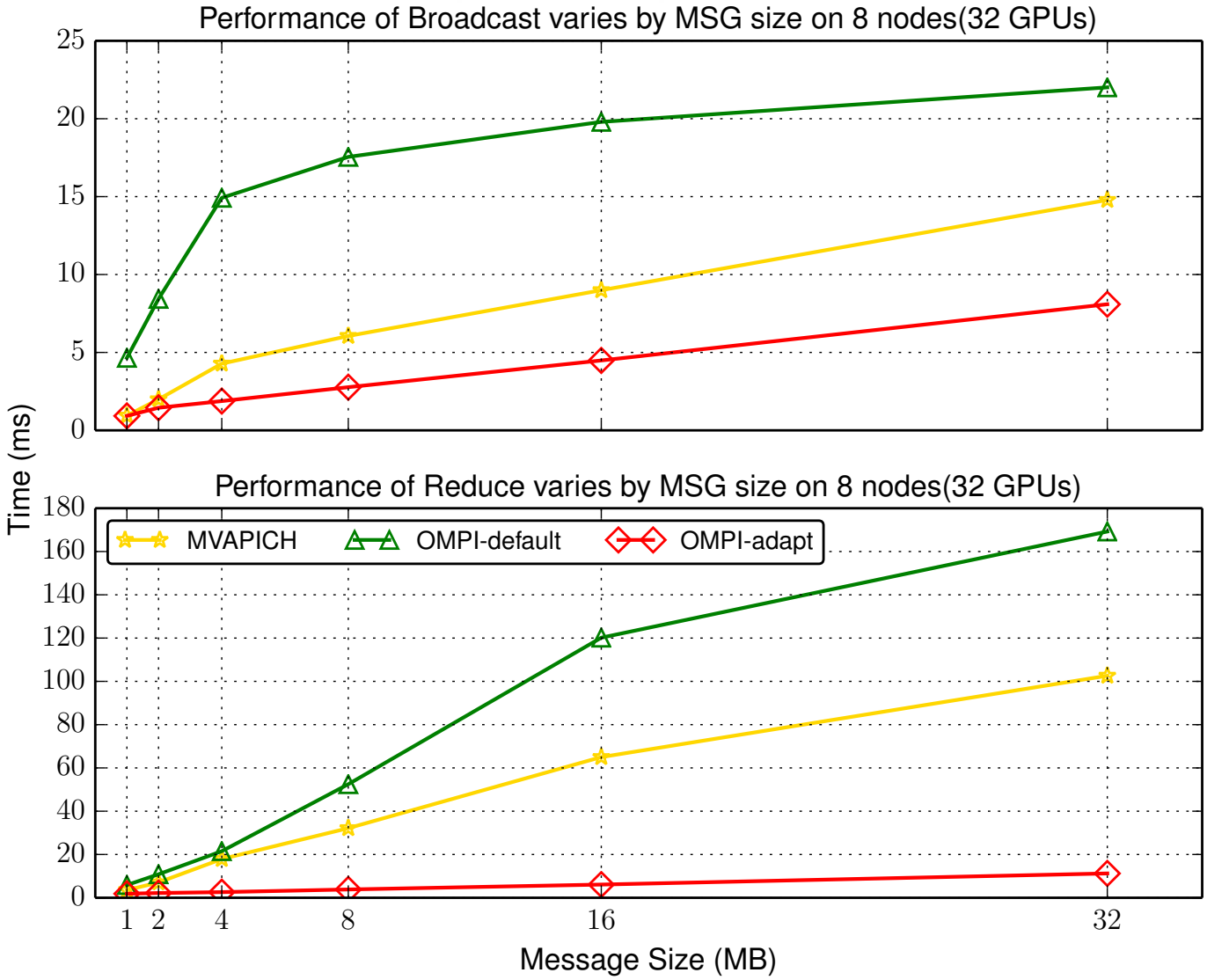
Experiments (topology-aware)



Experiments (topology-aware)



Experiments (GPU)



Experiments (Application)

ASP on Cori (1K cores, problem size 256K, message size 1MB)

	Cray MPI	Intel MPI	OMPI-ADAPT	OMPI-default
Communication (s)	2.98	15.26	1.99	14.18
Total Runtime (s)	6.20	18.46	5.21	17.40
Percentage (%)	48.06%	82.67%	38.20%	81.49%

Conclusion

- Conclusion
  - Alleviate the effects of noise by relaxing the synchronizations with an event-driven design.
  - Maximize the concurrent communications over different hardware levels to speed up collectives with topology-aware communication trees.
  - Enable highly efficient topology-aware NVIDIA GPU collective operations with optimizations.
- Future work
  - Support more type of collective operations.
  - Integrate with non-blocking collective operations.