CSCE 5300 Introduction to Big Data and Data Science

Introduction to High-performance Computing

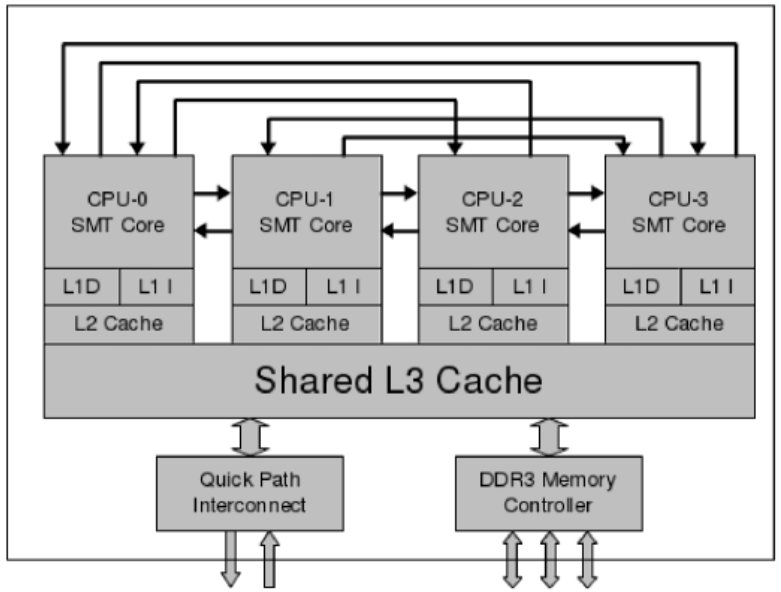
Xi Luo
Middleware Software Engineer, Intel

What's high-performance computing?

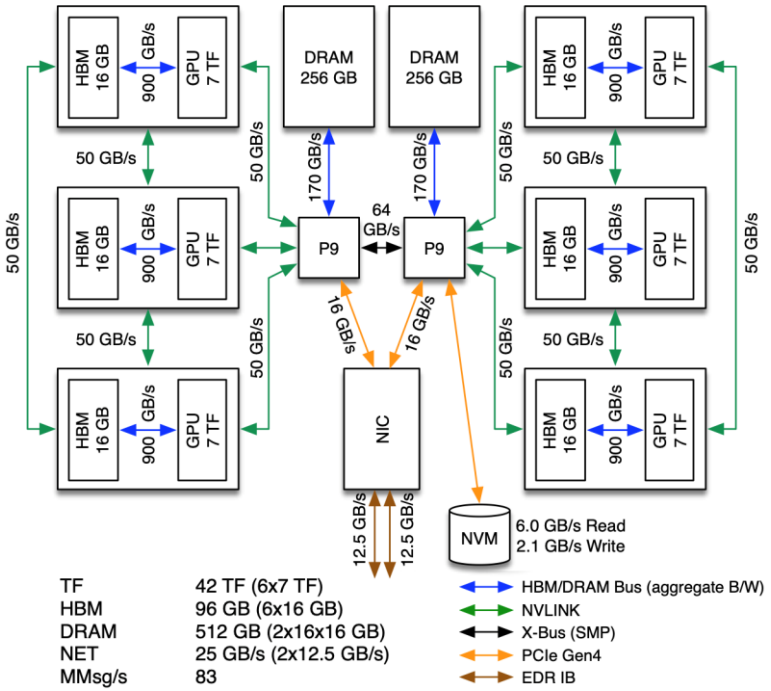
- High-performance computing (HPC) uses supercomputers and computer clusters to solve advanced computation problems
- **General**: high-performance + computing

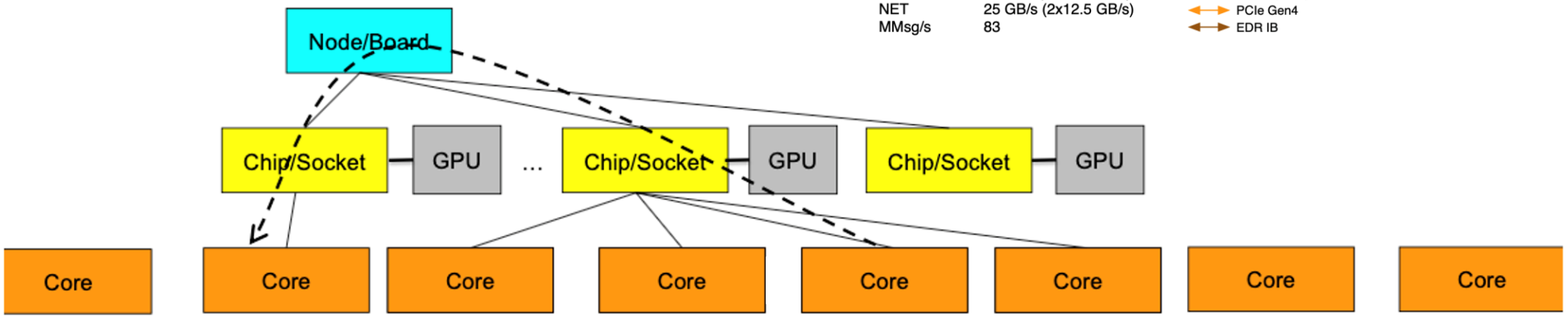
Example of a typical supercomputer

i9

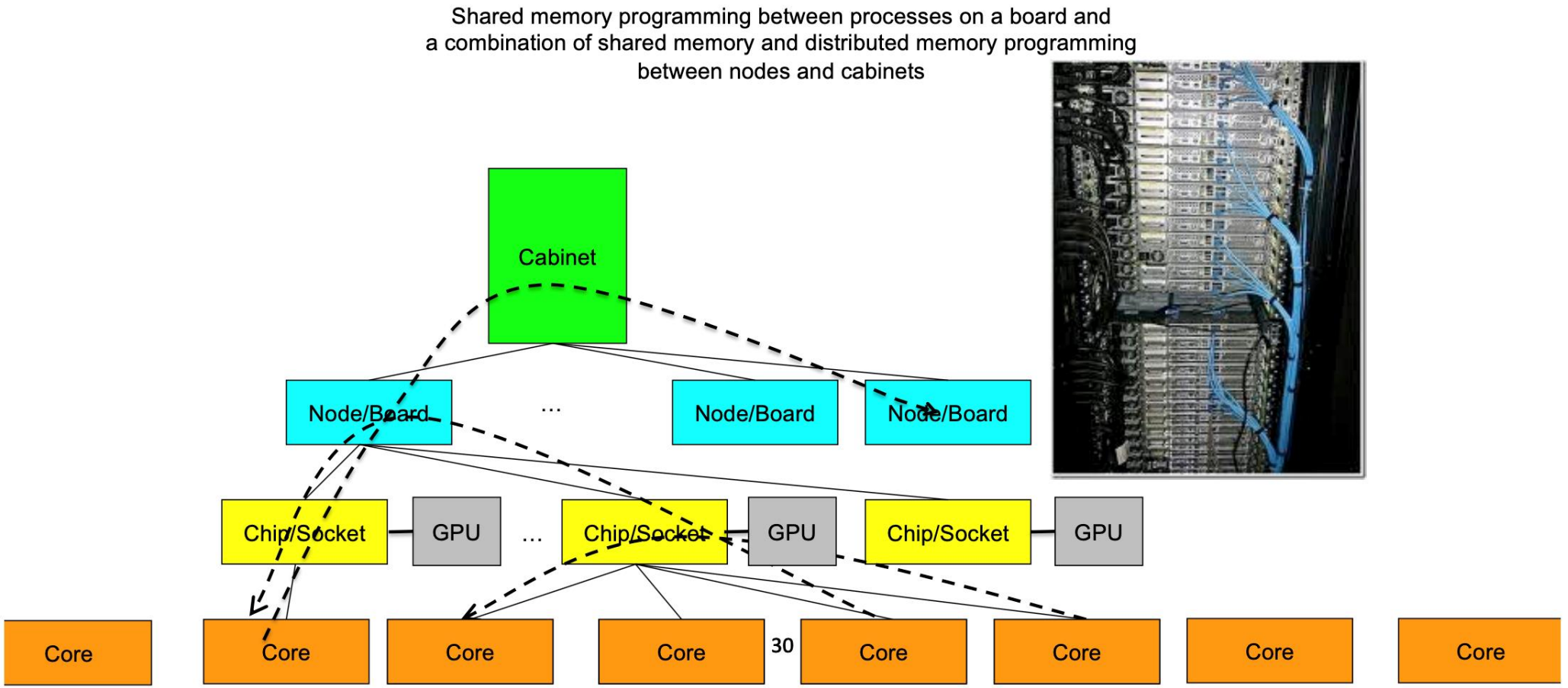


Summit





Example of a typical supercomputer



Example of a typical supercomputer

Importance of HPC

- HPC is essential for solving complex problems and making breakthrough discoveries in many fields of science, engineering, medicine, and industry.

- The ability of computing drives the evolution of science, engineering and technology.
  - ✓ In the 1980s, researchers began experimenting with neural networks that had multiple layers, allowing them to model more complex relationships in data. However, training these deep neural networks was difficult, and progress was slow.
  - ✓ In 2012, deep learning achieved a major breakthrough, as we enter the era of petascale computing
  - ✓ In 2023, a huge leap in large language model, e.g., ChatGPT, Bard, at the era of exascale computing

Today's HPC Environment

- Highly parallel
  - Distributed memory
  - MPI + Open-MPI programming model
- Heterogeneous
  - Commodity processors + GPU accelerators
- Communication between parts very expensive compared to computation
- Floating point hardware at 64, 32, 16 & 8 bit levels

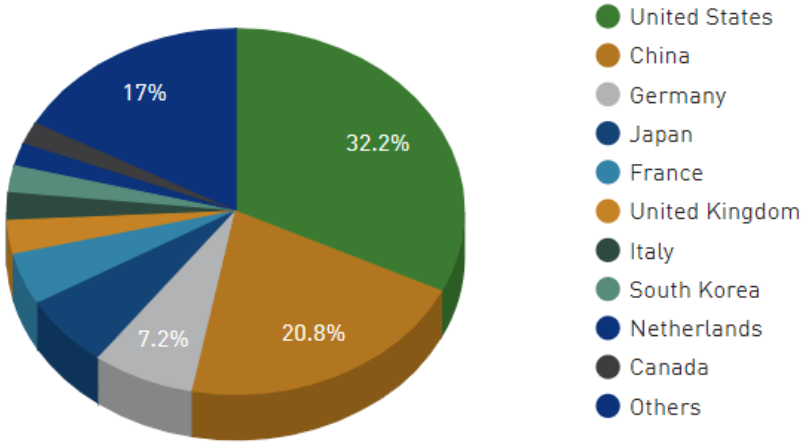
TOP500 list

- Listing of the 500 most powerful Computers in the World
- The first list, published in June 1993, included systems from the United States, Japan, Germany, and the United Kingdom
- Updated twice a year: (1) SC'[1] in the States in November and (2) ISC' in Germany in June
- The idea for the TOP500 list is to track the progress of supercomputing and create a way for researchers to compare their own systems with those of others around the world
- All data available from www.top500.org

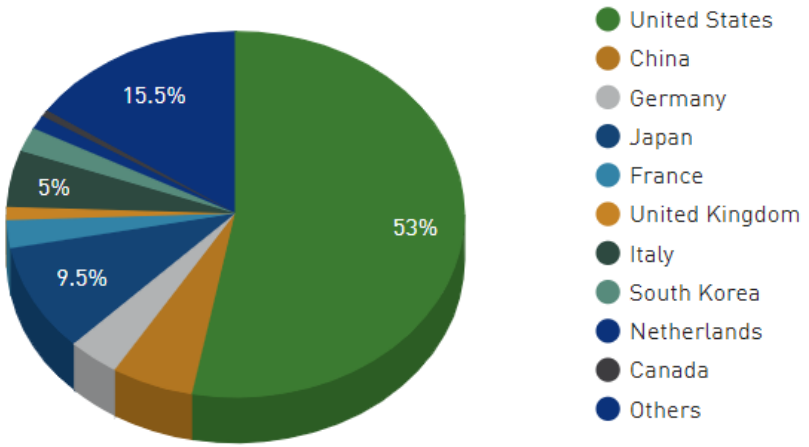
[1] SC: one of the largest supercomputer conferences. 2019, about 13,950 attendees; 2022, up to 15,000 attendees

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703
2	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	4,742,808	585.34	1,059.33	24,687
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Microsoft Azure United States	1,123,200	561.20	846.84	
4	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
5	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107

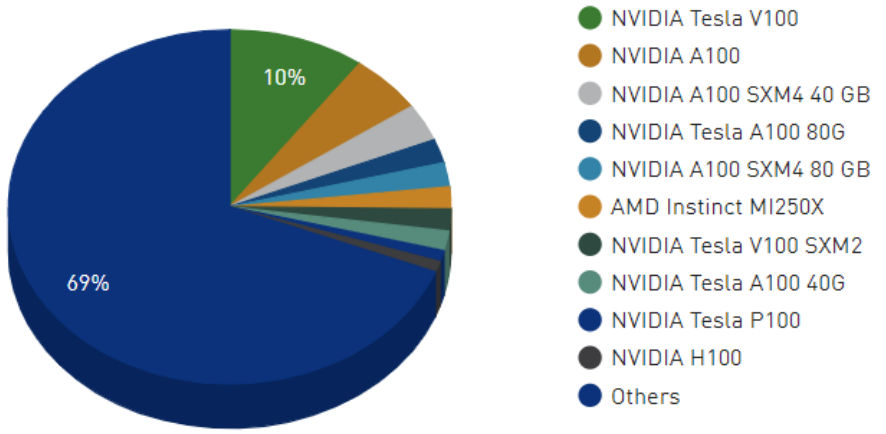
Countries System Share



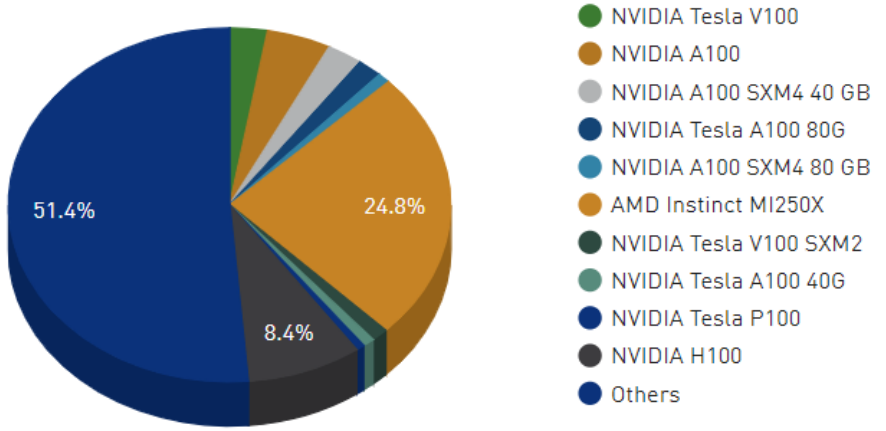
Countries Performance Share



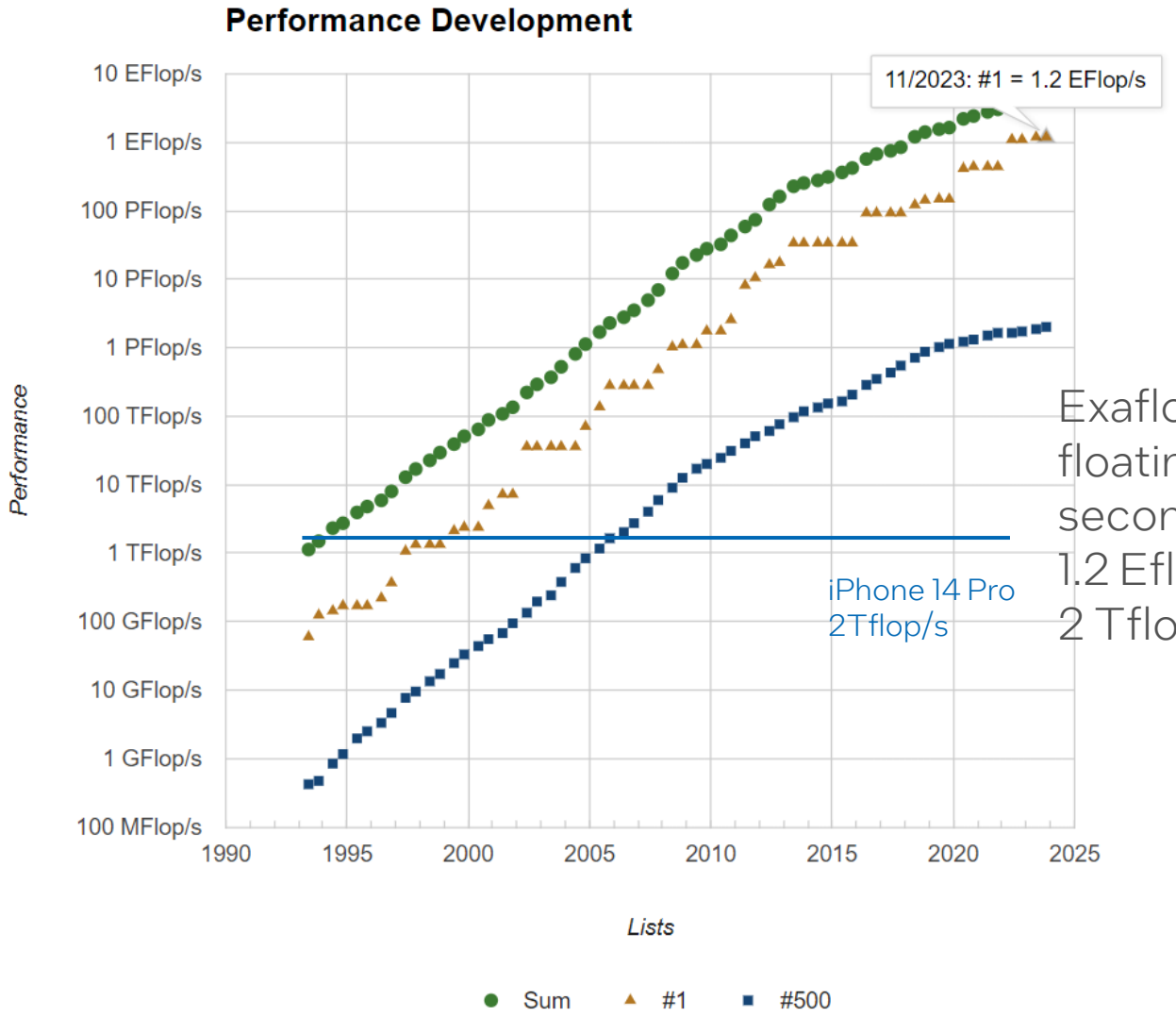
Accelerator/Co-Processor System Share



Accelerator/Co-Processor Performance Share



Performance development of HPC from TOP500



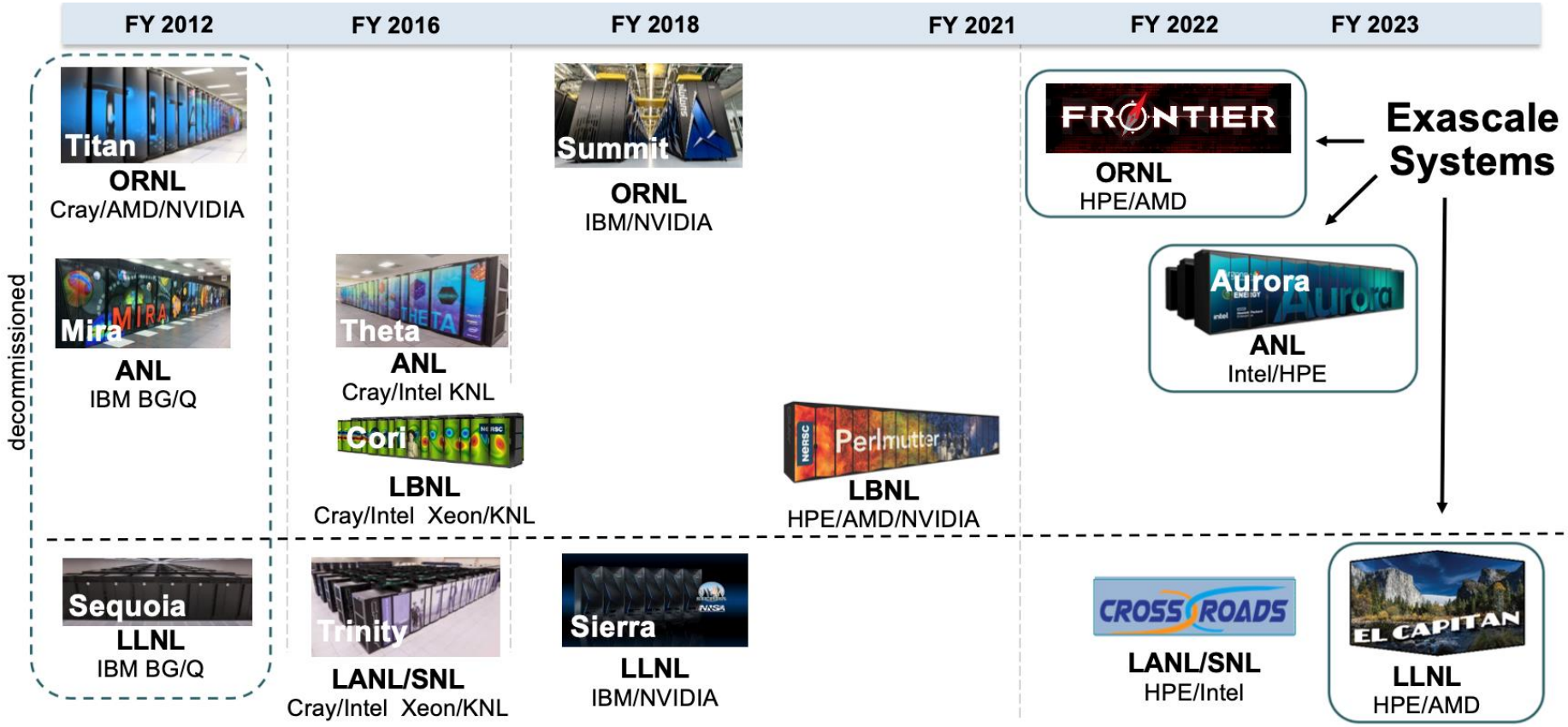
Exaflop is a billion-billion floating point operations per second.

1.2 Eflop/s $= 1.2*10^{18}$ flop/sec

2 Tflop/s $= 10^{12}$ flop/sec

The current status in US

- US Department of Energy Exascale Computing Program (ECP) has formulated a holistic approach that uses co-design and integration to achieve capable exascale.



Challenges of HPC

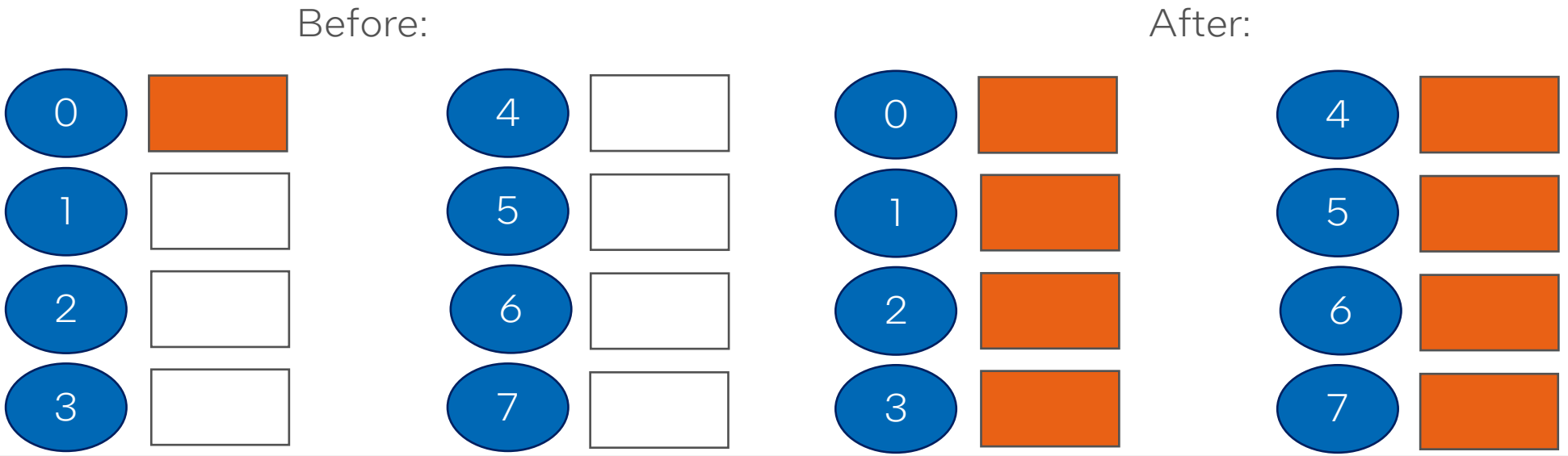
- Hardware complexity: HPC systems are made up of a large number of processors, nodes, and interconnects, and managing this hardware can be a major challenge. Ensuring that all components of the system are working together efficiently, and dealing with hardware failures, can be time-consuming and complex.
- Software complexity: Developing software for HPC systems can also be challenging. Parallel programming models, such as MPI and OpenMP, require specialized skills and can be difficult to learn. In addition, optimizing code for performance can be complex, requiring an understanding of low-level hardware details.
- Scalability: Ensuring that HPC applications can scale to large numbers of processors is critical for achieving high performance. However, achieving scalability can be challenging, requiring careful attention to issues such as load balancing, communication overhead, and data movement.
- Energy efficiency: HPC systems consume large amounts of power, and energy costs can be a significant factor in the operation of these systems. Finding ways to reduce energy consumption while maintaining performance is an ongoing challenge in HPC.
- Data management: HPC systems generate and process vast amounts of data, and managing this data can be a significant challenge. Ensuring that data is stored and accessed efficiently, and dealing with issues such as data locality and data movement, can be complex and time-consuming.
- Convergence of HPC and AI: A growing trend that has emerged as a result of the increasing demand for computational power in the field of AI and the development of new ML algorithms that can leverage HPC resources effectively. This convergence has significant implications for both research and industry, as it enables the development of more advanced AI models and the efficient processing of large-scale data.
- And more

Computation vs Communication

- Computation: the actual processing or calculations performed by the HPC system to solve a given problem.
- Optimizing computation involves strategies such as algorithm design, code optimization, and utilizing hardware accelerators like GPUs or specialized processors.
- Communication: the exchange of data and messages between processing units within a parallel system.
- Optimizing communication involves minimizing latency, reducing message contention, and overlapping communication with computation to hide latency.
- With the increasing scale of the HPC systems, more and more applications have bottlenecks on the communication part.

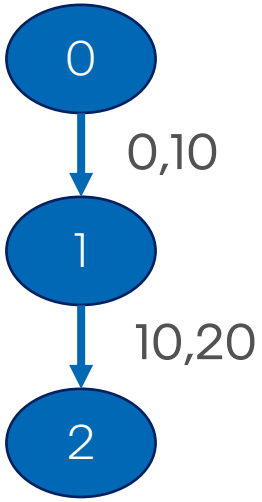
An example of communication optimization

- Message Passing Interface (MPI): a standardized message-passing system designed to provide portability and efficiency on a wide variety of parallel computers.
- MPI_Bcast: propagate messages from root to all the other processes.

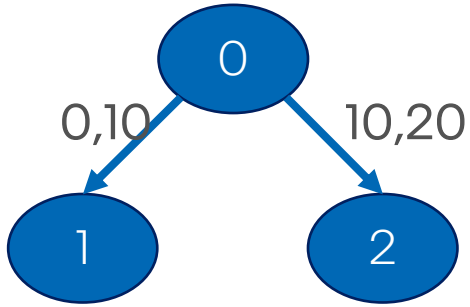


Assumption 1 (T)

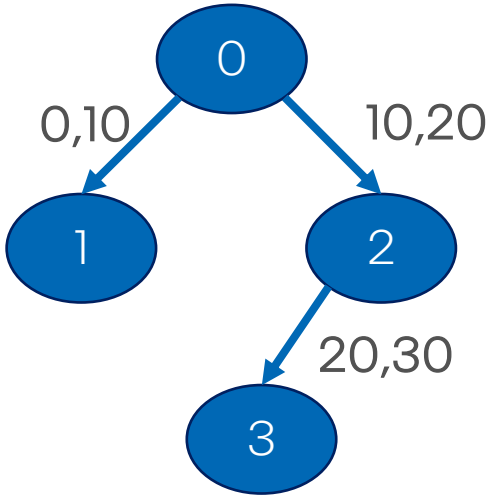
- Time of MPI_Bcast = the time when the last leaf node get the data
- Suppose it takes time T = 10 to transfer the data from one process to another, what is the fastest way to perform an MPI_Bcast on 8 processes in this case?
- For example: 0,10 means starts at timestamp 0 and ends at timestamp 10.



Total Time: 20



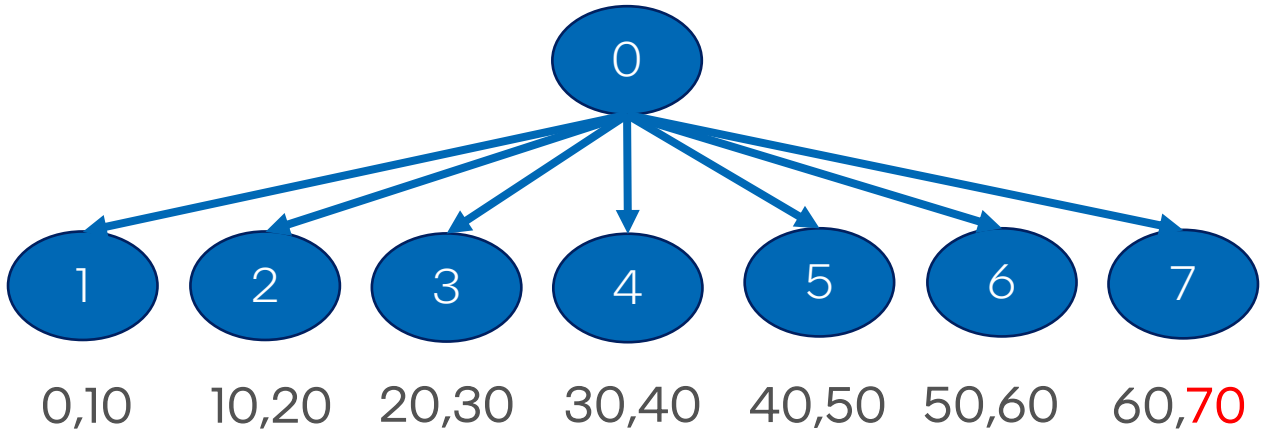
Total Time: 20



Total Time: 30

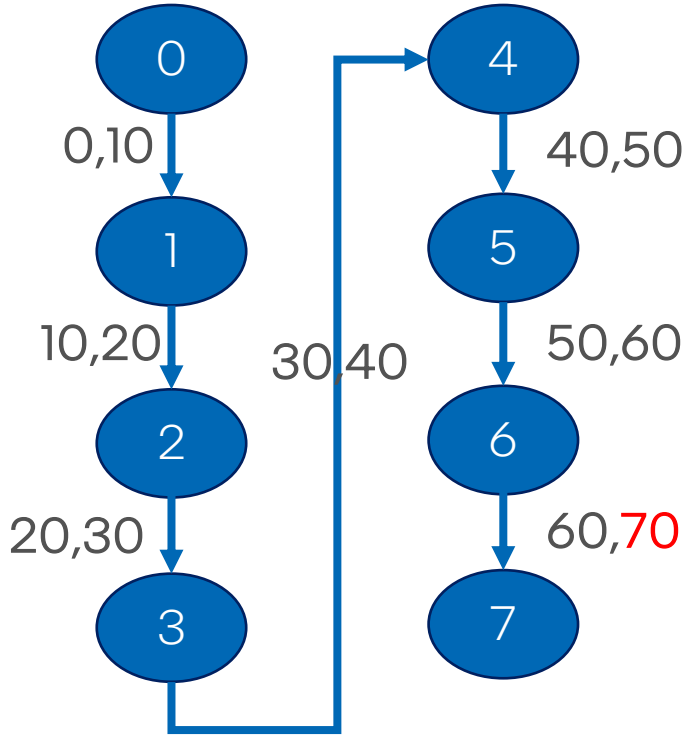
Assumption 1 (T)

Linear:



Total Time: 70

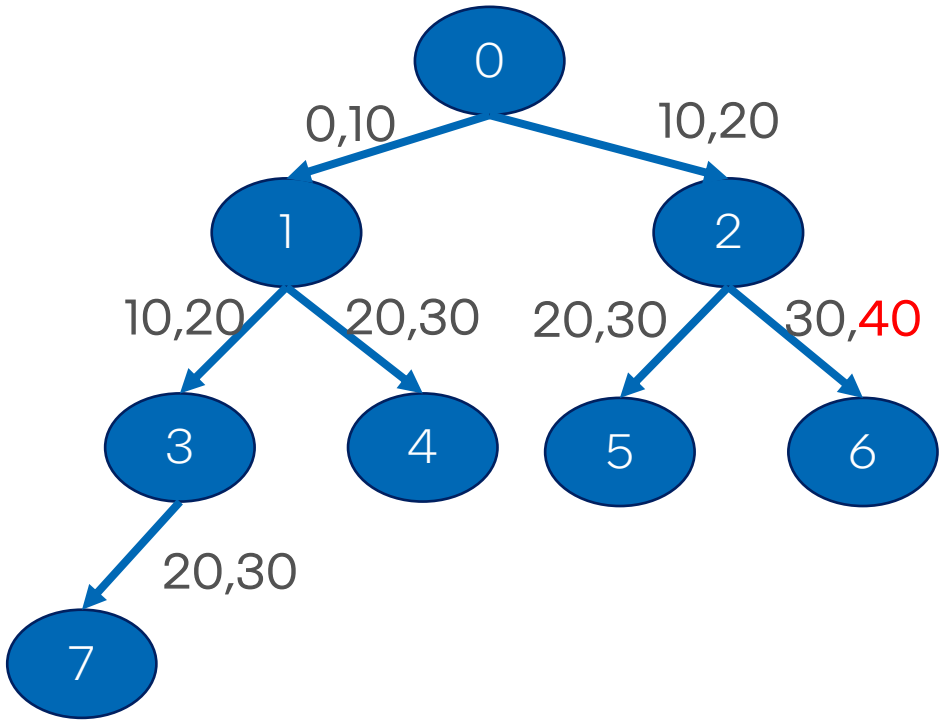
Chain:



Total Time: 70

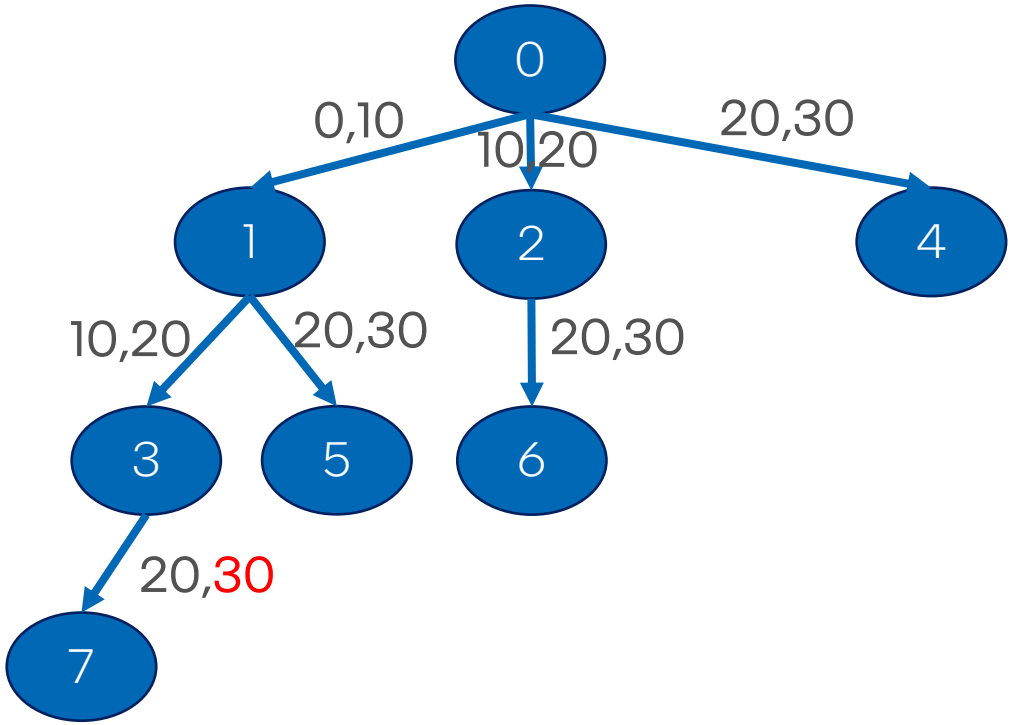
Assumption 1 (T)

Binary tree:



Total Time: 40

Binomial tree:



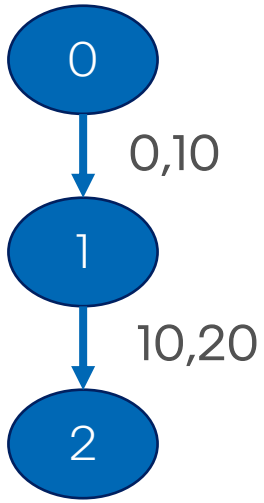
Total Time: 30

Assumption 2 (T = O + L)

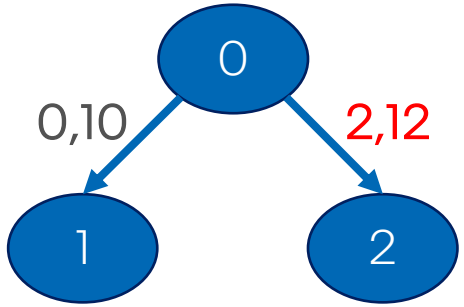
- Let's make the time T more realistic.
- T = O + L:
  - O: overhead, the time takes for preparing the data for communication (packing the data, move the data to the memory accessible from NIC…)
  - L: latency, the time takes for the data traveling the links between two processes (message queueing, possible resending, communication protocol overhead…).
  - O normally is much less than L on modern HPC systems.

Assumption 2 (T = O + L)

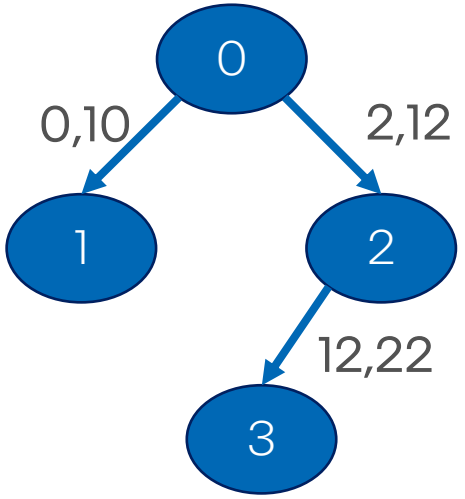
- Suppose O = 2, L = 8 and the network can handle multiple messages at the same time, what is the fastest way to perform an MPI_Bcast on 8 processes in this case?



Total Time: 20



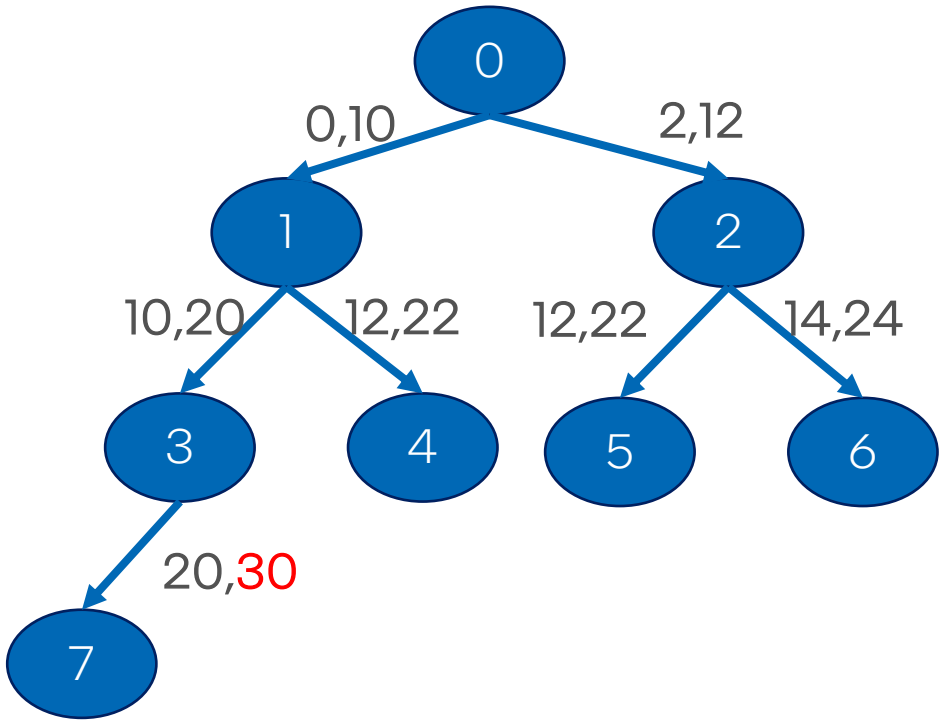
Total Time: 12



Total Time: 22

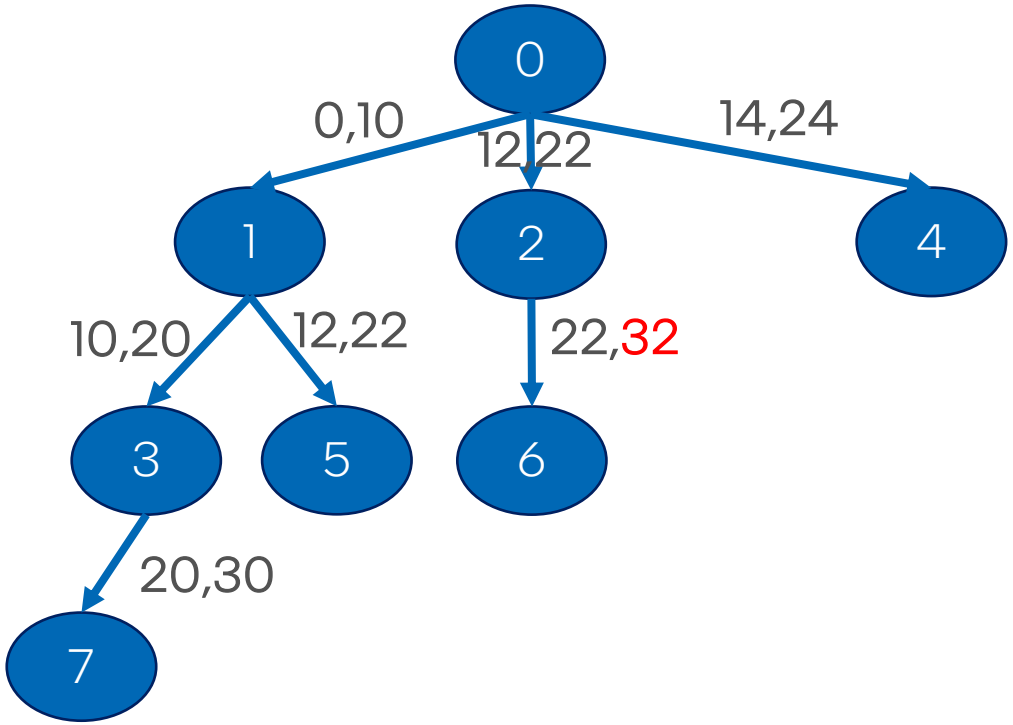
Assumption 2 (T = O + L)

Binary tree:



Total Time: 30

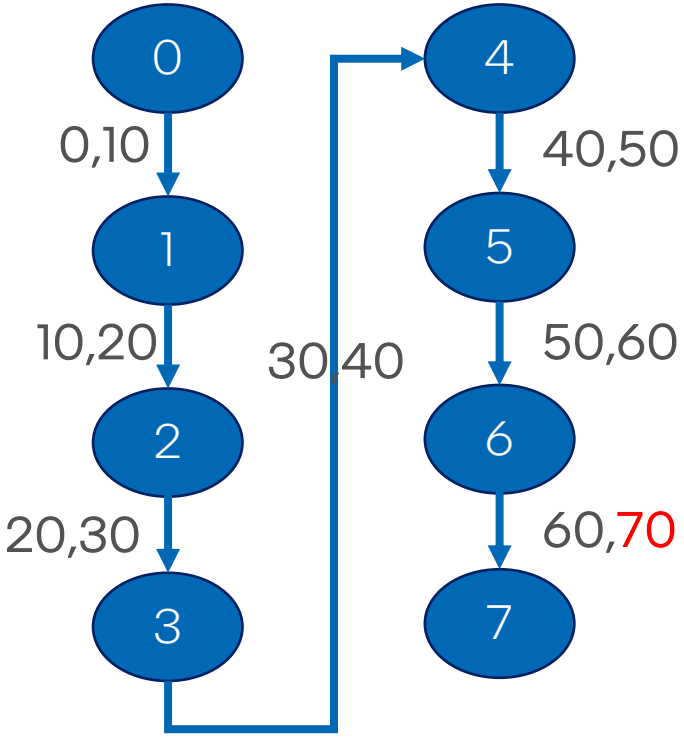
Binomial tree:



Total Time: 32

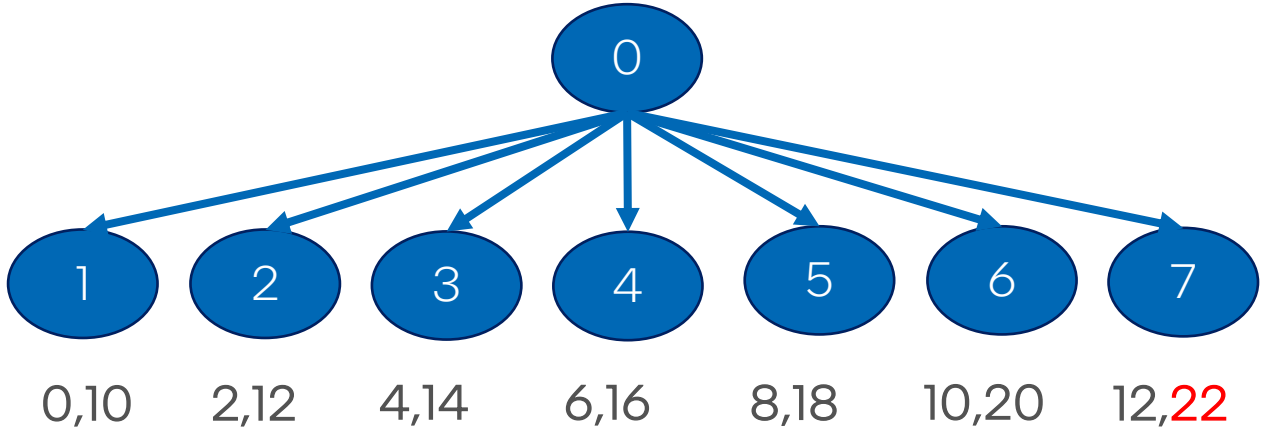
Assumption 2 (T = O + L)

Chain:



Total Time: 70

Linear:



Total Time: 22

Assumption 3 (T = O + L + M/B)

- Previous assumption T = O + L is reasonable for small messages, but what if messages are large?
- Large messages should take longer to transfer than small messages.
- T = O + L + M/B
  - M: message size.
  - B: bandwidth, represents the capacity or throughput of the channel and is typically measured in bytes per second (bps, Kbps, Mbps, Gbps).
  - O and L can be ignored for very large messages.

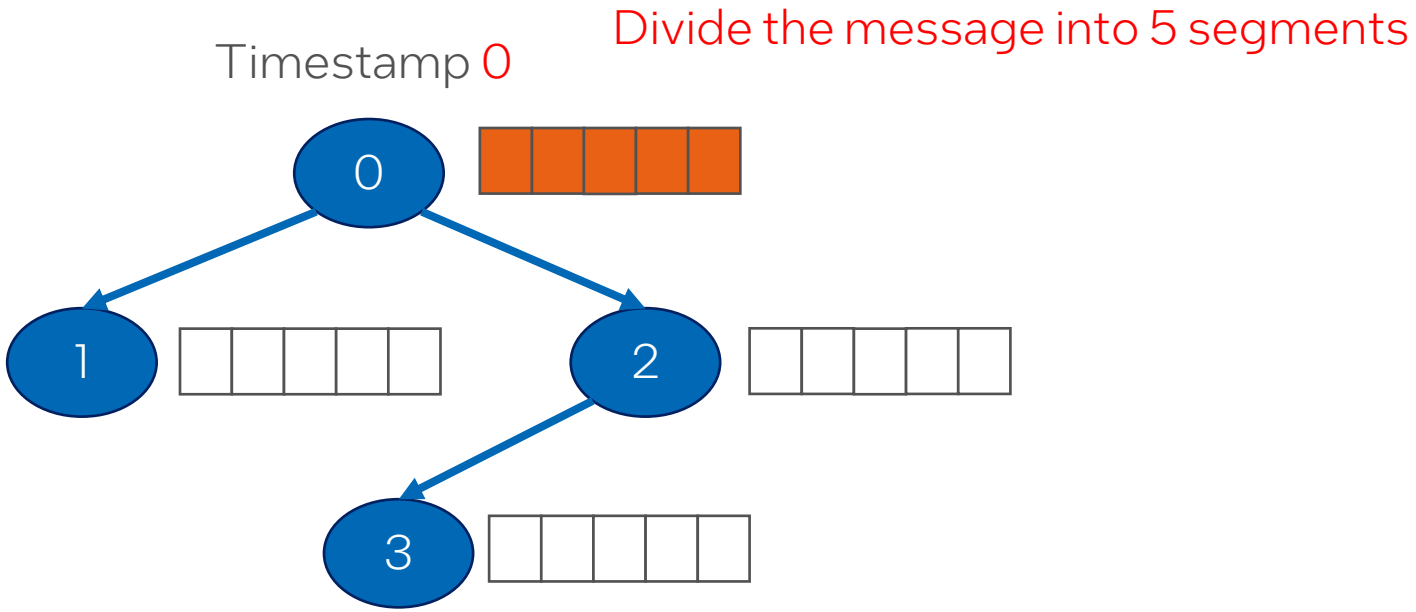
Assumption 3 (T = O + L + M/B)

- Suppose we are transferring very large messages:
  - M = 10, B = 1
  - O and L can be ignored
  - The communication are bound by the bandwidth

  what is the fastest way to perform an MPI_Bcast on 8 processes in this case?
- Hint: divide messages into segments and transferring segments one by one to overlap the communication.

Assumption 3 (T = O + L + M/B)

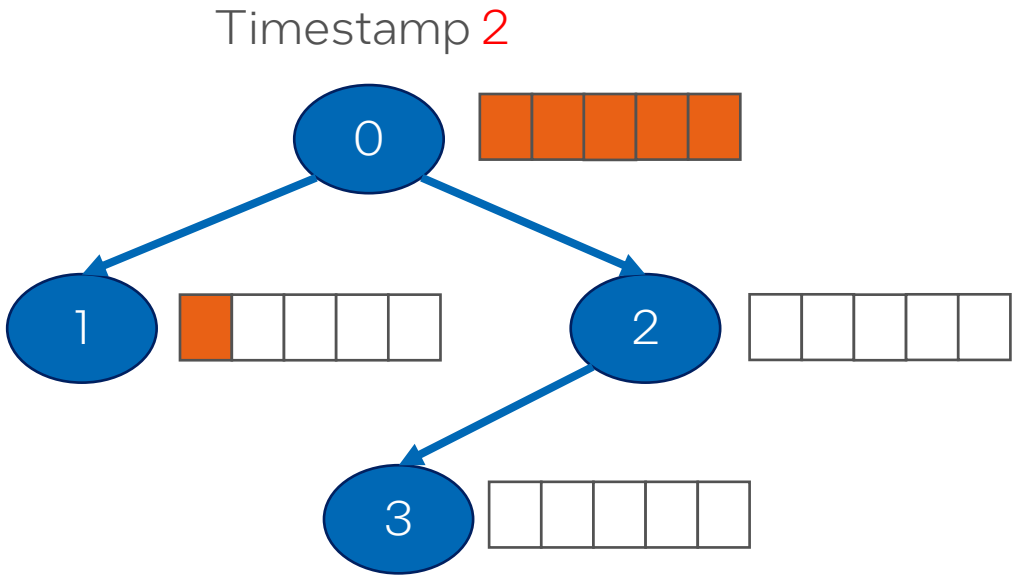
- Suppose:
  - M = 10, B = 1
  - O and L can be ignored
  - The communication are bound by the bandwidth



Assumption 3 (T = O + L + M/B)

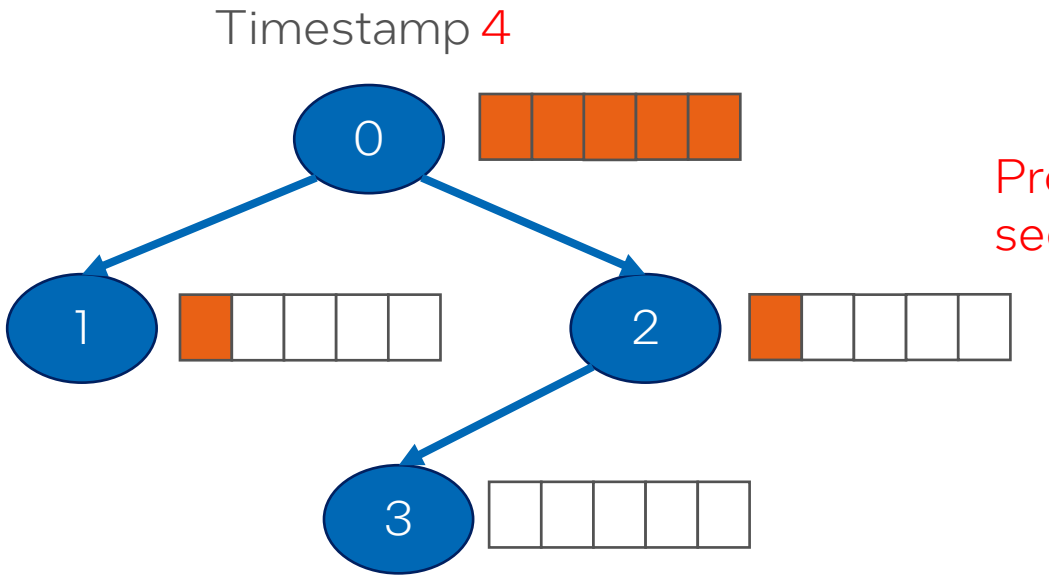
- Suppose:
  - M = 10, B = 1
  - O and L can be ignored
  - The communication are bound by the bandwidth

Process 1 gets the first segment on timestamp 2



Assumption 3 (T = O + L + M/B)

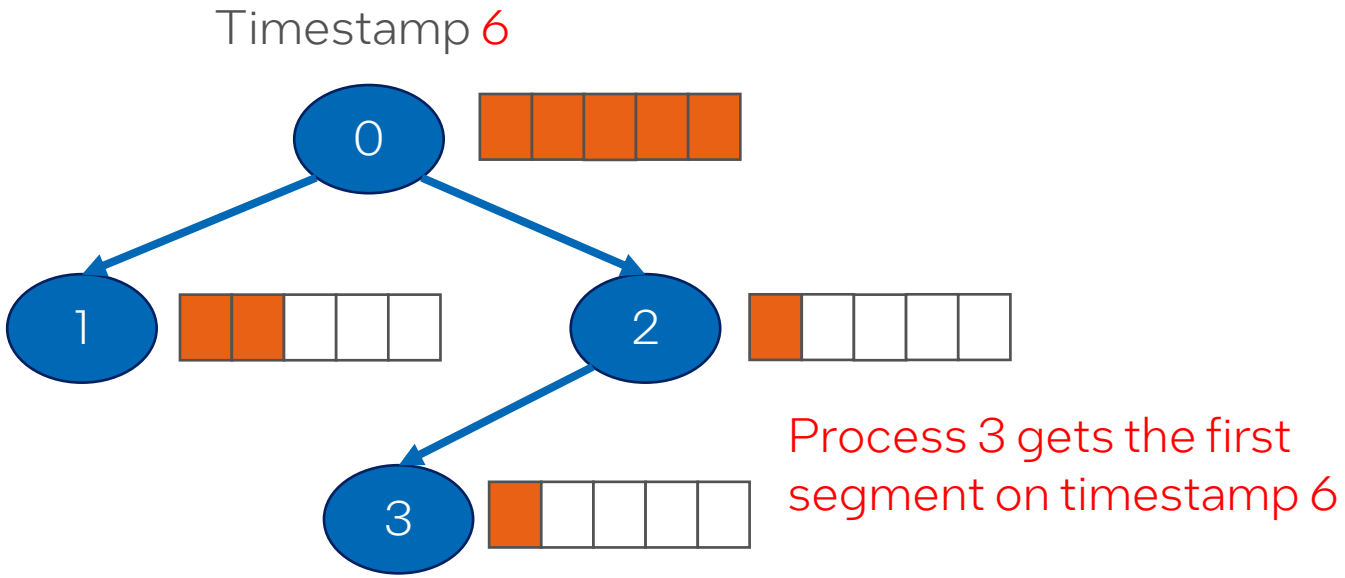
- Suppose:
  - M = 10, B = 1
  - O and L can be ignored
  - The communication are bound by the bandwidth



Process 2 gets the first segment on timestamp 4

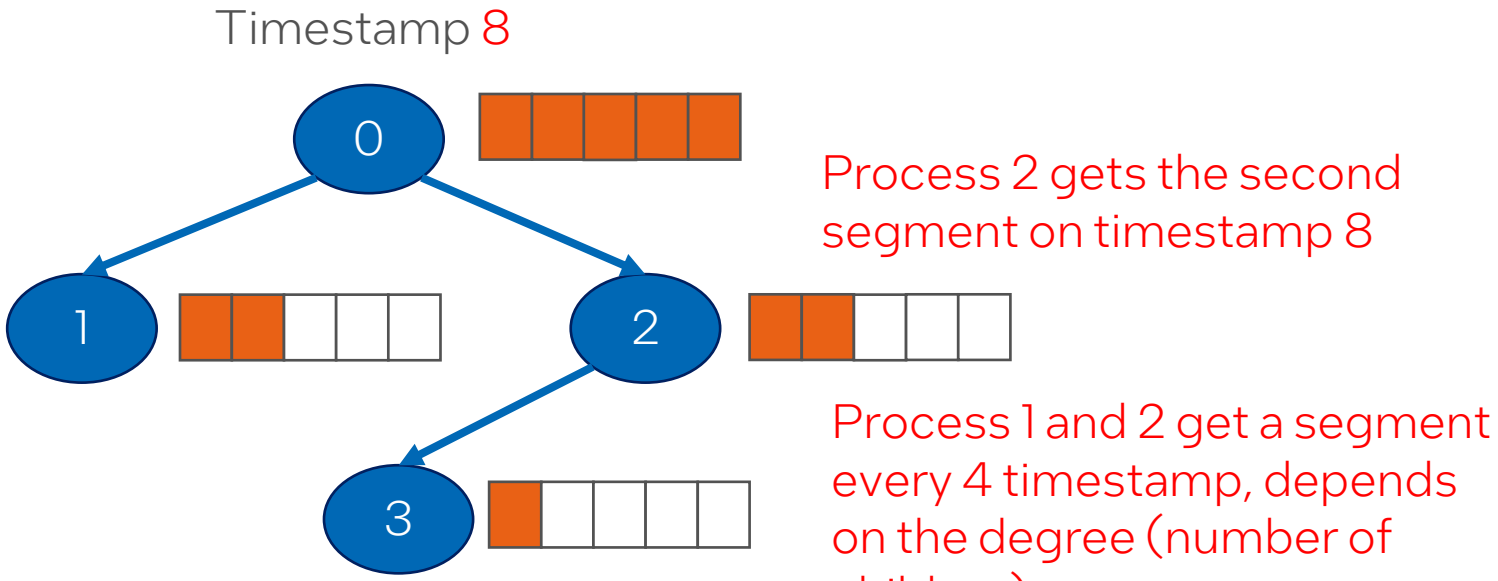
Assumption 3 (T = O + L + M/B)

- Suppose:
  - M = 10, B = 1
  - O and L can be ignored
  - The communication are bound by the bandwidth



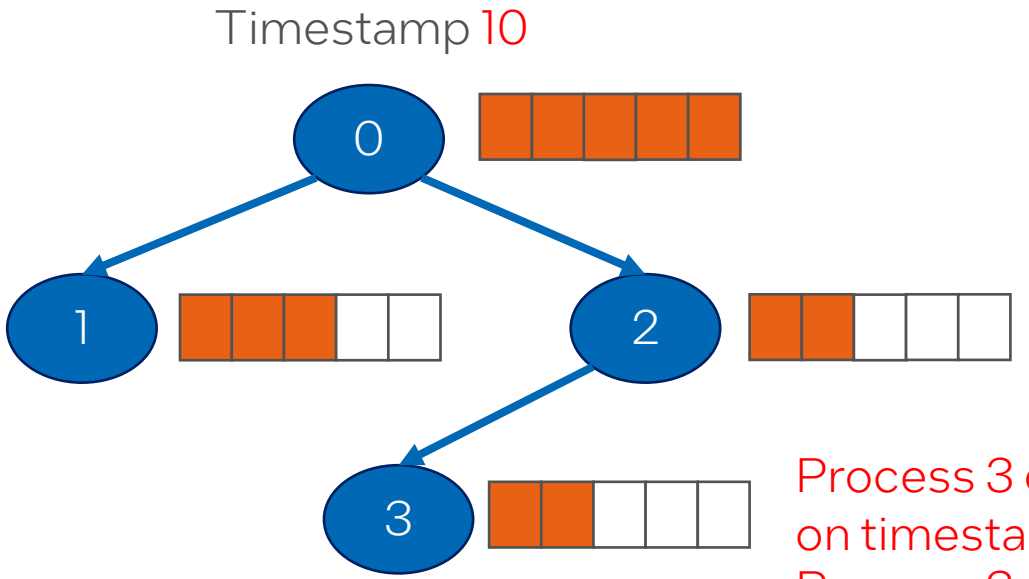
Assumption 3 (T = O + L + M/B)

- Suppose:
  - M = 10, B = 1
  - O and L can be ignored
  - The communication are bound by the bandwidth



Assumption 3 (T = O + L + M/B)

- Suppose:
  - M = 10, B = 1
  - O and L can be ignored
  - The communication are bound by the bandwidth

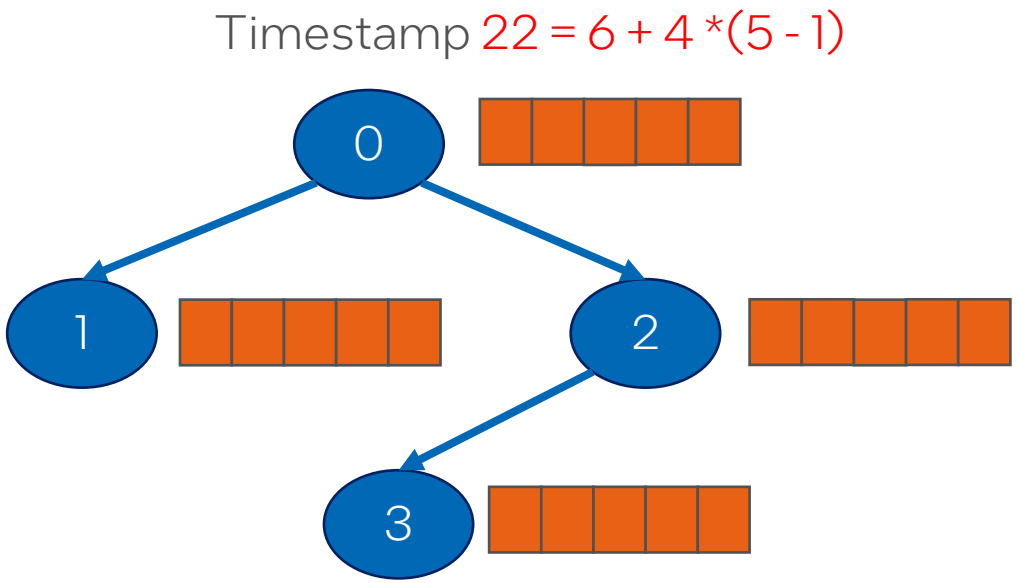


Process 1 gets the third segment on timestamp 10

Process 3 gets the second segment on timestamp 10
Process 3 gets a segment every 4 timestamp

Assumption 3 (T = O + L + M/B)

- Suppose:
  - M = 10, B = 1
  - O and L can be ignored
  - The communication are bound by the bandwidth

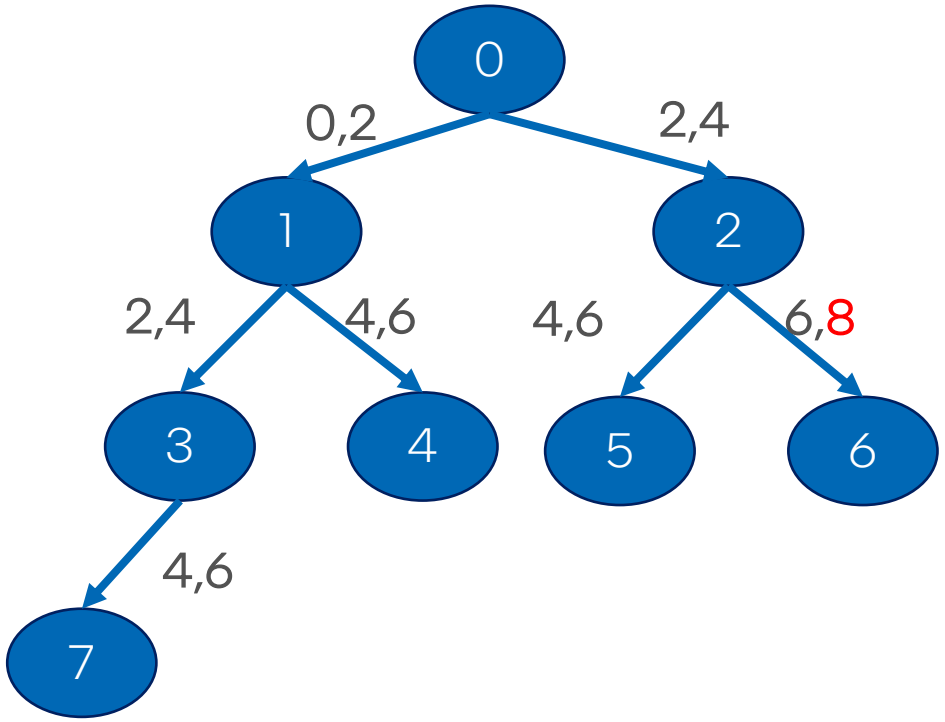


Assumption 3 (T = O + L + M/B)

- Suppose:
  - M = 10, B = 1
  - O and L can be ignored
  - The communication are bound by the bandwidth
  - The messages are divided into 5 segments
- What is the fastest way to perform an MPI_Bcast on 8 processes in this case?

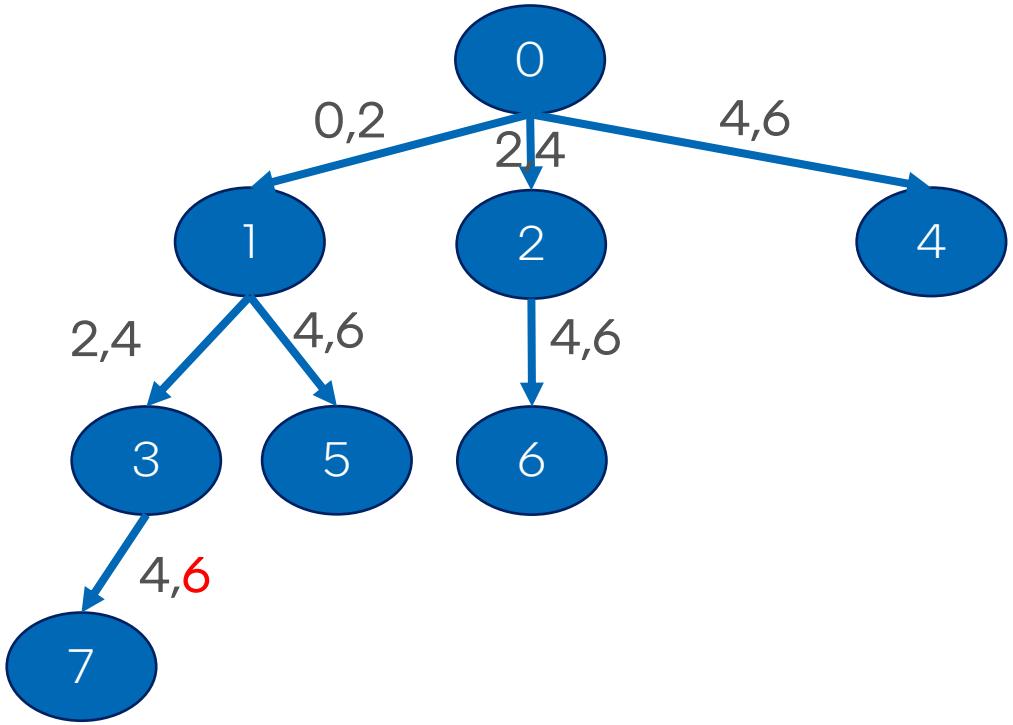
Assumption 3 (T = O + L + M/B)

Binary tree:



Rank 6 receives the first segment on timestamp 8
Rank 6 receives a segment every 4 timestamps
Total Time: 8 + 4 * (5 - 1) = 24

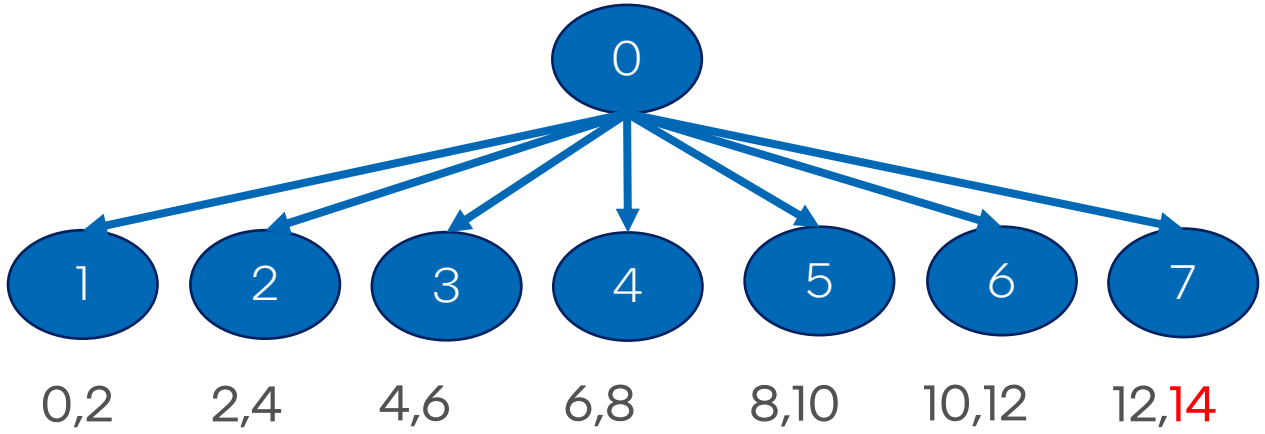
Binomial tree:



Rank 7 receives the first segment on timestamp 6
Rank 7 receives a segment every 4 timestamps
Total Time: 6 + 4 * (5 - 1) = 22

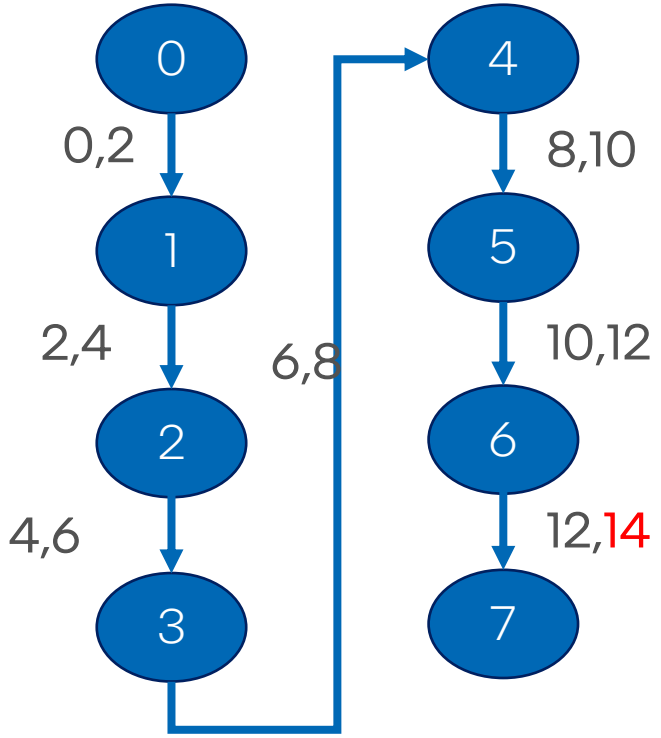
Assumption 3 (T = O + L + M/B)

Linear:



Rank 7 receives the first segment on timestamp 14
Rank 7 receives a segment every 7 timestamps
Total Time: 14 + 7 * (5 - 1) = 42

Chain:



Rank 7 receives the first segment on timestamp 14
Rank 7 receives a segment every 1 timestamps
Total Time: 14 + 1 * (5 - 1) = 18

Further optimizations

- Autotuning: select the best algorithm, segment size, … for a given collective.
- Process placement: where the processes are placed in the HPC systems matters. Minimize the slow communication.
- Hardware topology: design the algorithm based on the hardware topology to maximize concurrent communication on different hardware.
- Noise alleviation: small local noise can be propagated through the communications, especially on very large scale runs.
- ……


intel