Performance tuning of Graph500 benchmark on Supercomputer Fugaku

Masahiro Nakao (RIKEN R-CCS)

Outline

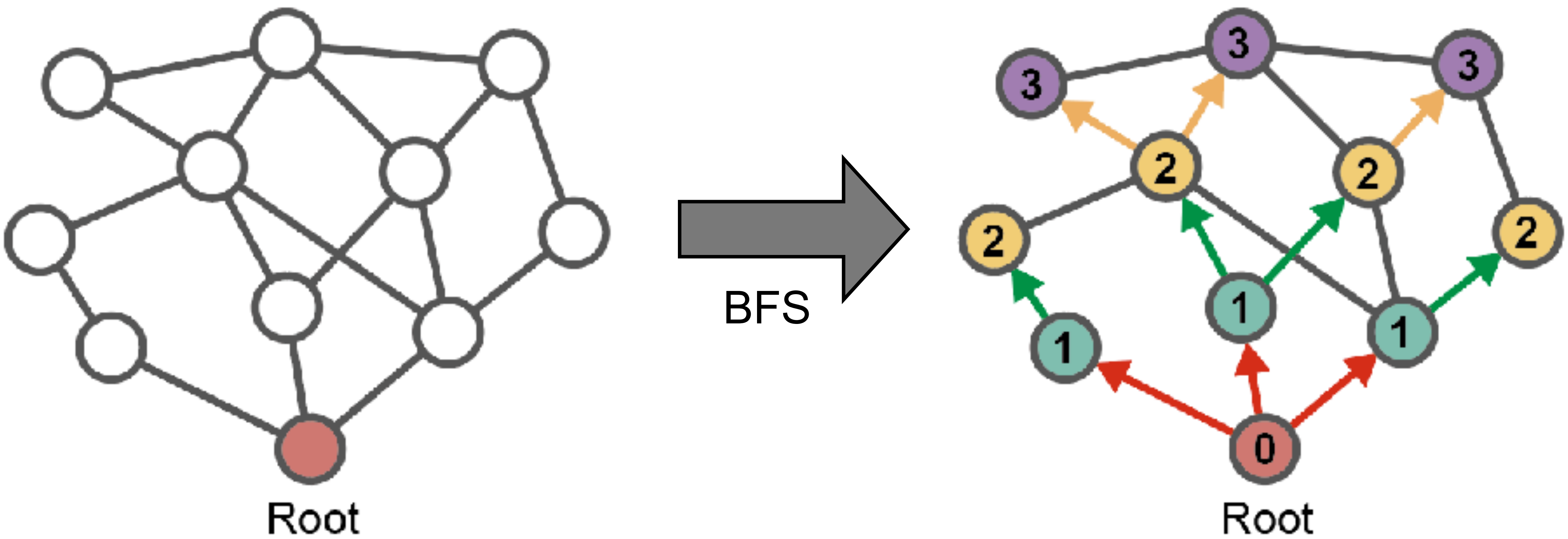
- Graph500 Benchmark
- Supercomputer Fugaku
- Tuning Graph500 Benchmark on Supercomputer Fugaku

Graph500

https://graph500.org

- Graph500 has started since 2010 as a competition for evaluating **performance of large-scale graph processing**
- The ranking is updated twice a year (June and November)
  - Fugaku won the awards twice in 2020
- One of kernels in Graph500 is BFS (Breadth-First Search)
- An artificial graph called the Kronecker graph is used
  - Some vertices are connected to many other vertices while numerous others are connected to only a few vertices
  - Social network is known to have a similar property

Overview of BFS



Input : graph and root

Output : BFS tree

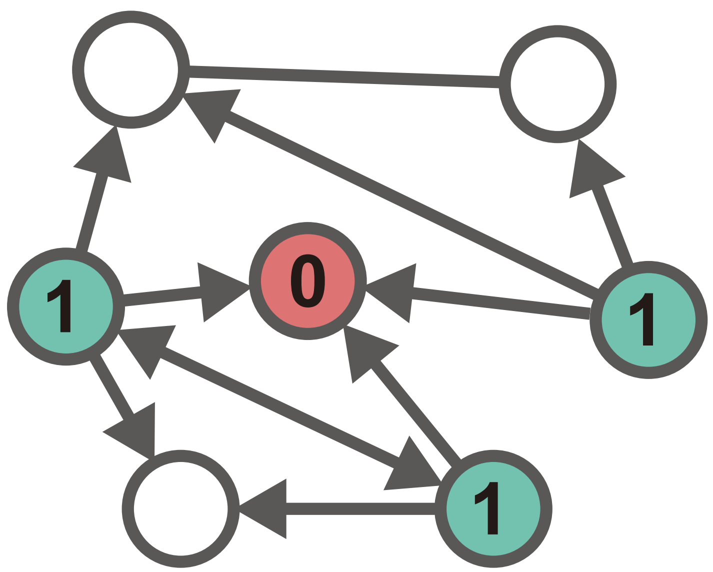
- Data structure and BFS algorithm are free

Hybrid-BFS

[Beamer, 2012] Scott Beamer et al. Direction-optimizing breadth-first search, SC ’12

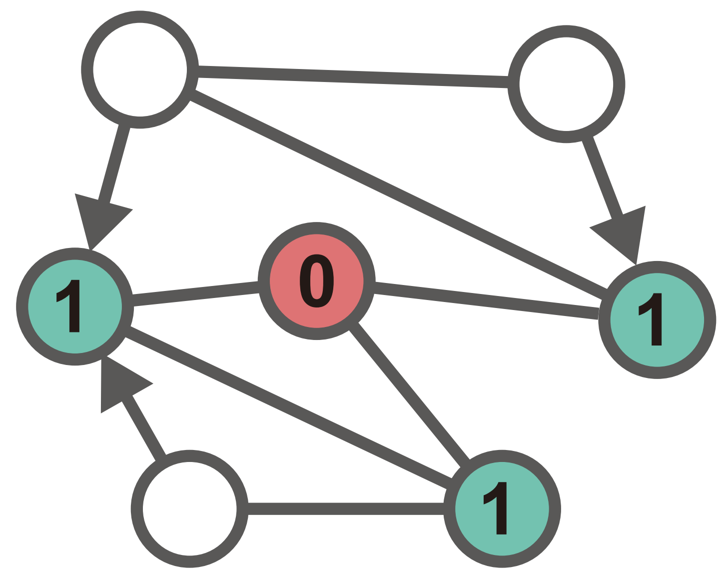
- It is suitable for small diameter graphs used in Graph500
- Perform BFS while switching between Top-down and Bottom-up
  - In the middle of BFS, the number of vertices being visited increases explosively, so it is inefficient in only Top-down

Top-down



Search for unvisited vertices from **visited vertices**

Bottom-up



Search for **visited vertices** from unvisited vertices

2D Hybrid-BFS

[Beamer, 2013] Scott Beamer, et. al. Distributed Memory Breadth-First Search Revisited: Enabling Bottom-Up Search. IPDPSW '13.

- Distribute the adjacency matrix to a 2D process grid (R x C)

$$A = \left(\begin{array}{c|c|c} A_{1,1} & \cdots & A_{1,C} \\ \hline \vdots & \ddots & \vdots \\ \hline A_{R,1} & \cdots & A_{R,C} \end{array}\right)$$

- Communication only within the column process and within the row process
  - Allgatherv, Alltoallv, isend/irecv/wait
- The closer the R and C values are, the smaller the total communication size

Outline

- Graph500 Benchmark
- Supercomputer Fugaku
- Tuning Graph500 Benchmark on Supercomputer Fugaku

Supercomputer Fugaku

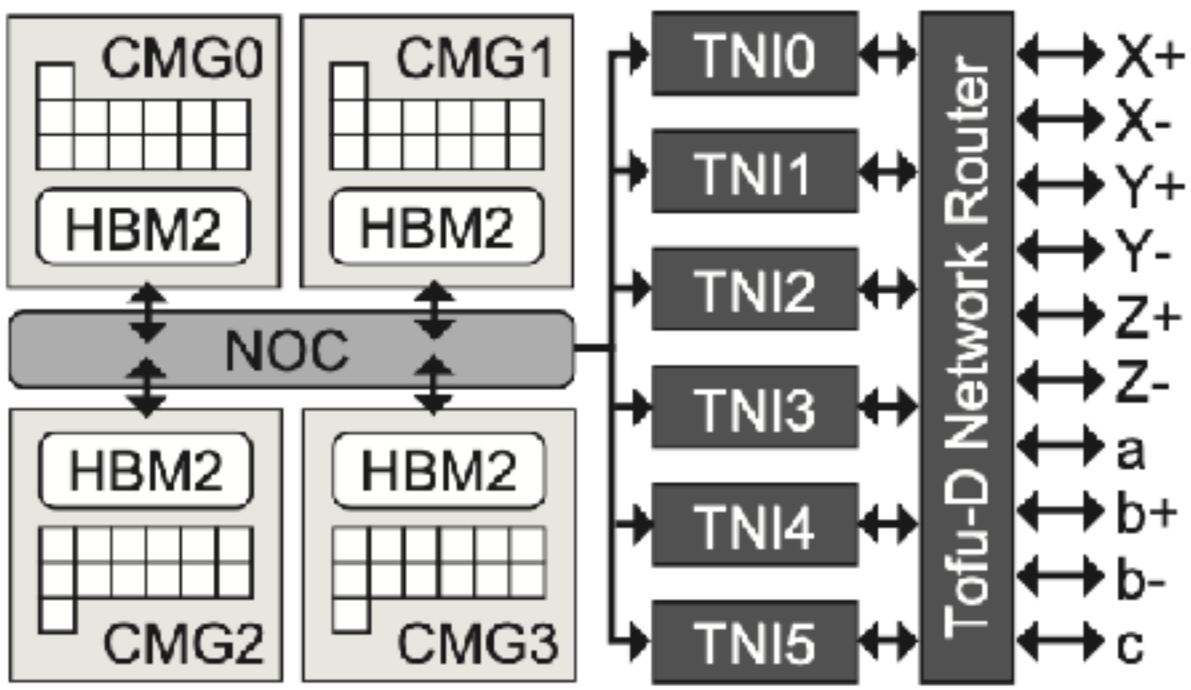
- RIKEN Center for Computational Science@KOBE, Japan
- 158,976 nodes, scheduled to commence sharing in 2021
- Note that the results of this presentation do not guarantee performance at the start of sharing as it is currently a pre-sharing evaluation environment

Node on Fugaku

Specification

CPU	A64FX, 48+2/4 cores, 2.0/2.2GHz, 3,072/3,379GFlops(DP)
Memory	HBM2 32 GiB, 1,024GB/s
Interconnect	Tofu-D, 6-dimensional mesh/torus 28.05Gbps × 2 lane ×10 ports

CPU (A64FX)

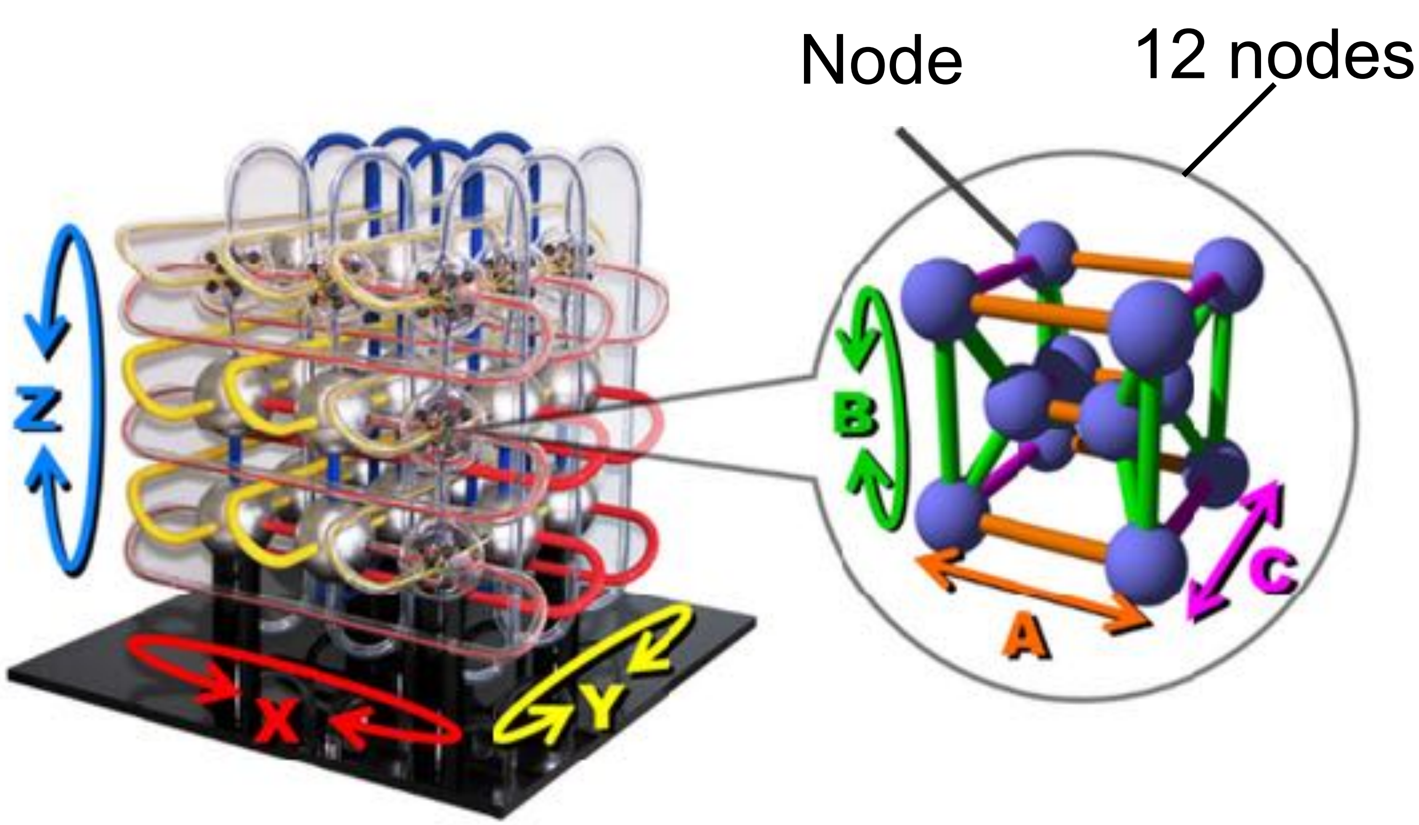


CMG : Core Memory Group
NOC : Network on Chip
TNI: Tofu Network Interface

- CPU has 48 compute cores
- and 2/4 assistant cores
    - Handle interrupts such as OS
- **2.0 GHz or 2.2 GHz for each job**
- CPU consists of 4 CMGs
- CMG consists of 12 + 1 cores and 8GiB HBM2
    - **The number of processes per CPU be a multiple of 4**
- Tofu Interconnect D (Tofu-D)
    - 6D mesh/torus
    - XYZabc-axis, 10 cables
    - 6 simultaneous communication

Network topology of Tofu-D

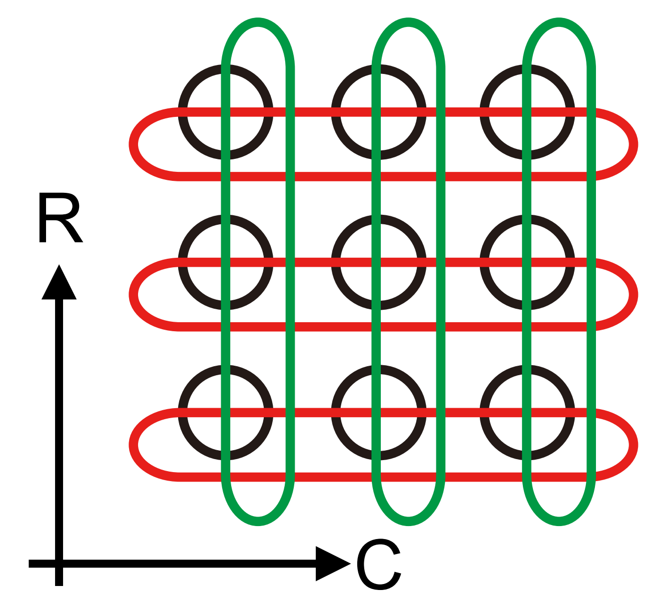
- 6D mesh/torus : XYZabc-axis
  - The size of abc is fixed (a,b,c) = (2,3,2)
  - The size of XYZ depends on the system
  - The size of XYZ of Fugaku is (24,23,24), so it has 24*23*24*2*3*2 = 158,976 nodes



https://pr.fujitsu.com/jp/news/2020/04/28.html

- Process Mapping
  - Discrete assignment
  - 1D torus or mesh
  - **2D torus** or mesh
  - 3D torus or mesh

$$A = \left(\begin{array}{c|c|c} A_{1,1} & \cdots & A_{1,C} \\ \hline \vdots & \ddots & \vdots \\ \hline A_{R,1} & \cdots & A_{R,C} \end{array}\right)$$

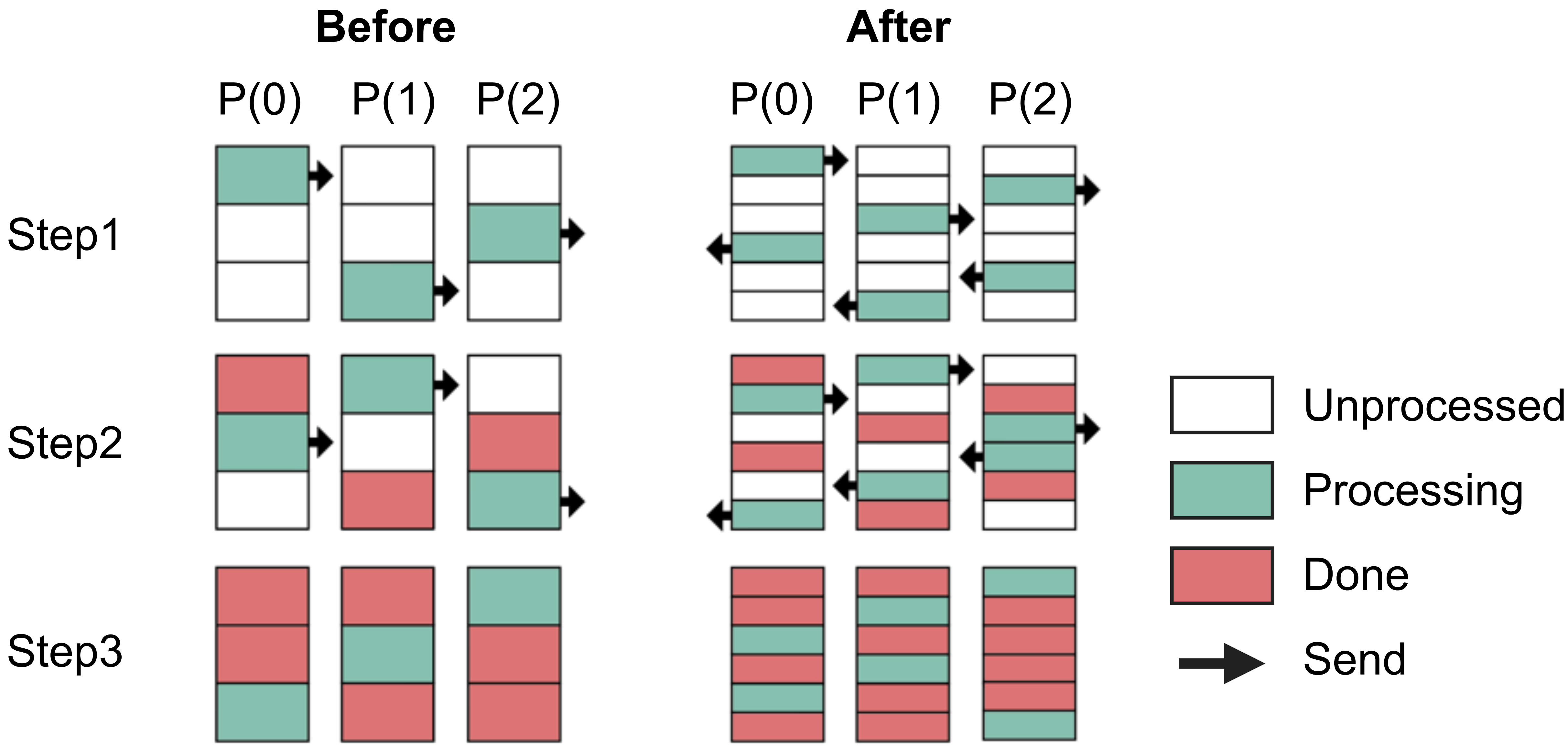


Outline

- Graph500 Benchmark
- Supercomputer Fugaku
- Tuning Graph500 Benchmark on Supercomputer Fugaku

Overlapping communication with computation

- Asynchronous send/recv neighborhood communication in two directions to effectively use torus direct network
- Communication and computation overlap by splitting processing

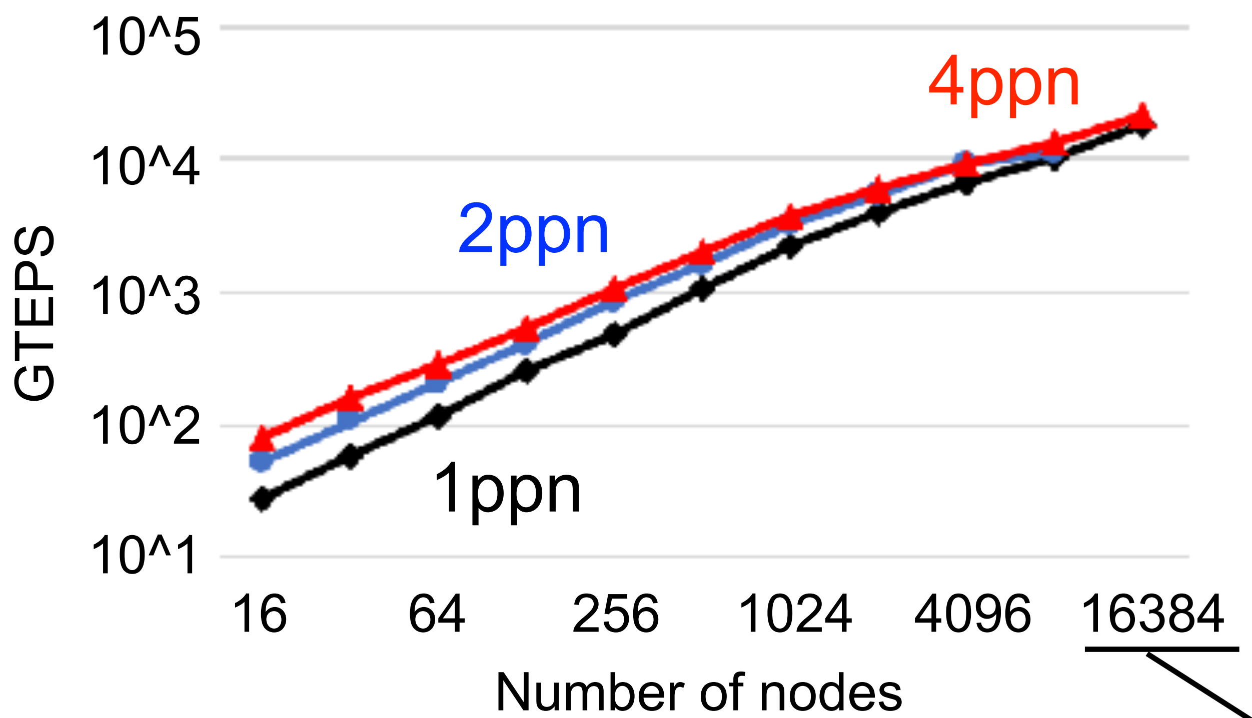


Number of processes per node (1/2)

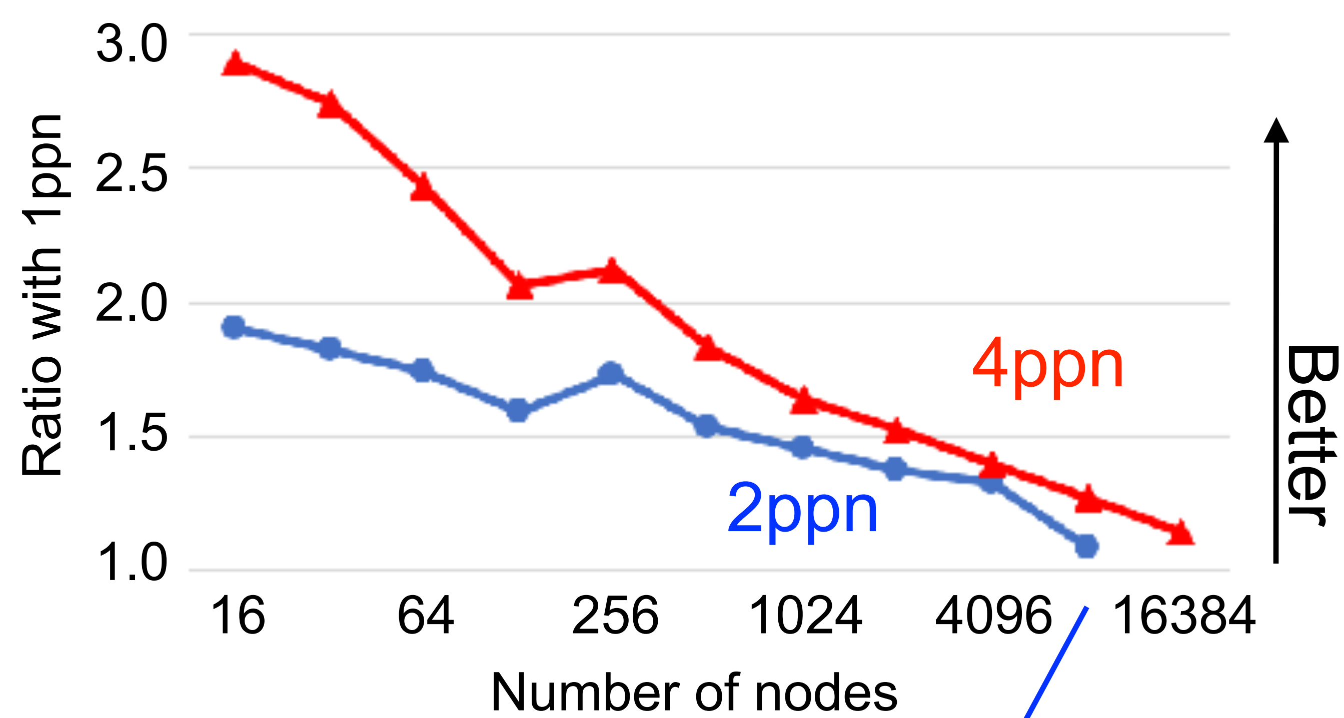
- Process per node (ppn)
  - 1 process 48 threads (1ppn)
  - 2 processes 24threads (2ppn)
  - 4 processes 12threads (4ppn)

The BFS performance unit is traversed edges per second (TEPS), which represents the number of edges searched per second

Performance



Performance Ratio with 1ppn

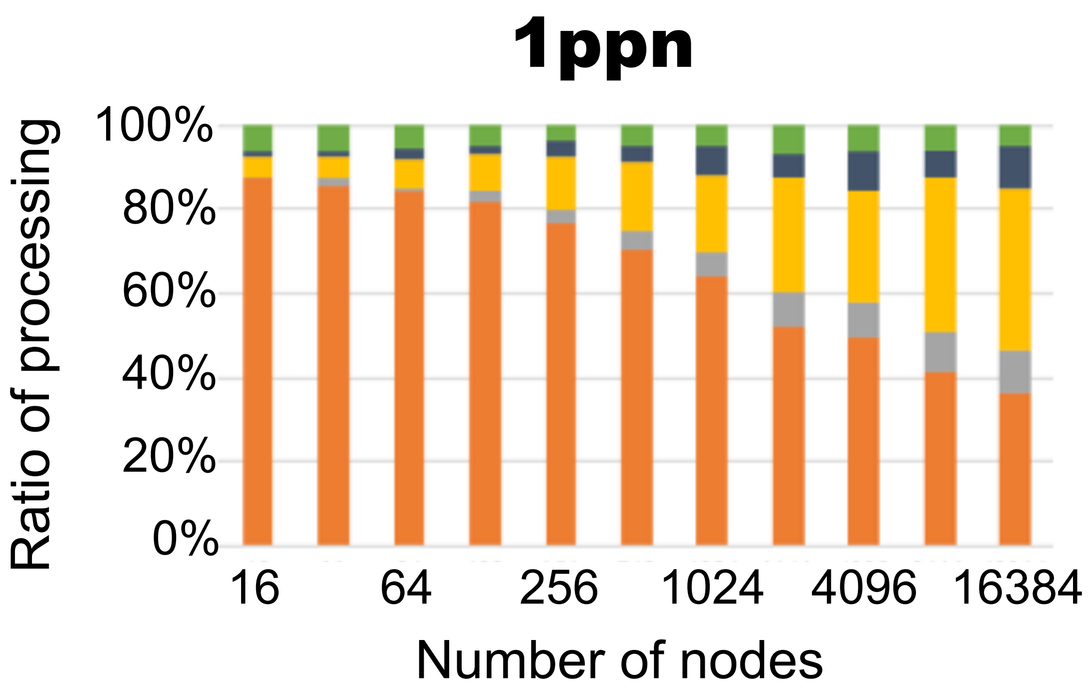


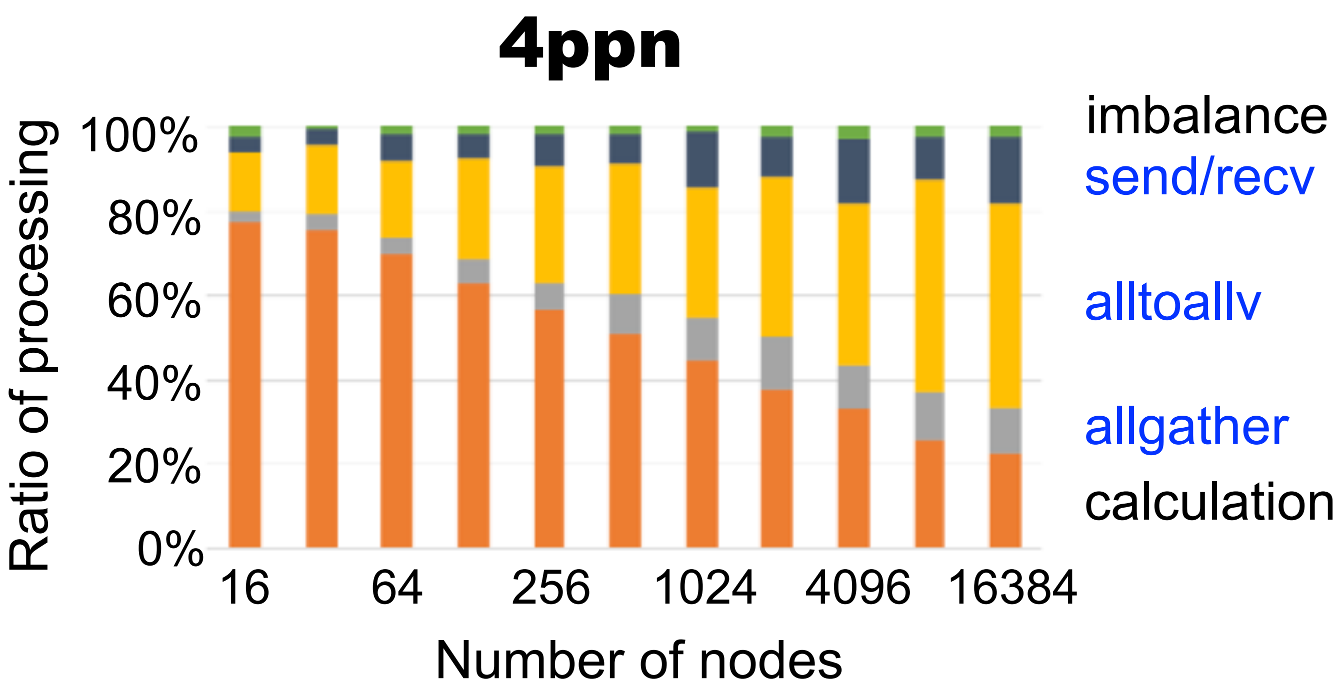
1ppn : R x C = 128 x 128
2ppn : R x C = 256 x 128
4ppn : R x C = 256 x 256

The result of 16384 nodes of 2ppn could not be obtained due to a system malfunction

- The larger the number of nodes, the smaller the performance difference

Number of processes per node (2/2)



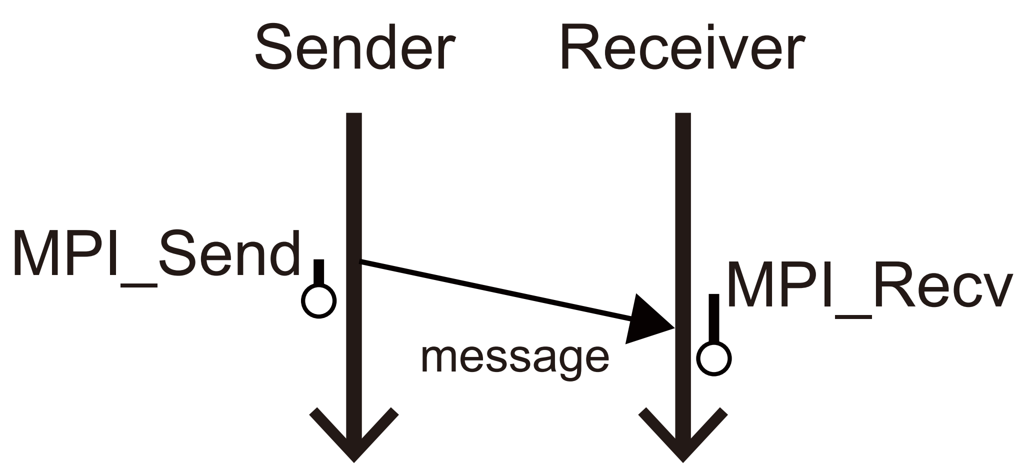


- As the number of nodes increases, the rate of communication increases
- 1ppn has a smaller rate of communication than 4ppn
- If the number of nodes is increased further, the communication ratio will increase. Thus, we will measure at 1 ppn, which can bring out the full communication performance
- The result for 16,384 nodes at 1 ppn was **18,450 GTEPS**

Use of Eager method (1/3)

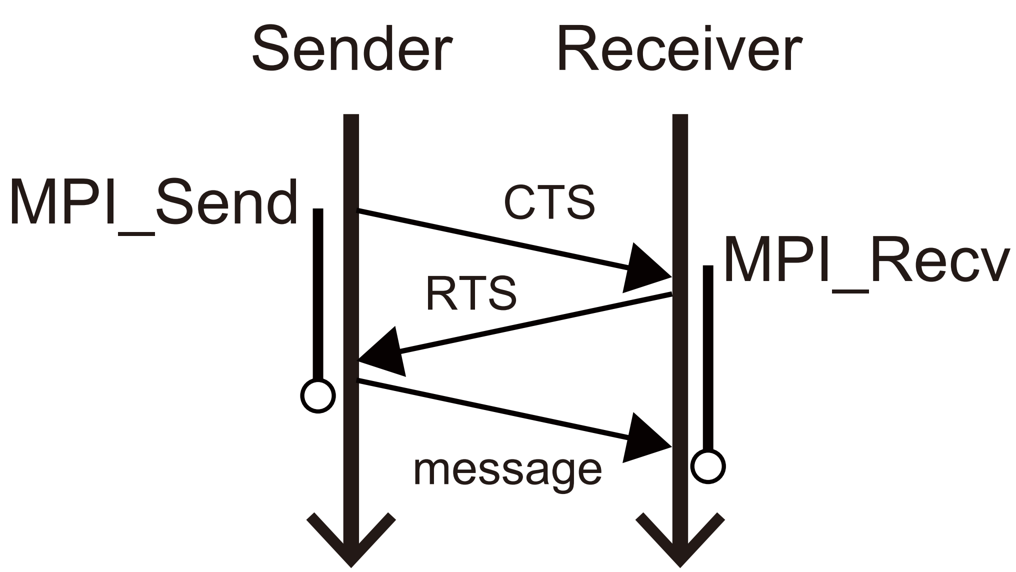
- In the point-to-point communication of most MPI implementations, the Eager and Rendezvous methods are implemented
- Eager is automatically selected when the size is small
- Rendezvous is automatically selected when the size is large

Eager



Asynchronous communication that can start/end the message sending process regardless of the state of the sending/receiving processes

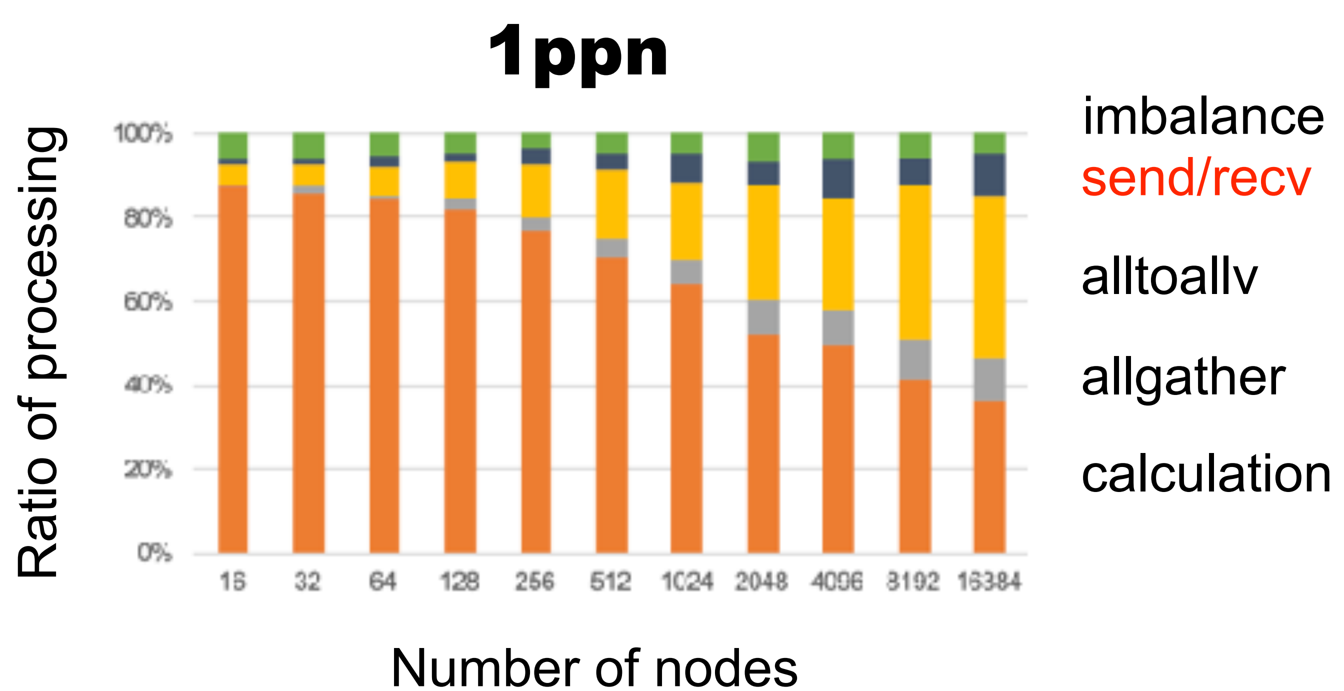
Rendezvous

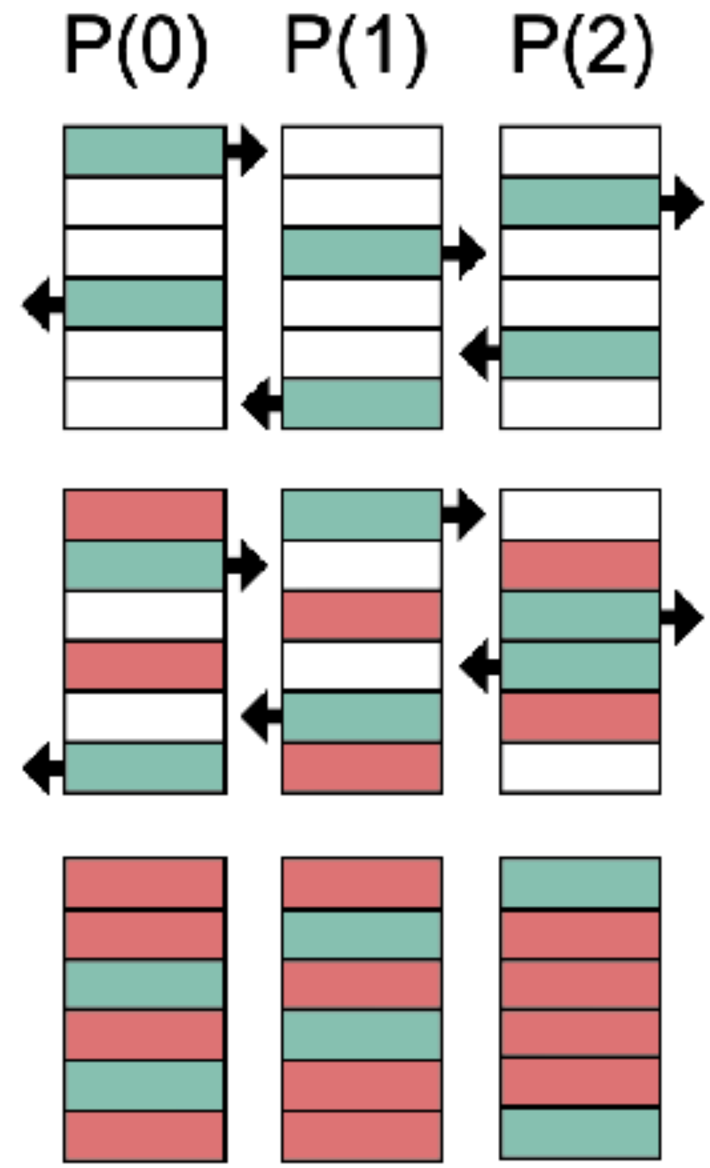


Synchronous communication to send/receive messages after both MPI processes are ready to communicate

Use of Eager method (2/3)

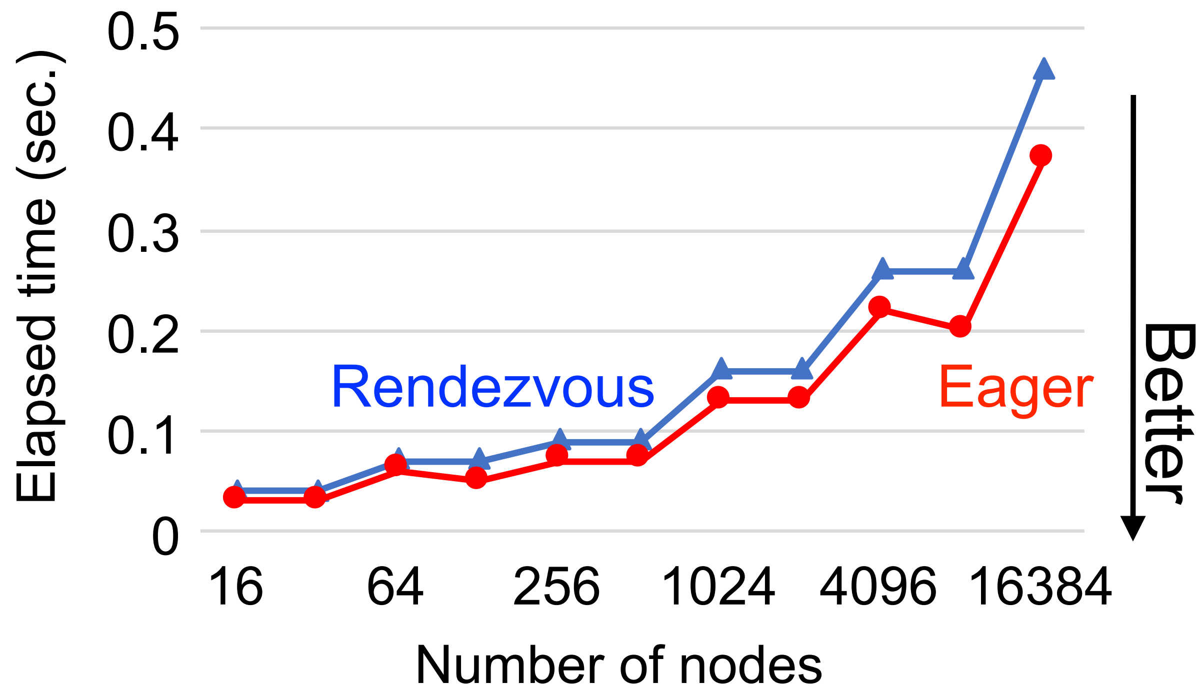
- The send/recv communication method used in the previous experiment was all Rendezvous
- If a node has enough memory and you want to promote asynchronous communication, you can increase the usage rate of the Eager method by passing a parameter (-mca btl_tofu_eager_limit) to mpiexec



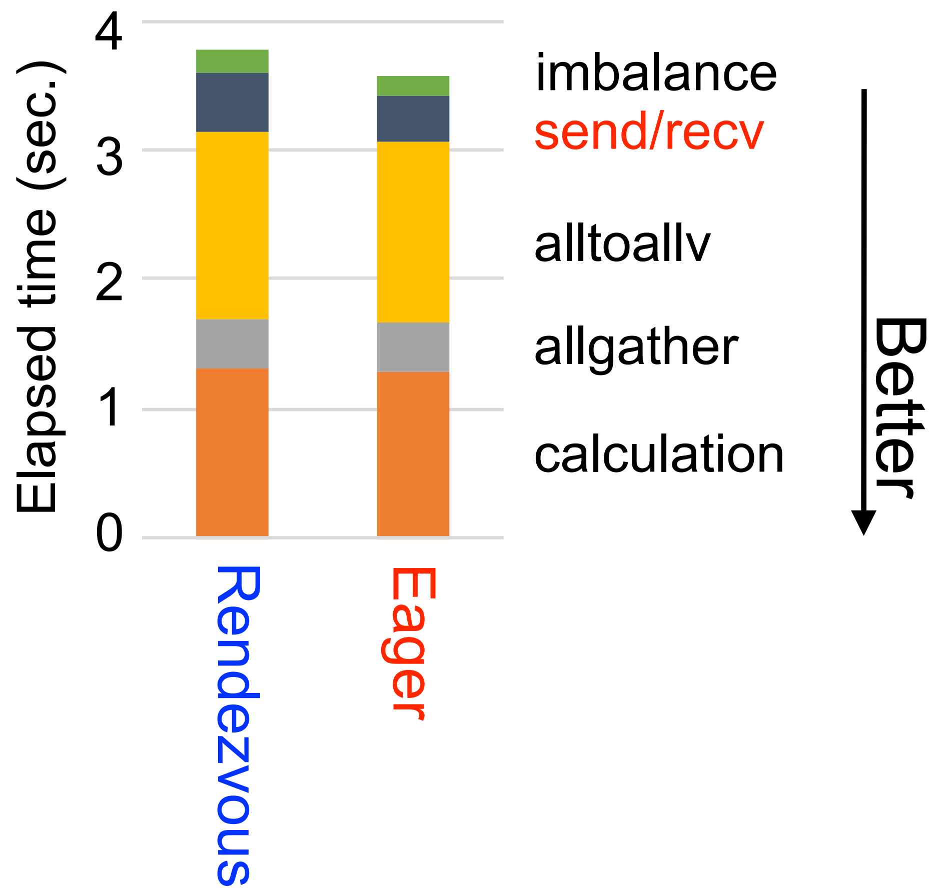


Use of Eager method (3/3)

- Time of send/recv



- Time in 16,384 nodes



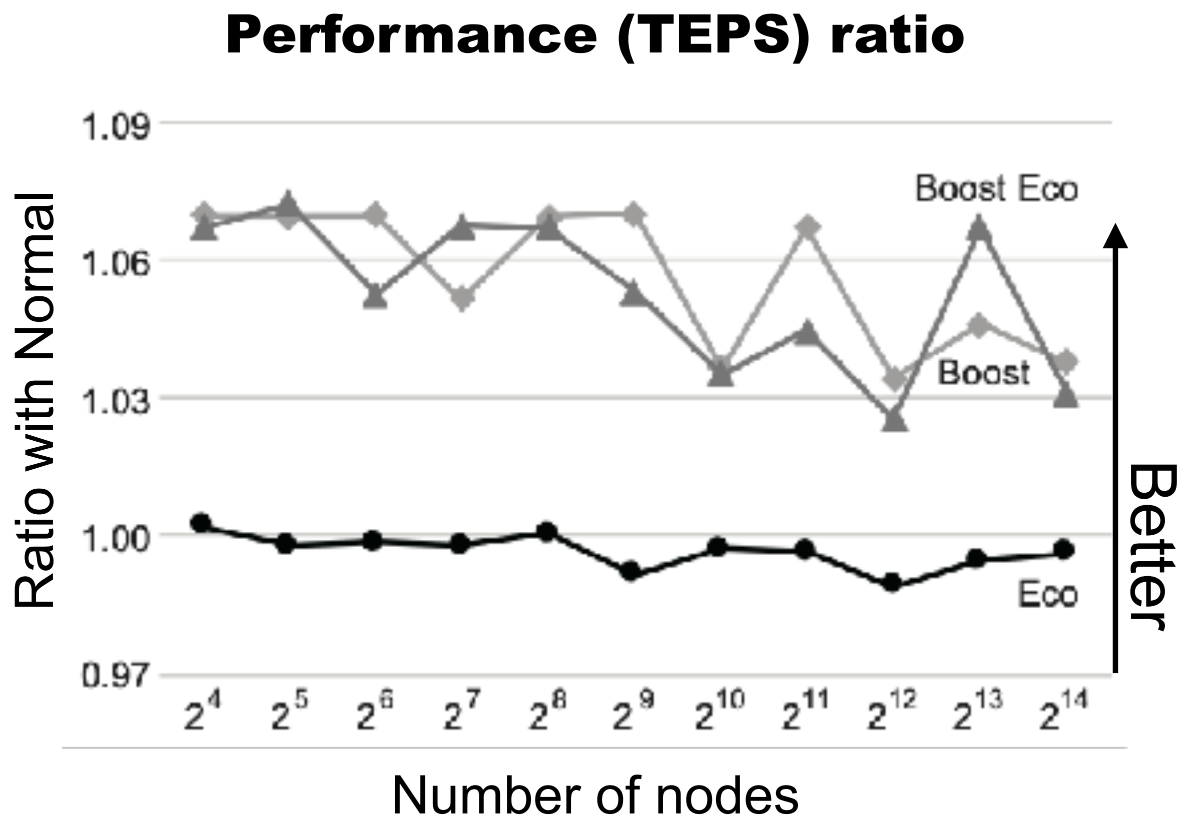
- The result of 16,384 nodes using **Eager method** is **19,496 GTEPS**, which is a 5.7% performance improvement over the result using **Rendezvous method (18,450 GTEPS)**
- In the following experiments, we will execute BFS using **Eager method**

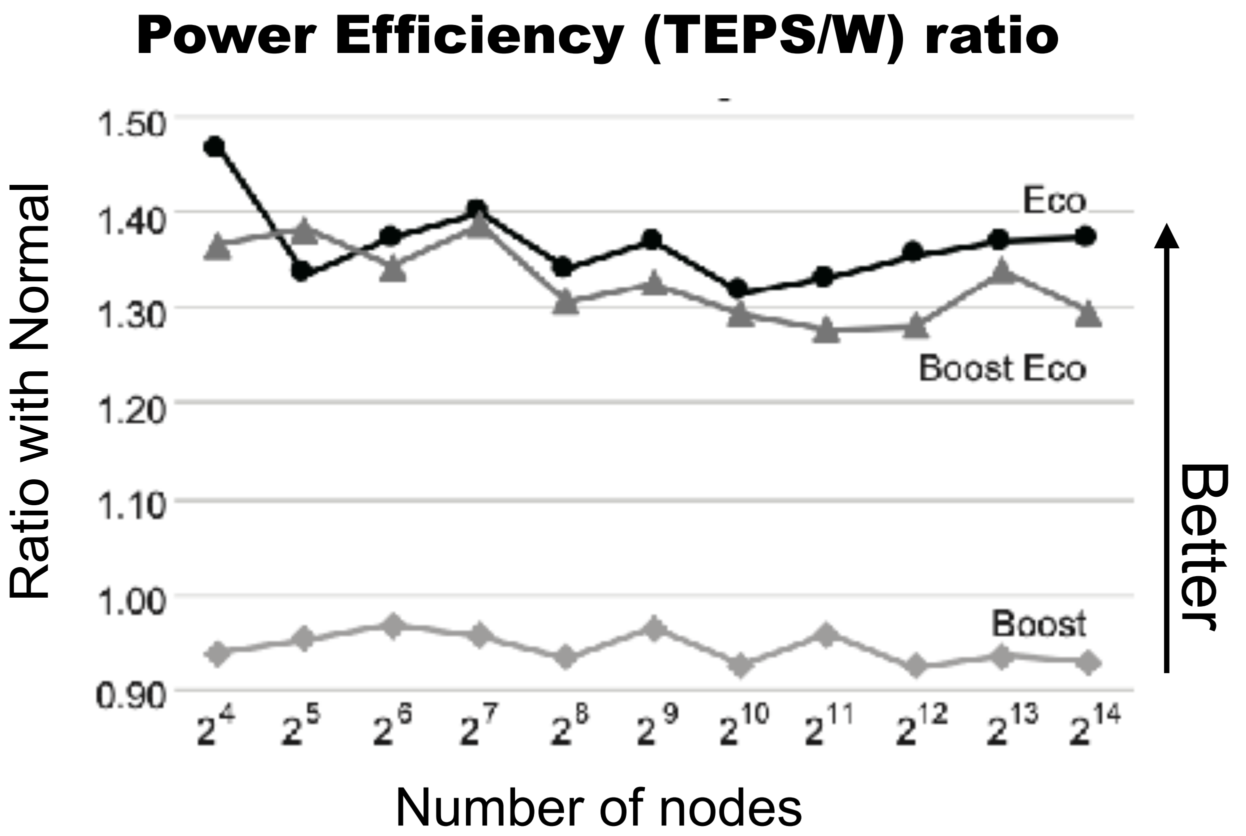
Boost mode and Eco mode (1/2)

- User can specify CPU frequency for each job
  - Normal mode : **2.0** GHz
  - Boost mode : **2.2** GHz
- Eco mode : **Two** floating-point arithmetic pipelines of A64FX are limited to **one**, and power control is performed according to the maximum power
  - Since BFS does not perform floating-point arithmetic, the use of Eco mode can be expected to reduce power consumption without affecting performance

- Normal : **2.0** GHz, **two** floating-point arithmetic pipelines
- Boost : **2.2** GHz, **two** floating-point arithmetic pipelines
- Normal Eco : **2.0** GHz **one** floating-point arithmetic pipeline
- Boost Eco : **2.2** GHz **one** floating-point arithmetic pipeline

Boost mode and Eco mode (2/2)



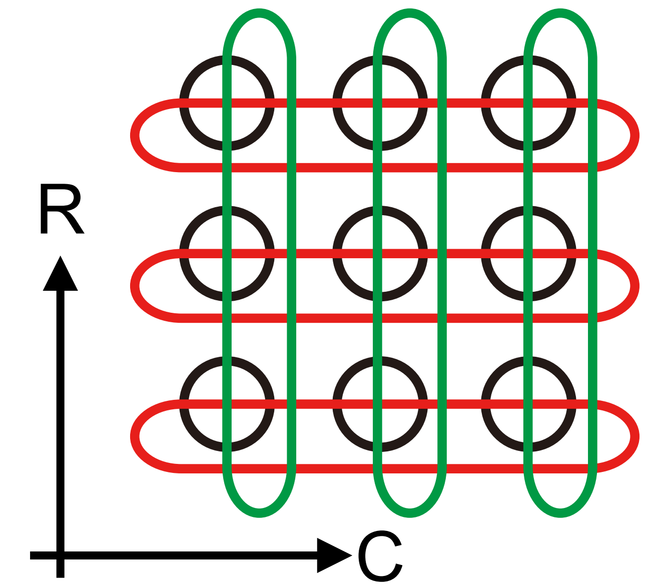


- The result of Normal mode is 1.00. Boost modes give high performance, Eco modes give high power efficiency
- Boost Eco mode has a good balance between performance and power efficiency. The result for 16,384 nodes is **20,098 GTEPS**, which is a 3.1% performance improvement over the previous result (**19,496 GTEPS**)

Six-dimensional process mapping (1/3)

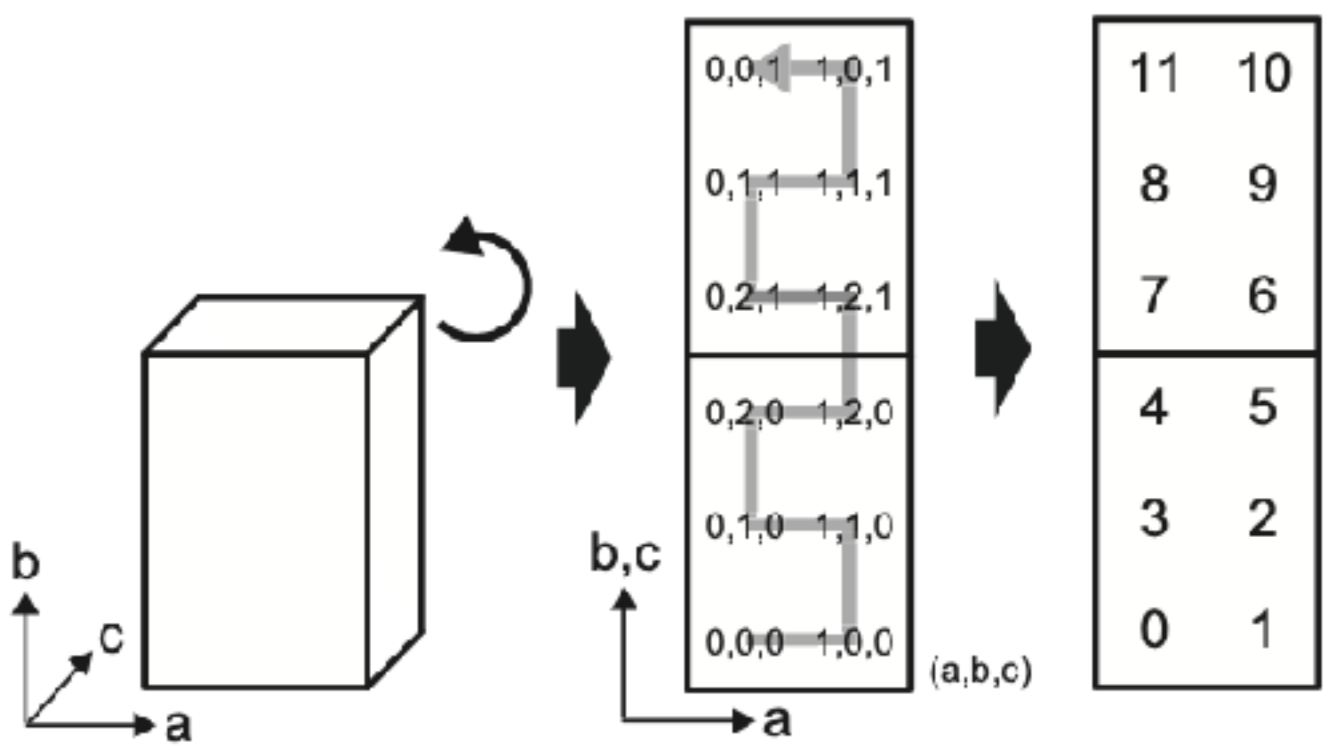
- Network topology in Fugaku
  - The size of six axes is (X, Y, Z, a, b, c) = (23, 24, 23, 2, 3, 2)
  - The maximum value of each axis when 2D process mapping is performed in the Fugaku job scheduler
    YZc × Xab = 1,104 x 114 = R x C
  - However, it is desirable that the values of R and C are close
- Fix BFS code to assign processes to any axis
  - The closest combination of R and C is
    XY × Zabc = 552 x 288 = R x C

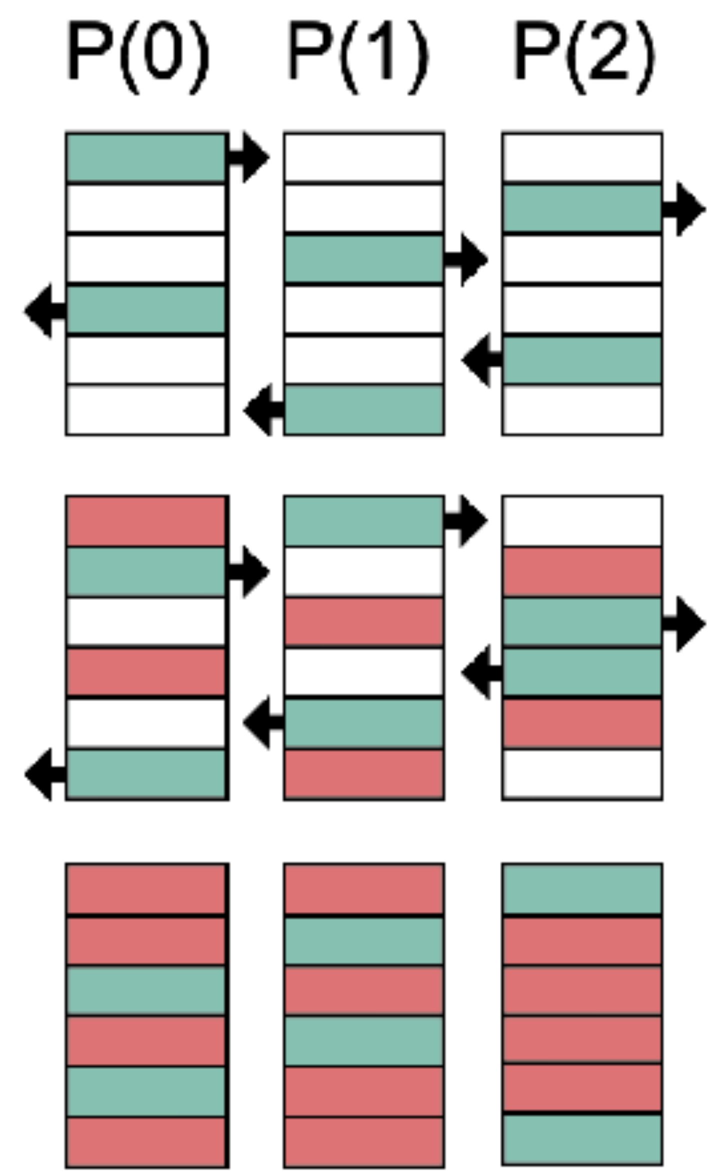
$$A = \left(\begin{array}{c|c|c} A_{1,1} & \cdots & A_{1,C} \\ \hline \vdots & \ddots & \vdots \\ \hline A_{R,1} & \cdots & A_{R,C} \end{array}\right)$$



Six-dimensional process mapping (2/3)

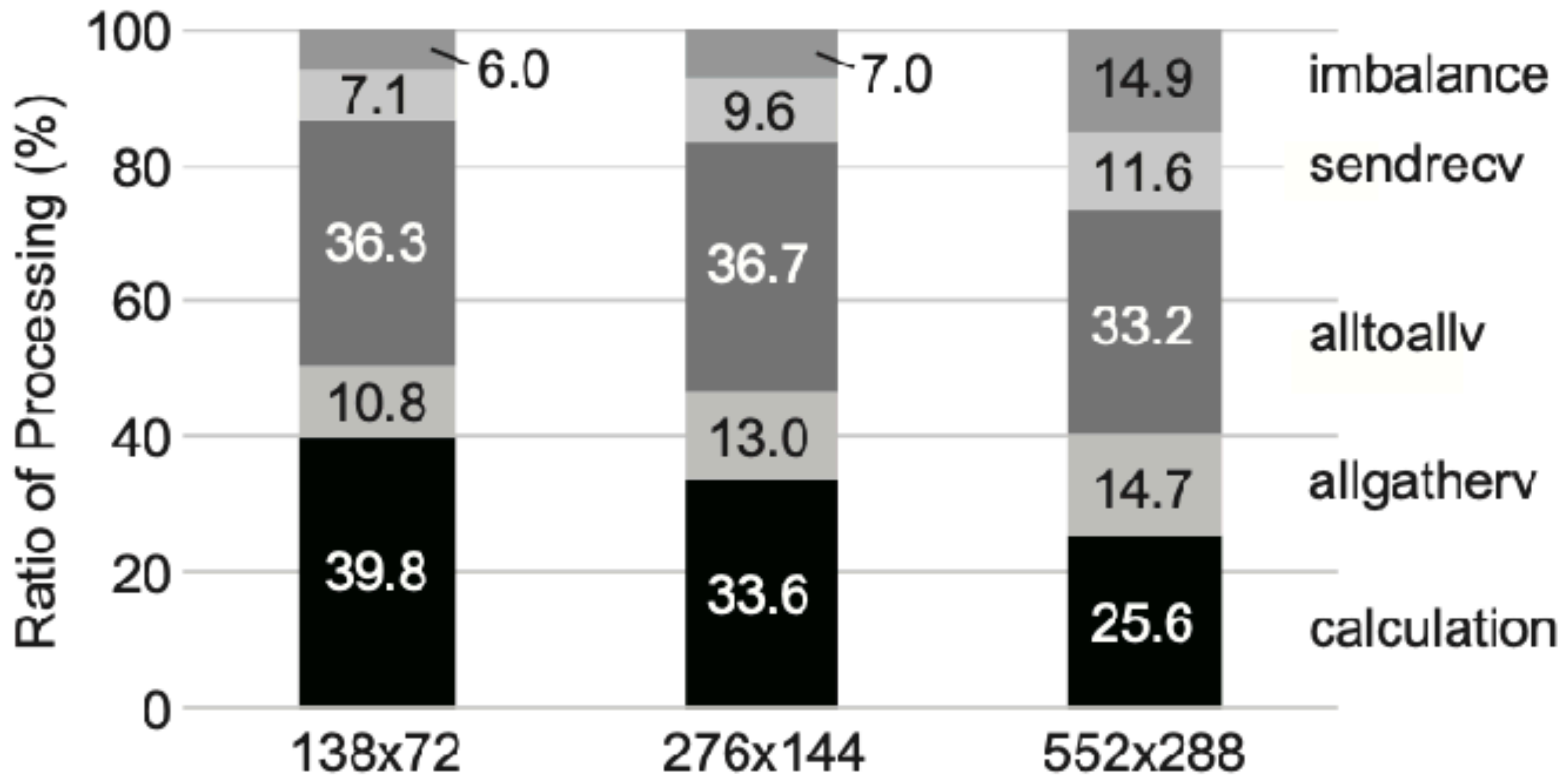
- C axis has high performance when all nodes are adjacent
- Example of assigning abc axis (2 x 3 x 2) to C axis
    - The horizontal is the first axis, and the vertical is the remaining axes
    - To make the first and last processes (0 and 11) adjacent physically





Six-dimensional process mapping (3/3)

- Measure BFS using 158,976 nodes (XY × Zabc = 552 x 288 = R x C)
- Boost Eco mode
- Performance: 102,955 GTEPS, Power: 14,961 kW, Efficiency: 6.9 MTEPS/W
- Performance is 3.3 times that of the K computer (82,944 nodes), and power efficiency is 1.9 times that of IBM Sequoia (Blue Gene/Q)



1/16 Fugaku 1/4 Fugaku 1/1 Fugaku

Summary

- Tune performance of BFS in Graph500
  - Overlap communication and calculation
  - Number of processes per node
  - Eager v.s. Rendezvous
  - Boost mode and Eco mode
  - Six-dimensional process mapping

- Future works
  - NUMA-aware optimization