SCA/HPCAsia 2026
January 26-29, 2026, Osaka, Japan

User Support for Full-System Execution on the Supercomputer Fugaku

Naoki Sueyasu, Tatsunobu Kokubo,
Yukihiro Ota, Masato Matsui, Hiroshi Kobayashi,
Yoshihisa Shizawa, Asako Terasawa, Eiji Tomiyama,
Gilles Gouaillardet, Daisuke Matsuoka

Research Organization for Information Science and Technology (RIST) ,
Kobe, Japan

Background and Motivation

- The Research Organization for Information Science and Technology (RIST) provides support for users of the High Performance Computing Infrastructure (HPCI) in Japan.
- As part of the support, RIST assists with large-scale executions on the full system of the supercomputer Fugaku. Fugaku is a massively parallel system consisting of 158976 nodes, each equipped with a single A64FX CPU based on the Arm architecture.
- A full-system execution refers to an event in which nearly all resources of Fugaku are allocated to a single project to run extremely large-scale jobs.
- These events are typically scheduled twice a year. Over the past two years, RIST has supported 8 of the 13 projects awarded for the full-system execution [1].

Additional Requirements Specific to Full-System Executions

- In addition to challenges associated with conventional jobs involving thousands of processes, full-system executions impose additional requirements specific to the extreme-scale computing:

R1. The application must exhibit extremely high inherent parallelism.
R2. Parallelism should be implemented through a hybrid MPI and OpenMP approach.
R3. Computational load must be evenly balanced across all processes.
R4. MPI communication must be optimized for large-scale execution.
R5. I/O operations should be distributed to avoid bottlenecks.
R6. Memory usage per node must be uniform.
R7. Hardware failures must be detected promptly.
R8. Performance metrics and various statistical information should be collected appropriately.

- For each requirement, we itemized concrete approaches and their associated verification aspects.
- Focusing on the optimization of MPI communication, we found that efficient MPI communication on Fugaku's Tofu-D interconnect needs verification of the following aspects:
  - ✓ whether the communication patterns are consistent with the process mapping;
  - ✓ whether appropriate collective communication algorithms are employed for MPI collective operations;
  - ✓ whether the communication mode achieves a balance between performance and memory usage; and
  - ✓ whether point-to-point communications are performed without unexpected messages.
- Over the past two years, RIST has supported eight projects awarded full-system execution. By applying the approaches summarized in Table 1, we have supported the successful completion of these full-system executions.

Table 1: RIST support for eight projects awarded full-system execution during 2024 - 2025

Project Name	Project ID	Date	Approaches of Support
Examination of effects from variety of libraries and compilers in huge-scale parallel simulation of eigenvalue problems	hp230114	February, 2024	R4. Optimized MPI communication - Balancing performance and memory usage - Reducing unexpected messages
	hp230532	July, 2024	R6. Uniform memory usage per node - Allocating non-uniform numbers of processes per node
Huge-scale molecular dynamics simulation of ultrasonic cavitation	hp240112	February, 2025	R4. Optimized MPI communication - Balancing performance and memory usage - Proper process mapping - Selecting appropriate collective communication algorithms
Galaxy simulation with AI	hp240219	February, 2025	R4. Optimized MPI communication - Proper process mapping - Selecting appropriate collective communication algorithms
	hp250226	July, 2025	
Direct numerical simulation of turbulent channel flow at world's largest Reynolds number using full-nodes on Fugaku	hp240171	February, 2025	R4. Optimized MPI communication - Proper process mapping by FugakuNodeMappingTools - Selecting appropriate collective communication algorithms R5. Distributed I/O operations - I/O striping and LLIO usage to maintain peak performance
Microscopic mouse whole cortex simulation	hp240214	February, 2025	R2. Hybrid MPI and OpenMP parallelism - Improving SIMD operations R4. Optimized MPI communication - Selecting appropriate collective communication algorithms
	hp250231	July, 2025	R5. Distributed I/O operations - Staging and LLIO usage to avoid I/O bottlenecks

Node Mapping Tools

- Through our support for project hp240171 [2], we found that the cost of MPI_Alltoall communication is dominant and requires optimization.
- To address this issue, we developed the FugakuNodeMappingTools to generate optimal process mappings that confine MPI_Alltoall operations within neighboring Tofu units.
- FugakuNodeMappingTools automatically generates efficient process mappings based on the communication characteristics of Fugaku's Tofu-D interconnect.
- It is designed for three-dimensional applications in which two dimensions (XY) involve intensive communication and the remaining dimension (Z) involves relatively sparse communication.
- The creation of a rank map file consists of the following two steps:
  1. Retrieving the default process placement using dump_data.f90
  2. Generating the rank map file using main_tofu.f90

1. **Retrieving the default process placement using dump_data.f90**
   - A job is executed without using a rank map file to obtain the default process mapping.
   - By using the FJMPI_TOPOLOGY_GET_COORDS subroutine provided by FJMPI [4], the following information is obtained:
     - A) Logical node coordinates (XL, YL, ZL)
     - B) Tofu coordinates of the node (physical coordinates) (XP, YP, ZP, AP, BP, CP)
     - C) Tofu coordinates of the node (relative coordinates with respect to process #0) (XR, YR, ZR, AR, BR, CR)

Example of dump_data job script:

```
#!/bin/sh -l
#PJM -L node=20x12x24:torus:strict
#PJM -L elapse=0:05:00
#PJM -L rscunit=rscunit_ft01
#PJM -L rscgrp=large
#PJM -L proc-core=unlimited
#PJM --mpi "proc=5760"
#PJM --mpi "rank-map-bynode"
#PJM --mpi "assign-online-node"
#PJM -g xxxxxxx
#PJM -S
#
llio_transfer ./dump_data.exe
###
mpiexec ../dump_data.exe -Wl,-T
```

Example of dump_data output:

```
#1_node  #2_dim  #3_X #4_Y #5_Z
   5760       3   20   12   24
      0    0    0    0   17   12    4    0    0    0      0    0    0    0    0    0
      1    1    0    0   18   12    4    0    0    0      1    0    0    0    0    0
      2    2    0    0   19   12    4    0    0    0      2    0    0    0    0    0
     ...
   5757   17   11   23   19   12    4    1    1    1      2    0    0    1    1    1
   5758   18   11   23   18   12    4    1    1    1      1    0    0    1    1    1
   5759   19   11   23   17   12    4    1    1    1      0    0    0    1    1    1
```

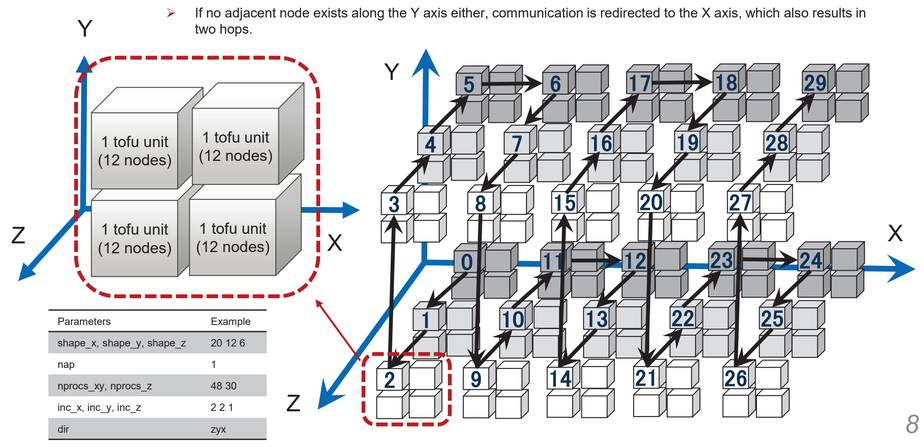

2. **Generating the rank map file using main_tofu.f90**
   - This program does not require job execution and runs on the login node.
   - This program generates a rank map file by taking as input the default process placement obtained by dump_data.f90 and the desired process topology specified by the following parameters.

Example of main_tofu execution:

```
echo 20 12 24 > shape.txt
echo 1 > nap.txt
echo 48 120 > xy_z.txt
echo 2 2 1 > inc_xyz.txt
echo zyx > dir.txt
../a.tofu > output_20x12x24_1_48x120_2x2x1_yxz.txt
```

Parameter file	Parameters	Example	Description
shape.txt	shape_x, shape_y, shape_z	20 12 24	3D shape of the application
nap.txt	nap	1	Number of processes per node
xy_z.txt	nprocs_xy, nprocs_z	48 120	Size of the XY and Z dimension of the application
inc_xyz.txt	inc_x, inc_y, inc_z	2 2 1	Size of rectangular region along the XY dimension (in Tofu units).
dir.txt	dir	zyx	Routing priority for the Z dimension of the application

- **Example of the generated rank map file**
  - Rank map file generation policy
    1. For the dimensions with intensive communication (XY), a block of four Tofu units (shape of 2 x 2 x 1) is allocated.
       - ➢ For example, when an all-to-all communication is performed within this block, the use of a high-performance MPI collective communication algorithm is ensured.
    2. For the dimension with sparse communication (Z), processes are assigned to adjacent nodes so as to minimize communication distance (i.e., the number of hops) as much as possible.
       - ➢ The routing priority for the application's Z dimension is Z > Y > X.
       - ➢ Adjacent communication along the Z dimension requires one hop.
       - ➢ If no adjacent node exists along the Z axis, communication is redirected to the Y axis, resulting in two hops.
       - ➢ If no adjacent node exists along the Y axis either, communication is redirected to the X axis, which also results in two hops.



Parameters	Example
shape_x, shape_y, shape_z	20 12 6
nap	1
nprocs_xy, nprocs_z	48 30
inc_x, inc_y, inc_z	2 2 1
dir	zyx

Node Mapping Results

- The use of process mappings generated by FugakuNodeMappingTools
  - ✓ ensured the use of appropriate collective communication algorithms for MPI_Alltoall;
  - ✓ eliminated interference from other MPI_Alltoall operations; and
  - ✓ assigned processes to adjacent nodes to minimize the communication distance.
- As a result, this approach achieved performance improvements of 3.0 to 5.5 times [2].
- The tool for generating rank map files is publicly available as FugakuNodeMappingTools [3].

Conclusions and Future Work

- We have provided support for full-system executions on Fugaku and established several methodologies during this process.
- In this poster, we highlighted the rank mapping tool as one example of optimization methodologies. We developed FugakuNodeMappingTools and achieved performance improvements of 3.0 to 5.5 times by generating optimal process mappings.
- In addition, we are also developing additional methodologies, including approaches for ensuring uniform memory usage per node.

ACKNOWLEDGMENTS

- This research used computational resources of the supercomputer Fugaku provided by the RIKEN Center for Computational Science.
- The development of FugakuNodeMappingTools was carried out as part of the Advanced User Support Program for the HPCI System Research Project (Project ID: hp230138 and hp240171).

REFERENCES

[1] HPCI Awarded Projects: https://www.hpci-office.jp/en/using_hpci/awarding_results
[2] Project Report: https://www.hpci-office.jp/output/hp230138/outcome.pdf
[3] FugakuNodeMappingTools: https://github.com/rist-kobe/FugakuNodeMappingTools
[4] Fujitsu Software Technical Computing Suite V4.0L20 Development Studio MPI User's Guide: https://www.r-ccs.riken.jp/fugaku/docs/manual/en/lang/mpi/j2ul-2565-01enz0.pdf