FUJITSU

shaping tomorrow with you

MPI Forum May 2012 Meeting @ Tsukuba:

Fujitsu extensions of Open MPI for K computer

Takahiro Kawashima, Tomoya Adachi, Shinji Sumimoto
Fujitsu Limited

Outline of This Talk

FUJITSU

- RIKEN and Fujitsu are jointly developing K computer at RIKEN AICS, Kobe, Japan
  - Public operation will start on autumn of 2012

- Outline of This Talk
  - K computer and its interconnect Tofu
  - Our challenges to MPI implementation
  - Performance evaluation

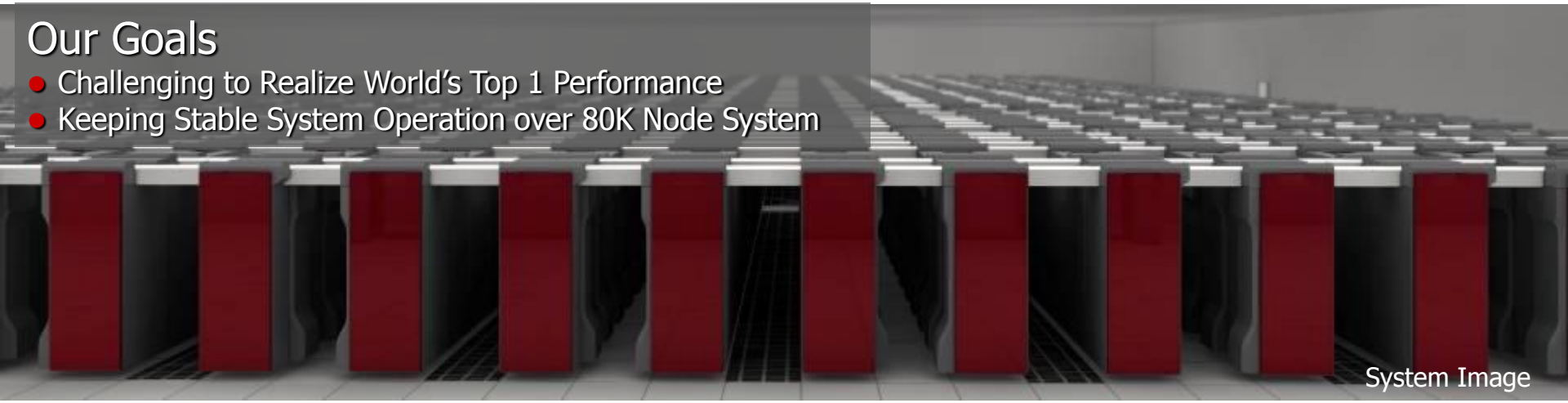
K computer

FUJITSU

- 88K nodes connected by Tofu interconnect
- Each node equips:
  - one SPARC64 VIIIfx processor (8 cores, 2.0GHz)
  - one Tofu interconnect controller
- We got:
  - No.1 of Nov. 2011 TOP500 (10.51PFLOPS with 93.2% efficiency)
  - No.1 in Four benchmarks at 2011 HPC Challenge Awards
  - Gordon Bell Prize for Peak Performance at SC11

京

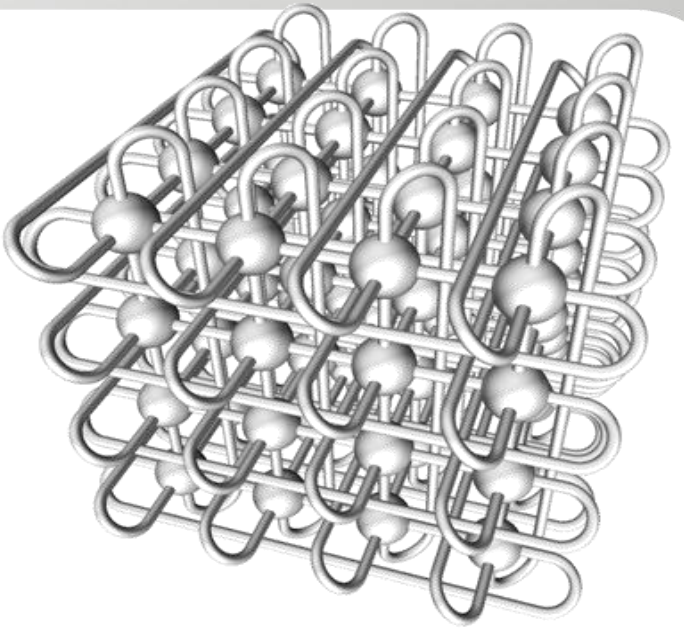
K computer

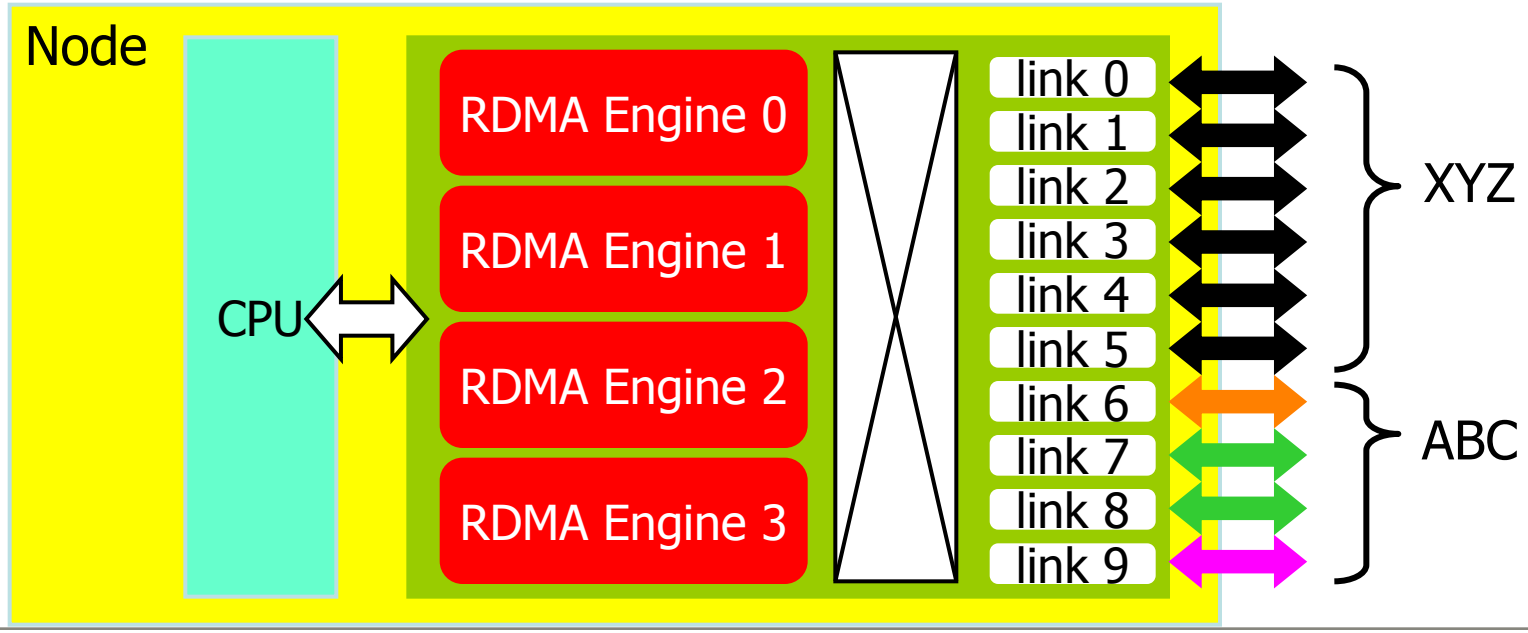


Software View of Tofu Interconnect

FUJITSU

- 3D Torus View for User's Job
  - Combining 6D torus/mesh axes to 3 rings
- 4 RDMA Engines (NIC) per Node
  - RDMA Interface: Put and Get
  - Minimum latency to neighbor nodes: < 1μs
  - Unidirectional bandwidth per engine: 5GB/s
    (Total bidirectional bandwidth: 40GB/s)





Fujitsu MPI and Our Challenges

FUJITSU

- Fujitsu MPI for the K computer: Based on Open MPI with
  - Original Process Management and High Performance Point-to-Point Communication for Tofu Interconnect
  - Enhanced Collective Communication
- Our Challenges
  - High Performance
    - High Bandwidth
    - Low Latency
    - Effective Collective Communication
    - Simple Tofu RDMA Interface

    } Topics of This Talk
  - Reduced Memory Consumption
    - 12KiB/process communication buffer needs 1GiB for 88K processes!
    - Trade-off with communication performance
  - Usability
    - Hide complex 6D torus/mesh view and provide arbitrary-sized 1D/2D/3D torus view

Low Latency: Issues and Our Solution

FUJITSU

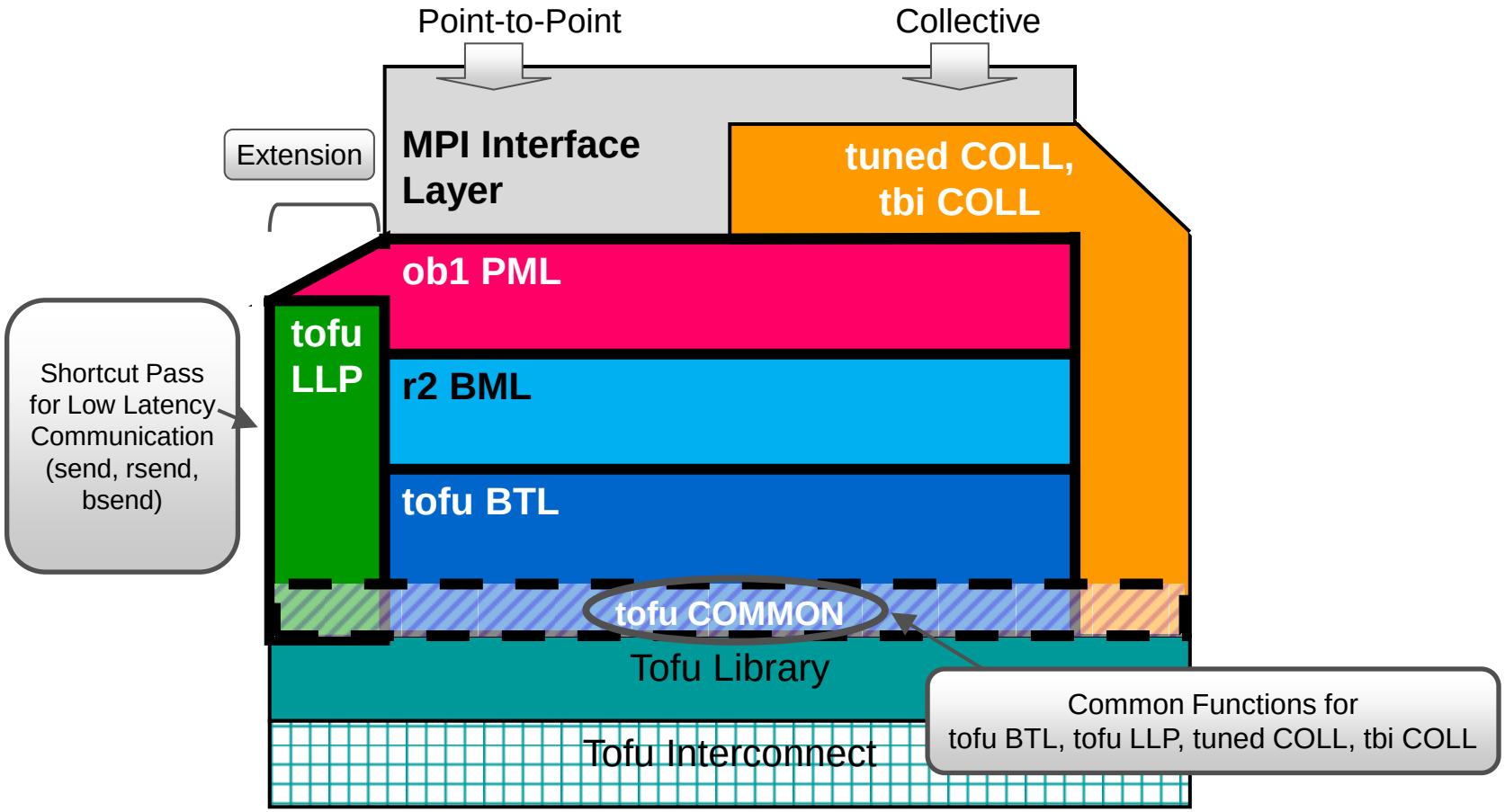
- Issues
  - Software overhead should be << 1μs for short message PingPong in order to utilize Tofu hardware latency (< 1μs).
  - Rich functionalities of MPI (communicator, datatype, blocking/non-blocking, ...) involve software overhead.

- Our Solution: Tofu LLP (Tofu Low Latency Path)
  - Optimized path dedicated to blocking send of short & contiguous message.
  - In the Open MPI terms, we created new “LLP framework” and “tofu LLP component”.

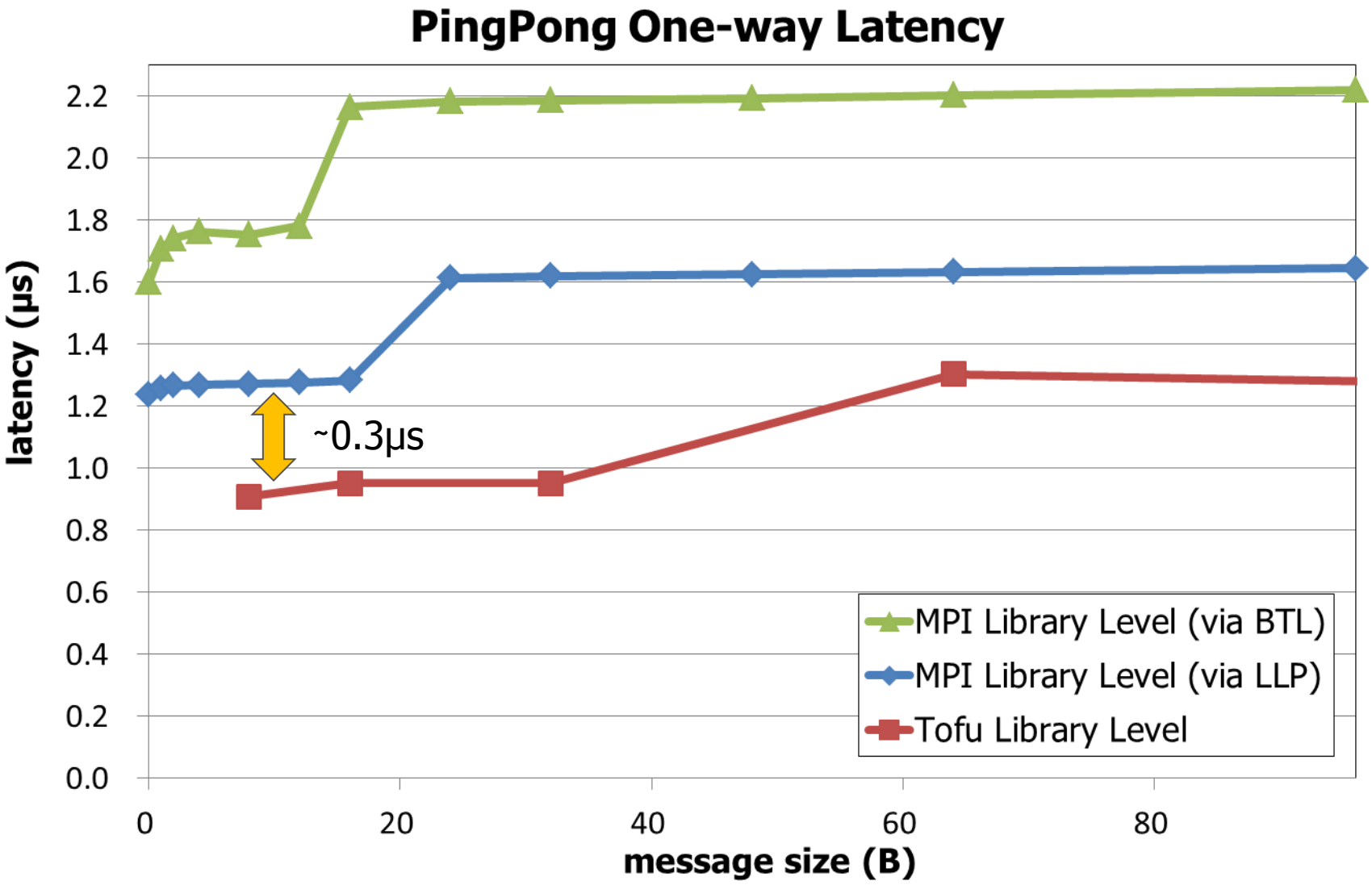
Low Latency: Open MPI Stacks of K computer

FUJITSU

- Tofu LLP bypasses creating a request object, packing data for complex datatype, creating a BTL descriptor, ...
- Falls back to normal path if condition doesn't meet.



Low Latency: PingPong Latency



Software overhead is relatively small
(though we are not satisfied yet)

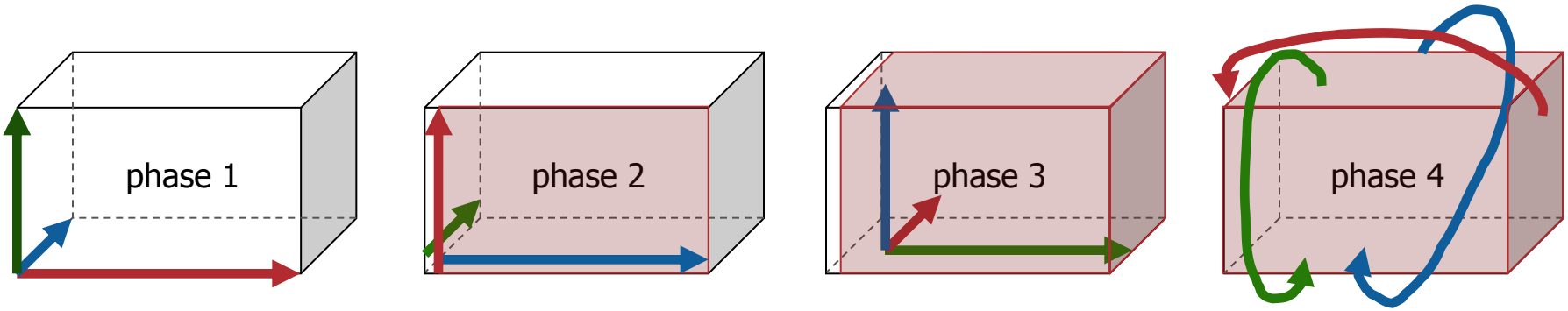
Effective Collective Communication: Issues and Our Solution

FUJITSU

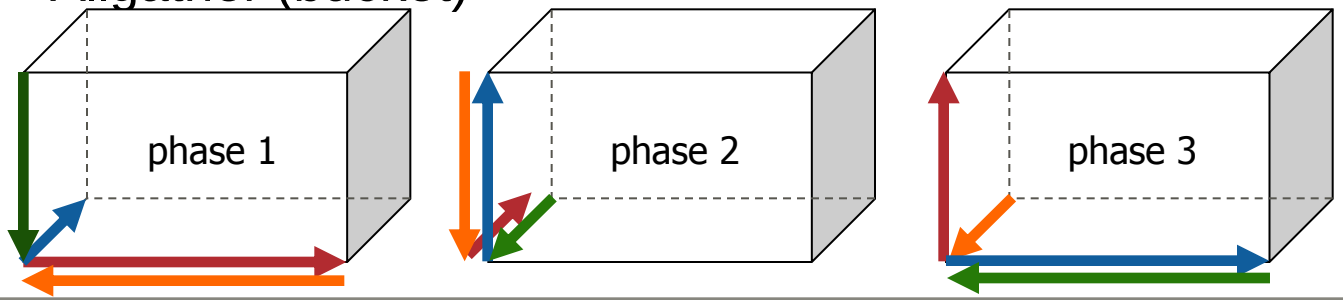
- Issues of Collective Communication Implementation of Open MPI:
  - Not topology-aware, but rank-based
  - Not multi-NIC-aware
    - →Frequent message collision (results in low bandwidth)
  - Send/Receive model
    - →Large software overhead
      (e.g. rendezvous on each pipeline segment)
- Our Solution:
  - Using topology- and multi-NIC-aware algorithms with one-sided RDMA-based communication

Effective Collective Communication: Topology- and multi-NIC-aware algorithms

- Collision-freeness: Communicating only with neighbor nodes in a pipelined or bucket manner
- Multi-NIC-awareness: Dividing messages into multiple parts and transferring via different paths
- Examples
  - Bcast (pipeline)

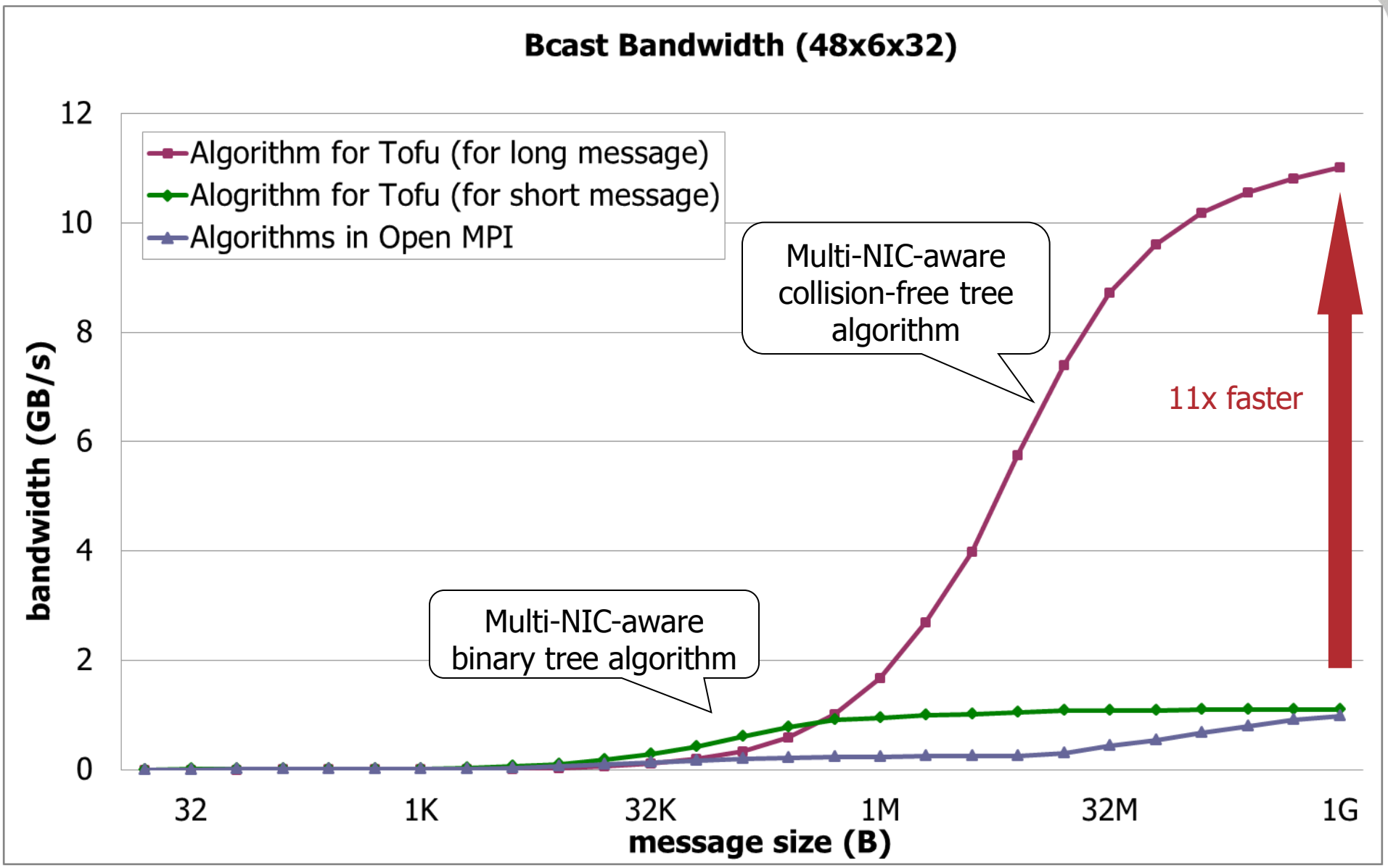


  - Allgather (bucket)



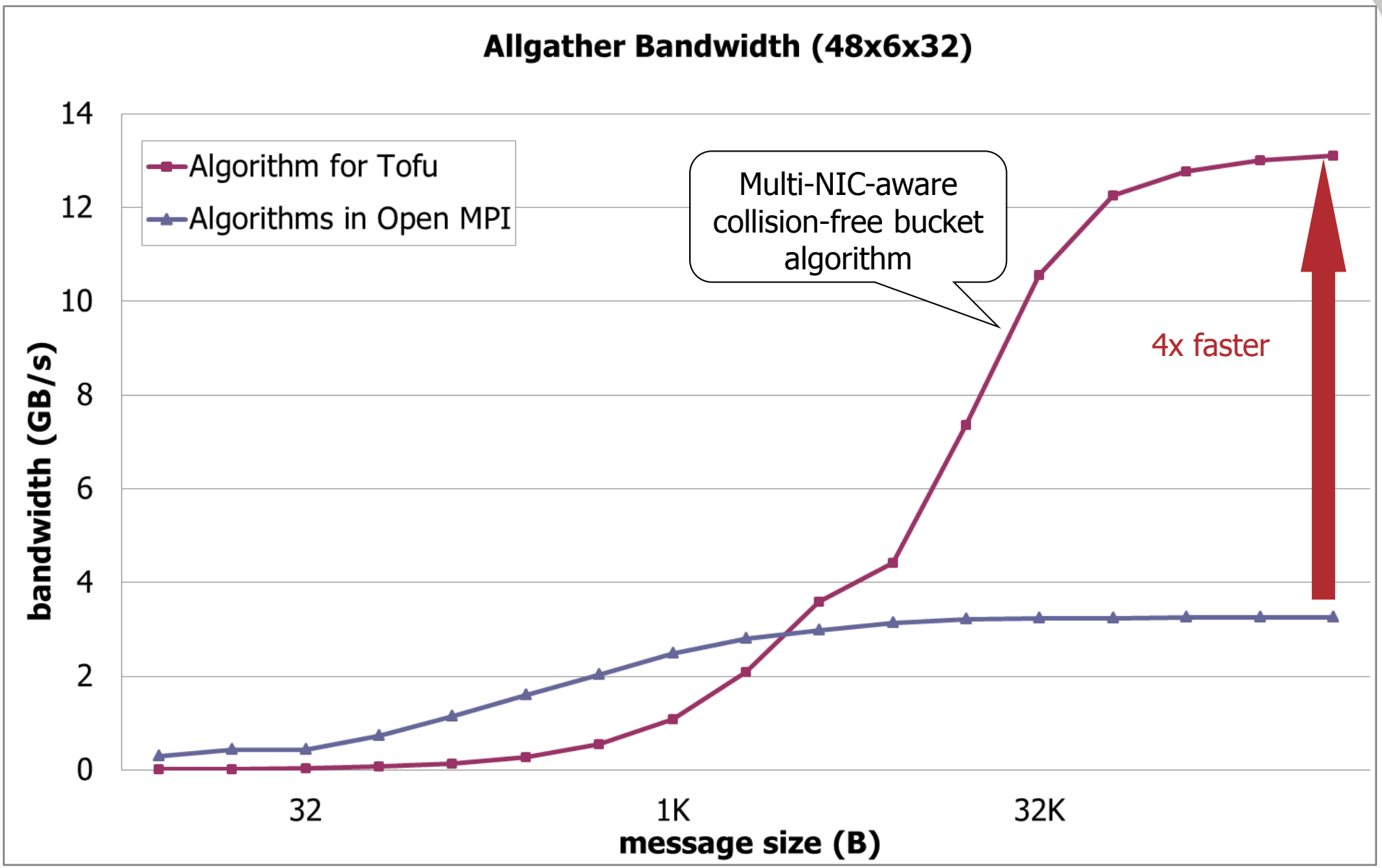
Effective Collective Communication: Bcast Bandwidth

FUJITSU



Effective Collective Communication: Allgather Bandwidth

FUJITSU



Simple Tofu RDMA Interface: Issues and Our Solution

FUJITSU

- Issues:
  - Rich and portable functionalities of MPI point-to-point communication and one-sided communication involve various overheads.
  - Users cannot fully control communication by MPI calls.
    - True RDMA
    - Multi-NIC
    - Specific Hardware Feature
    - Communication Path in Torus/Mesh
  - Applications may get more performance by low-level hardware control.
- Our Solution: Simple Tofu Specific RDMA Interface
  - Fujitsu-specific API (FJMPI_Rdma_ prefix)
  - Low-level RDMA communication
    - Able to control NIC, communication path, and memory registration directly.
    - Able to use Tofu specific feature (remote process notification on RDMA)
  - Simplified API; only RDMA (Put/Get), no communicators, no datatypes.
  - Abstract API; can be implemented for widely-used InfiniBand.

Simple Tofu RDMA Interface: Application Performance

FUJITSU

- Global RandomAccess, modified to use extended RDMA interface, shows good performance.
  - K computer got No.1 on 2011 HPC Challenge G-RandomAccess with it.

Rank	Achieved (GUPS)	System	Affiliation	Peak (TFLOPS)	Processes
1	121	K computer (1/5 scale)	RIKEN AICS	2359	18432
2	117	IBM BG/P	LLNL	446	32768
3	103	IBM BG/P	ANL	557	32768
4	38	Cray XT5	ORNL	2320	111556

http://www.hpcchallenge.org/

- Application-specific tuning with extended RDMA interface in real application is expected.

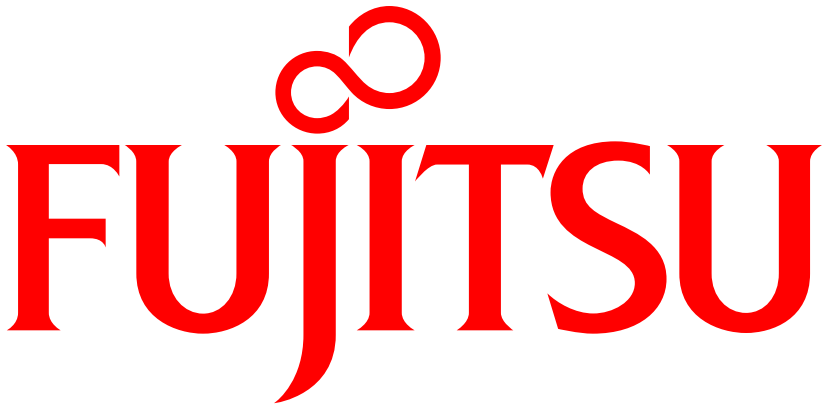
Summary

FUJITSU

- MPI scaled to 88K nodes and achieved LINPACK 10PFLOPS on K computer.
- Fujitsu MPI enhanced implementation of Open MPI to utilize the performance of Tofu interconnect.
- Application-specific tuning with extended interface shows good performance on Global RandomAccess.

- We thank Open MPI development team very much for providing very stable MPI software.

- We would like to make some contribution to Open MPI community. (undergoing)



shaping tomorrow with you