# Tracing Data Movements within MPI Collectives


Kevin A. Brown
brown.k.aa@m.titech.ac.jp

Jens Domke
domke.j.aa@m.titech.ac.jp

Satoshi Matsuoka
matsu@is.titech.ac.jp

Tokyo Institute of Technology



## Abstract

We propose extending common performance measurement and visualization tools to identify network bottlenecks within MPI collectives. By creating additional trace points in the Peruse utility of Open MPI, we track low-level InfiniBand (IB) communication events and then visualize the communication profile in Boxfish for a more comprehensive analysis. The proposed tool-chain is non-intrusive and incurs less than 0.1% runtime overhead with the NAS Parallel FT benchmark.


## Introduction

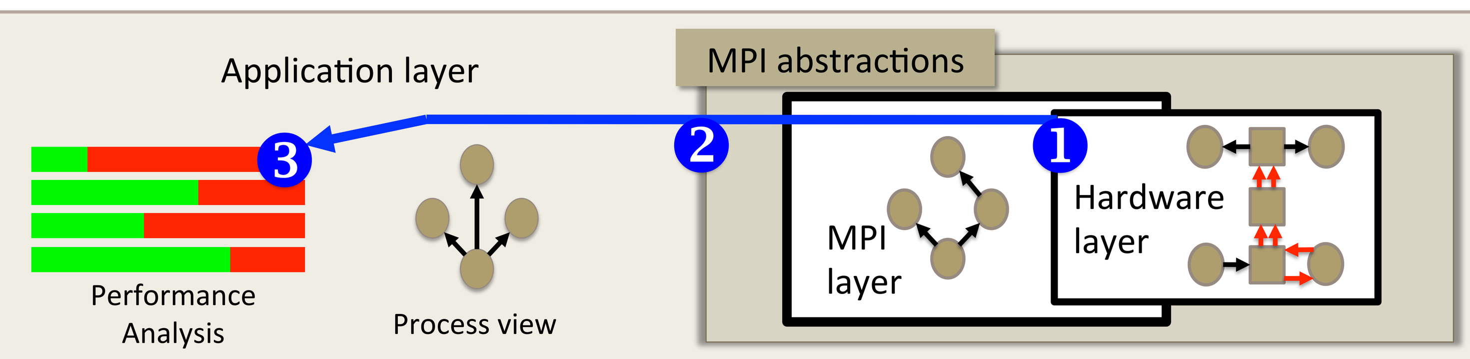
### Background

- High performance computing (HPC) systems are growing in physical size and complexity
- The Message Passing Interface (MPI) is used to perform inter-process communication on HPC systems
- Performance analysis is usually process-centric
  - typical done using the PMPI profiling interface in the application layer

### Motivations

- MPI's hardware abstraction hinders exposing performance from within the MPI library and from within the network layer
- Communication bottlenecks within collectives are hidden from PMPI-based analysis tools
- Hardware-centric analysis tools provide limited support for various network architectures and topologies

## Our Goals




❶ Expose MPI's internal performance in a portable manner
❷ Develop a lightweight profiler to capture low-level metrics
❸ Enable the visual, hardware-centric analysis of performance metrics

## Design and Implementation

### ❶ Exposing Low-level Performance in Open MPI

We used the Peruse interface[1] in Open MPI

- Peruse defines an interface for exposing the internal performance of MPI libraries
- User-supplied callback functions can be attached to internal MPI events

We created a new Peruse event in Open MPI named **PERUSE_OPENIB_SEND**. This event:

- Represents points when data is sent over InfiniBand (IB) interfaces
- Can be queried, activated and used via the standard Peruse Interface

### ❷ Non-intrusively Collect Low-level Metrics

We created a non-intrusive profiler named ibprof, which:

- Uses the **PERUSE_OPENIB_SEND** event to aggregate messages sent from each local IB interface to each remote interface
- Writes the network communication profile to Open Trace Format (OTF) files
- Supports the profiling of all communication, specific collective(s), and specific code section(s)
- Can be joined to an application at runtime or at link time

### ❸ Visualizing Network Communication

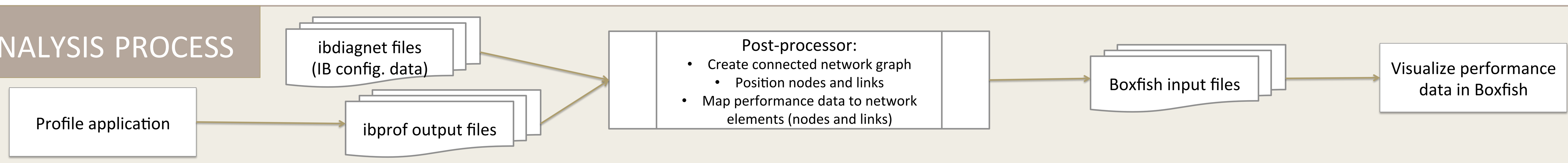
We used Boxfish[2], which:

- Is a hardware-centric analysis tool
- Allows visualizing performance data on network links and nodes (switches, compute nodes, etc.)

We created a visualization module for Boxfish, named **Fat Tree**. Our module:

- Is based on the **Torus 3D** module that is distributed with Boxfish
- Can natively visualize any 2D network topology and can be extend to support all topologies
- Uses bicoloured network links to accurately represent bidirectional traffic flow

## Analysis Process




## Results

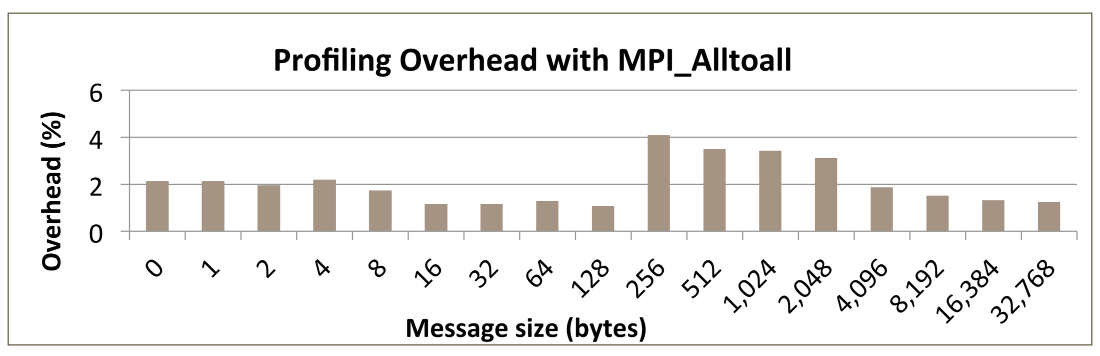
### Profiling Overhead

Experiment environment: TSUBAME-KFC

- Used 32 nodes, two IB FDR switches
- Open MPI 1.6.3 with our Peruse enhancements

Experiment 1: MPI_Alltoall microbenchmark

- 30 profiled and 30 un-profiled trials
- 20,000 collective calls (19,998 measured) per trial




- **Average communication overhead = 4.08%***

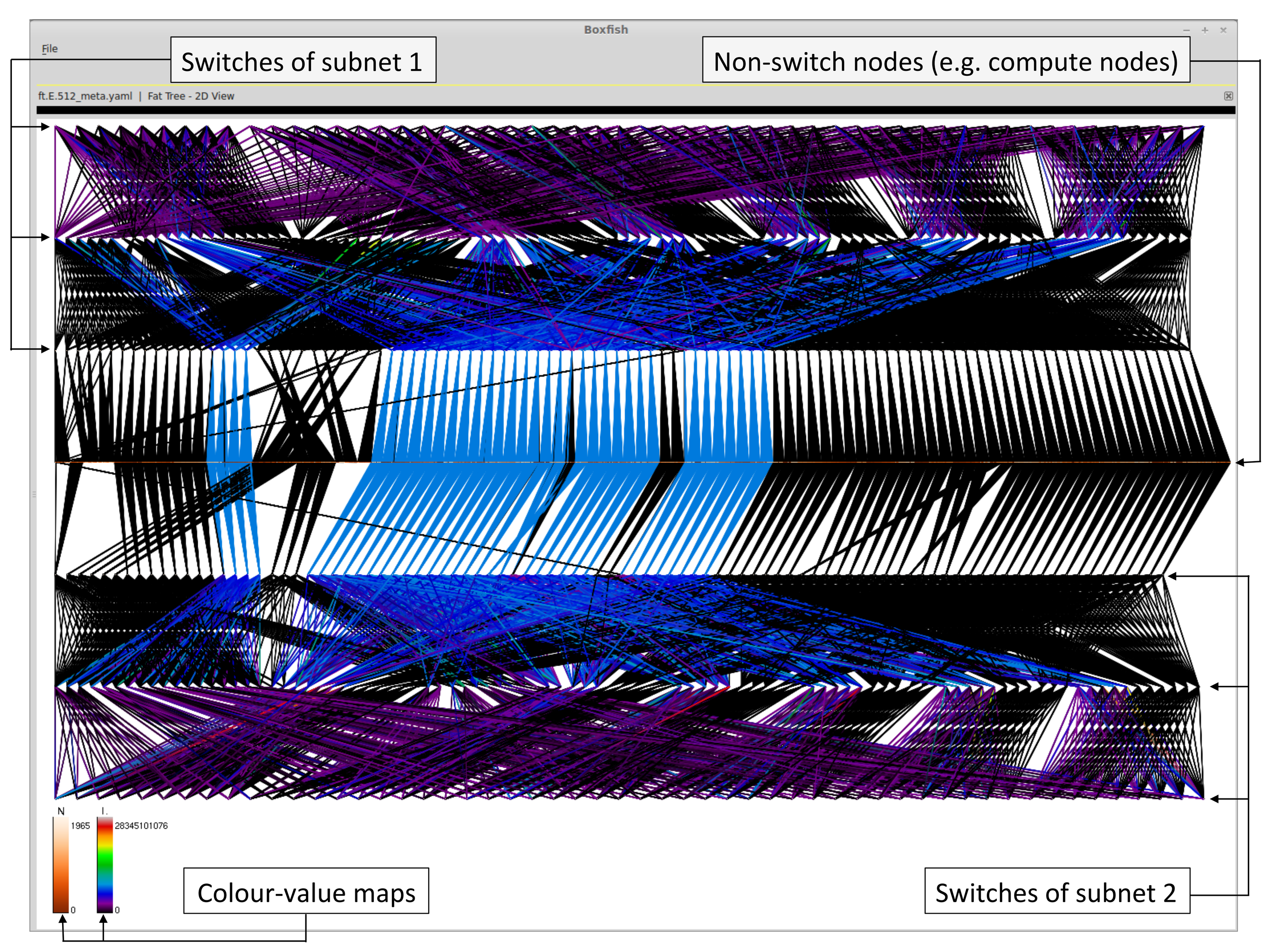
Experiment 2: NAS Parallel FT Benchmark[3]

- 30 profiled and 30 un-profiled trials
- Class C problem size
- **Average runtime overhead = 0.205%***

** Time to write OTF file, which was constant at ≈13ms, is not captured in our results*

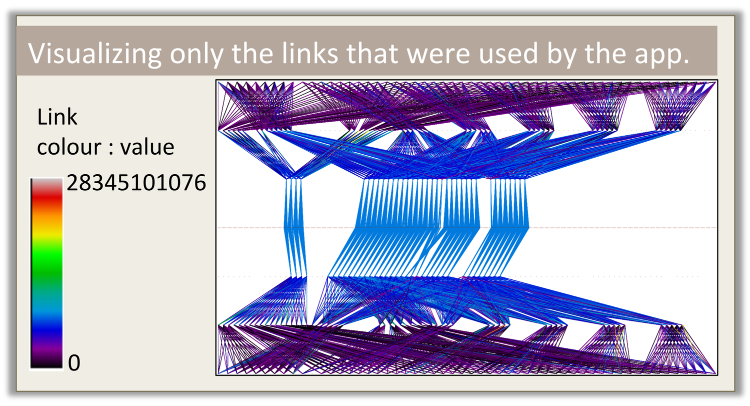
### Large-scale Visualization Using our **Fat Tree** Module

Visualizing the NPB FT benchmark (problem class: E) running on 512 nodes of TSUBAME2.5

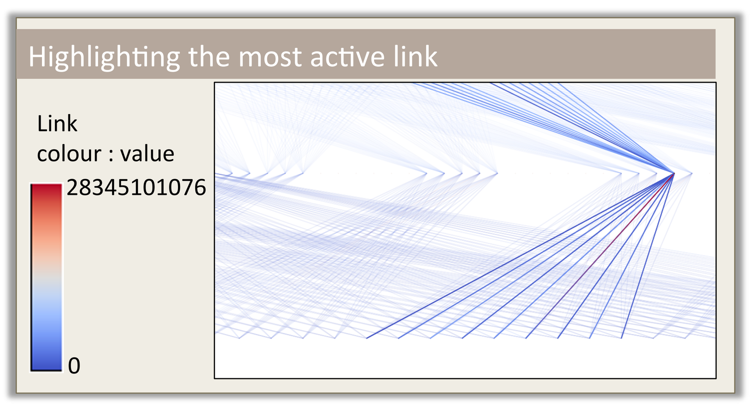



- Link colours represent application traffic (bytes)
- Node colours do not represent any metrics
- The visualization can be filtered, zoomed, and panned to locate network hotspots

Visualizing only the links that were used by the app.




Highlighting the most active link




## References


1. R. Keller, G. Bosilca, G. Fagg, M. Resch, and J. J. Dongarra. Implementation and Usage of the PERUSE-Interface in Open MPI. In *Proceedings of the 13th European PVM/MPI User's Group Meeting*, 2006.
2. K. E. Isaacs, A. G. Landge, T. Gamblin, P.-T. Bremer, V. Pascucci, and B. Hamann. Exploring Performance Data with Boxfish. In *Proceedings of the 2012 SC Companion: High Performance Computing, Networking, Storage and Analysis (SC'12)*, 2012.
3. D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga. The NAS Parallel Benchmarks. Technical Report RNR-94-007, NASA Ames Research Center, Mar 1994.


*EuroMPI/ASIA `14, Sep. 9-12 2014, Kyoto, Japan*