

# Programmability and Performance: OpenMP Implementation on Cluster Systems

Jie Cai

## Research Goal

The goal of this project is to provide a software platform that allows user using an easy programming model to achieve reasonable performance for some parallel applications on cluster systems. The sub-goals for the projects are :

- Evaluate the performance of the existing cluster-enabled OpenMP implementations.
- Improve the performance of the existing cluster-enabled OpenMP implementations on cluster systems.

## OpenMP on Cluster Computing?

The dominant programming model on cluster systems is message passing interface (MPI).

- **Problem of this programming model is it is too hard to program for most cluster users.**

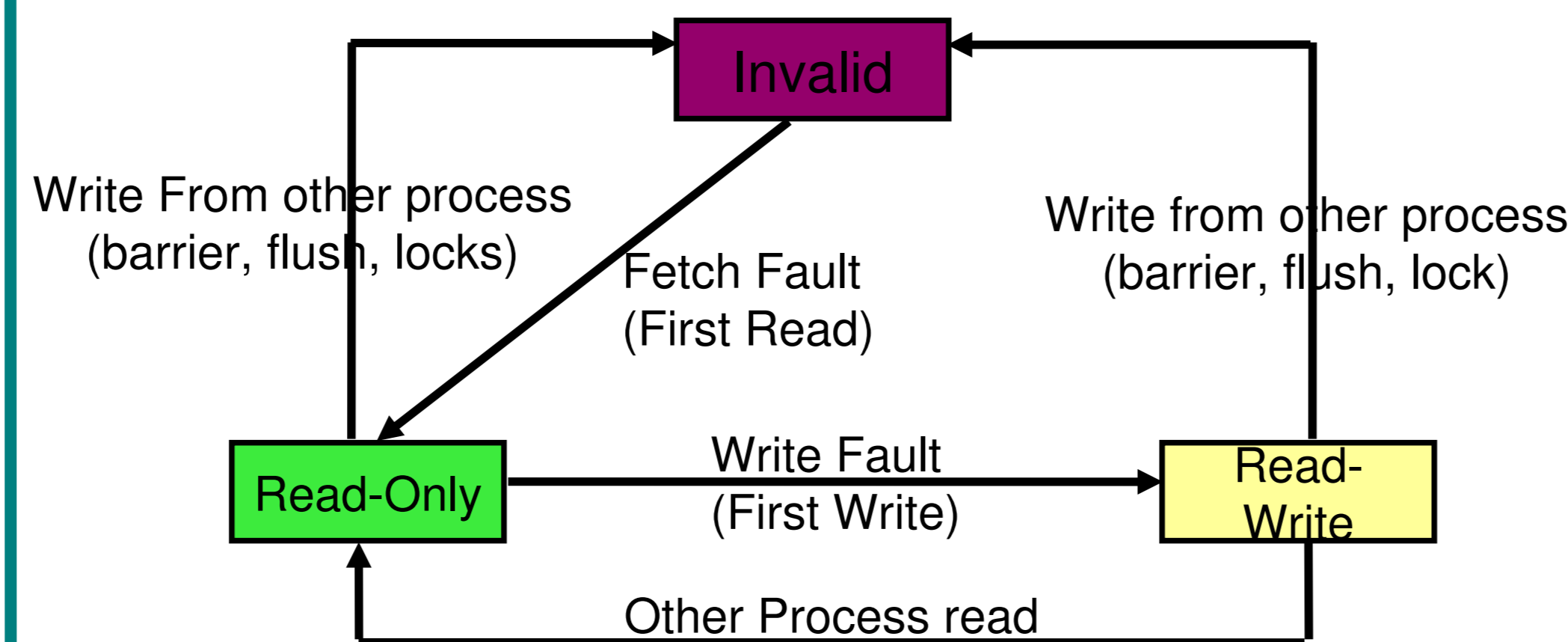
**Idea:** Using easy shared memory programming model, *OpenMP*, over clusters to achieve good programmability.

**Attempts:** utilizing software distributed shared memory systems (sDSM) with OpenMP programming model. This system is known as **Cluster OpenMP systems**.

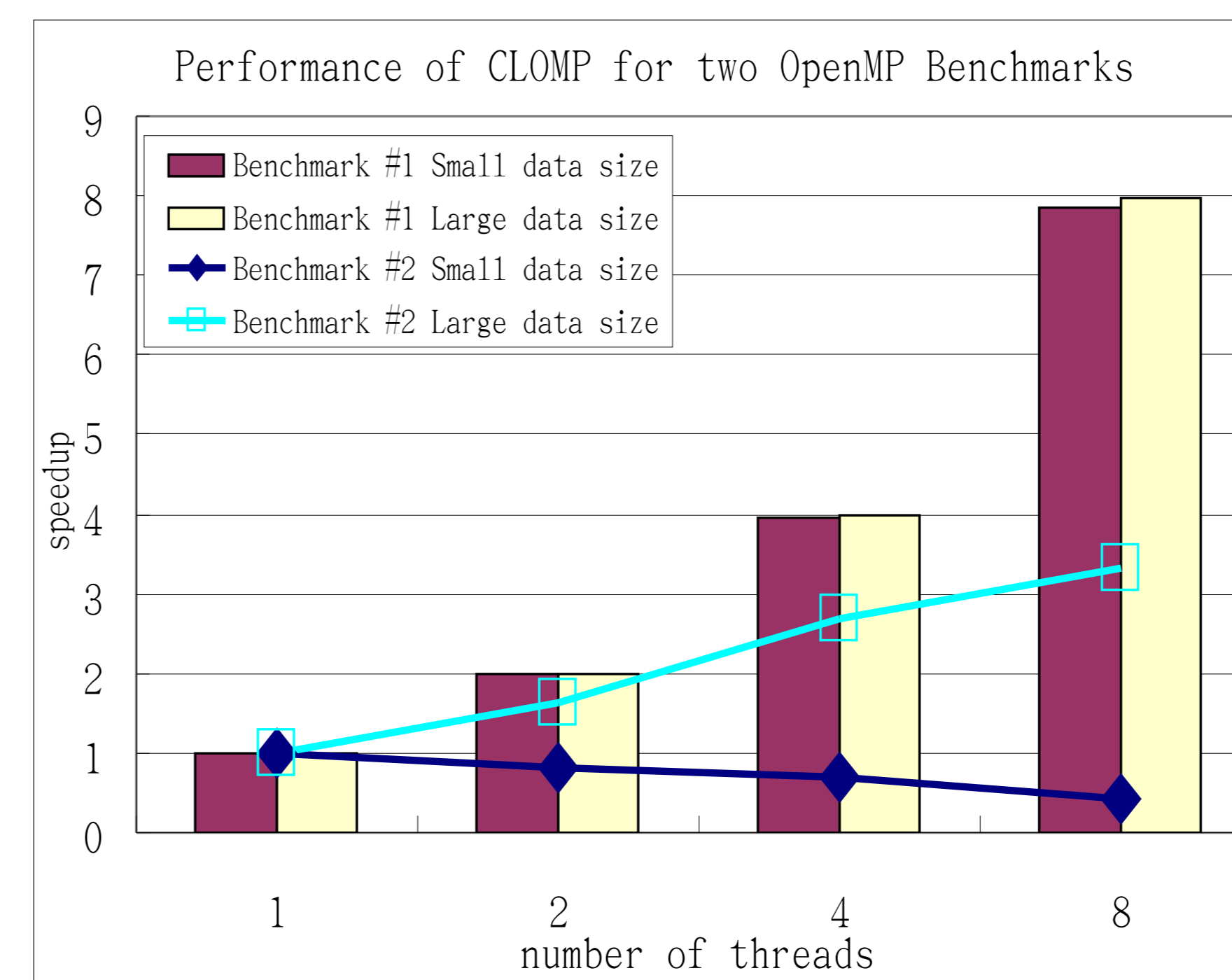
## Cluster OpenMP Performance

Most currently active cluster OpenMP systems are page-based system, such as Intel Cluster OpenMP (CLOMP).

The shared memory space is managed in fixed block of size, called page. Each node of the cluster maintained a local view of all shared pages which is kept consistent via page fault detecting and servicing.



The page fault service (sending page) is the major overhead of the cluster OpenMP systems. We evaluated the performance of cluster OpenMP systems via running two OpenMP benchmarks with CLOMP on an 8 node AMD dual core cluster.



## Performance analysis:

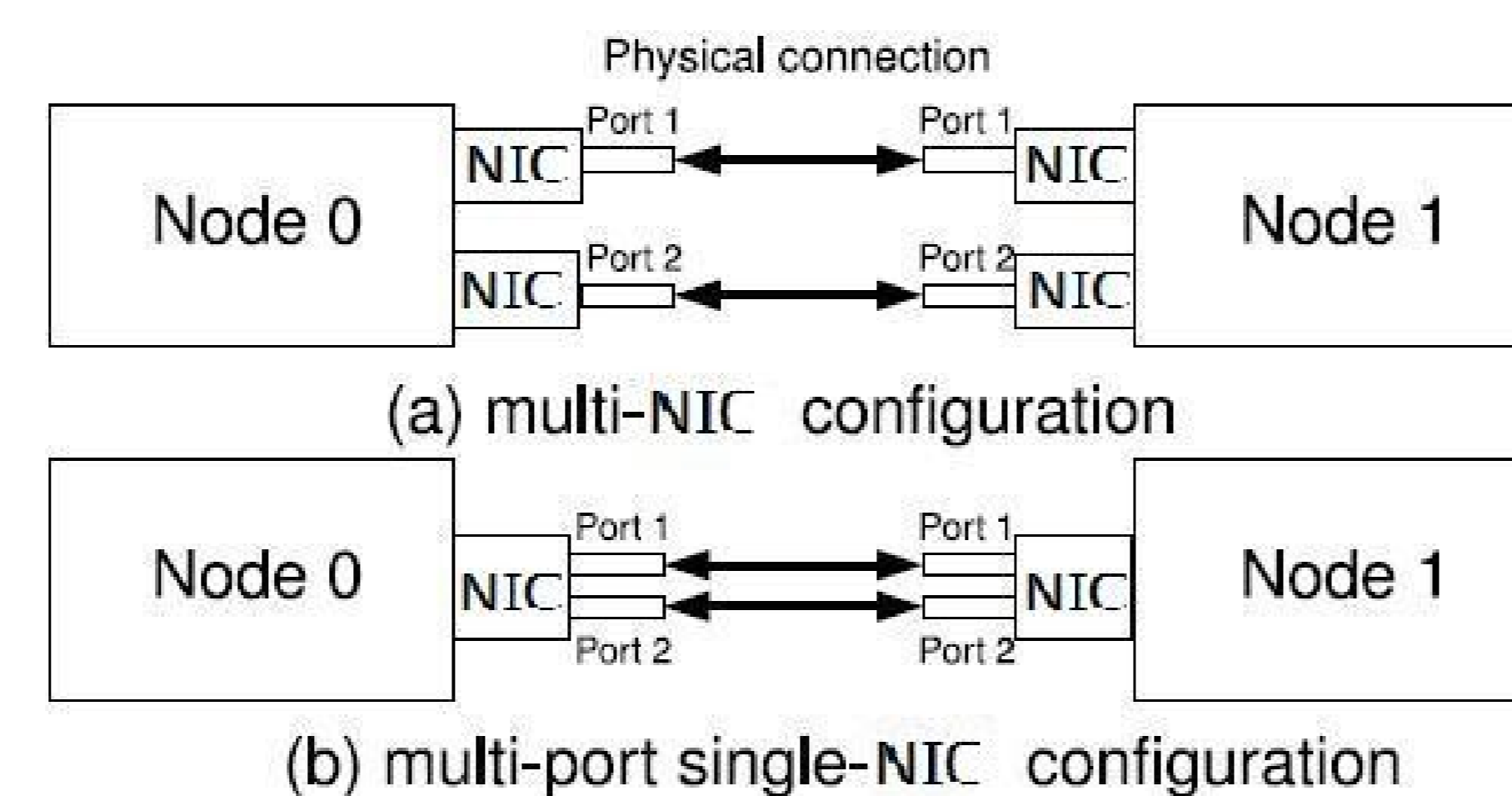
- Benchmark #1: Good performance!
  - ✓ Due to almost no page faults.
- Benchmark #2: Slowdown for small data size and relative better speedup for large data size.
  - ✗ Larger number of page faults;
  - ✗ Transferring page dominates the performance of small problem size;
  - ✓ Computation dominates the performance of large problem size.

**Reducing the page fault servicing cost is the key to improve performance of cluster OpenMP systems.**

## Utilizing Multirail Network to Improve the Performance

How to reduce the cost of page transfers? The major issue is the network communication efficiency.

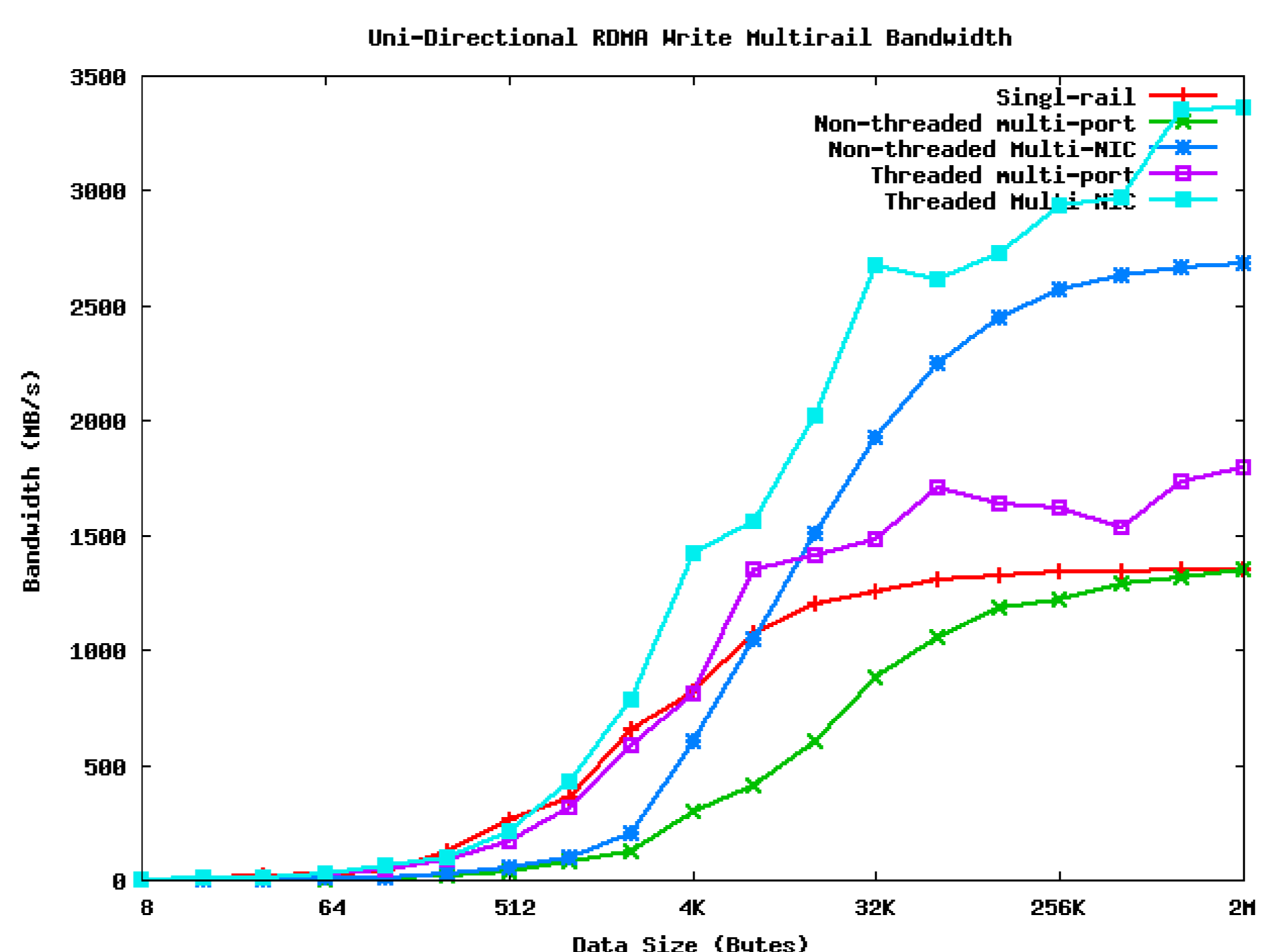
**SOLUTION:** Utilizing a "multirail network" is a possible solution.



The wide use of multi-core machines brought more variety. It is important to find out the best way to achieve the bandwidth benefit of multirail networks.

Two different approaches are designed to explore the network bandwidth, non-threaded and threaded approaches.

*In general, threaded approach with multi-NIC configurations shows best improvement.*



## Conclusion and Future Work

We evaluated the performance of current cluster OpenMP systems, and analyzed the major overhead of the current systems. **The first research goal is achieved!**

Two approaches, non-threaded and threaded, to achieve more efficient network communications over multirail networks are designed. **The second research goal is partially done!**

In the near future, we will design multirail implementation for CLOMP and utilize pre-fetch to aggregate messages for the better multirail performance.



OpenMP: EASY! include header and insert ONE line directive!!  
`#pragma omp parallel for default (shared) reduction(+:s)`  
`for(i = 0; i < N; i++)`  
`s += a[i];`

MPI: I need to scatter the data to different node first, then calculate the sub-sum at each node, then doing a reduction to sum up the total.  
`MPI_Scatter(sendbuf, N/nprocs,`  
`MPI_INT, recvbuf, N/nprocs,`  
`MPI_INT, 0, comm);`  
`for(i = 0; i < N/nprocs, i++)`  
`ss += rbuf[i];`  
`MPI_Reduce(&ss, &ss, 1,`  
`MPI_INT, MPI_SUM, 0, comm);`

MPI: Ooops, I am out of space!! --!

How to parallelize a for loop?  
`for(i = 0; i < N; i++)`  
`s += a[i];`

Why OpenMP ???

Supervised by Dr. Peter Strazdins and Dr. Alistair Rendell, in co-operation with Intel.



Computer Systems Research Group  
 College of Engineering & Computer Science