

# Performance Evaluation of Intel Cluster OpenMP

Speaker: Jie Cai

Collaborator: H'Sien Jin Wong

Supervisor: Dr. Peter Strazdins and A/Prof. Alistair Rendell

Department of Computer Science  
The Australian National University

APAC07 Student Forum

8 October 2007



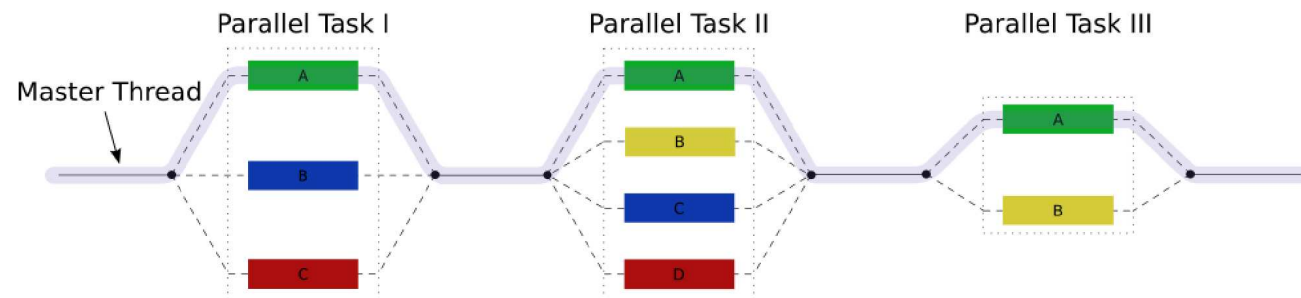
# Outline

- What is OpenMP?
- What is Cluster OpenMP?
- Performance Evaluation
- Summary
- Future Work

# What is OpenMP?

OpenMP stands for **Open** specifications for **M**ultiple **P**rocessing.

- shared memory parallelism programming model.
  - Fork-join approach to create threads and execute in parallel.
  - Synchronization between threads
    - \* Locks, critical sections, barriers, flushes, etc..
  - Creation variables private to a thread and shared among threads
- Easy to program

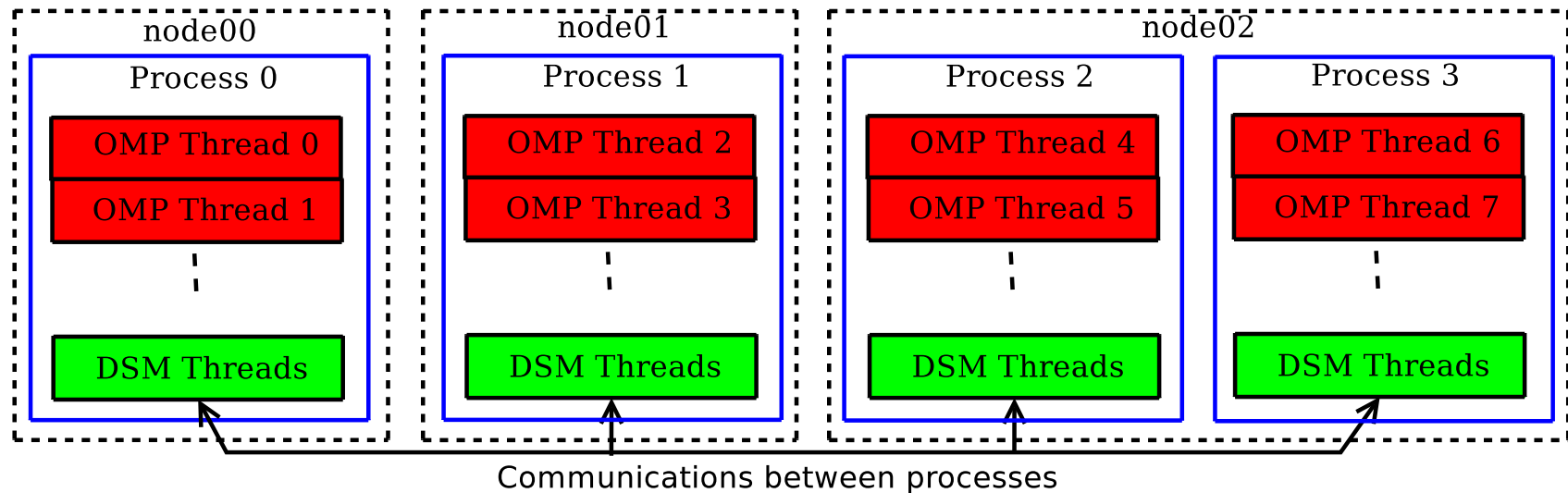


Graph source: [www.wikimedia.org](http://www.wikimedia.org)

# What is Cluster OpenMP (CLOMP)?

- Compiler extension and a run-time library that supports running an OpenMP program on a cluster.
  - Released as a part of Intel Compiler at May 2006.
  - Implemented using a page based software Distributed Shared Memory system (DSM).
- DSM mimic shared memory environment on physical distributed memory system.
  - Relies on underlying virtual memory system.
  - Data that shared across multiple threads placed on certain pages which each process has a copy of them.
  - mprotect system call is used to determine whether the page accessed is modified by some other processes.
- Advantages:
  - Extending OpenMP program on normal clusters.

# Running CLOMP



- OpenMP Threads

- Master thread, which starts executing the CLOMP program
- Worker threads, the rest of the threads started in a parallel region

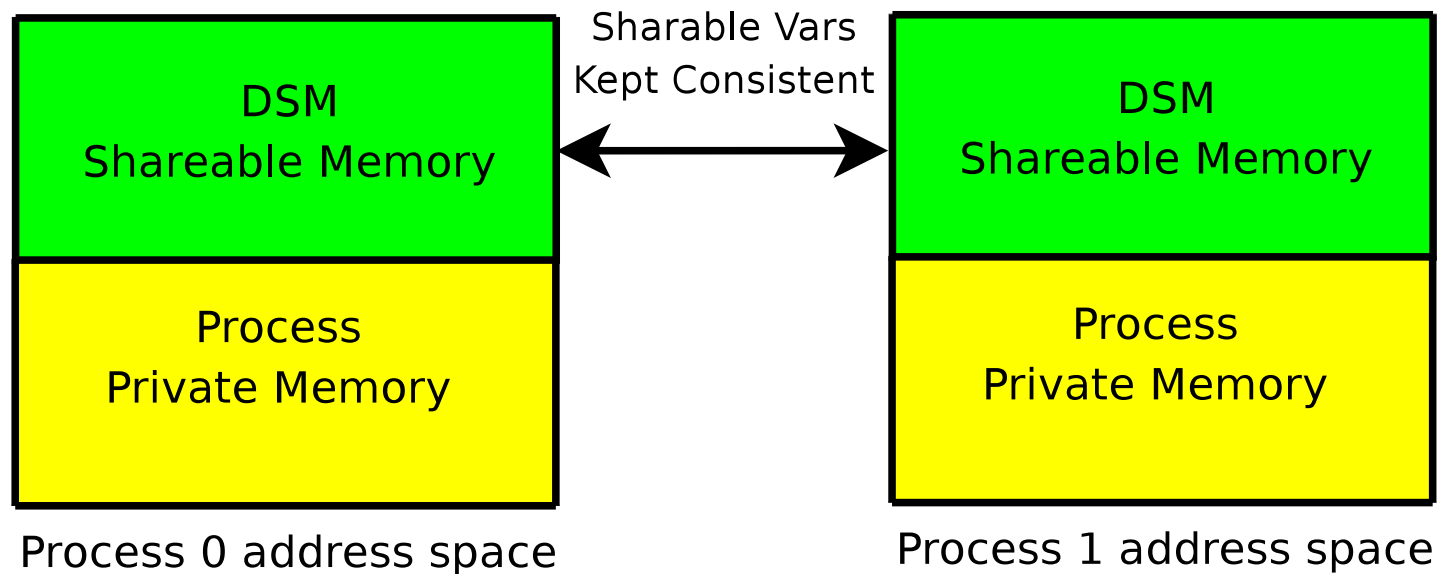
- DSM Support Threads

- handle asynchronous communication for processes, activated by messages that come to the process from other processes
- heartbeat thread to detect fault

## CLOMP vs. OpenMP

The sharable directive is introduced in CLOMP to indicate that a variable that would be accessed by multiple OpenMP threads from private variables.

- `kmp_sharable_malloc` is the memory allocation call to allocate memory for sharable variables.
  - Standard OpenMP data defaults to be shared.
  - It would not be difficult to implement this in cluster, therefore an explicit tag `sharable` is required.



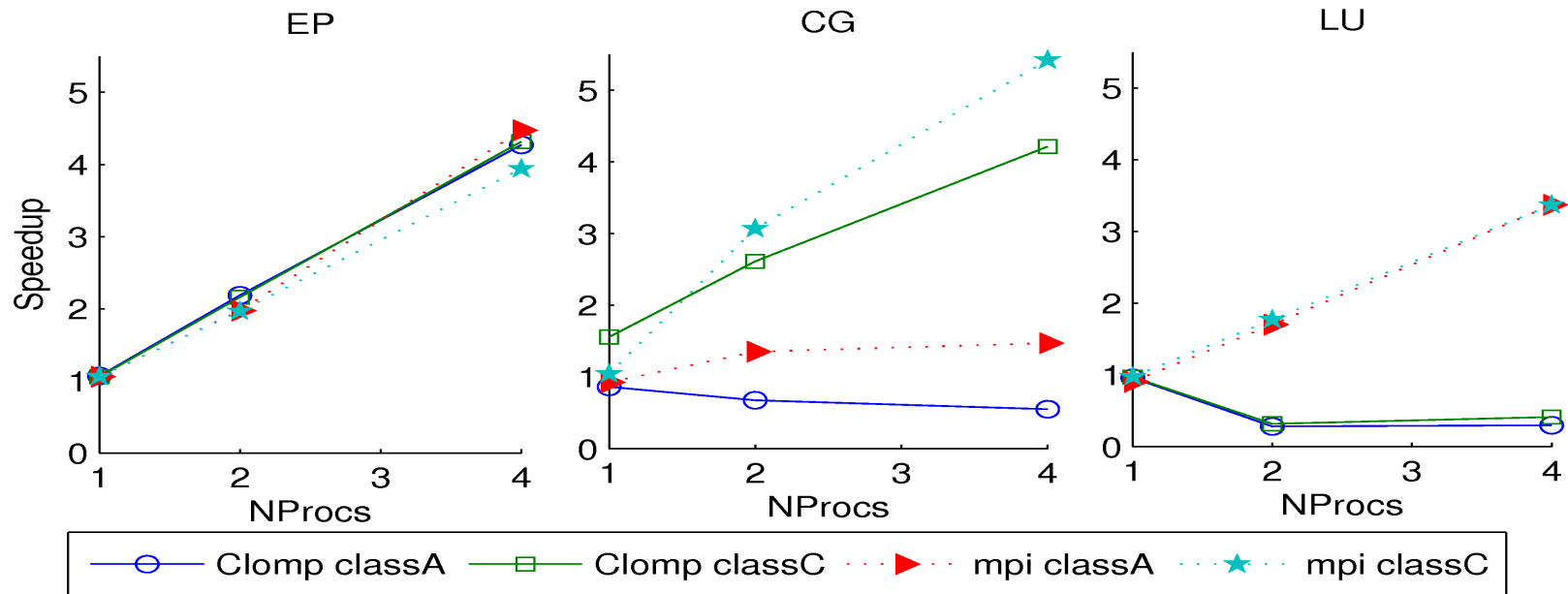
# Performance Evaluation

- NAS Parallel Benchmark (NPB) OpenMP and MPI version are run on CLOMP version 9.1. Class A and C of the EP, CG and LU are used. Class A refers to smaller problem size.
  - EP is an **E**mbarassingly **P**arallel benchmark;
  - CG is a **C**onjugate **G**radient method to compute an approximation to the smallest eigenvalue of a large, sparse, unstructured matrix;
  - LU refers to finite-difference discretization of Navier-Stokes equation in 3-D and then use symmetric successive over-relaxation (SSOR) method to solve a resulting seven-block-diagonal system by splitting it into block **L**ower and **U**pper triangular systems.

Table 1: Hardware and Cluster Specification

Cluster	# of nodes	CPU	L2 cache	Network	SPEC FP
AMD	8	Dual core 2.2GHz	512KB per core	GigaBit Ethernet	1476
Intel	4	Core2 2.13GHz	2MB shared	GigaBit Ethernet	2168

## NPB Comparison for CLOMP and MPI on AMD cluster



**Remark:** The speedup is calculated based on the elapsed time of NPB-OMP version compiled without -openmp flag.

- EP: CLOMP shows better speedup on class C than MPI
- CG: CLOMP have no speedup on class A, but have speedup on class C (although still worse than MPI).
- LU: CLOMP have no speedup because of a large number of synchronization operations (flush, barriers).

## Performance on Heterogeneous Cluster

The following equation represents the best-case performance based on load balancing according to processor speed.

$$T_{Best} = \frac{1}{\frac{f}{T_i} + \frac{1-f}{T_a}} \quad (1)$$

where  $T_i$  and  $T_a$  represent the elapsed time on the Intel and the AMD cluster with same number of nodes as mixed cluster,  $f$  stands for the fraction of Intel (fast) nodes.

Table 2: Heterogeneity ability test on class C NPB EP and CG benchmarks for Elapsed time (sec)

	number of threads [ $nodes \times threads$ ]					
	$2 \times 2$		$4 \times 2$		$8 \times 2$	
	Mixed(% over $Est_{best}$ )	$Est_{best}$	Mixed(% over $Est_{best}$ )	$Est_{best}$	Mixed(% over $Est_{best}$ )	$Est_{best}$
EP	120.4%	92.0	119.6%	46.4	123.1%	22.9
CG	183.2%	428.6	151.8%	387.1	126.5%	330.5

- There is no clear pattern for the observed performance related to predicted best case performance, due to OpenMP does not support scheduling on cluster.
- Developing a heterogeneous cluster scheduler might be an interesting research topic in my PhD thesis.
- Another interesting topic is remote memory access (RMA) for CLOMP to utilize those functions provided by some network technology, such as Infiniband.

## Summary

- CLOMP provide a easier program paradigm compared to MPI.
- CLOMP performs better on large problem size (coarse-grained) application.
- CLOMP currently suits for the program using synchronization and data sharing sparingly.
- Performance of CLOMP might be improved with support from faster network interconnection, such as Infiniband.

## Future Works

- Heterogeneous load balance issue.
- Remote memory access issue.

# Questions?

## Acknowledgment

This research is funded by ARC Linkage  
Grant LP0669726 with support from

Alexander Technology and Intel Corporation.