

Min Si, Yutaka Ishikawa (Advisor) from University of Tokyo, Pavan Balaji (Advisor) from Argonne National Laboratory

My doctoral research focuses on exploiting the capabilities of *massively threaded many-core architectures* on widely used MPI implementations.

I will investigate the characteristics of MPI and develop various design strategies that allow MPI implementations to *perform efficiently for different kinds of user programming models*.

The research plan is broadly divided into three related pieces for various thread modes.



## Many core

- Massively parallel environment
- Intel® Xeon Phi co-processor
- Blue Gene/Q

## MPI\_THREAD\_FUNNELED

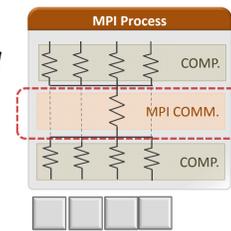
- Multiple user threads
- Only master thread is allowed to make MPI calls

## MPI\_THREAD\_SERIALIZED

- Multiple user threads
- Only a single thread is allowed to make MPI calls

### Problem

- Threads are **IDLE** during MPI calls

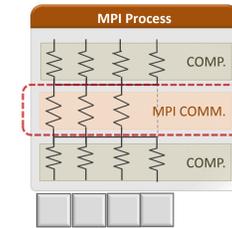


## MPI\_THREAD\_MULTIPLE

- Multiple user threads
- All the threads are allowed to make MPI calls by sharing MPI resources

### Problem

- Overhead to keep **internal state consistent** (lock, memory barrier etc.) between hundreds of threads

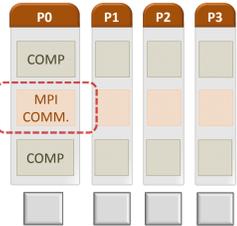


## MPI\_THREAD\_SINGLE

- MPI process per core, no threads

### Problem

- Core is **IDLE** if the MPI process is in blocking wait during MPI calls.



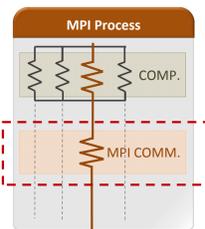
## Research I. Internal Multithreading in MPI

### Problem: Idle Resources during MPI Calls

#### Funneled Mode

```
#pragma omp parallel
{ /* User Computation */ }

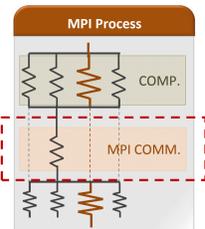
MPI_Function ();
```



#### Serialized Mode

```
#pragma omp parallel
{ /* User Computation */ }

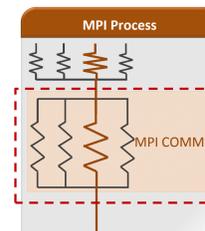
#pragma omp single
MPI_Function ();
```



### Solution: Sharing Idle Threads with Application inside MPI

```
#pragma omp parallel
{ /* User Computation */ }

MPI_Function() {
  #pragma omp parallel
  { /* Internal Processing */ }
}
```



### Challenges

- The **number of IDLE threads** required by some parallel algorithms is **UNKNOWN**.
- **Nested parallelism**
  - Creates **new Pthreads**
  - **Offloads thread scheduling to OS, causes threads OVERRUNNING issue.**

### OpenMP Runtime Extensions

- Getting the number of idle threads

```
#pragma omp parallel
#pragma omp single
{
  #pragma omp parallel ¥
  num_threads( get_num_idle_threads() )
  { ... }
}
```

Threads waiting in the barrier are doing **SPIN LOOP** until timeout!

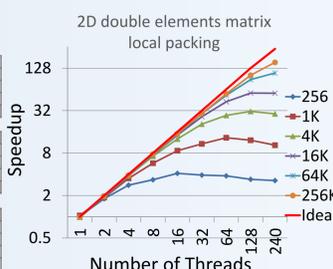
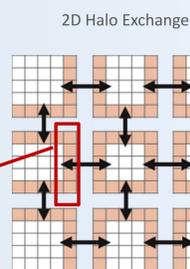
- Forcing waiting threads to enter in a passive wait immediately

```
#pragma omp parallel
#pragma omp single
{
  set_fast_yield( 1);
  #pragma omp parallel
  { ... }
}
```

### MPI Internal Parallelism

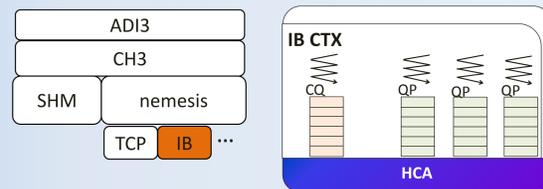
- Derived Data Type Packing Processing

- MPI\_Pack / MPI\_Unpack
- Communication using Derived Data Type
  - Transfers non-contiguous data
  - Packs / unpacks data internally
- Parallelism
  - Packs/unpacks non-contiguous elements in parallel

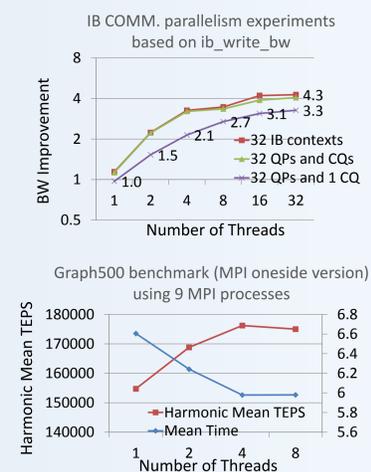


### Netmod

- Parallel InfiniBand Communication

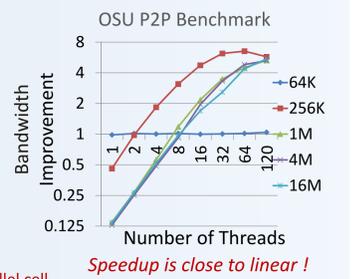
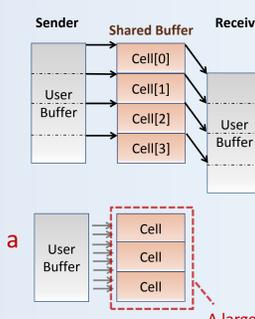


- Onside applications using parallel IB netmod
  - Large amount of small non-blocking RMA operations
  - Wait for all operations to complete before returning from the second synchronization call
  - Parallelize messages sent to different targets

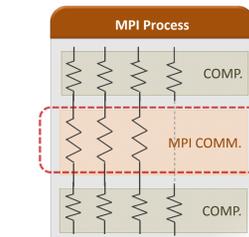


### Intra-node Large Message Communication

- Original sequential algorithm
  - Shared user space buffer
  - Pipelining data movement between sender and receiver
- Parallelism
  - Reserves many available cells as a large cell
  - Moves this large cell in parallel

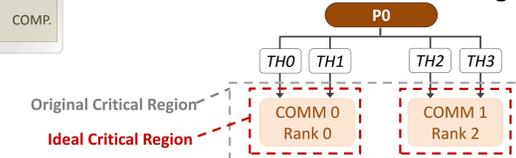


## Research II. Fine-Grained Consistency Management in MPI

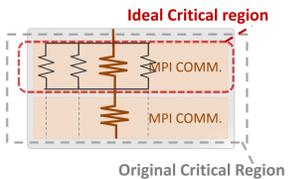


**PROBLEM** Heavy Synchronization & Memory Barrier between hundreds of threads

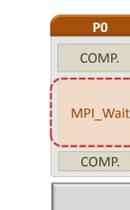
- Small collection of threads sharing



- Runtime dynamic thread safety reduces the scope of Multithreading region

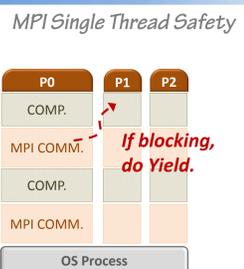


## Research III. Fiber level MPI processes



**PROBLEM** Core is IDLE while blocking wait!

- How to keep the core **always BUSY** ?
  - Multiple fiber level MPI processes per core
  - Fine-grained scheduling between these fibers



\* This research is based on FG-MPI. We plan to improve various issues based on the current design, such as the task load balancing, and the critical section for shared resources.

This work has been partially supported by the CREST project of the Japan Science and Technology Agency (JST), and the National Project of MEXT called Feasibility Study on Advanced and Efficient Latency Core Architecture. It has been also supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357