On the Energy Efficiency of MPI Intra-node Communication

Hyun-Wook Jin    Keon-Woo Kim
System Software Laboratory
Dept. of Computer Science and Engineering
Konkuk University
jinh@konkuk.ac.kr
MUG’20

• Kernel-Level Support for MPI Intra-Node Communication (Post-LiMIC2): Project Overview
  – Power efficiency

<table>
<thead>
<tr>
<th></th>
<th>Blocking APIs( (\text{MPI_Send, MPI_Recv}) )</th>
<th>Nonblocking APIs( (\text{MPI_Wait, MPI_Waitall}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eager</td>
<td><img src="true" alt="Eager" /></td>
<td></td>
</tr>
<tr>
<td>Rendezvous</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

– Skew tolerance
– Better manageability
• Kernel-Level Support for MPI Intra-Node Communication (Post-LiMIC2): Project Overview
  – Power efficiency

<table>
<thead>
<tr>
<th></th>
<th>Blocking APIs (MPI_Send, MPI_Recv)</th>
<th>Nonblocking APIs (MPI_Wait, MPI_Waitall)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eager</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Rendezvous</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

– Skew tolerance
– Better manageability
Contents

• Background and motivation

• A framework for energy-efficient MPI
  – Overall design
  – Performance evaluation

• Concluding remark
CPU Scheduling in HPC

- **One-to-one mapping between processes and CPU cores**
  - In HPC systems, the runtime solely dedicates a CPU core to each parallel process
  - Parallel programming libraries are optimized on the assumption that a parallel process occupies an entire CPU core

- **MPI libraries**
  - Perform *busy-waiting* to check the completion of outstanding communications
  - Using busy-waiting is acceptable and can provide low latency, as a CPU core runs only a single process
Busy-Waiting and Energy Consumption

• The longer the busy-waiting time, the higher the energy consumption

• Causes of longer busy-waiting time
  – Nonuniformity of network latency
  – Asynchronous semantics in communications
  – Load imbalance
Busy-Waiting in **MPI_Bcast**

- **Busy-waiting in MPI_Bcast**
  - Experiment system
    - 16-node KNL cluster
      - Intel Xeon KNL 7290 1.50GHz
      - InfiniBand EDR 100Gb/s
  - Measurement results
    - Busy-waiting time is quite random regardless of the number of processes
    - A real application may suffer from a larger busy-waiting time

![Graph showing busy-waiting time vs number of processes](image)
Energy Efficient MPIs

• **Dynamic Voltage and Frequency Scaling (DVFS)**
  – Provides different levels of voltage and frequency for operating processors
  – P-states (ACPI)
    • P0: Maximum power and frequency
    • Pn: Less than P(n−1) voltage and frequency scaled

• **Core-Idling**
  – Turns off hardware components of idle cores
  – C-states (ACPI)
    • C0: Active
    • C1: Halt
    • C2: Stop-clock
    • C3: Sleep
Energy Efficient MPIs

- **Decision policies**
  - Determine when and which energy-saving mode to enter

  - **EAM, SC’15**
    - Estimates the duration of MPI and communication phases based on temporal execution patterns
    - Interrupt-based core-idling
  
  - **COUNTDOWN, ToC 2021**
    - Intercepts MPI calls and uses a time-out strategy for DVFS
    - Countdown Slack, TPDS 2020

  - **EAR/EARL, Cluster 2020**
    - Detects iterative regions and maintains application signatures by intercepting MPI calls
    - Decides the CPU frequency based on an energy model
Motivation

• Decision policies can gain much insight if the MPI library provides internal information/features

• MPI library has a better idea of when to trigger the decision algorithm
  – Separation of mechanism and policy
A FRAMEWORK
MPI Communication Channels

• **Inter-node communication channels**
  – Interfaces for InfiniBand, Omni-Path, Ethernet, ...

• **Intra-node communication channels**
  – Shared memory channel
    • Moves messages from source to destination via a shared memory region
    • Small messages based on eager protocol
  – Memory mapping channel
    • Directly moves messages from source to destination without intermediate copies by means of a kernel level support
    • Large messages based on rendezvous protocol
    • CMA, LiMIC2, XPMEM, ...
Our Goal

• We aim to provide a framework that efficiently supports energy-aware decision policies over multiple MPI communication channels

• First step
  – Intra-node communication channels
    • Shared memory
    • Memory mapping
  – Decision policy
    • Energy-saving mode: core-idling
    • Static: busy-waiting -> core-idling
Simple Policy of Core-Idling

Sender

Receiver

Call MPI_Send

Calls MPI_Recv

MPi_Recv returns

Busy-Waiting

Message

Shared memory
Memory mapping

Suspend

Resume

Calls MPI_Recv

MPi_Recv returns

SLEEP

RUN
Suspending and Resuming Points

- **Shared memory channel**
  - Suspending points
    - When a shared buffer is not available
    - When there is no received message
  
  ![Diagram of shared memory channel with suspend and resume points]

- Resuming points
  - When a shared buffer becomes available
  - When a new message is arrived

![Diagram of shared memory channel with suspend and resume points]
SUSPENDING AND RESUMING POINTS

- **Memory mapping channel**
  - Suspending points
    - When there is no corresponding control message of rendezvous protocol
  - Resuming points
    - When a control message of rendezvous protocol arrives
Implementation Methodologies

• **CPU dependent implementation**
  – Assembly instructions (e.g., `mwait`)

• **CPU independent implementation**
  – Timers: only for coarse-grained controls
  – Semaphores: deadlock-prone
  – Signals: lossy
    • Easy to support callback functions
    • Flexible enough to support the inter-node communication channel
    • Able to leverage existing decision policies used in DVFS and core-idling approaches
PERFORMANCE EVALUATION
Point-to-Point Communication

• **Experiment system**
  – Intel Core i7-8700 3.20GHz processor (6 cores)
  – Linux kernel v.5.3.7
  – MVAPICH2

• **Measurement tools**
  – OSU microbenchmark
  – RAPL

---

1: `procedure` OSU_LATENCY
2: Start measuring Latency
3: Start measuring Power consumption
4: `for` Number of iterations `do`
5: `if` rank is 0 `then`
6: `Delay` for N micro seconds
7: `MPI_Send`(to rank 1)
8: `MPI_Recv`(from rank1)
9: `end` `if`
10: `if` rank is 1 `then`
11: `MPI_Recv`(from rank 0)
12: `Delay` for N micro seconds
13: `MPI_Send`(to rank 0)
14: `end` `if`
15: `end` `for`
16: End measuring Power Consumption
17: End measuring Latency
18: `end` `procedure`
Point-to-Point Communication

- **Eager protocol**
  - Message size: 8KB
  - Energy consumption: 93% saving (10,000us)
  - Latency: 43% increase (0us)
Point-to-Point Communication

- **Rendezvous protocol**
  - Message size: 8MB
  - Energy consumption: 86% saving (10,000us)
  - Latency: 2~7% increase

[Graphs showing latency and energy consumption for different delay times with labels: BusyWaiting, Core-idling, Core-idling + BusyWaits(500).]
NAS Parallel Benchmarks

• **Experiment system**
  – Two Intel Xeon Ivy Bridge 2.8GHz processors (10 cores x 2)
  – Linux kernel v.5.3.7
  – MVAPICH2

• **Measurement tools**
  – NPB Class C
  – RAPL
  – cpupower
NAS Parallel Benchmarks

• 16 processes (# of processes ≤ # of cores)
  – Energy consumption: 12% saving (MG)
  – C6 state usage time: 17% increase (MG)
  – Execution time: 2% degradation (IS)
NAS Parallel Benchmarks

- **32 processes (# of processes > # of cores)**
  - Energy consumption: 91% saving (CG)
  - C6 state usage time: 17% increase (CG)
  - Execution time: 91% improvement (CG)
CONCLUDING REMARK
Conclusions

• A framework for better supports for energy-aware decision policies over multiple MPI communication channels

• Signaling-based framework with core-idling policy
  – Identified when the decision policy should be triggered on intra-node communication channels
  – Presented preliminary implementation in MVAPICH2
    • Could reduce the energy consumption of NPB Class C
      – Up to 12% in an undersubscribed case
      – Up to 91% in an oversubscribed case
Future Work

• **Additional analyses**
  – Various CPU architectures
    • Large-scale NUMA
      – Eight Intel Xeon E7 processors (24 cores x 8)
    • Many-core CPUs
      – Intel Xeon Phi KNL (72 cores)
      – ARMv8 (32 cores)
  – Various applications
    • QAND
    • Etc.
Future Work

• Integration with MVAPICH2-EA
Thank You!

Ministry of Science and ICT