The LiMIC Strikes Back

Hyun-Wook Jin
System Software Laboratory
Dept. of Computer Science and Engineering
Konkuk University
jinh@konkuk.ac.kr
Contents

• MPI Intra-node Communication
• Introduction of LiMIC
• Optimization for Huge Pages
• Conclusions and Future Work
## Multi/Many-core Processors

<table>
<thead>
<tr>
<th>Xeon 5100 Series (Woodcrest)</th>
<th>Xeon 5500 Series (Gainestown)</th>
<th>Xeon 5600 Series (Westmere-EP)</th>
<th>Xeon E5-2600 Series (Sandy Bridge-EP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Xeon E5-2600 v2 Series (Ivy Bridge-EP)</th>
<th>Xeon E5-2600 v3 Series (Haswell-EP)</th>
<th>Xeon E7 v4 Family (Broadwell)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>18</td>
<td>24</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Xeon Phi X100 Series (Knights Corner)</th>
<th>Xeon Phi 7200 Series (Knights Landing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>72</td>
</tr>
</tbody>
</table>
MPI Intra-node Communication

• **Loopback**
  - NIC and TCP/IP provide a loopback path
  - Two DMAs or two data copies + system calls

• **Shared Memory**
  - Communicate through a memory area shared between MPI processes
  - Two data copies
MPI Intra-node Communication

• Memory Mapping
  – Single data copy by means of memory mapping
    • Beneficial for large messages
  – LiMIC (Linux kernel module for MPI Intra-node Communication) is the first implementation of the memory mapping mechanism
History of LiMIC

• 1st Generation LiMIC
  – A stand-alone kernel module for intra-node comm.
  – Could improve the latency and bandwidth up to 71% and 313%, respectively
  – Internal queues had several scalability and stability issues

History of LiMIC

- **2nd Generation LiMIC (LiMIC2)**
  - Provided primitives to handle memory mapping
  - Removed in-kernel message queues
  - Could improve the latency and bandwidth up to 81% and 358%, respectively

History of LiMIC

• **MVAPICH2 with LiMIC2**
  – (Jun. 2009) LiMIC2-0.5 was publically released with MVAPICH2-1.4RC1
  – (Current) LiMIC2-0.5.6 is being released with the latest MVAPICH2 version
    • mvapich2-src]$ ./configure --with-limic2 [omit other configure options]
    • mvapich2-src]$ mpirun_rsh -np 4 -hostfile ~/hosts MV2_SMP_USE_LIMIC2=1 [path to application]
History of LiMIC

• LiMIC2 Has Inspired Several Studies
  – KNEM
    • Addressed a security issue
    • OpenMPI includes KNEM support
  – CMA
    • In-kernel implementation + New system calls
    • Default intra-node communication channel for large messages in MVAPICH2
No More Significant Performance Optimizations?

• MPI Point-to-Point Communication
  – LiMIC2 and CMA show almost the same performance
Overhead Breakdown

Per-page overheads occupy about 40%
Large Pages in x86-64

- **CR3**
  - **PML4E**
    - **PDPE**
      - **Page Directory Pointer Table**
        - **Page Directory**
          - **PDE**
            - **Page Table**
              - Normal Page
              - Normal Page
              - Normal Page
              - Normal Page

- **Page Map Level 4**
  - **Large Page**
    - **2MB**
      - **1GB**
    - **Large Page**
      - **4KB**
Large Pages in Linux

• **Persistent Huge Pages**
  – Provide a persistent pool for huge pages that are not split into normal pages
  – hugetlbfs
    • `~]$ echo [# of huge page] > /proc/sys/vm/nr_hugepages`
    • `get_huge_pages()` and `free_huge_pages()` in libhugetlbfs

• **Transparent Huge Pages**
  – Transparent to user applications in allocation and use
  – Can be fallen back to normal pages if necessary
Current Implementations

• **Performance with Huge Pages**
  – Using huge pages improves the bandwidth up to 37% with LiMIC2 and 35% with CMA, respectively
  – Most of benefits come from less number of TLB misses
Current Implementations

• Assume Normal Page
  – Offset calculation
    • Uses PAGE_SIZE
  – get_user_pages()
    • Returns the corresponding page structures for a given virtual address
Support for Huge Pages

• Checking If a Page is Huge
  – Persistent huge pages
  – Transparent huge pages

• Offset Calculation for Less Iteration
  – Uses HPAGE_SIZE for buffers backed by huge pages

• Efficient Usage of `get_user_pages()`
  – The first `page` structure is sufficient to map all pages that belong to the huge page
Performance Evaluation

• Experimental System
  – Intel KNL 7210
    • 64 cores
    • 16GB MCDRAM
      – Flat mode
    • Quadrant mode
  – 96GB DDR4

• Performance Measurement
  – OSU-microbenchmarks
    • osu_bw
    • osu_latency
    • osu_alltoall
Performance Evaluation

• Point-to-Point Bandwidth
  – Enhanced LiMIC2 could improve the bandwidth up to 44% with huge pages
Performance Evaluation

- **Point-to-Point Latency**
  - Enhanced LiMIC2 could reduce the latency up to 14% with huge pages
Performance Evaluation

- All-to-All Collective
  
  Enhanced LiMIC2 could reduce the collective latency up to 14% with huge pages

- Efficient buffer allocation for on-package memory (i.e., MCDRAM)
- Enhancement for huge pages
- Evaluation results for concurrent point-to-point communications
Concluding Remark

• **Support for Huge Pages**
  – Existing implementations for intra-node communication could not fully benefit from huge pages
  – Suggested an enhanced memory mapping scheme for huge pages
    • Improved the bandwidth and latency up to 44% and 14%, respectively

• **Future Work**
  – Application-level evaluation
  – Patches for LiMIC2
Thank You!