OMB-UM: Design, Implementation, and Evaluation of CUDA Unified Memory Aware MPI Benchmarks

PMBS ‘19

Karthik Vadambacheri Manian, Ching-Hsiang Chu, Ammar Ahmad Awan, Kawthar Shafie Khorassani, Hari Subramoni, and Dhabaleswar K. Panda

Network Based Computing Laboratory (NBCL)
Dept. of Computer Science and Engineering
The Ohio State University
{vadambacherimanian.1, chu.368, awan.10, shafiekhorassani.1, subramoni.1, panda.2}@osu.edu
Agenda

• Introduction
• Motivation
• Research Challenges
• Design
• Evaluation
• Discussion
• Conclusion
• Helps to characterize a system
• Provides various options for experimentation
• Benchmark results should be unambiguous
• MPI, OpenSHMEM, UPC & UPC++ benchmarks
• Pt2Pt, Collective & One-sided
  – Blocking & Non-blocking
• Support for CUDA & OpenACC extensions
  – Support for CUDA Managed/Unified Memory
Unified Memory
Dramatically Lower Developer Effort

Developer View Today
System Memory
GPU Memory

Developer View With Unified Memory
Unified Memory

Courtesy: NVI DIA developer blogs
void sortfile(FILE *fp, int N) {
    char *data;
    data = (char *)malloc(N);
    fread(data, 1, N, fp);
    qsort(data, N, 1, compare);
    use_data(data);
    free(data);
}

void sortfile(FILE *fp, int N) {
    char *data;
    cudaMallocManaged(&data, N);
    fread(data, 1, N, fp);
    qsort<<<...>>>(data, N, 1, compare);
    cudaDeviceSynchronize();
    use_data(data);
    cudaFree(data);
}
• Unified Memory (UM) can be either on the host or device

• **Effective location** of UM is the location where UM currently resides
Agenda

• Introduction
• Motivation
• Research Challenges
• Design
• Evaluation
• Discussion
• Conclusion
for (timestep = 0; ...) {
    compute_interior_kernel <<<..., interior_stream>>> (...) 
    cudaStreamSynchronize(boundary_stream)
    MPI_Irecv(...)
    MPI_Isend(...)
    MPI_Waitall(...)
    compute_xboundary_kernel <<<..., boundary_stream>>> (...) 
    compute_yboundary_kernel <<<..., boundary_stream>>> (...) 
    cudaDeviceSynchronize(...)
}
```c
cudaMemset(s_buf,...)
cudaMemset(r_buf,...)

if (my_rank == 0) {
    t_start = MPI_Wtime();
    for (iter = 0; iter < max_iter; iter++) {
        MPI_Send(s_buf,...)
        MPI_Recv(r_buf,...)
    }
    t_end = MPI_Wtime();
} else {
    for (iter = 0; iter < max_iter; iter++) {
        MPI_Recv(r_buf,...)
        MPI_Send(s_buf,...)
    }
}

latency = (t_end - t_start)/2.0 * max_iter
```
Limitations in current state of the art

• Oblivious to the effective location of UM buffers
• No provision to set the 4 possible UM effective locations
  ➢ MH-MH
  ➢ MD-MH
  ➢ MH-MD
  ➢ MD-MD
• In conclusion, there is a need for properly benchmarking middleware libraries on UM buffers
Agenda

• Introduction
• Motivation
• **Research Challenges**
  • Design
  • Evaluation
  • Discussion
• Conclusion
How can a full-fledged UM Aware OMB (OMB-UM) be designed to provide the facility to set the four possible effective locations for UM buffers leading to the full characterization of UM aware MPI on modern GPU clusters?
Research Challenges

- Can the performance of UM aware MPI be characterized fully?
- How to achieve the different data placements for UM buffer?
- What are the characteristics of the CUDA kernels employed for UM data placement?

Let’s design OMB-UM
Agenda

• Introduction
• Motivation
• Research Challenges
• **Design**
  • Evaluation
  • Discussion
• Conclusion
UM Buffer Placements

### MH – MH
- **Process 0**: MPI_Send(sendbuf, ...)
- **Process 1**: MPI_Recv(recvbuf, ...)

### MD – MD
- **Common**: kernel_touch(sendbuf)
- **Process 0**: MPI_Send(sendbuf, ...)
- **Process 1**: MPI_Recv(recvbuf, ...)
- **Common**: kernel_touch(recvbuf)

### MD – MH
- **Common**: kernel_touch(sendbuf)
- **Process 0**: MPI_Send(sendbuf, ...)
- **Process 1**: MPI_Recv(recvbuf, ...)

### MH – MD
- **Process 0**: MPI_Send(sendbuf, ...)
- **Process 1**: MPI_Recv(recvbuf, ...)
- **Common**: kernel_touch(recvbuf)

All the buffers involved are managed buffers.
Proposed Latency Benchmark (MD-MD)

Latency_{MD-MD} = \frac{(t_{End} - t_{start} - 2 \times t_{Kernel Launch})}{2}
Bandwidth\textsubscript{MD-MD} = \frac{(M \times \text{window\_size})}{(t_{bw} - t_{Kernel\_Launch})}
Agenda

• Introduction
• Motivation
• Research Challenges
• Design
• **Evaluation**
• Discussion
• Conclusion
<table>
<thead>
<tr>
<th>CPU</th>
<th>GPU</th>
<th>Interconnect</th>
<th>NVLink</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sandy Bridge E5-2670</td>
<td>Volta V100</td>
<td>EDR</td>
<td>No</td>
<td>RHEL 7.5.1804</td>
</tr>
<tr>
<td>Haswell E5-2687W</td>
<td>Volta V100</td>
<td>EDR</td>
<td>No</td>
<td>RHEL 7.5.1804</td>
</tr>
<tr>
<td>OpenPOWER POWER9</td>
<td>Volta V100</td>
<td>EDR</td>
<td>Yes</td>
<td>RHEL 7.6</td>
</tr>
</tbody>
</table>
• Latency MH MH has bump due to advanced managed memory designs.
• Performance of managed buffers on par with device and host buffers
• intra-node & inter-node MD-MD small to medium message bandwidth needs improvement
  – Caused by excessive movement of UM buffers between host & device
  – Performance worsens when the size of the message buffer increases

Managed Buffer Page Faults

<table>
<thead>
<tr>
<th></th>
<th>On GPU</th>
<th>On CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>65293</td>
<td>101630</td>
</tr>
</tbody>
</table>
• intra-node bibw: OpenMPI needs improvement

<table>
<thead>
<tr>
<th>MPI Library</th>
<th>On GPU</th>
<th>On CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenMPI+UCX</td>
<td>284557</td>
<td>295680</td>
</tr>
<tr>
<td>SpectrumMPI</td>
<td>1248</td>
<td>---</td>
</tr>
</tbody>
</table>

Managed Buffer Page Faults

Bandwidth MD MD

Bi-Bandwidth MD MD

Latency MD MD
Network Based Computing

OpenPOWER Intra-node Evaluation

Broadcast on managed buffers
• inter-node latency: SpectrumMPI needs improvement

**Managed Page Faults**

<table>
<thead>
<tr>
<th>MPI Library</th>
<th>On GPU</th>
<th>On CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenMPI+UCX</td>
<td>70445</td>
<td>74020</td>
</tr>
<tr>
<td>SpectrumMPI+UCX</td>
<td>351864</td>
<td>390526</td>
</tr>
</tbody>
</table>

[Graphs showing bandwidth and latency measurements for different MPI libraries.]
• CUDA Unified Memory greatly improves the user productivity
• Hardware support for UM in latest Pascal/Volta GPUs greatly improved the UM performance
• Current state of the art UM-Aware benchmarks do not accurately capture the effective location of UM buffer
• The proposed OMB-UM benchmarks provides necessary options to set the effective location of UM buffer
• Various insights obtained from evaluating OMB-UM are discussed along with potential solutions
Thank You!

{vadambacherimanian.1, chu.368, awan.10, shafiekhorrassani.1, subramoni.1, panda.2}@osu.edu

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/