Characterizing CUDA Unified Memory (UM)-Aware MPI Designs on Modern GPU Architectures

SC ’19 Booth Talk

Karthik Vadambacheri Manian, Ammar Ahmad Awan, Amit Ruhela, Ching-Hsiang Chu, Hari Subramoni, and Dhabaleswar K. Panda

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

- Introduction
- Motivation
- Research Challenges
- UM-Aware MPI Performance Characterization
- Conclusion
GPUs in HPC

• **GPUs are growing faster in the HPC arena**

• **NVIDIA GPUs - main driving force for**
  – **Accelerating traditional HPC applications**
  – **Accelerating AI thru faster training of Deep Neural Networks (DNNs)**

https://www.top500.org (Nov ‘18)
Managed/Unified Memory

Dramatically Lower Developer Effort

System Memory
GPU Memory
Unified Memory

Courtesy: NVIDIA developer blogs
Managed/Unified Memory (Contd)

```
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);
}
```

Courtesy: NVIDIA developer blogs
Message Passing Interface (MPI)

- Popular parallel programming model to harness the power of nodes in a clusters
- Data is explicitly sent by one process and received by another process in a cooperative fashion.
- MPI libraries can be
  - CUDA Aware
  - UM Aware
Agenda

• Introduction

• Motivation

• Research Challenges

• UM-Aware MPI Performance Characterization

• Conclusion
Basic Unified Memory (UM) Awareness in MPI

- CUDA Unified Memory (UM) => no memory pin down
  - No IPC support for intra-node communication
  - No GDR support for Inter-node communication
- Initial and basic support in MVAPICH2-GDR MPI library
  - For both intra- and inter-nodes use “pipeline through” host memory

D. Banerjee, K. Hamidouche, and D. K Panda, CUDA Designing High Performance Communication Runtime for GPU Managed Memory: Early Experiences, GPGPU-9
Enhanced Support for UM Awareness in MVAPICH2

- Designs are motivated due to the poor UM performance on Kepler series of NVIDIA GPUs

K. Hamidouche, A. Awan, A. Venkatesh, and D. K Panda, CUDA M3: Designing Efficient CUDA Managed Memory-aware MPI by Exploiting GDR and IPC, HiPC ‘16
Enhanced H/W Support for Unified Memory on Pascal/Volta

• GPU page faulting hardware support introduced in Pascal/Volta
  – Only faulting pages need to be migrated on-demand

• Hardware access counters makes only the most needed pages migrated on-demand

• On IBM Power systems, new Address Translation Services (ATS) allows a GPU to access CPU’s page table directly
Enhanced API Support for Unified Memory on Pascal/Volta

- Hints like `cudaMemAdvise` and `cudaMemPrefetchAsync()` are very useful

**USER HINTS**

**Prefetching**

```c
char *data;
cudaMallocManaged(&data, N);
init_data(data, N);

cudaMemPrefetchAsync(data, N, myGpuId, s);
mykernel<<<..., s>>>(data, N);
cudaMemPrefetchAsync(data, N, cudaCpuDeviceId, s);
cudaStreamSynchronize(s);

use_data(data, N);
cudaFree(data);
```

Page faults can be expensive and they stall SM execution

**Avoid faults by prefetching data to the accessing processor**

**Preferred Location**

```c
char *data;
cudaMallocManaged(&data, N);
init_data(data, N);

cudaMemAdvise(data, N, .PreferredLocation, cudaCpuDeviceId);
mykernel<<<..., s>>>(data, N);

use_data(data, N);
cudaFree(data);
```

Here the kernel will page fault and generate direct mapping to data on the CPU

The driver will “resist” migrating data away from the preferred location

---

Time to Revisit UM-Aware Designs in MVAPICH2?

- The substantial improvement in UM performance in latest Pascal/Volta GPUs motivates one to:
  - Explore whether UM-Aware designs in MVAPICH2 done for Kepler era are still valid for Pascal/Volta?
    - A thorough UM-Aware MPI library characterization will get the answer
Agenda

• Introduction

• Motivation

• **Research Challenges**

• UM-Aware MPI Performance Characterization

• Conclusion
Broad Challenge

*How can UM-Aware MPI runtimes deal with diverse GPU architectures and maintain efficient and high-performance designs for UM data movement across GPU generations?*
Research Challenges

Let’s characterize the performance of UM-Aware MPI

Does the UM designs done during Kepler era still valid for Volta/Pascal GPUs?

What is the fundamental H/W changes in Volta over Kepler wrt UM performance?

Can the performance of UM aware MPI be characterized completely?

How advances in UM need to be dealt with in the context of MPI on different GPUs?
Agenda

• Introduction
• Motivation
• Research Challenges
• **UM-Aware MPI Performance Characterization**
• Conclusion
Evaluation Platforms

- Experiments were performed on a local cluster containing three generations of GPUs

1. **Kepler K-80** hosted on a node containing 2 Intel Broadwell E5 v2680 processors running at 2.4 GHz with 125 GB RAM

2. **Pascal P100** hosted on a node containing 2 Intel Haswell E5-2687W processors running at 3.1 GHz with 62 GB RAM

3. **Volta V100** are hosted on following systems:
   - A node containing 2 Intel Haswell E5-2687W processors running at 3.1 GHz with 62 GB RAM
   - A node containing 2 POWER9 processors running at 3.8 GHz each connected to 2 V100s using NVLink2
Experimental Setup

- Experiments were conducted using OSU Micro Benchmark (OMB) suite\(^*\) of benchmarks.

- OMB suite contains benchmarks for analyzing the following types of MPI operations:
  - Point to Point
  - Collectives
  - One-sided

- ‘osu_latency’ benchmark is chosen from the OMB suite\(^*\) for UM characterization study

*OSUMicrobenchmarks, http://mvapich.cse.ohio-state.edu/benchmarks/
Experimental Setup: OMB

- ‘osu_latency’ benchmark measures latency by sending messages in a pingpong fashion on a UM buffer
- It sets the effective location of UM buffer as ‘Host’ by default
- Not much useful in characterizing the UM Aware MPI designs
Experimental Setup: MOMB

- OMB is modified to make the effective location of UM buffer as ‘Device’
- Achieved by launching a sender-side CUDA kernel during every pingpong iteration
- The CUDA kernel also modifies the UM buffer during every CUDA kernel launch
- More successful in characterizing the UM Aware MPI
Experimental design

• Both the benchmarks are evaluated for:
  – Host to Host transfer (HH)
  – Device to Device transfer (DD)
  – Unified (Managed) to Unified (Managed) buffer transfer (MM)

• Benchmark performance is evaluated using:
  – MVAPICH2-GDR & OpenMPI (Intel systems)
  – Spectrum MPI (IBM Power systems)
UM-Aware MPI Characterization on V100 with OMB

MM and HH overlap in both these graphs shows OMB’s improper characterization of UM
UM-Aware MPI Characterization on K80 with MOMB

Proper characterization of UM on K80 using MOMB: UM performance is worse than device memory
UM-Aware MPI Characterization on V100 with MOMB

Proper characterization of UM on V100 using MOMB: UM performance is on par with device memory
UM-Aware MPI Characterization on V100 + Power9 + NVLink with MOMB (Power Systems)

Proper characterization of UM on V100 using MOMB: UM performance is better than device memory
Let’s Dig Deeper

### Page Faults while running MOMB

<table>
<thead>
<tr>
<th>GPU Architecture</th>
<th>Host Architecture</th>
<th>GPU Page Faults</th>
<th>CPU Page Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-80</td>
<td>Intel X86</td>
<td>N/A</td>
<td>282800</td>
</tr>
<tr>
<td>Volta</td>
<td>Intel X86</td>
<td>31653</td>
<td>6550</td>
</tr>
<tr>
<td>Volta</td>
<td>POWER9</td>
<td>259</td>
<td>335</td>
</tr>
</tbody>
</table>
Gains From Enhanced UM Support on K80

Enhanced UM designs certainly improved performance during Kepler era.
Gains From Enhanced UM Support on V100

The enhanced UM designs from Kepler era still provides benefits for Volta GPUs!!
Gains From Enhanced UM Support on V100 + Power9 + NVLink (Power Systems)

The enhanced UM designs provides benefits for Volta GPUs on Power systems too!!
Gains from Enhanced UM Support on V100: MVAPICH2 Vs OpenMPI

The enhanced UM designs in MVAPICH2 fares better than OpenMPI.
Agenda

• Introduction
• Background
• Research Challenges
• UM-Aware MPI Performance Characterization
• Conclusion
Conclusion

• Hardware support for UM in latest Pascal/Volta GPUs greatly improved the UM performance

• This does not mean that the UM aware designs conceived in MPI libraries during Kepler era are not valid any more

• The conducted UM characterization study clearly shows that previous UM aware designs still improve the UM performance of latest Pascal/Volta GPUs

• MVAPICH2-OPT containing enhanced UM aware designs outperforms regular MVAPICH2-GDR by 4.2x and OpenMPI by 2.8x for Intel Systems
Thank You!

{vadambacherimanian.1,awan.10,ruhela.2,chu.368,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/
Backup Slides
MVAPICH2-GDR Vs OpenMPI: Pure Device Buffer Performance

![Graph showing latency (us) vs message size (Bytes) for MVAPICH2-GDR vs OpenMPI]