MVAPICH2-GDR: Pushing the Frontier of Designing MPILibraries Enabling GPUDirect Technologies

Presentation at NVIDIA Booth (SC’15)
Dhabaleswar K. (DK) Panda - The Ohio State University
panda@cse.ohio-state.edu
• Communication on InfiniBand Clusters with GPUs
• MVAPICH2-GPU with GPUDirect-RDMA (GDR)
• New Enhancements
  • Non-Blocking Collectives (MPI3 NBC) support
  • Multi-Stream for efficient MPI Datatype Processing
• On-Going Work (GDR-Async and Managed Memory)
• OpenACC-Aware support
• Conclusions
MVAPICH2-GPU: CUDA-Aware MPI

- Before CUDA 4: Additional copies
  - Low performance and low productivity
- After CUDA 4: Host-based pipeline
  - Unified Virtual Address
  - Pipeline CUDA copies with IB transfers
  - High performance and high productivity
- After CUDA 5.5: GPUDirect-RDMA support
  - GPU to GPU direct transfer
  - Bypass the host memory
  - Hybrid design to avoid PCI bottlenecks
Overview of MVAPICH2 Software

• High Performance MPI and PGAS Library for InfiniBand, 10-40Gig/iWARP, and RDMA over Converged Enhanced Ethernet (RoCE) and Accelerators

  MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.0), Available since 2002
  MVAPICH2-X (MPI + PGAS), Available since 2011

Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014

Support for Virtualization (MVAPICH2-Virt), Available since 2015

Support for Energy-Awareness (MVAPICH2-EA), Available since 2015

Used by more than 2,475 organizations in 76 countries

More than 307,000 downloads from the OSU site directly

Empowering many TOP500 clusters (Nov ‘15 ranking)

• 10th ranked 519,640-core cluster (Stampede) at TACC

• 13th ranked 185,344-core cluster (Pleiades) at NASA

• 25th ranked 76,032-core cluster (Tsubame 2.5) at Tokyo Institute of Technology and many others

• Available with software stacks of many vendors and Linux Distros (RedHat and SuSE)

http://mvapich.cse.ohio-state.edu
Optimizing MPI Data Movement on GPU Clusters

- Connected as PCIe devices – Flexibility but Complexity

1. Intra-GPU
2. Intra-Socket GPU-GPU
3. Inter-Socket GPU-GPU
4. Inter-Node GPU-GPU
5. Intra-Socket GPU-Host
6. Inter-Socket GPU-Host
7. Inter-Node GPU-Host

8. Inter-Node GPU-GPU with IB adapter on remote socket and more . . .

- For each path different schemes: Shared_mem, IPC, GPUDirect RDMA, pipeline ...
- Critical for runtimes to optimize data movement while hiding the complexity
Outline

• Communication on InfiniBand Clusters with GPUs
• MVAPICH2-GPU with GPUDirect-RDMA (GDR)
• New Enhancements
  • Non-Blocking Collectives (MPI3 NBC) support
  • Multi-Stream for efficient MPI Datatype Processing
• On-Going Work (GDR-Async and Managed Memory)
• OpenACC-Aware support
• Conclusions
GPUDirect RDMA (GDR) with CUDA

- Hybrid design using GPUDirect RDMA
  - GPUDirect RDMA and Host-based pipelining
  - Alleviates P2P bandwidth bottlenecks on SandyBridge and IvyBridge
- Support for communication using multi-rail
- Support for Mellanox Connect-IB and ConnectX VPI adapters
- Support for RoCE with Mellanox ConnectX VPI adapters

S. Potluri, K. Hamidouche, A. Venkatesh, D. Bureddy and D. K. Panda, Efficient Inter-node MPI Communication using GPUDirect RDMA for InfiniBand Clusters with NVIDIA GPUs, Int'l Conference on Parallel Processing (ICPP ’13)
Using MVAPICH2-GPUDirect Version

- MVAPICH2-2.2a with GDR support can be downloaded from
  https://mvapich.cse.ohio-state.edu/download/mvapich2gdr/

- System software requirements
  - Mellanox OFED 2.1 or later
  - NVIDIA Driver 331.20 or later
  - NVIDIA CUDA Toolkit 6.5 or later
  - Plugin for GPUDirect RDMA

  **Strongly Recommended**: use the new GDRCOPY module from NVIDIA

- Has optimized designs for point-to-point and collective communication using GDR
- Contact MVAPICH help list with any questions related to the package
  mvapich-help@cse.ohio-state.edu
Performance of MVAPICH2-GDR with GPU-Direct-RDMA

**GPU-GPU Internode MPI Latency**

- MV2-GDR2.1
- MV2-GDR2.0b
- MV2 w/o GDR

**GPU-GPU Internode MPI Uni-Bandwidth**

- MV2-GDR2.1

**Intel Ivy Bridge (E5-2680 v2) node - 20 cores**

**NVIDIA Tesla K40c GPU**

**Mellanox Connect-IB Dual-FDR HCA**

**CUDA 7**

**Mellanox OFED 2.4 with GPU-Direct-RDMA**
Application-Level Evaluation (HOOMD-blue)

- Platform: Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- HOOMDBlue Version 1.0.5
  - GDRCOPY enabled: MV2_USE_CUDA=1 MV2_IBA_HCA=mlx5_0 MV2_IBA_EAGER_THRESHOLD=32768 MV2_VBUF_TOTAL_SIZE=32768 MV2_USE_GPUDIRECT_LOOPBACK_LIMIT=32768 MV2_USE_GPUDIRECT_GDRCOPY=1 MV2_USE_GPUDIRECT_GDRCOPY_LIMIT=16384
Performance of MVAPICH2 with GPU-Direct-RDMA: MPI-3 RMA

GPU-GPU Internode MPI Put latency (RMA put operation Device to Device)

MPI-3 RMA provides flexible synchronization and completion primitives

Small Message Latency

- MV2-GDR2.1
- MV2-GDR2.0b

Latency (us)

Message Size (bytes)

MVAPICH2-GDR-2.1
Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 7, Mellanox OFED 2.4 with GPU-Direct-RDMA

2.88 us

7X
Performance of MVAPICH2-GDR with GPU-Direct-RDMA and Multi-Rail Support

**GPU-GPU Internode MPI Uni-Directional Bandwidth**

- **MV2-GDR 2.1**: 8,759 MB/s
- **MV2-GDR 2.1 RC2**: 40% lower

**GPU-GPU Internode Bi-directional Bandwidth**

- **MV2-GDR 2.1**: 15,111 MB/s
- **MV2-GDR 2.1 RC2**: 20% lower

**Message Size (bytes)**

- 1, 4, 16, 64, 256, 1K, 4K, 16K, 64K, 256K, 1M, 4M

**Bandwidth (MB/s)**

- 0, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000

**Bi-Bandwidth (MB/s)**

- 0, 2000, 4000, 6000, 8000, 10000, 12000, 14000, 16000

**Equipment Details**

- Intel Ivy Bridge (E5-2680 v2) node - 20 cores
- NVIDIA Tesla K40c GPU
- Mellanox Connect-IB Dual-FDR HCA CUDA 7
- Mellanox OFED 2.4 with GPU-Direct-RDMA
Outline

• Communication on InfiniBand Clusters with GPUs
• MVAPICH2-GPU with GPUDirect-RDMA (GDR)
• New Enhancements
  • Non-Blocking Collectives (MPI3 NBC) support
  • Multi-Stream for efficient MPI Datatype Processing
• On-Going Work (GDR-Async and Managed Memory)
• OpenACC-Aware support
• Conclusions
Non-blocking collectives (NBC) using network tasks offload mechanism (CORE-Direct)

- MPI NBC decouple initiation(Ialltoall) and completion(Wait) phases and provide overlap potential (Ialltoall + compute + Wait) but CPU drives progress largely in Wait (=> 0 overlap)
- CORE-Direct feature provides true overlap capabilities by providing a priori specification of a list of network-tasks which is progressed by the NIC instead of the CPU (hence freeing it)
- We propose a design that combines GPUDirect RDMA and Core-Direct features to provide efficient support of CUDA-Aware NBC collectives on GPU buffers
  - Overlap communication with CPU computation
    - and
  - Overlap communication with GPU computation
- Extend OMB with CUDA-Aware NBC benchmarks to evaluate
  - Latency
  - Overlap on both CPU and GPU

CUDA-Aware NonBlocking collectives

Platform: Wilkes: Intel Ivy Bridge
NVIDIA Tesla K20c + Mellanox Connect-IB
MVAPICH2-GDR 2.2a
Outline

• Communication on InfiniBand Clusters with GPUs
• MVAPICH2-GPU with GPUDirect-RDMA (GDR)
• New Enhancements
  • Non-Blocking Collectives (MPI3 NBC) support
  • Multi-Stream for efficient MPI Datatype Processing
• On-Going Work (GDR-Async and Managed Memory)
• OpenACC-Aware support
• Conclusions
Non-contiguous Data Exchange

- Multi-dimensional data
  - Row based organization
  - Contiguous on one dimension
  - Non-contiguous on other dimensions

- Halo data exchange
  - Duplicate the boundary
  - Exchange the boundary in each iteration
MPI Datatype Processing (Computation Optimization)

- Comprehensive support
  - Targeted kernels for regular datatypes - vector, subarray, indexed_block
  - Generic kernels for all other irregular datatypes

- Separate non-blocking stream for kernels launched by MPI library
  - Avoids stream conflicts with application kernels

- Flexible set of parameters for users to tune kernels
  - Vector
    - MV2_CUDA_KERNEL_VECTOR_TIDBLK_SIZE
    - MV2_CUDA_KERNEL_VECTOR_YSIZE
  - Subarray
    - MV2_CUDA_KERNEL_SUBARR_TIDBLK_SIZE
    - MV2_CUDA_KERNEL_SUBARR_XDIM
    - MV2_CUDA_KERNEL_SUBARR_YDIM
    - MV2_CUDA_KERNEL_SUBARR_ZDIM
  - Indexed_block
    - MV2_CUDA_KERNEL_IDXBLK_XDIM
Common Scenario

MPI_Isend (A,.. Datatype,…)  
MPI_Isend (B,.. Datatype,…)  
MPI_Isend (C,.. Datatype,…)  
MPI_Isend (D,.. Datatype,…)  
…

MPI_Waitall (...);

Waste of computing resources on CPU and GPU

Expected Benefits
Application-Level Evaluation (HaloExchange - Cosmo)

- 2X improvement on 32 GPUs nodes
- 30% improvement on 96 GPU nodes (8 GPUs/node)

Outline

• Communication on InfiniBand Clusters with GPUs
• MVAPICH2-GPU with GPUDirect-RDMA (GDR)
• New Enhancements
  • Non-Blocking Collectives (MPI3 NBC) support
  • Multi-Stream for efficient MPI Datatype Processing
• On-Going Work (GDR-Async and Managed Memory)
• OpenACC-Aware support
• Conclusions
On Going Work

- **Support for GDR Async feature (GPUDirect RDMA Family 4)**
  - Offload control flow to the GPU
  - Issue the communication operation from/to GPU
  - Free CPU and remove from critical path
  - Hide the overhead of launching CUDA Kernels and keep the GPU busy
  - Extend OMB with GDR Async semantics

- **Initial Support for Managed Memory**
  - MPI-Aware managed memory
  - Transparently handle the data movement of managed memory at MPI level
  - High productivity (unique pointer for both CPU and GPU work)
  - Extend OMB with managed memory semantics
• Communication on InfiniBand Clusters with GPUs
• MVAPICH2-GPU with GPUDirect-RDMA (GDR)
• New Enhancements
  • Non-Blocking Collectives (MPI3 NBC) support
  • Multi-Stream for efficient MPI Datatype Processing
• On-Going Work (GDR-Async and Managed Memory)
  • OpenACC-Aware support
• Conclusions
OpenACC-Aware MPI

- **acc_malloc** to allocate device memory
  - No changes to MPI calls
  - MVAPICH2 detects the device pointer and optimizes data movement
- **acc_deviceptr** to get device pointer (in OpenACC 2.0)
  - Enables MPI communication from memory allocated by compiler when it is available in OpenACC 2.0 implementations
  - MVAPICH2 will detect the device pointer and optimize communication

Delivers the same performance as with CUDA

```c
A = acc_malloc(sizeof(int) * N);
......
#pragma acc parallel loop deviceptr(A) . . .
//compute for loop
MPI_Send (A, N, MPI_INT, 0, 1, MPI_COMM_WORLD);
......
acc_free(A);
```

```c
A = malloc(sizeof(int) * N);
......
#pragma acc data copyin(A) . . .
{
#pragma acc parallel loop . . .
//compute for loop
MPI_Send(acc_deviceptr(A), N, MPI_INT, 0, 1, MPI_COMM_WORLD);
}
......
free(A);
```
• Communication on InfiniBand Clusters with GPUs
• MVAPICH2-GPU with GPUDirect-RDMA (GDR)
• New Enhancements
  • Non-Blocking Collectives (MPI3 NBC) support
  • Multi-Stream for efficient MPI Datatype Processing
• On-Going Work (GDR-Async and Managed Memory)
• OpenACC-Aware support
• Conclusions
Conclusions

- MVAPICH2 optimizes MPI communication on InfiniBand clusters with GPUs
- Provides optimized designs for point-to-point two-sided and one-sided communication, datatype processing and collective operations
- Efficient and maximal overlap for MPI3 NBC collectives
- Takes advantage of CUDA features like IPC and GPUDirect RDMA families
- Delivers
  - High performance
  - High productivity
- With support for latest NVIDIA GPUs and InfiniBand Adapters
- Users are strongly encouraged to use MVAPICH2-GDR 2.2a
Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

MVAPICH Web Page
http://mvapich.cse.ohio-state.edu/