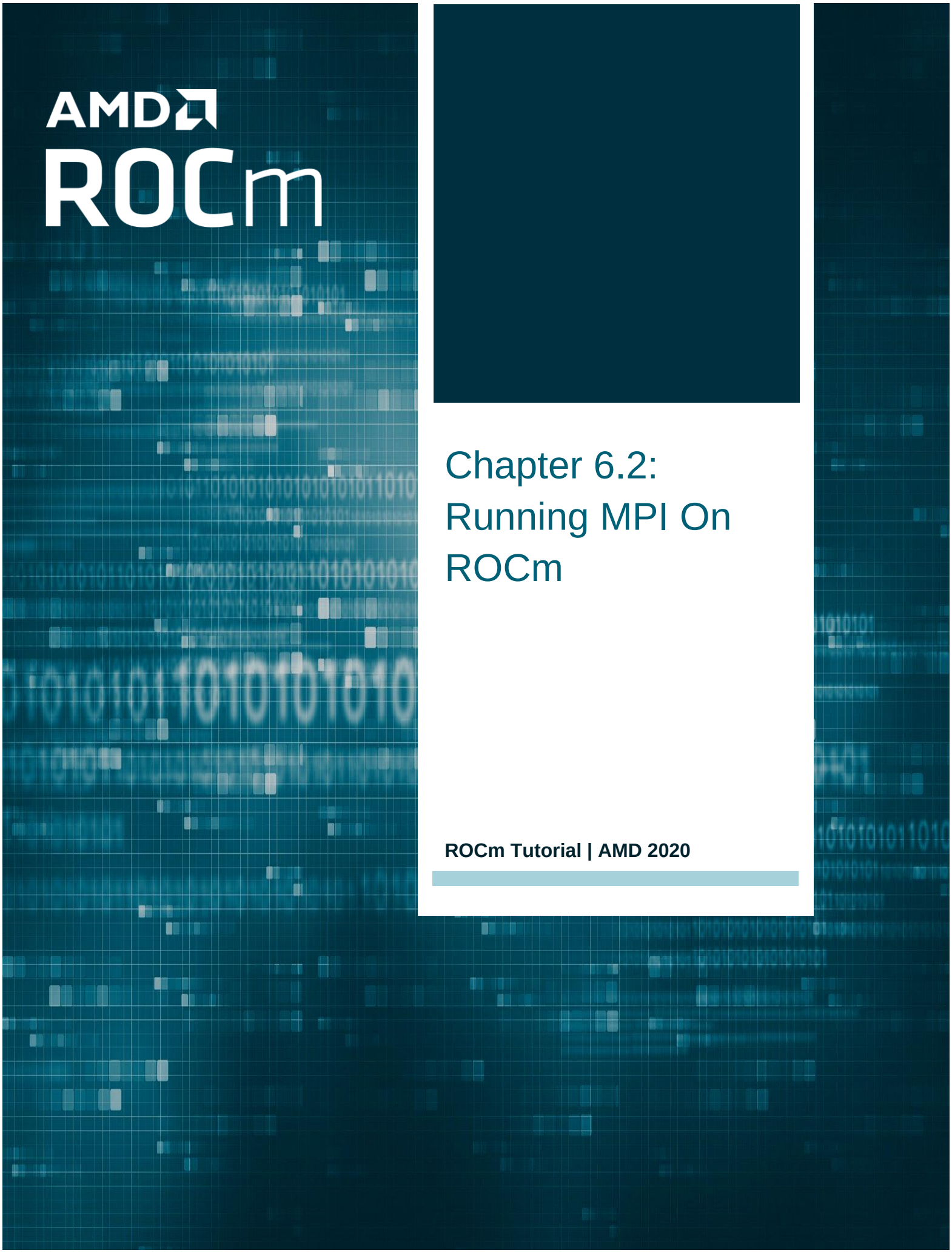

Chapter 6.2: Running MPI On ROCm

ROCm Tutorial | AMD 2020

AMD
ROCm

Table of Contents

AMD ROCm

Chapter 6.2: Running MPI On ROCm

This hands-on tutorial shows we can write a multi-GPU MPI application on ROCm.

Preparation

1. Ensure you have a multi-GPU system to run this example.

2. Ensure ROCm and HIP are installed correctly.

3. Ensure that you have a working MPI installation:
   mpixx --showme show return the paths to your MPI installation
   (where xx is cc or c++)

   If MPI is not installed, run “sudo apt-get install libopenmpi-dev”

4. Change to the working directory:

   cd Chapter6/02_MultiGPU_MPI

Application Logic

1. The application starts out creating a array of random numbers

2. These numbers are distributed to each GPU using MPIScatter from the root node

3. Each GPU receives its set of numbers and runs a kernel that squares the received data.

4. Data is copied back to the CPU of each node.

5. Using MPIGather, the data is collected by the root node.
6. The root node than runs a reduce kernel on the GPU that does a reduction on the received data and finally produces the final result

Running the Example

1. We will be using the HIP compiler “hipcc” to compile the example.

2. But first we need to add the MPI include and library paths so that during compilation and linking the appropriate headers and libraries can be found.

3. Run mpixx --showme(where xx is cc or c++).
   For example running mpicc --showme on our system shows the following output:

   - gcc -I/usr/lib/x86_64-linux-gnu/openmpi/include/openmpi -I/usr/lib/x86_64-linux-gnu/openmpi/include/openmpi/opal/mca/event/libevent2022/libevent -I/usr/lib/x86_64-linux-gnu/openmpi/include/openmpi/opal/mca/event/libevent2022/libevent/include -I/usr/lib/x86_64-linux-gnu/openmpi/include -pthread -L/usr//lib -L/usr/lib/x86_64-linux-gnu/openmpi/lib -lmpi

   - As you can see, this contains the include paths for the MPI headers and the paths to libraries for linking

4. So to compile use

   “/opt/rocm/bin/hipcc -I/opt/rocm/include -I/usr/lib/x86_64-linux-gnu/openmpi/include/openmpi -I/usr/lib/x86_64-linux-gnu/openmpi/include/openmpi/opal/mca/event/libevent2022/libevent -I/usr/lib/x86_64-linux-gnu/openmpi/include/openmpi/opal/mca/event/libevent2022/libevent/include -I/usr/lib/x86_64-linux-gnu/openmpi/include -L/usr//lib -L/usr/lib/x86_64-linux-gnu/openmpi/lib -lmpi_cxx -lmpi -lstdc++ multigpu_mpi.cc -o multigpu_mpi ”

5. To run use
    - “mpirun -np p ./multigpu_mpi N” where “p” is the number of processes you want to launch, “N” is the count of numbers distributed to each process in the beginning
    - Ideally number of processes should be set to number of GPUs
    - For example if we have 2 GPUs and we want each GPU to process 10 thousand numbers we will run “mpirun -np 2 ./multigpu_mpi 10000”

```
Root node scattering data
GPU 1 completed its squaring. Copying data back to local CPU
GPU 0 completed its squaring. Copying data back to local CPU
Root node finished gathering data. Running reduce kernel
Reduce sum 1546907
```

Figure 1 : Trace of the processing running on a two GPU system. Your output result might be different due to different random numbers that are generated at runtime