AMD

ROCm

Chapter 3.2: Matrix Copy Kernel Analysis

ROCm Tutorial | AMD 2020

AMD
ROCm

Table of Contents

AMD ROCm

Chapter 3.2: Matrix Copy Kernel Analysis

In this tutorial, we will analyze the matrix copy application using the rocProf profiler tool from ROCm

Preparation

1. First in the tutorial repository go to the directory:

   cd 02_Matrix_Transpose

2. The application code is in copy.cpp

Compiling and Executing

1. Compile the program

   hipcc copy.cpp -o copy

2. Execute the program without profiler

   ./copy

3. Note that we are not printing any output from the matrices as the matrices are large. But you can add print code if desired

AMD ROCm

Profiling

1. Now we will analyze the application through the profiler
2. Recall that we have two profiler modes:
    a. Performance Measurement Mode to get the kernel execution time
    b. Performance Counter Mode to get the desired hardware counters

3. First let us collect the kernel execution time using the performance measurement mode. Run the following command:

    rocprof --stats ./copy

    - You will get the output in a file results.csv
    - The metrics are self-explanatory. We are interested in the kernel execution time. So take a record of the Duration(ns)

4. Now we are going to collect some hardware performance counters. Specifically we are interested in the number of reads and writes of this application

5. In performance counter mode, rocProf requires the user to provide a metric collection file. For this application we have provided this file in “metrics_copy_kernel.txt” as shown below

```
pmc: TCC_EA_WRREQ_sum, TCC_EA_RDREQ_sum
range: 0:1
gpu: 0
kernel: copy_kernel
```

6. The first row “pmc” defines the metrics we are collecting. Note that if we are collecting a lot of metrics, you would have to split the metrics into groups as defined here in section 2.1.2.2: https://github.com/ROCm-Developer-Tools/rocprofiler/blob/amd-master/doc/rocprof.md. The two metrics here are for the number of reads and writes made by the kernel

7. The second row “range” defines the number range of the kernel dispatch, In this case, since we are only launching one kernel it is defined as “0:1”

8. The third row “gpu” defines the GPU on which we want to collect the HW performance counters from. We have set it as 0 since GPU 0 is the selected device by the application by default.

9. And finally,the fourth row “kernel” defines the name of the kernel we want to profile which is the “copy_kernel” in copy.cpp. If you have multiple kernels just add the names and separate them with a space (no commas).

10. Now we will run the application in performance counter mode using our defined metric file:

    rocprof -i metrics_copy_kernel.txt -o metrics_copy.csv ./copy

    This will output the results of the HW performance counters in metrics_copy.csv.

11. Keep a record of the results obtained for this kernel. For our case, we obained the results below. Your results might be different depending on the GPU you are on:

    Kernel time(ns):280798
    TCC_EA_RDREQ_sum: 524312
    TCC_EA_WRREQ_sum: 524288