Updates on heterogeneous tracing with ROCm

Arnaud Fiorini

Polytechnique Montréal
Laboratoire DORSAL

Agenda

1. Background
   - GPU Programming model
   - Tracing pipeline
2. Advances
   - Backward dependencies (wait dependencies)
   - Interval format
3. Challenges and Future work
   - Trace size
   - Distributed system

Background – GPU Programming model

- Terminology
  - Host The CPU
  - Device The GPU
  - Kernel Routine compiled for accelerators

Background – GPU Programming model

- Device code, also called kernel function

```
1 __global__ void helloworld(char* in, char* out)
2 {
3     int num = hipThreadIdx_x + hipBlockDim_x * hipBlockIdx_x;
4     out[num] = in[num] + 1;
5 }
```

Background – GPU Programming model

- Host code

```
1  int main(int argc, char* argv[])
2  {
3      const char* input = "GdkknVnqkc";
4      size_t strlength = strlen(input);
5      char *output = (char*) malloc(strlength + 1);
6
7      char* inputBuffer;
8      char* outputBuffer;
9
10     hipMalloc((void**)&inputBuffer, (strlength + 1) * sizeof(char));
11     hipMalloc((void**)&outputBuffer, (strlength + 1) * sizeof(char));
12     hipMemcpy(inputBuffer, input, (strlength + 1) * sizeof(char), hipMemcpyHostToDevice);
13
14     hipLaunchKernelGGL(helloworld, dim3(1), dim3(strlength), 0, 0, inputBuffer, outputBuffer);
15
16     hipMemcpy(output, outputBuffer,(strlength + 1) * sizeof(char), hipMemcpyDeviceToHost);
17     hipFree(inputBuffer);
18     hipFree(outputBuffer);
19
20     free(output);
21 }
```

Background – GPU Programming model

1. Allocating space on the device
2. Copy your memory over to the device
3. Perform a kernel launch with the kernel function
4. Copy back the result to the host
5. Free the space on the device

These steps are done by the user with the HIP API (CUDA-like API)

Background – GPU Programming model

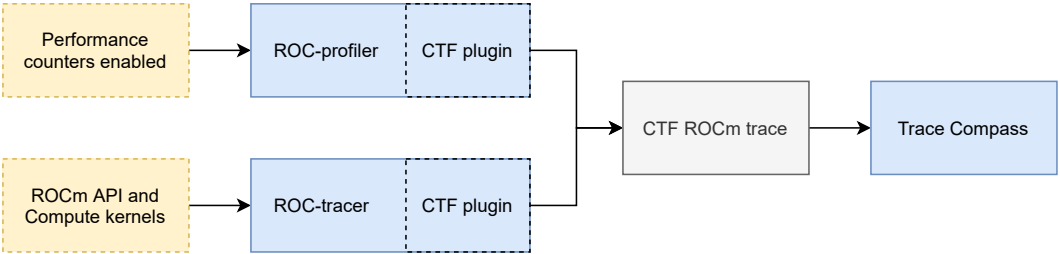
- We have seen how the user will queue operations for the GPU
- In the next slide we will see how this is implemented in the runtime using the HSA API.

Background – GPU Programming model

1. Creating a packet
2. Placing the packet into synchronized memory between CPU and GPU
3. Send a bell signal
4. The GPU will either pick up the packet or receive the signal and execute the packet

These steps are done by the runtime in user mode with the HSA API

Background – Tracing pipeline



Advances – Interval format

Changing the trace format from monotonic events representing start and end of a function to interval events:

- Improve trace size by 14 %
- Improve trace analysis by 27 %
- But the analysis of the trace has to be redone using future events

Advances – Backward dependencies

Advances – Backward dependencies

Demo

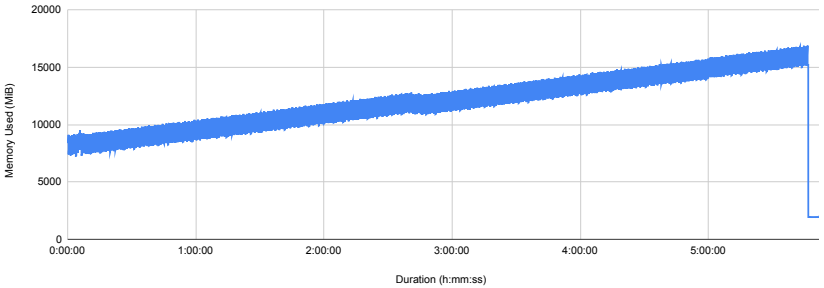
Challenges and Future work – Trace size

A stress test has been done with a program executing around 200 millions kernel calls. Here are some statistics and challenges:

- The trace generated was 146 GB and had 1.6 billion events.
- The statistics and ROCm state systems were 52 GB and 174 GB respectively.
- Running the indexing and analysis took 4 hours 45 minutes.

Challenges and Future work – Trace size

A stress test has been done with a program executing around 200 millions kernel calls. Here are some statistics and challenges:



Challenges and Future work – Trace size

These results can be improved by:

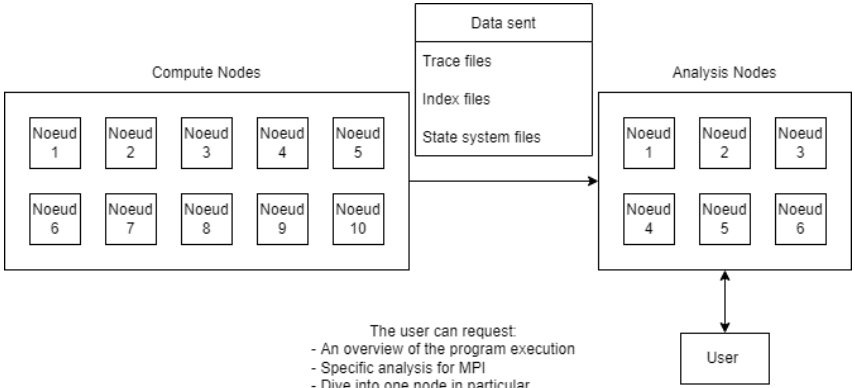
- Not recording some events that are not useful for the analysis.
- Using the partial state system that will reduce drastically the size of the state system.
- Some optimization to reduce contention on the Trace Compass server (e.g. running the indexing separately).

Challenges and Future work – Distributed system

GPU programs are often times using MPI to parallelize the workload on a number of nodes. This makes analyzing the resulting program more difficult for a couple of reasons:

- MPI has its own event that warrant specialized analyzes
- We use preprocessing to index and analyze the traces generated.
- The traces will be further analyzed by Trace Compass in a distributed manner to give an overview of the program execution.
- We saw that even if we generate the state system in advance, we still have to copy the traces back to our analysis nodes. To alleviate this issue, we can filter, aggregate or compress the trace before copying it.

Challenges and Future work – Distributed system



Thank You