Support: Debugging and Profiling

Rupesh Nasre.

Debugging

- Debugging parallel programs is difficult.
  - Non-determinism due to thread-scheduling
  - Output can be different
  - Correct intermediate values may be large

- cuda-gdb
  - for debugging CUDA programs on real hardware
  - Extension to gdb
  - Allows breakpoints, single-step, read/write memory contents.

Sample Error

```
#include <cuda.h>
#include <stdio.h>

__global__ void K(int *x) {
    *x = 0;
}
int main() {
    int *x;
    K<<<2, 10>>>(x);
    cudaDeviceSynchronize();

    return 0;
}
```

Sample Error

```
#include <cuda.h>
#include <stdio.h>


__global__ void K(int *x) {
    *x = 0;
    printf("%d\n", *x);      // does not print anything.
}
int main() {
    int *x;
    K<<<2, 10>>>(x);
    cudaDeviceSynchronize();

    return 0;
}
```

Sample Error

```
#include <cuda.h>
#include <stdio.h>

__global__ void K(int *x) {
    *x = 0;
    printf("%d\n", *x);
}
int main() {
    int *x;
    K<<<2, 10>>>(x);                          error=77, cudaErrorIllegalAddress,
    cudaDeviceSynchronize();                  an illegal memory access was encountered
    cudaError_t err = cudaGetLastError();
    printf("error=%d, %s, %s\n", err, cudaGetErrorName(err),
                                 cudaGetErrorString(err));
    return 0;
}
```

5

CUDA Errors

- cudaSuccess = 0, /// No errors
- cudaErrorMissingConfiguration = 1, /// Missing configuration error
- cudaErrorMemoryAllocation = 2, /// Memory allocation error
- cudaErrorInitializationError = 3, /// Initialization error
- cudaErrorLaunchFailure = 4, /// Launch failure
- cudaErrorPriorLaunchFailure = 5, /// Prior launch failure
- cudaErrorLaunchTimeout = 6, /// Launch timeout error
- cudaErrorLaunchOutOfResources = 7, /// Launch out of resources
- cudaErrorInvalidDeviceFunction = 8, /// Invalid device function
- cudaErrorInvalidConfiguration = 9, /// Invalid configuration
- cudaErrorInvalidDevice = 10, /// Invalid device
- ...

Homework: Write programs to invoke these errors.

cuda-gdb

- Generate debug information

  - nvcc -g -G file.cu
  - Disables optimizations, inserts symbol information.

- Run with cuda-gdb

  - cuda-gdb a.out

  - > run

- May have to stop the windows manager.

- Due to lots of threads, cuda-gdb works with a focus (current thread).

```
__global__ void K(int *x)
{
    *x = 0;
    printf("%d\n", *x);
}
```

(cuda-gdb) **run**
Starting program: ..../a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
[New Thread 0x7ffff7396700 (LWP 10305)]
[New Thread 0x7ffff696d700 (LWP 10306)]

CUDA Exception: Device Illegal Address
The exception was triggered in device 0.

Program received signal CUDA_EXCEPTION_10, Device Illegal Address.
[Switching focus to CUDA kernel 0, grid 1, block (1,0,0), thread (0,0,0), device 0, sm 13, warp 0, lane 0]
0x0000000000aa9510 in K<<<(2,1,1),(10,1,1)>>> (x=0x0) at gdb2.cu:6
6   printf("%d\n", *x);
(cuda-gdb)

cuda-gdb

(cuda-gdb) **info cuda kernels**

Kernel Parent Dev Grid Status   SMs Mask GridDim BlockDim Invocation

*   0   -   0   1   Active 0x00006000 (2,1,1)   (10,1,1)   K(x=0x0)

(cuda-gdb) **info threads**

Id   Target Id   Frame

3   Thread 0x7ffff696d700 (LWP 10497) "a.out" 0x00000038db4df113 in poll ()
 from /lib64/libc.so.6

2   Thread 0x7ffff7396700 (LWP 10496) "a.out" 0x00000038db4eac6f in accept4
    () from /lib64/libc.so.6

* 1   Thread 0x7ffff7fca720 (LWP 10487) "a.out" 0x00007ffff77a2118 in cudbgApiDetach () from /usr/lib64/libcuda.so.1

cuda-gdb

(cuda-gdb) **info cuda threads**

  BlockIdx ThreadIdx To BlockIdx ThreadIdx Count   Virtual PC Filename Line

Kernel 0

* (0,0,0) (0,0,0)   (1,0,0) (9,0,0)   20 0x0000000000aa9f50 gdb2.cu   6

(cuda-gdb) **cuda kernel block thread**

kernel 0, block (0,0,0), thread (0,0,0)

(cuda-gdb) **cuda block 1 thread 0**

[Switching focus to CUDA kernel 0, grid 1, block (1,0,0), thread (0,0,0), device 0, sm 13, warp 0, lane 0]

0x0000000000aa9510   6   printf("%d\n", *x);

(cuda-gdb) **cuda kernel block thread**

kernel 0, block (1,0,0), thread (0,0,0)

10

Breakpoints

- break `main`// first instruction in main

- break file.cu:223   // file:line

- set cuda break_on_launch application
      // kernel entry breakpoint

- break file.cu:23 if threadIdx.x == 1 && i < 5
      // conditional breakpoint

Step

- Once at a breakpoint, you can single-step
  - `step`, `s` or `<enter>`

```
(cuda-gdb) info cuda sms
  SM  Active Warps Mask
Device 0
   0 0x0000000000000000
   1 0x0000000000000000
   2 0x0000000000000000
   3 0x0000000000000000
   4 0x0000000000000000
   5 0x0000000000000000
   6 0x0000000000000000
   7 0x0000000000000000
   8 0x0000000000000000
   9 0x0000000000000000
  10 0x0000000000000000
  11 0x0000000000000000
  12 0x0000000000000000
  13 0x0000000000000001
* 14 0x0000000000000001
```

(cuda-gdb) **info cuda warps**
  Wp Active Lanes Mask Divergent Lanes Mask Active Physical PC Kernel BlockIdx First Active ThreadIdx
Device 0 SM 14

Wp	Active Lanes Mask	Divergent Lanes Mask	Active Physical PC	Kernel	BlockIdx	First Active ThreadIdx
* 0	0x000003ff	0x00000000	0x0000000000000110	0	(0,0,0)	(0,0,0)
1	0x00000000	0x00000000	n/a	n/a	n/a	n/a
2	0x00000000	0x00000000	n/a	n/a	n/a	n/a
3	0x00000000	0x00000000	n/a	n/a	n/a	n/a
4	0x00000000	0x00000000	n/a	n/a	n/a	n/a
5	0x00000000	0x00000000	n/a	n/a	n/a	n/a
6	0x00000000	0x00000000	n/a	n/a	n/a	n/a
7	0x00000000	0x00000000	n/a	n/a	n/a	n/a
8	0x00000000	0x00000000	n/a	n/a	n/a	n/a
9	0x00000000	0x00000000	n/a	n/a	n/a	n/a
10	0x00000000	0x00000000	n/a	n/a	n/a	n/a

---Type <return> to continue, or q <return> to quit---

(cuda-gdb) **info cuda lanes**

Ln	State	Physical PC	ThreadIdx	Exception
Device 0 SM 14 Warp 0				
* 0	active	0x0000000000000110	(0,0,0)	Device Illegal Address
1	active	0x0000000000000110	(1,0,0)	Device Illegal Address
2	active	0x0000000000000110	(2,0,0)	Device Illegal Address
3	active	0x0000000000000110	(3,0,0)	Device Illegal Address
4	active	0x0000000000000110	(4,0,0)	Device Illegal Address
5	active	0x0000000000000110	(5,0,0)	Device Illegal Address
6	active	0x0000000000000110	(6,0,0)	Device Illegal Address
7	active	0x0000000000000110	(7,0,0)	Device Illegal Address
8	active	0x0000000000000110	(8,0,0)	Device Illegal Address
9	active	0x0000000000000110	(9,0,0)	Device Illegal Address
10	inactive	n/a	n/a	n/a
11	inactive	n/a	n/a	n/a
12	inactive	n/a	n/a	n/a
...				
29	inactive	n/a	n/a	n/a
30	inactive	n/a	n/a	n/a
31	inactive	n/a	n/a	n/a

Homework

For the given program, what sequence of *cuda-gdb* commands would you use to identify the error?

```
__global__ void K(int *p) {
    *p = 0;
    printf("%d\n", *p);
}
int main() {
    int *x, *y;
    cudaMalloc(&x, sizeof(int));

    K<<<2, 10>>>(x);
    cudaDeviceSynchronize();

    y = x;
    cudaFree(y);

    K<<<2, 10>>>(x);
    cudaDeviceSynchronize();
    return 0;
}
```

Profiling

- Measuring "indicators" of performance

  – Time taken by various kernels

  – Memory utilization

  – Number of cache misses

  – Degree of divergence

  – Degree of coalescing

  – …

- Intrusive versus non-intrusive

CUDA Profiler

- **nvprof**: command-line

- **nvvp**, **nsight**: Visual Profilers

- An **event** is a measurable activity on a device. It corresponds to a hardware counter value.
  - About 140 events
  - tex0_cache_sector_queries
  - gld_inst_8bit
  - inst_executed
  - ...

nvprof

- No changes required to the binary. Uses defaults.

    - nvprof a.out

- To profile part of a program, use *cudaProfilerStart()* and *Stop()*.

    - Include *cuda_profiler_api.h*

    - nvprof --profile-from-start off a.out

```
__global__ void K1(int num) {
    num += num;
    ++num;
}
__device__ int sum = 0;
__global__ void K2(int num) {
    atomicAdd(&sum, num);
}

__global__ void K3(int num) {
    __shared__ int sum;
    sum = 0;
    __syncthreads();
    sum += num;
}
int main() {
    for (unsigned ii = 0; ii < 100; ++ii) {
        K1<<<5, 32>>>(ii);       cudaDeviceSynchronize();
    }
    for (unsigned ii = 0; ii < 100; ++ii) {
        K2<<<5, 32>>>(ii);       cudaDeviceSynchronize();
    }
    for (unsigned ii = 0; ii < 100; ++ii) {
        K3<<<5, 32>>>(ii);       cudaDeviceSynchronize();
    }
    return 0;
}
```

Which kernel should you optimize?
(Which kernel consumes more time?)

```
$ nvprof  a.out


==26519== NVPROF is profiling process 26519, command: a.out
==26519== Profiling application: a.out
==26519== Profiling result:
Time(%)      Time     Calls      Avg      Min      Max  Name
 39.46%  191.46us       100  1.9140us  1.8880us  2.1440us  K2(int)
 33.86%  164.26us       100  1.6420us  1.6000us  1.8880us  K3(int)
 26.68%  129.44us       100  1.2940us  1.2480us  1.5360us  K1(int)


==26519== API calls:
Time(%)      Time     Calls      Avg      Min      Max  Name
 95.75%  369.08ms       300  1.2303ms  10.560us  364.03ms  cudaLaunch
  2.33%  8.9986ms       728  12.360us    186ns  619.78us  cuDeviceGetAttribute
  0.91%  3.5039ms         8  437.98us  396.85us  450.61us  cuDeviceTotalMem
  0.73%  2.8134ms       300  9.3780us  6.4650us  32.547us  cudaDeviceSynchronize
  0.18%  699.99us         8  87.498us  85.431us  90.737us  cuDeviceGetName
  0.05%  194.20us       300    647ns    339ns  10.694us  cudaConfigureCall
  0.04%  156.27us       300    520ns    292ns  2.2700us  cudaSetupArgument
  0.00%  9.4130us        24    392ns    186ns    862ns  cuDeviceGet
  0.00%  5.7760us         3  1.9250us    317ns  4.7490us  cuDeviceGetCount
```

```cuda
__global__ void K1(int num) {
    num += num;
    ++num;
}
__device__ int sum = 0;
__global__ void K2(int num) {
    atomicAdd(&sum, num);
}

__global__ void K3(int num) {
    __shared__ int sum;
    sum = 0;
    __syncthreads();
    sum += num;
}
int main() {
    for (unsigned ii = 0; ii < 100; ++ii) {
        K1<<<5, 32>>>(ii);
        K2<<<5, 32>>>(ii);
        K3<<<5, 32>>>(ii);
        cudaDeviceSynchronize();
    }
    return 0;
}
```

One natural optimization, subject to application semantics, is to merge the loops.

Loop fusion

Doesn't change output much.
We need kernel fusion.

```
__device__ int sumg = 0;
__global__ void K(int num) {
    num += num;
    ++num;
    atomicAdd(&sumg, num);
    __shared__ int sum;
    sum = 0;
    __syncthreads();
    sum += num;
}
int main() {
    for (unsigned ii = 0; ii < 100; ++ii) {
        K<<<5, 32>>>(ii);
        cudaDeviceSynchronize();
    }
    return 0;
}
```

```
        Time(%)    Time   Calls    Avg      Min      Max  Name
        100.00%  194.81us     100  1.9480us  1.9200us  2.3040us  K(int)

        ==26721== API calls:
        Time(%)    Time   Calls    Avg      Min      Max  Name
        96.26%  375.85ms     100  3.7585ms  18.175us  373.93ms  cudaLaunch
         2.33%  9.1124ms     728  12.517us    183ns  661.26us  cuDeviceGetAttribute
        ...
```

Time(%)	Time	Calls	Avg	Min	Max	Name
39.46%	191.46us	100	1.9140us	1.8880us	2.1440us	K2(int)
33.86%	164.26us	100	1.6420us	1.6000us	1.8880us	K3(int)
26.68%	129.44us	100	1.2940us	1.2480us	1.5360us	K1(int)

==26519== API calls:

Time(%)	Time	Calls	Avg	Min	Max	Name
95.75%	369.08ms	300	1.2303ms	10.560us	364.03ms	cudaLaunch
2.33%	8.9986ms	728	12.360us	186ns	619.78us	cuDeviceGetAttribute

Time(%)	Time	Calls	Avg	Min	Max	Name
100.00%	194.81us	100	1.9480us	1.9200us	2.3040us	K(int)

==26721== API calls:

Time(%)	Time	Calls	Avg	Min	Max	Name
96.26%	375.85ms	100	3.7585ms	18.175us	373.93ms	cudaLaunch
2.33%	9.1124ms	728	12.517us	183ns	661.26us	cuDeviceGetAttribute
...						

```cuda
__device__ int sumg = 0;
__global__ void K(int num) {
    int num = blockIdx.x * blockDim.x + threadIdx.x;
    num += num;
    ++num;
    atomicAdd(&sumg, num);
    __shared__ int sum;
    sum = 0;
    __syncthreads();
    sum += num;
}
int main() {
    K<<<5*100, 32>>>(ii);
    cudaDeviceSynchronize();
    return 0;
}
```

> What if the launch configuration is changed to <<<100, 5*32>>>?

```cuda
int main() {
    for (unsigned ii = 0; ii < 100; ++ii) {
        K<<<5, 32>>>(ii);
        cudaDeviceSynchronize();
    }
    return 0;
}
```

```
        Time(%)    Time   Calls    Avg       Min      Max  Name
        100.00%  3.7120us      1  3.7120us  3.7120us  3.7120us  K(void)

==26862== API calls:
        Time(%)    Time   Calls    Avg       Min      Max  Name
        96.91%  369.68ms      1  369.68ms  369.68ms  369.68ms  cudaLaunch
         2.16%  8.2407ms    728  11.319us    142ns  461.62us  cuDeviceGetAttribute
```

nvprof

- Supports device-specific profiling

- Supports remote profiling

- Output can be dumped to files as a .csv

- ...