3 回复 由 LeeHowes 于 2011-7-18 下午9:15 最新回复

asy1502 2011-7-18 下午5:02

Multiple GPU OpenCL kernel execution being serialized

I have been working on multiple GPUs (2x Firepro 3D 7800) on Ubuntu 10.04 x86_64. I have created two simple examples 1 a vector addition and one that creates a negative of an image.
I have them executing successfully and the time returned from the queue event shows perfect speed-up over a single GPU. The problem is that I have timers around the execute kernel statement. From those timers, I see no speed-up and usually a speed decrease.
I began printing out queue submit and queue start times. I found that the second gpu kernel won't begin execution until the first ends.
I have the latest driver 8.86.5.
Below is my kernel launch code:

```
timers[timer_name[timer_num+1]]->start();
#pragma omp parallel for private(i)//, schedule(static,1)
for(i = 0; i<num_gpus; i++)
{
   try
   {
      cli->err = cli->queue.enqueueNDRangeKernel(kernels,cl::NullRange,  cl::NDRange(x,y/num_gpus),cl::NullRange , NULL, &event_execute);
   }
   catch (cl::Error er)
   {
      printf("j = %d, num_gpus = %d, i = %d\n",j,num_gpus,i);
      printf("ERROR: %s(%s)\n", er.what(), oclErrorString(er.err()));
   }
}
for(i = 0; i<num_gpus; i++)
{
   cli->queue.finish();
```

```
        }
        timers[timer_name[timer_num+1]]->stop();
```

Here are my printouts from the negative image kernel:

```
Negative calculation on GPU # 1 of 1:
    Submit Time:        248512122.967647999525070
    Queue Time:         248512122.961299985647202
    Start Time:         248512123.145188987255096
    End Time:           248512126.960956990718842
    Minimum Time:         2.731202000000000
    Maximum Time:         3.815768000000000
    Average Time:         2.865470800000000
    Total Time:          28.654707999999996
    Count:        10

Negative calculation on GPU # 1 of 2:
    Submit Time:        248512260.716033995151520
    Queue Time:         248512260.707136988639832
    Start Time:         248512260.910378992557526
    End Time:           248512262.280068993568420
    Minimum Time:         1.368542000000000
    Maximum Time:         2.042697000000000
    Average Time:         1.528395200000000
    Total Time:          15.283951999999999
    Count:        10

Negative calculation on GPU # 2 of 2:
    Submit Time:        248512262.304941982030869
    Queue Time:         248512260.690681993961334
    Start Time:         248512262.438205987215042
    End Time:           248512263.875981003046036
    Minimum Time:         1.368398000000000
    Maximum Time:         2.052437000000000
    Average Time:         1.572541900000000
    Total Time:          15.725419000000000
    Count:        10
```

```
manipulating img on 1 GPUs    | avg:   4.4911 | tot:   44.9110 | count=   10
manipulating img on 2 GPUs    | avg:   3.8605 | tot:   38.6050 | count=   10
```

1059 查看　　标签：

nou 2011-7-18 下午5:35 （回复 asy1502）

Multiple GPU OpenCL kernel execution being serialized

you call finish() which is wrong in single-thread multi device enviroment. when you enqueue kernel it dont start execution. you must call flush() on queue to start execution. after that you can call some blocking call like finish()

操作　　喜欢 (0)

asy1502 2011-7-18 下午5:39 （回复 nou）

Multiple GPU OpenCL kernel execution being serialized

Thanks!! I had the misconception that finish() implicitly called flush()!

操作　　喜欢 (0)

LeeHowes 2011-7-18 下午9:15 （回复 asy1502）

Multiple GPU OpenCL kernel execution being serialized

It does call flush. What it does not do is call flush on every queue. So in your loop you finish on one queue - you flush and block on that queue. Then you flush and block on the next. Of course, as you didn't flush the second when you blocked on the first you serialised.
Split your code. Loop to flush. Loop to finish. Or, more cleanly, maybe, build an event list with the last event in each queue and wait on the event set and block on all queues at once.

操作　　喜欢 (0)

主页 | 页面顶部 | 帮助