Trevor Stahl

CS 475 Spring 2019

Project 5

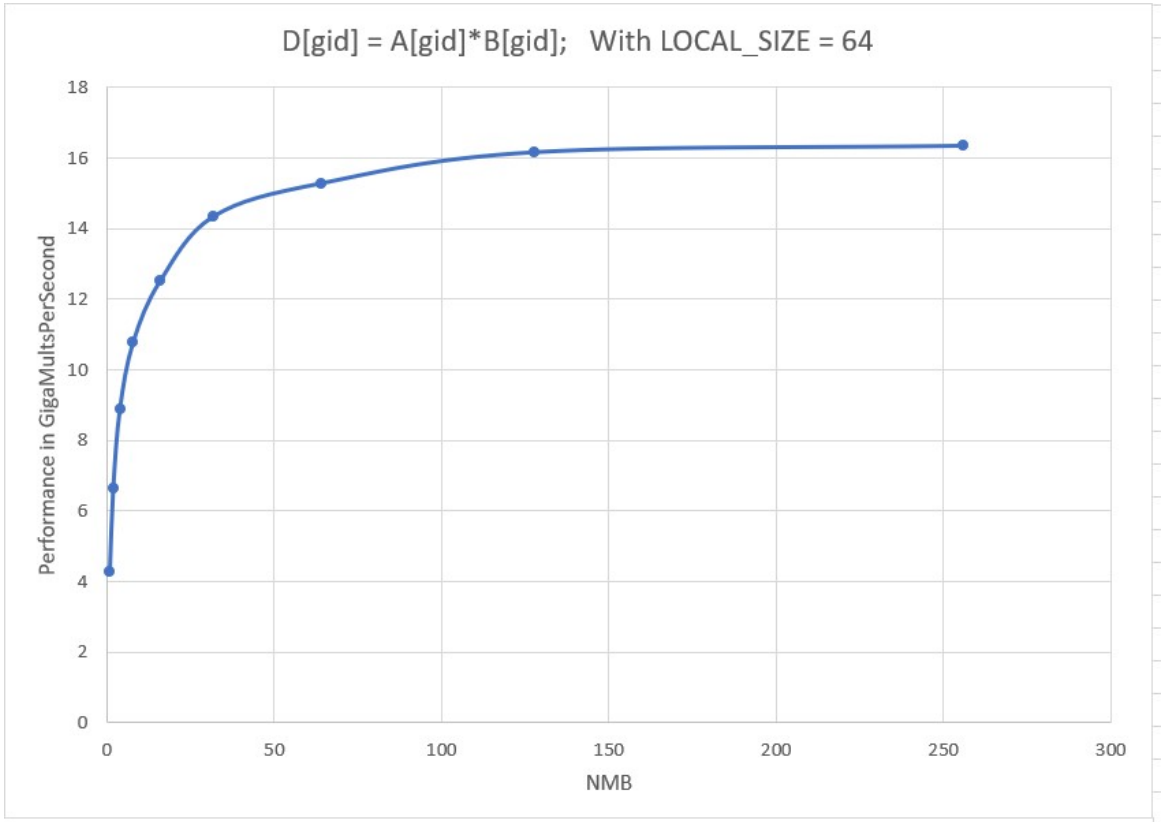
Professor Bailey

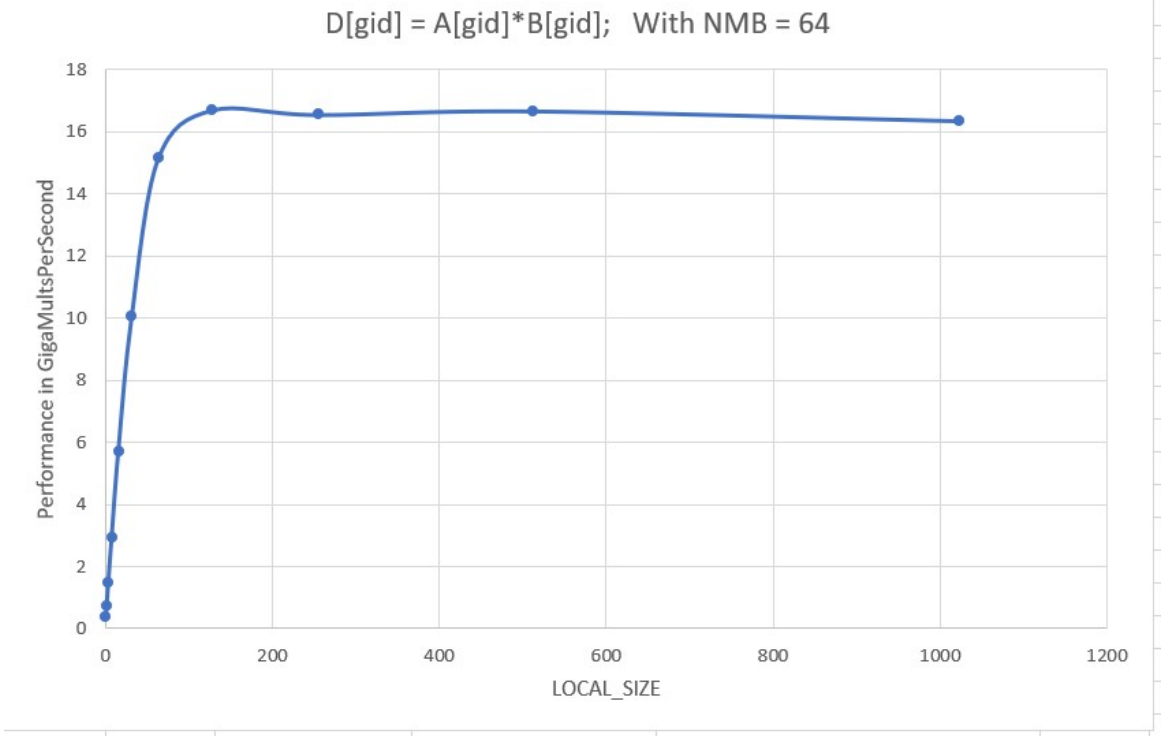
What machine you ran this on:

I ran my code on rabbit. Uptime load averages were low before and after. I ran in the morning on day after due date so it was less active. I am using a bonus day of course.

Show the tables and graphs for parts 1 and 2:
Part 1

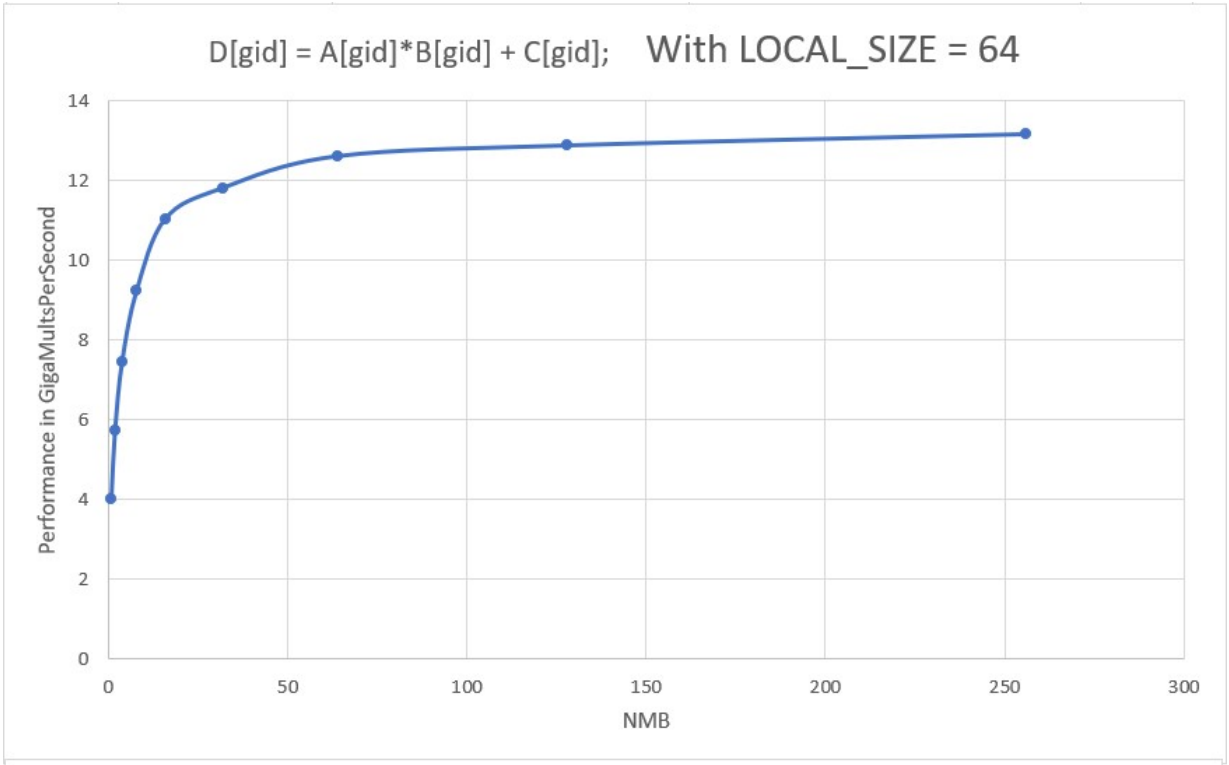
		Part 1	
NMB	LOCAL_SIZE	NUM_WORK_GROUPS	Performance in GigaMultsPerSecond
64	1	67108864	0.378
64	2	33554432	0.74
64	4	16777216	1.486
64	8	8388608	2.945
64	16	4194304	5.712
64	32	2097152	10.033
64	64	1048576	15.134
64	128	524288	16.684
64	256	262144	16.541
64	512	131072	16.658
64	1024	65536	16.343
1	64	16384	4.289
2	64	32768	6.619
4	64	65536	8.886
8	64	131072	10.789
16	64	262144	12.503
32	64	524288	14.337
64	64	1048576	15.267
128	64	2097152	16.15
256	64	4194304	16.332

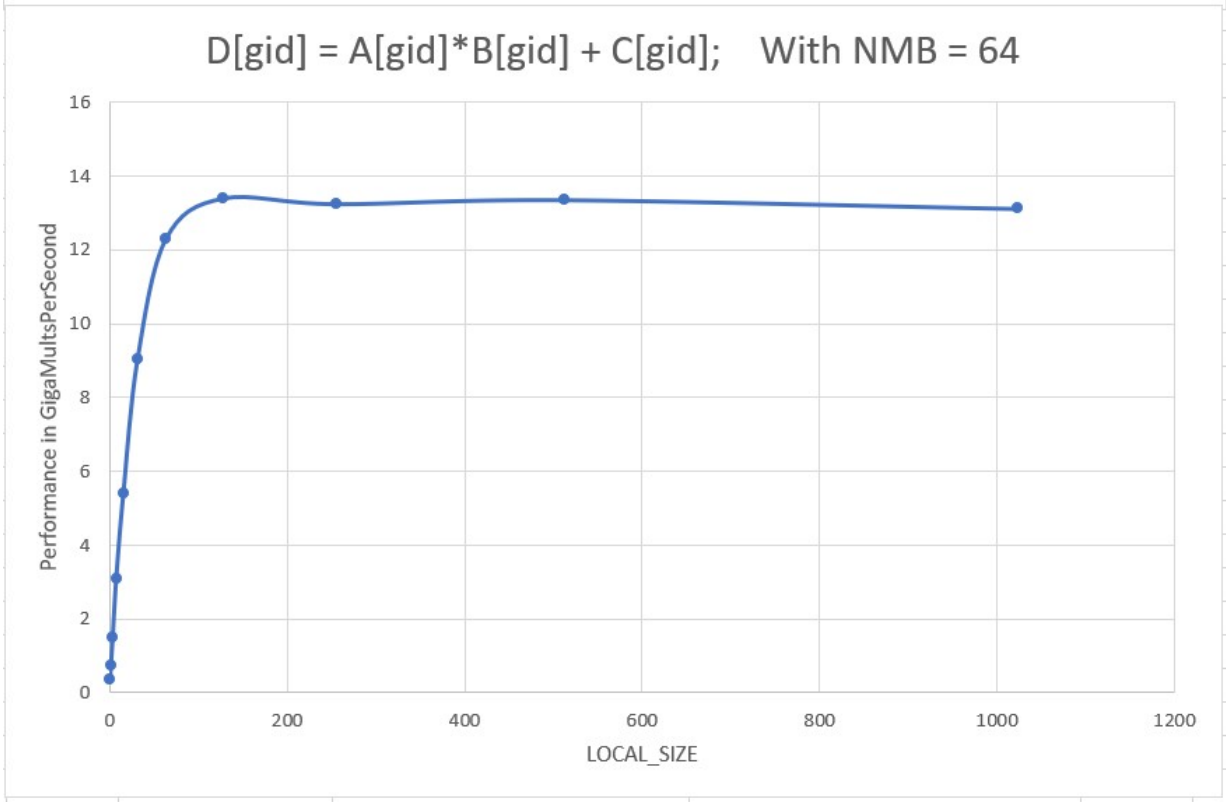

D[gid] = A[gid]*B[gid]; With LOCAL_SIZE = 64
Performance in GigaMultsPerSecond
NMB
0
2
4
6
8
10
12
14
16
18
0
50
100
150
200
250
300


D[gid] = A[gid]*B[gid]; With NMB = 64
Performance in GigaMultsPerSecond
LOCAL_SIZE
0
2
4
6
8
10
12
14
16
18
0
200
400
600
800
1000
1200

Part 2

		Part 2	
NMB	LOCAL_SIZE	NUM_WORK_GROUPS	Performance in GigaMultsPerSecond
64	1	67108864	0.375
64	2	33554432	0.738
64	4	16777216	1.493
64	8	8388608	3.105
64	16	4194304	5.401
64	32	2097152	9.046
64	64	1048576	12.312
64	128	524288	13.41
64	256	262144	13.259
64	512	131072	13.367
64	1024	65536	13.128
1	64	16384	4.032
2	64	32768	5.728
4	64	65536	7.472
8	64	131072	9.237
16	64	262144	11.043
32	64	524288	11.816
64	64	1048576	12.611
128	64	2097152	12.88
256	64	4194304	13.162


D[gid] = A[gid]*B[gid] + C[gid]; With LOCAL_SIZE = 64
Performance in GigaMultsPerSecond
NMB
0
2
4
6
8
10
12
14
0
50
100
150
200
250
300


D[gid] = A[gid]*B[gid] + C[gid]; With NMB = 64
Performance in GigaMultsPerSecond
LOCAL_SIZE
0
2
4
6
8
10
12
14
16
0
200
400
600
800
1000
1200

What patterns are you seeing in the performance curves?

-Performance starts out very low. Then as the values for NMB/LOCAL_SIZE increase the performance rapidly increases until it hits a plateau.

-When NMB is held constant at 64 however, we see the best performance towards the left hand side. Once it peeks it dips down a bit and stays about there.

-On the other hand, when LOCAL_SIZE is held constant the best performance is the right most point. There is no point on these curves where slope is negative.

Why do you think the patterns look this way?
I am not sure why there is a dip back down when holding NMB constant. ~~Given how local work group and global work size are related this makes some sense, because if NMB is held constant but local_size is increased decreasing need for communication b~~

The curves overall though have expected appearances. When the values are too small there is not enough work to make going to the gpu and back worth it. Kind of like spinning up threads. So this causes a sharp increase in the far left side of the graphs starting from basically 0, the bottom. The curves shoots up until it hits a plateau where adding more data does not increase how much is done per second.

What is the performance difference between doing a Multiply and doing a Multiply-Add?
MultAdd tops out at around 13 for me. While regular mult tops out at around 16. This means normal multiply is faster, of course. With Reduction we have much slower performance.

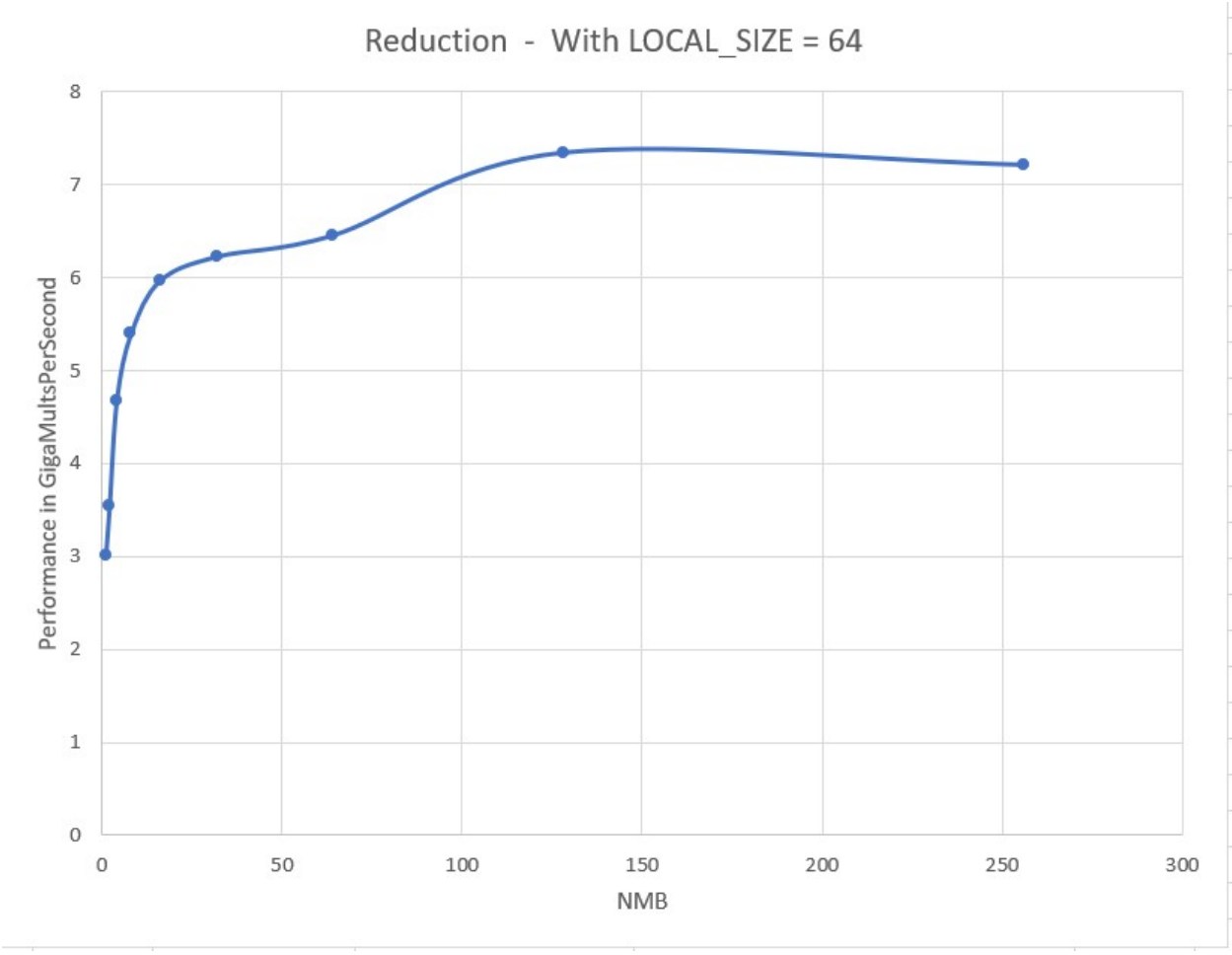
What does that mean for the proper use of GPU parallel computing?
~~This would suggest to me that the proper use for the GPU is to have one instruction~~

This would suggest to me that the best use of the GPU is to put as many instructions/ steps on it as possible. When simply multiplying we get 16 GigaMults. When we multiply and then add we get 13.5 or so. This is less, however, we are doing more because there is another parameter with another instruction. This is probably not twice as much work but is more than the change in performance. Meaning more work is done. Basically, there is the cost to go to and from the GPU and all the CL setup. If we increase what is being done on the GPU then overall everything is more efficient.

Also make sure there is enough data to make using the GPU worth it.

Show the table and graph for part 3:

		Part 3	
NMB	LOCAL_SIZE	NUM_WORK_GROUPS	Performance in GigaMultsPerSecond
1	64	16384	3.014
2	64	32768	3.542
4	64	65536	4.674
8	64	131072	5.4
16	64	262144	5.964
32	64	524288	6.226
64	64	1048576	6.454
128	64	2097152	7.345
256	64	4194304	7.214



What pattern are you seeing in this performance curve?

- There is the typical and expected sharp rise in the left most area of graph.
- There are two humps.
- The right most point is not the point with the highest performance.

Why do you think the pattern looks this way?

- The sharp rise and low start on the left side of graph is because there is not enough data to make setting things up and going to the GPU worth it yet.

- I am also not sure why the reduction graph has two humps, or why the right most point is not the highest performance point.

What does that mean for the proper use of GPU parallel computing?

- Make sure there is enough data to make using the GPU worth it.
- I am not sure what else this could mean.