The purpose of this assignment was to test and demonstrate performance trends of GPUs when multiplying, multiplying and adding, and multiplying and reducing large arrays of numbers by using OpenCL.

Question 1

What machine did you run the experiment on?

The experiment was run on Rabbit at a time when the load average was approximately 0.75.

Question 2

Show tables and graphs of the array multiplication and array multiplication with addition timing experiments.

Graphs of the computation speed versus the global size and local work group size can be found in Figures 1 and 2. A table of the values used to generate these graphs can be found in Table 1 of Appendix 1.1.

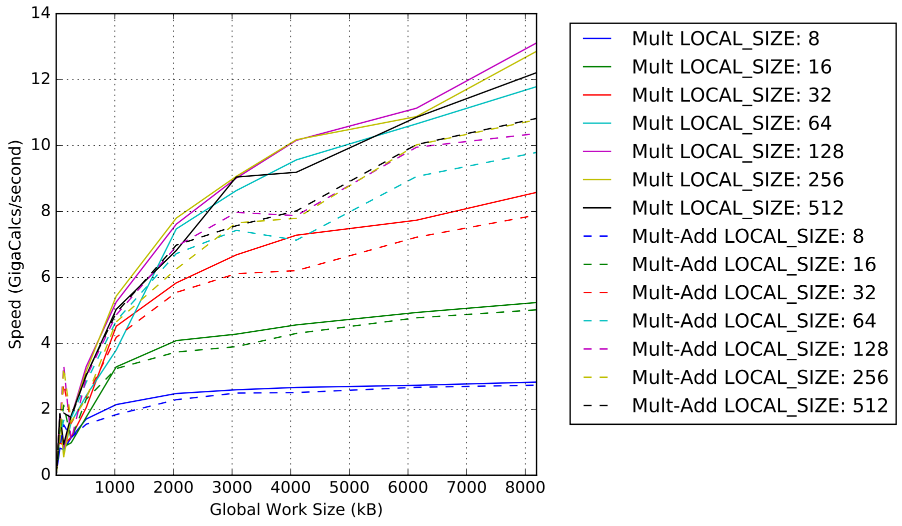

Figure 1: Speed of the multiplication and multiplication with addition experiments vs. global size. Note that the solid lines signify multiplication, while the dashed lines signify multiplication with addition.

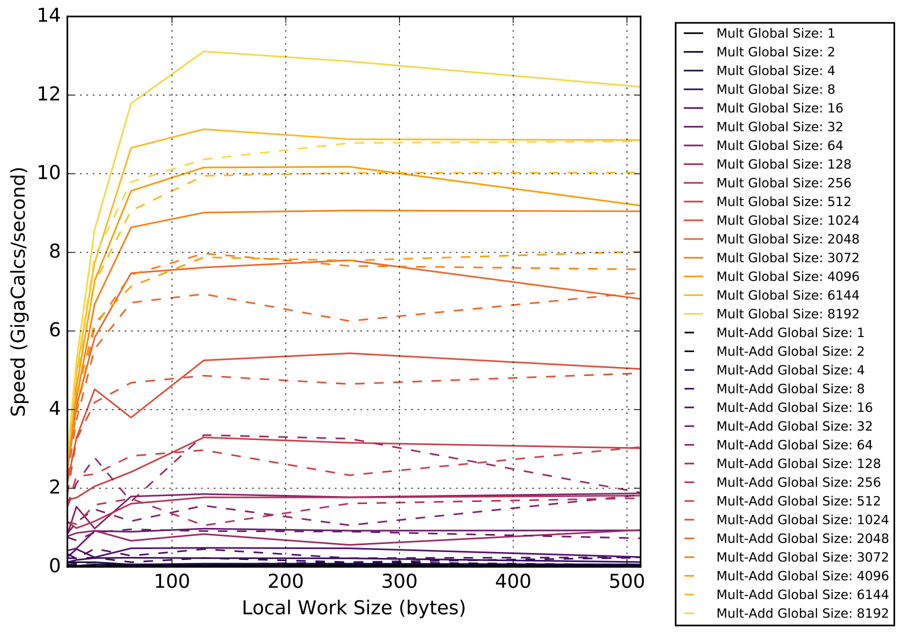

Figure 2: Speed of the multiplication and multiplication with addition experiments vs. local size. Note that the solid lines signify multiplication, while the dashed lines signify multiplication with addition.

Question 3

What patterns are you seeing in the performance curves of array multiplication and array multiplication-addition?

As seen in Figure 1, as the global work size increases, the performance increases for almost all values of the local work size. For some values of the local work size, particularly smaller values (`LOCAL_WORK_SIZE` $<\sim 16$), the performance begins to hit a plateau as the global work size increased. It is possible that with global work sizes much greater than 8MB, the performance will plateau for all of the larger values of `LOCAL_WORK_SIZE`.

As seen in Figure 2, the performance is largely flat beyond a `LOCAL_WORK_SIZE` of approximately 128 bytes, and may slightly degrade beyond a `LOCAL_WORK_SIZE` of approximately 256. Thus, there may be an optimal value of the local work size for each value of the global work size, however more testing would be needed to verify these results.

When either very small global work sizes or local work sizes were selected, performance was very poor.

Question 4

Why do you think the patterns of array multiplication and array multiplication-addition look this way?

The performance increases as the global work size increases because the experiment may be easily run in parallel. There is a significant amount of computational overhead in setting up the problem to be run on a GPU versus a CPU, and therefore at small values of the global work size the parallelism is not worth the setup cost. However, as the global work size gets bigger, the parallel fraction increases, making the problem more optimal for a GPU setup with OpenCL.

Having a local work size too small is not efficient because the GPU only has a limited number of processing elements, and with too small a work group each work group will be assigned a small amount of work at a given point in time. It will therefore need to have have work reassigned to it often, which degrades performance. With larger local work sizes, this will not be an issue. Performance remains relatively flat as the local work size increases because the overhead becomes a relatively or even insignificantly small portion of the overall computational cost. However, performance may begin to degrade as the local work size gets to be large due to issues storing large amounts of data in memory on the GPU.

Question 5

What is the performance difference between doing a multiply and doing a multiply-add?

The performance of multiplication was in all cases slightly greater than the performance of multiplication with addition for the same global work size and local work size. The number of calculations per second does not take into account that multiplication operations consisted of only one calculation, while operations of multiplication with addition consisted of two calculations. Thus, although the performance dropped slightly with multiplication with addition versus multiplication alone, the actual number of calculations per second may be greater due to the increased amount of work done. This may be due to the increased amount of instructions associated with a similar overhead, or it may have to do with the fact that GPU hardware supports simultaneous multiplication and addition in some circumstances.

Question 6

What does the difference in performance between doing a multiply and a multiply-add mean for the proper use of GPU parallel computing?

Although multiplication alone is faster than multiplication with addition, the difference between the two is not huge. Because multiplication with addition involves more computations than multiplication alone, it may be more efficient to do multiplication with addition than multiplication alone with addition separate, and the two should be done together whenever both operations are needed.

Question 7

Show a table and graph for the performance of the multiplication-reduction experiment.

Graphs of the computation speed versus the global size and local work group size can be found in Figures 3 and 4. A table of the values used to generate these graphs can be found along with the values for multiplication and multiplication with addition in Table 1 of Appendix 1.1.

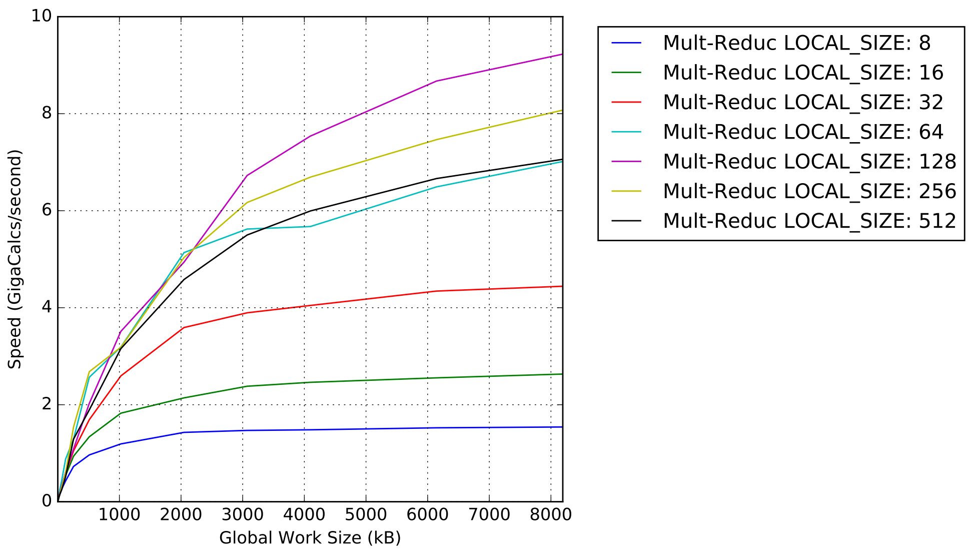

Figure 3: Speed of the multiplication with reduction experiment vs. global size.

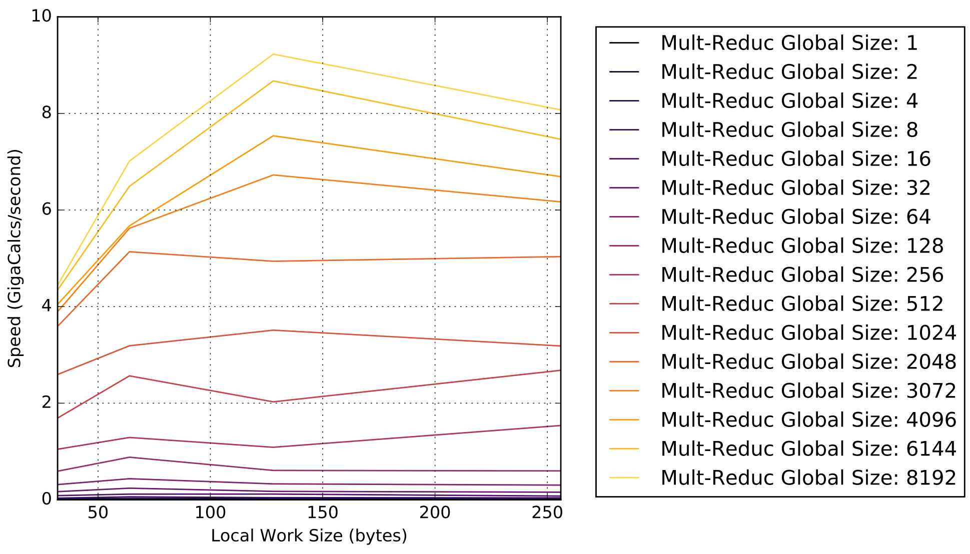

Figure 4: Speed of the multiplication with reduction experiment vs. local size.

Question 8

What patterns are you seeing in the performance curves of the multiplication-reduction experiment?

The performance curves of the multiplication with reduction in both Figures 3 and 4 are largely similar to the curves seen earlier in Figures 1 and 2. The performance appears to plateau slightly more with multiplication with reduction than it does for either multiplication or multiplication with addition as the global work size gets larger.

The performance of the multiplication-reduction experiment appears to peak more with a `LOCAL_WORK_SIZE` of 128 than it did with either the multiplication or the multiplication with addition experiments.

Overall, the performance of the multiplication-reduction experiment was overall similar to the performance of the multiplication with addition experiment.

Question 9

Why do you think the patterns of the multiplication-reduction experiment look this way?

The reasons that the patterns of the multiplication-reduction experiment look the way they do are similar to the reasons that the multiplication and multiplication with addition experiments exhibit the same trends.

The reasoning for the increased tendency of the multiplication-reduction experiment to plateau may be because the data eventually becomes less and less parallel as the reduction progresses. With data that is not highly parallel, the performance of the GPU diminishes when compared to the performance of a CPU.

Question 10

What do the patterns in the multiplication-reduction experiment mean for the proper use of GPU parallel computing?

The GPU can be used to efficiently multiply and reduce a set of data. The high performance of the operation means that with large sets of data to be handled in this manner, a GPU should be employed, with larger sets of data corresponding to an increase in performance.

1 Appendices

1.1 Tables of Generated Data

Table 1: Raw Data Generated

Mode	NKB	LOCAL_SIZE	NUM_WORK_GROUPS	GigaCalcPerSecond
Multiply	1	8	128	0.028
Multiply	2	8	256	0.057
Multiply	4	8	512	0.120
Multiply	8	8	1024	0.118
Multiply	16	8	2048	0.249
Multiply	32	8	4096	0.414
Multiply	64	8	8192	0.822
Multiply	128	8	16384	0.777
Multiply	256	8	32768	1.139
Multiply	512	8	65536	1.705
Multiply	1024	8	131072	2.143
Multiply	2048	8	262144	2.479
Multiply	3072	8	393216	2.592
Multiply	4096	8	524288	2.663
Multiply	6144	8	786432	2.728
Multiply	8192	8	1048576	2.825
Multiply	1	16	64	0.029
Multiply	2	16	128	0.027
Multiply	4	16	256	0.119
Multiply	8	16	512	0.176
Multiply	16	16	1024	0.452
Multiply	32	16	2048	0.471
Multiply	64	16	4096	1.532
Multiply	128	16	8192	0.861
Multiply	256	16	16384	0.985
Multiply	512	16	32768	1.760
Multiply	1024	16	65536	3.286
Multiply	2048	16	131072	4.084
Multiply	3072	16	196608	4.280
Multiply	4096	16	262144	4.560
Multiply	6144	16	393216	4.936
Multiply	8192	16	524288	5.238
Multiply	1	32	32	0.017
Multiply	2	32	64	0.061
Multiply	4	32	128	0.118
Multiply	8	32	256	0.225
Multiply	16	32	512	0.243
Multiply	32	32	1024	0.914
Multiply	64	32	2048	0.976
Multiply	128	32	4096	0.920
Multiply	256	32	8192	1.160
Multiply	512	32	16384	2.048
Multiply	1024	32	32768	4.519
Multiply	2048	32	65536	5.834
Multiply	3072	32	98304	6.684
Multiply	4096	32	131072	7.281
Multiply	6144	32	196608	7.735
Multiply	8192	32	262144	8.579
Multiply	1	64	16	0.030

Multiply	2	64	32	0.061
Multiply	4	64	64	0.061
Multiply	8	64	128	0.232
Multiply	16	64	256	0.473
Multiply	32	64	512	0.894
Multiply	64	64	1024	1.790
Multiply	128	64	2048	0.664
Multiply	256	64	4096	1.607
Multiply	512	64	8192	2.413
Multiply	1024	64	16384	3.799
Multiply	2048	64	32768	7.471
Multiply	3072	64	49152	8.634
Multiply	4096	64	65536	9.565
Multiply	6144	64	98304	10.657
Multiply	8192	64	131072	11.786
Multiply	1	128	8	0.028
Multiply	2	128	16	0.058
Multiply	4	128	32	0.082
Multiply	8	128	64	0.223
Multiply	16	128	128	0.483
Multiply	32	128	256	0.972
Multiply	64	128	512	1.853
Multiply	128	128	1024	0.834
Multiply	256	128	2048	1.768
Multiply	512	128	4096	3.294
Multiply	1024	128	8192	5.253
Multiply	2048	128	16384	7.615
Multiply	3072	128	24576	9.012
Multiply	4096	128	32768	10.159
Multiply	6144	128	49152	11.132
Multiply	8192	128	65536	13.112
Multiply	1	256	4	0.030
Multiply	2	256	8	0.060
Multiply	4	256	16	0.082
Multiply	8	256	32	0.217
Multiply	16	256	64	0.472
Multiply	32	256	128	0.921
Multiply	64	256	256	1.773
Multiply	128	256	512	0.559
Multiply	256	256	1024	1.766
Multiply	512	256	2048	3.157
Multiply	1024	256	4096	5.433
Multiply	2048	256	8192	7.799
Multiply	3072	256	12288	9.065
Multiply	4096	256	16384	10.177
Multiply	6144	256	24576	10.878
Multiply	8192	256	32768	12.858
Multiply	1	512	2	0.029
Multiply	2	512	4	0.059
Multiply	4	512	8	0.060
Multiply	8	512	16	0.123
Multiply	16	512	32	0.250
Multiply	32	512	64	0.928
Multiply	64	512	128	1.870
Multiply	128	512	256	0.939
Multiply	256	512	512	1.812

Multiply	512	512	1024	3.019
Multiply	1024	512	2048	5.034
Multiply	2048	512	4096	6.816
Multiply	3072	512	6144	9.045
Multiply	4096	512	8192	9.190
Multiply	6144	512	12288	10.857
Multiply	8192	512	16384	12.210
Multiply-Add	1	8	128	0.031
Multiply-Add	2	8	256	0.057
Multiply-Add	4	8	512	0.112
Multiply-Add	8	8	1024	0.219
Multiply-Add	16	8	2048	0.333
Multiply-Add	32	8	4096	0.753
Multiply-Add	64	8	8192	0.847
Multiply-Add	128	8	16384	1.530
Multiply-Add	256	8	32768	1.143
Multiply-Add	512	8	65536	1.547
Multiply-Add	1024	8	131072	1.840
Multiply-Add	2048	8	262144	2.292
Multiply-Add	3072	8	393216	2.492
Multiply-Add	4096	8	524288	2.508
Multiply-Add	6144	8	786432	2.667
Multiply-Add	8192	8	1048576	2.732
Multiply-Add	1	16	64	0.029
Multiply-Add	2	16	128	0.049
Multiply-Add	4	16	256	0.089
Multiply-Add	8	16	512	0.221
Multiply-Add	16	16	1024	0.209
Multiply-Add	32	16	2048	0.673
Multiply-Add	64	16	4096	1.136
Multiply-Add	128	16	8192	2.126
Multiply-Add	256	16	16384	1.085
Multiply-Add	512	16	32768	2.304
Multiply-Add	1024	16	65536	3.235
Multiply-Add	2048	16	131072	3.737
Multiply-Add	3072	16	196608	3.901
Multiply-Add	4096	16	262144	4.302
Multiply-Add	6144	16	393216	4.775
Multiply-Add	8192	16	524288	5.017
Multiply-Add	1	32	32	0.030
Multiply-Add	2	32	64	0.030
Multiply-Add	4	32	128	0.115
Multiply-Add	8	32	256	0.234
Multiply-Add	16	32	512	0.465
Multiply-Add	32	32	1024	0.877
Multiply-Add	64	32	2048	1.466
Multiply-Add	128	32	4096	2.779
Multiply-Add	256	32	8192	1.578
Multiply-Add	512	32	16384	2.358
Multiply-Add	1024	32	32768	4.183
Multiply-Add	2048	32	65536	5.545
Multiply-Add	3072	32	98304	6.114
Multiply-Add	4096	32	131072	6.206
Multiply-Add	6144	32	196608	7.219
Multiply-Add	8192	32	262144	7.892
Multiply-Add	1	64	16	0.015

Multiply-Add	2	64	32	0.061
Multiply-Add	4	64	64	0.058
Multiply-Add	8	64	128	0.120
Multiply-Add	16	64	256	0.284
Multiply-Add	32	64	512	0.955
Multiply-Add	64	64	1024	1.163
Multiply-Add	128	64	2048	1.664
Multiply-Add	256	64	4096	1.736
Multiply-Add	512	64	8192	2.821
Multiply-Add	1024	64	16384	4.685
Multiply-Add	2048	64	32768	6.721
Multiply-Add	3072	64	49152	7.428
Multiply-Add	4096	64	65536	7.131
Multiply-Add	6144	64	98304	9.060
Multiply-Add	8192	64	131072	9.791
Multiply-Add	1	128	8	0.027
Multiply-Add	2	128	16	0.057
Multiply-Add	4	128	32	0.058
Multiply-Add	8	128	64	0.224
Multiply-Add	16	128	128	0.451
Multiply-Add	32	128	256	0.911
Multiply-Add	64	128	512	1.558
Multiply-Add	128	128	1024	3.354
Multiply-Add	256	128	2048	1.055
Multiply-Add	512	128	4096	2.969
Multiply-Add	1024	128	8192	4.863
Multiply-Add	2048	128	16384	6.940
Multiply-Add	3072	128	24576	7.972
Multiply-Add	4096	128	32768	7.875
Multiply-Add	6144	128	49152	9.948
Multiply-Add	8192	128	65536	10.364
Multiply-Add	1	256	4	0.030
Multiply-Add	2	256	8	0.059
Multiply-Add	4	256	16	0.119
Multiply-Add	8	256	32	0.120
Multiply-Add	16	256	64	0.222
Multiply-Add	32	256	128	0.898
Multiply-Add	64	256	256	1.059
Multiply-Add	128	256	512	3.261
Multiply-Add	256	256	1024	1.612
Multiply-Add	512	256	2048	2.331
Multiply-Add	1024	256	4096	4.651
Multiply-Add	2048	256	8192	6.253
Multiply-Add	3072	256	12288	7.655
Multiply-Add	4096	256	16384	7.793
Multiply-Add	6144	256	24576	10.020
Multiply-Add	8192	256	32768	10.782
Multiply-Add	1	512	2	0.031
Multiply-Add	2	512	4	0.059
Multiply-Add	4	512	8	0.053
Multiply-Add	8	512	16	0.235
Multiply-Add	16	512	32	0.243
Multiply-Add	32	512	64	0.727
Multiply-Add	64	512	128	1.832
Multiply-Add	128	512	256	1.888
Multiply-Add	256	512	512	1.748

Multiply-Add	512	512	1024	3.048
Multiply-Add	1024	512	2048	4.926
Multiply-Add	2048	512	4096	6.979
Multiply-Add	3072	512	6144	7.568
Multiply-Add	4096	512	8192	8.018
Multiply-Add	6144	512	12288	10.029
Multiply-Add	8192	512	16384	10.823
Multiply+Reduction	1	8	128	0.005
Multiply+Reduction	2	8	256	0.011
Multiply+Reduction	4	8	512	0.018
Multiply+Reduction	8	8	1024	0.035
Multiply+Reduction	16	8	2048	0.075
Multiply+Reduction	32	8	4096	0.138
Multiply+Reduction	64	8	8192	0.260
Multiply+Reduction	128	8	16384	0.430
Multiply+Reduction	256	8	32768	0.727
Multiply+Reduction	512	8	65536	0.965
Multiply+Reduction	1024	8	131072	1.192
Multiply+Reduction	2048	8	262144	1.430
Multiply+Reduction	3072	8	393216	1.470
Multiply+Reduction	4096	8	524288	1.483
Multiply+Reduction	6144	8	786432	1.524
Multiply+Reduction	8192	8	1048576	1.541
Multiply+Reduction	1	16	64	0.005
Multiply+Reduction	2	16	128	0.010
Multiply+Reduction	4	16	256	0.020
Multiply+Reduction	8	16	512	0.038
Multiply+Reduction	16	16	1024	0.083
Multiply+Reduction	32	16	2048	0.156
Multiply+Reduction	64	16	4096	0.281
Multiply+Reduction	128	16	8192	0.540
Multiply+Reduction	256	16	16384	0.938
Multiply+Reduction	512	16	32768	1.339
Multiply+Reduction	1024	16	65536	1.826
Multiply+Reduction	2048	16	131072	2.141
Multiply+Reduction	3072	16	196608	2.381
Multiply+Reduction	4096	16	262144	2.462
Multiply+Reduction	6144	16	393216	2.554
Multiply+Reduction	8192	16	524288	2.632
Multiply+Reduction	1	32	32	0.005
Multiply+Reduction	2	32	64	0.010
Multiply+Reduction	4	32	128	0.022
Multiply+Reduction	8	32	256	0.034
Multiply+Reduction	16	32	512	0.086
Multiply+Reduction	32	32	1024	0.168
Multiply+Reduction	64	32	2048	0.315
Multiply+Reduction	128	32	4096	0.592
Multiply+Reduction	256	32	8192	1.047
Multiply+Reduction	512	32	16384	1.692
Multiply+Reduction	1024	32	32768	2.594
Multiply+Reduction	2048	32	65536	3.591
Multiply+Reduction	3072	32	98304	3.895
Multiply+Reduction	4096	32	131072	4.046
Multiply+Reduction	6144	32	196608	4.343
Multiply+Reduction	8192	32	262144	4.442
Multiply+Reduction	1	64	16	0.005

Multiply+Reduction	2	64	32	0.015
Multiply+Reduction	4	64	64	0.027
Multiply+Reduction	8	64	128	0.060
Multiply+Reduction	16	64	256	0.117
Multiply+Reduction	32	64	512	0.240
Multiply+Reduction	64	64	1024	0.436
Multiply+Reduction	128	64	2048	0.881
Multiply+Reduction	256	64	4096	1.290
Multiply+Reduction	512	64	8192	2.566
Multiply+Reduction	1024	64	16384	3.189
Multiply+Reduction	2048	64	32768	5.136
Multiply+Reduction	3072	64	49152	5.622
Multiply+Reduction	4096	64	65536	5.673
Multiply+Reduction	6144	64	98304	6.491
Multiply+Reduction	8192	64	131072	7.014
Multiply+Reduction	1	128	8	0.005
Multiply+Reduction	2	128	16	0.014
Multiply+Reduction	4	128	32	0.031
Multiply+Reduction	8	128	64	0.044
Multiply+Reduction	16	128	128	0.118
Multiply+Reduction	32	128	256	0.174
Multiply+Reduction	64	128	512	0.329
Multiply+Reduction	128	128	1024	0.608
Multiply+Reduction	256	128	2048	1.087
Multiply+Reduction	512	128	4096	2.028
Multiply+Reduction	1024	128	8192	3.512
Multiply+Reduction	2048	128	16384	4.939
Multiply+Reduction	3072	128	24576	6.726
Multiply+Reduction	4096	128	32768	7.537
Multiply+Reduction	6144	128	49152	8.673
Multiply+Reduction	8192	128	65536	9.228
Multiply+Reduction	1	256	4	0.005
Multiply+Reduction	2	256	8	0.009
Multiply+Reduction	4	256	16	0.018
Multiply+Reduction	8	256	32	0.039
Multiply+Reduction	16	256	64	0.077
Multiply+Reduction	32	256	128	0.157
Multiply+Reduction	64	256	256	0.304
Multiply+Reduction	128	256	512	0.598
Multiply+Reduction	256	256	1024	1.538
Multiply+Reduction	512	256	2048	2.681
Multiply+Reduction	1024	256	4096	3.185
Multiply+Reduction	2048	256	8192	5.035
Multiply+Reduction	3072	256	12288	6.169
Multiply+Reduction	4096	256	16384	6.691
Multiply+Reduction	6144	256	24576	7.465
Multiply+Reduction	8192	256	32768	8.074
Multiply+Reduction	1	512	2	0.005
Multiply+Reduction	2	512	4	0.010
Multiply+Reduction	4	512	8	0.020
Multiply+Reduction	8	512	16	0.038
Multiply+Reduction	16	512	32	0.077
Multiply+Reduction	32	512	64	0.149
Multiply+Reduction	64	512	128	0.246
Multiply+Reduction	128	512	256	0.517
Multiply+Reduction	256	512	512	1.290

Multiply+Reduction	512	512	1024	1.894
Multiply+Reduction	1024	512	2048	3.156
Multiply+Reduction	2048	512	4096	4.581
Multiply+Reduction	3072	512	6144	5.499
Multiply+Reduction	4096	512	8192	5.994
Multiply+Reduction	6144	512	12288	6.663
Multiply+Reduction	8192	512	16384	7.060