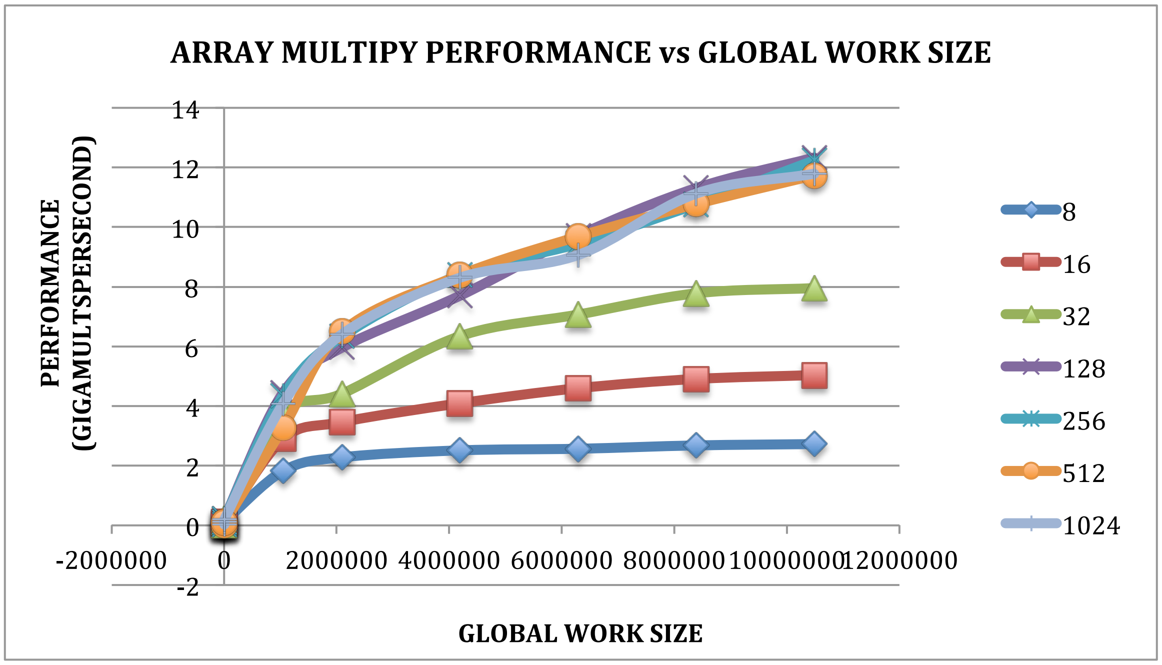
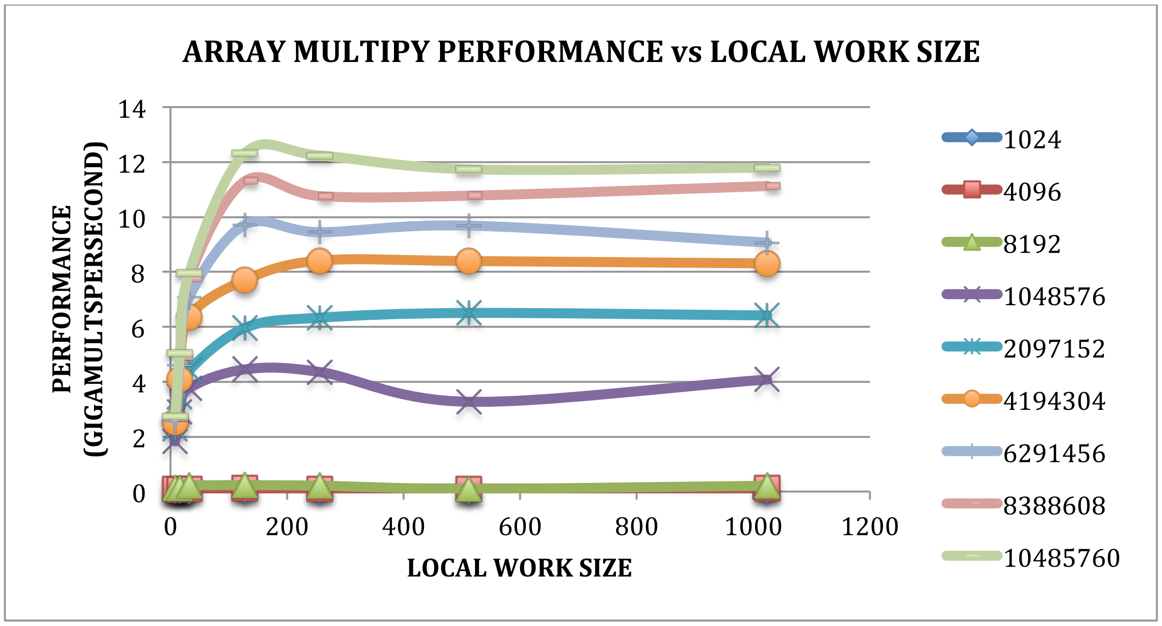
Project 5: OpenCL Array Multiply, Multiply-Add, and Multiply-Reduce

Array Multiply and the Array Multiply-Add:

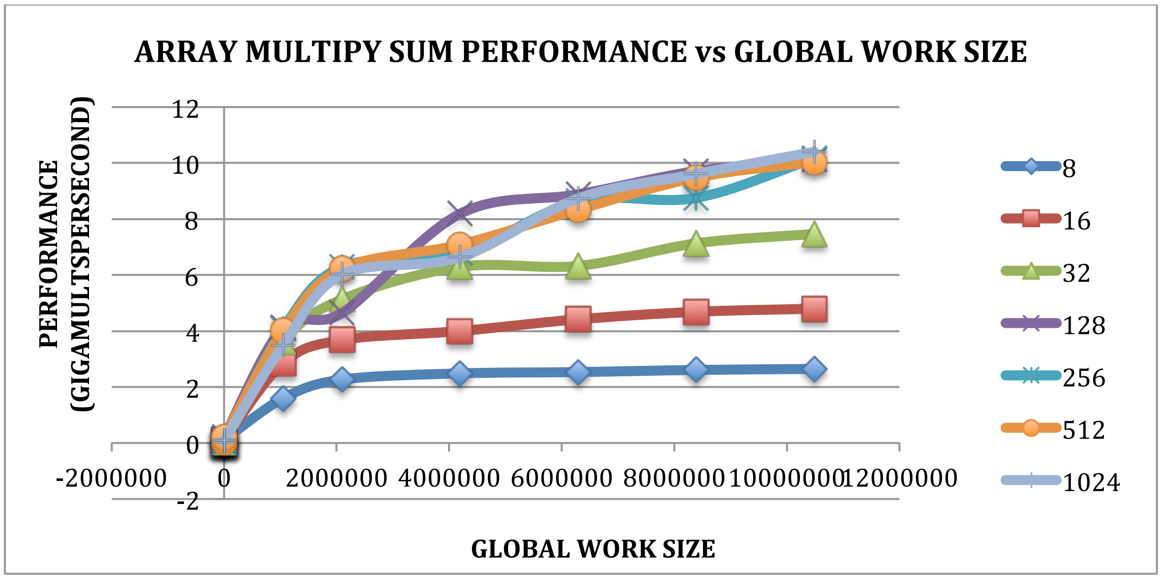
1. **What machine you ran this on.**
   - Rabbit
2. **Show the tables and graphs:**

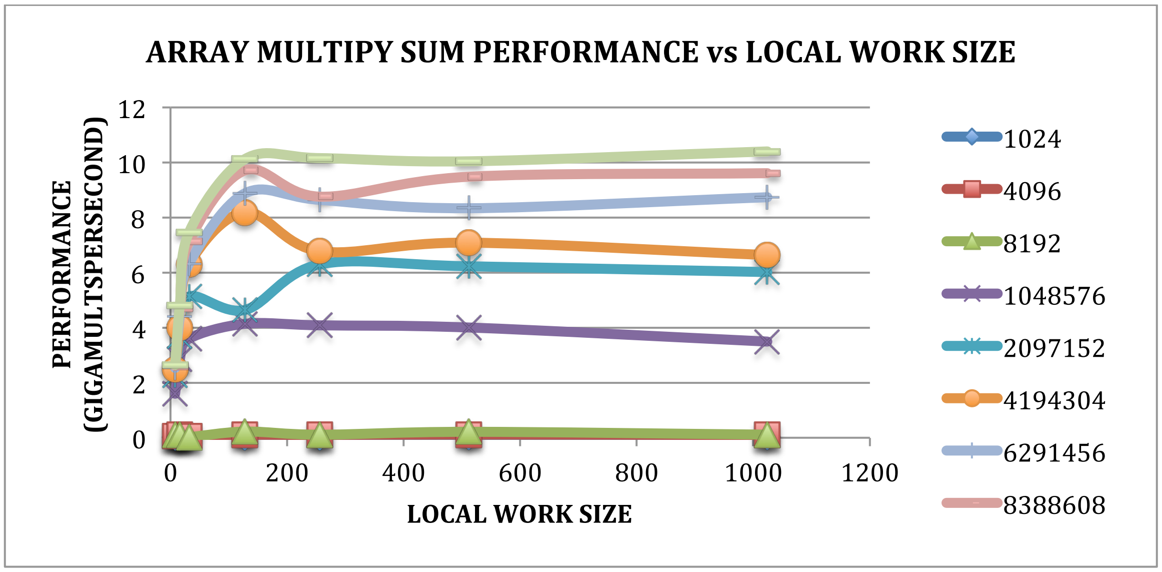
ARRAY MULTIPLY									
Local Work Size	Global Work Size(GigaMultsPerSecond)								
	1024	4096	8192	1048576	2097152	4194304	6291456	8388608	10485760
8	0.028	0.107	0.116	1.825	2.283	2.515	2.569	2.681	2.726
16	0.03	0.076	0.112	2.897	3.465	4.095	4.599	4.909	5.038
32	0.012	0.106	0.219	3.764	4.416	6.347	7.068	7.781	7.955
128	0.026	0.118	0.231	4.45	5.964	7.704	9.719	11.309	12.332
256	0.027	0.107	0.214	4.354	6.329	8.409	9.446	10.762	12.239
512	0.012	0.11	0.101	3.269	6.504	8.394	9.684	10.774	11.735
1024	0.028	0.112	0.211	4.079	6.413	8.306	9.067	11.124	11.79





ARRAY MULTIPLY-ADD									
Local Work Size	Global Work Size(GigaMultsPerSecond)								
	1024	**4096**	**8192**	**1048576**	**2097152**	**4194304**	**6291456**	**8388608**	**10485760**
8	0.027	0.067	0.111	1.597	2.267	2.487	2.535	2.61	2.647
16	0.013	0.106	0.106	2.859	3.683	4	4.42	4.687	4.799
32	0.028	0.052	0.027	3.604	5.133	6.294	6.326	7.133	7.47
128	0.027	0.107	0.224	4.14	4.644	8.191	8.882	9.715	10.127
256	0.027	0.104	0.111	4.084	6.298	6.776	8.642	8.758	10.159
512	0.028	0.116	0.216	4.005	6.229	7.087	8.338	9.483	10.043
1024	0.028	0.101	0.113	3.494	6.019	6.637	8.737	9.611	10.402





3. **What patterns are you seeing in the performance curves?**

   As Global work size increases, the performance improves for both array multiply and array multiply-sum. Larger work size values yield better performance. Similarly, with large array sizes performance improves as lesser computational units are required for performing the operation. Initial dips and rises are mostly due to small array sizes.

4. **Why do you think the patterns look this way?**

   Large local size performs better as less number of work groups are allocated in GPU for the operation. The performance improves with more local size, means increasing data size as the data is distributed across work groups.

Each work-group has multiple work items which enhances the performance as the data sets become larger and larger.

5. **What is the performance difference between doing a Multiply and doing a Multiply-Add?**

   Additional computation required along with multiplications adds up more time to complete each task in work groups. Hence, overall multiply seems to be performing better than the multiply add.

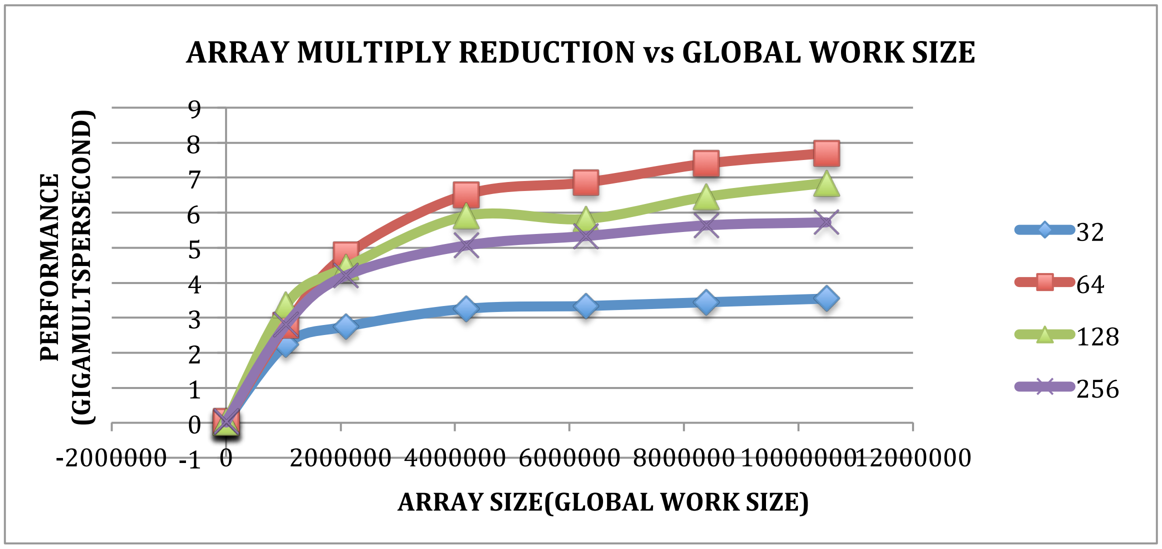
6. **What does that mean for the proper use of GPU parallel computing?**

   The proper use of GPU will be for larger array sizes as observed by the graph and the performances for small array sizes. If we have single operation, the performance is improved in GPU as the work is divided parallel in an optimal way.

Multiply-Reduce:

1. **Show this table and graph**

ARRAY MULTIPLY-REDUCTION									
Local Work Size	Global Work Size(GigaMultsPerSecond)								
	1024	4096	8192	1048576	2097152	4194304	6291456	8388608	10485760
32	0.003	0.016	0.037	2.239	2.748	3.257	3.336	3.443	3.546
64	0.005	0.014	0.037	2.777	4.79	6.517	6.867	7.403	7.702
128	0.003	0.014	0.034	3.377	4.44	5.907	5.822	6.456	6.849
256	0.001	0.017	0.041	2.797	4.208	5.064	5.335	5.639	5.725



2. **What pattern are you seeing in this performance curve?**

   Performance seems to increase with increase in array size. Interestingly, the performance seems to be improved for 64 work size better than the 256.

3. **Why do you think the pattern looks this way?**

   The reason maybe that since we are waiting more and more in large work sizes with ever barrier the performance may take a hit. All the threads would need to wait more and more with larger work sizes and would need to wait to calculate the sum. Hence, this may be the reason why 64 works better than 256 work size.

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

   It means that by experimenting, we can maybe find a good combination of number of work groups and work items for optimally utilizing the GPU parallel computing. After observation, it seems like having large data items distributed between local work groups seems a good option to efficiently utilize the GPU with reduction.