CS 575 Parallel Programming

Project #6

OpenCL Quadratic Regression

Name: Rachel Xing (#934229189)

Email Address: xingru@oregonstate.edu

Description: This project uses OpenCL to perform a quadratic regression to obtain the best-fit parabola equation ($y = Qx^2 + Lx + C$) on an x-y pair dataset with a size of 4 million. To perform this based on the least squares method, we need to compute seven summations $\sum x^4, \sum x^3, \sum x^2, \sum x, \sum x^2y, \sum xy, \sum y$. The procedure for accumulating these sums is written in a .cl program and run on the GPU. The other parts of regression computation are written in C++ and run on the host. It is run with five different local sizes (i.e., work-group sizes) of 8, 16, 32, 64, 128, 256 combined with data sizes of 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288, 1048576, 2097152, 4194304 (maximum).

Commentary:

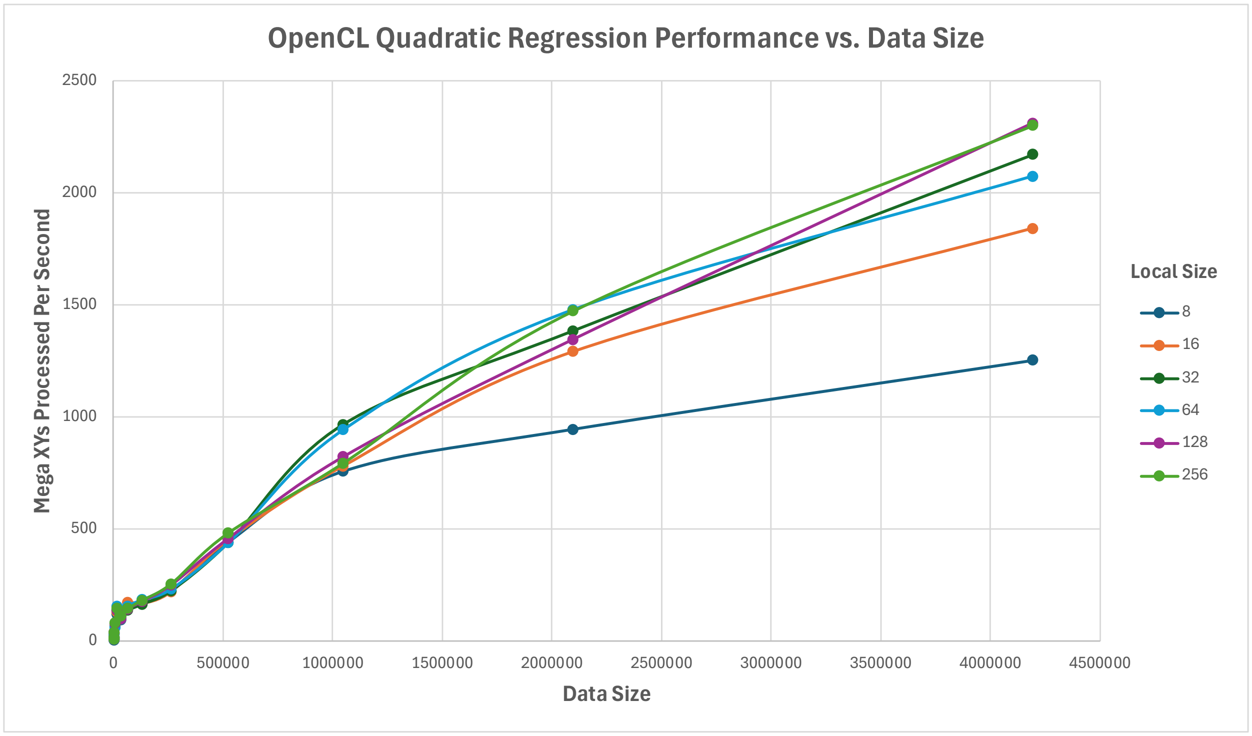
1. ***Tell what machine you ran this on***

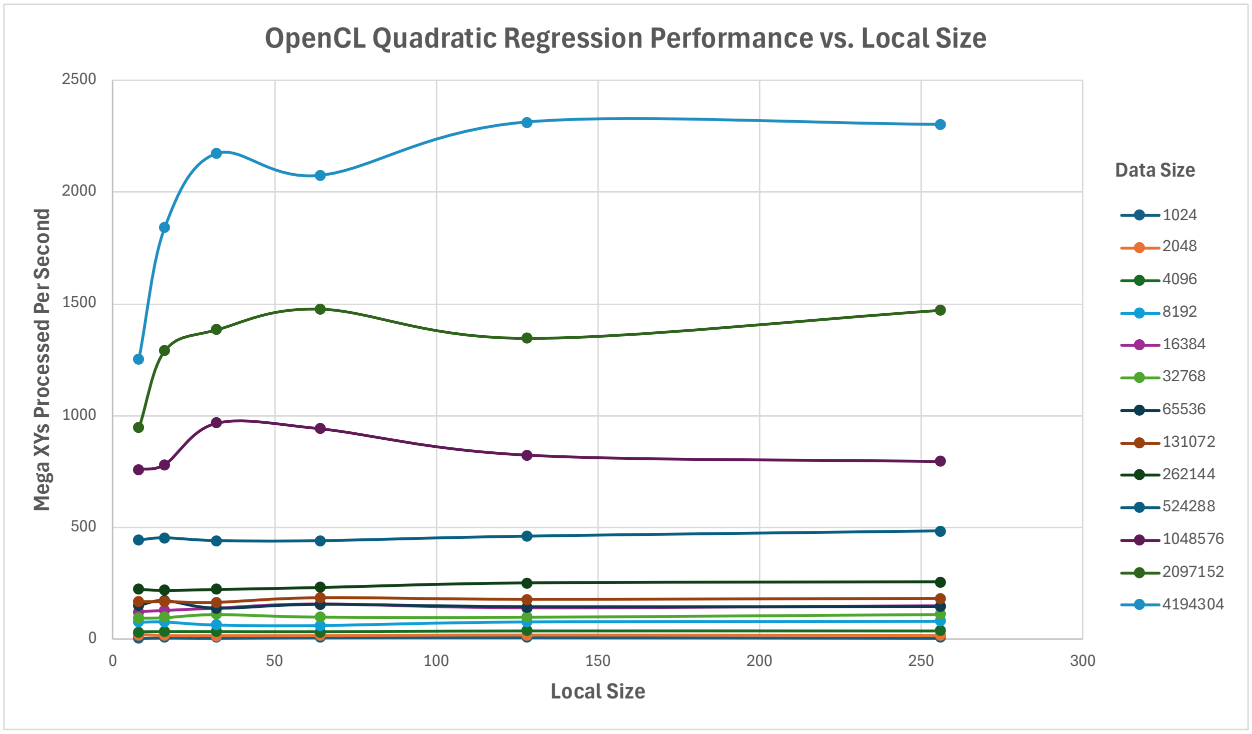
   I ran this project on the **OSU Rabbit server**, which is equipped with Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz and runs **CentOS Linux 7 (Core)**. The compiler I used is the **GNU C++ Compiler** (OpenMP enabled) and **CUDA Toolkit 10.1**.

2. ***Show the table and graphs***

Performance Table (MegaXYsProcessed / Second)

Data Size / Local Size	1024	2048	4096	8192	16384	32768	65536	131072	262144	524288	1048576	2097152	4194304
8	8.14	**21.30**	32.83	78.18	124.09	96.84	151.44	169.56	225.48	443.49	759.52	947.00	1254.61
16	8.91	17.21	34.61	79.06	129.26	99.15	**173.25**	169.49	221.17	452.77	782.30	1292.17	1841.95
32	8.48	17.03	34.63	66.04	139.53	112.68	138.71	166.42	224.91	440.42	**968.30**	1385.89	2171.46
64	9.44	17.18	33.80	63.83	**157.88**	101.80	156.24	**186.50**	232.95	439.94	943.53	**1478.94**	2073.95
128	**9.71**	19.53	**38.79**	79.47	141.22	101.23	145.33	179.57	251.60	460.20	824.42	1347.94	**2312.48**
256	8.46	17.58	38.68	**82.03**	150.11	**113.05**	146.32	183.59	**255.90**	**483.74**	795.51	1473.54	2301.72





3. ***What patterns are you seeing in the performance curves? What difference does the amount of data you process make? What difference does the size of each work-group make?***

- In terms of data sizes, for all work-group sizes, we can observe that the overall performance increases as the amount of processed data increases. The work-group sizes of 8 and 16 yield lower overall performance than larger work-group sizes.
- In terms of work-group sizes, for **small data sizes (less than 1 million x-y pairs)**, there is no significant performance changes when increasing the work-group sizes from 8 to 256. However, for **larger data sizes that have more than 1 million x-y pairs**, performance differences among different work-group sizes become more noticeable. Specifically, there is a significant performance boost when increasing the work-group size from 8 to 32. Beyond the work-group size of 32, performance either degrades (as seen with a data size of 1,048,576) or fluctuates around a plateau (as seen with data sizes of 2,097,152 and 4,194,304).

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

"In terms of data sizes, for all work-group sizes, we can observe that the overall performance increases as the amount of processed data increases."

Reason: This is because when increasing the number of x-y pairs to be processed for a fixed work-group size, there are more GPU threads to be scheduled and executed in parallel. Thus, as the problem size grows, the parallel portion of the task increases, so OpenCL parallelization can yield greater performance gains. This observation aligns with Gustafson's Law and what we observed in CUDA parallelization from Project #5.

"The work-group sizes of 8 and 16 yield lower overall performance than larger work-group sizes.

Reason: This is because a warp consists of 32 threads where each executes the same instruction at the same time. When the work-group size is smaller than 32, only a portion of threads in a warp are active while the rest remain idle during the runtime. This leads to inefficient use of GPU resources. Also, work-group sizes of 8 or 16 mean only one-fourth or half of a warp is scheduled per work-group. It not only reduces execution efficiency but also results in lower occupancy, so the GPU's ability to hide memory latency by warp switching cannot be fully utilized. Besides, smaller work-group size means larger number of total work-groups, which can result in more scheduling and management overhead and further degrade performance.

"In terms of work-group sizes, for small data sizes (less than 1 million x-y pairs), there is no significant performance changes when increasing the work-group sizes from 8 to 256."

Reason: This is because for smaller data sizes, the overheads associated with OpenCL parallelization outweighs the performance gains from it, so we cannot observe the performance boost by parallelization no matter the work-group sizes.

"However, for larger data sizes that have more than 1 million x-y pairs, performance differences among different work-group sizes become more noticeable. Specifically, there is a significant performance boost when increasing the work-group size from 8 to 32. Beyond the work-group size of 32, performance either degrades (as seen with a data size of 1,048,576) or fluctuates around a plateau (as seen with data sizes of 2,097,152 and 4,194,304)."

Reason:

- For larger work-group sizes, the parallelizable portion of the task becomes large enough, so the benefit from OpenCL parallelization can outweigh its associated overheads. Thus, performance gains brought by different work-group sizes become more distinctive.
- For work-group sizes smaller than 32, larger sizes can utilize GPU resources more efficiently by leaving fewer threads idle in a warp during the execution, leading to a performance boost when increasing work-group sizes to 32.
- For data sizes like 1 million x-y pairs, a work-group size of 32 can achieve the best performance, and further increasing the work-group size can degrade performance. That is because it reduces the number of work-groups that can be executed in parallel, thus limiting parallelism at the work-group level and reducing the performance gain.
- For large data sizes such as 2 million and 4 million x-y pairs, larger work-group sizes like 64 (2 warps per group) and 128 (4 warps per group) yield the optimal performance respectively. This is because the problem size is large enough to keep the GPU fully occupied, and the larger work-groups can reduce memory access latency through warp scheduling when there are more warps per work-group. Also, using a larger work-group size can result in fewer total number of work-groups, which allows better utilization of shared memory within each group, especially for huge computational loads.

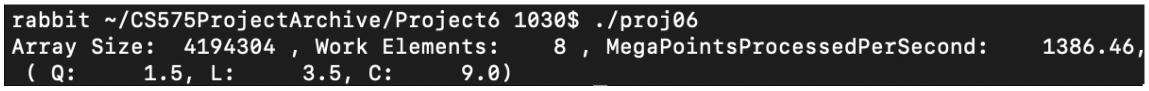
5. ***What did you get for Q, L, and C? (Hint: the actual values either end in .0 or .5 -- you don't need to report any more than one decimal digit.)***

Answer:

$$\boldsymbol{Q = 1.5, L = 3.5, C = 9.0}$$

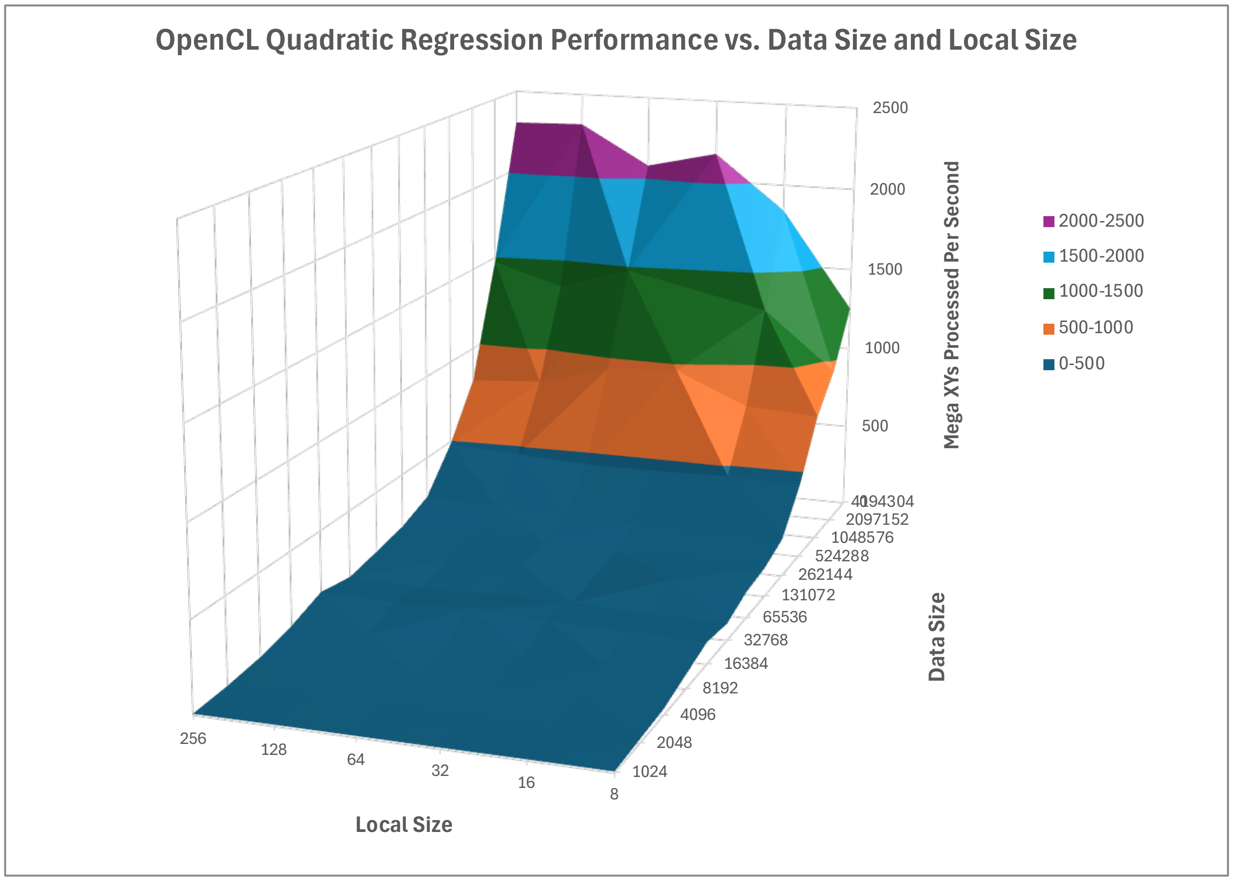
Best-fit parabola equation: $y = 1.5x^2 + 3.5x + 9.0$

Screenshot:



```
rabbit ~/CS575ProjectArchive/Project6 1030$ ./proj06
Array Size:  4194304 , Work Elements:    8 , MegaPointsProcessedPerSecond:    1386.46,
 ( Q:     1.5, L:     3.5, C:     9.0)
```

[*Extra Graph*: 3D visualization of OpenCL performance with respect to DATASIZE and LOCALSIZE in the quadratic regression]


OpenCL Quadratic Regression Performance vs. Data Size and Local Size
Mega XYs Processed Per Second
Data Size
Local Size