I wrote several kernels for matrix vector multiplications for serial, naive and shared memory (LDS) algorithms. The benchmark results are shown below:

2. Matrix of 128x128

First element of output vector	Execution time for serial implementation (ms)	Execution time for HIP kernel without LDS (ms)	Execution time for HIP kernel with LDS (ms)
256	0.055	20.6449	20.68

3. Matrix of 1024x1024

First element of output vector	Execution time for serial implementation (ms)	Execution time for HIP kernel without LDS (ms)	Execution time for HIP kernel with LDS (ms)
0	3.277	21.3156	20.9316

4. Matrix of 2048x2048

First element of output vector	Execution time for serial implementation (ms)	Execution time for HIP kernel without LDS (ms)	Execution time for HIP kernel with LDS (ms)
2096128	12.966	21.8481	21.2644

5.1 Matrix of 128x64 and vector of 64

First element of output vector	Execution time for serial implementation (ms)	Execution time for HIP kernel without LDS (ms)	Execution time for HIP kernel with LDS (ms)
128	0.031	20.6531	20.6541

5.2 Matrix of 1024x256 and vector of 256

First element of output vector	Execution time for serial implementation (ms)	Execution time for HIP kernel without LDS (ms)	Execution time for HIP kernel with LDS (ms)
1024	1.017	20.8824	2.5594e-41

5.3 Matrix of 1000x49 and vector of 49

First element of output vector	Execution time for serial implementation (ms)	Execution time for HIP kernel without LDS (ms)	Execution time for HIP kernel with LDS (ms)
1024	0.163	20.7124	4.57748e-41

The results clearly illustrate the advantages and disadvantages of both serial and parallel implementations. Specifically, for a low number of computations, serial implementations on the CPU can be completed in less time compared to parallel implementations on the GPU. However, this trend reverses for tasks with a high number of computations. Moreover, LDS implementations significantly reduce the time required to perform the computations.