PCS5120 Homework

Giuliano A. F. Belinassi[1]

[1]Institute of Mathematics and Statistics (IME) – University of São Paulo (USP)
Rua do Matão, 1010 – São Paulo – SP – Brazil

In various research fields where numerical linear algebra is required eighter because of its facility or inexisence of analytical solutions, can take advantage of packages such as Lapack or BLAS. One major concern about these libraries is its performance, subject discussed in this report. Here we target the routine designed to multiply two matrices in single precision floating point (`SGEMM`).

Althrough implementing a matrix-matrix multiplication seems trivial by its concept, it is fairly difficult to provide an efficient code because of various reasons, such as cache usage. The use of techniques such as block matrix multiplication can yield better results due to a better cache usage, but a question that can be asked from this approach is what is the block size that maximizes performance?

1. The database

We used the database "*SGEMM GPU kernel performance Data Set*" provided by *UCI machine learning repository*[1]. Briefly, it has timings of multiplication of two matrices in *ms*, each one of size 2048×2048, using a combination of 14 parameters, totalizing 241601 lines in the database.

2. Data analysis

We used Orange in our analysis. Our first objective was to find the distribuition of the average of the four executions per sample to check possible improvements or deterioration of the performance. For creating a column with the average of four executions, we used the Orange's Feature Constructor. Unfortunately, there is no average function implemented here, so we had to calculate such function by the definition $\left(\frac{1}{n}\sum x_i\right)$.

Once we had the average column, we plotted the distribution, as illustrated by 1. Notice that most of the averages concentrate under 200ms, and there are some data around 2400ms. Such high timing can be caused by the parameters itself or be an outlier because there were other programs running on the computer.

Once we got the distribuition of timings, we focused on what parameters yielded best results. Orange's Data Table let us find the combination of parameters because how straightfoward the table sorting feature is implemented.

Maybe for pratical reasons, the showed results is enough to provide a setup for an efficient implementation, but we can explore the provided data in order to create projections for perhaps even better timings.

3. Conclusions

[1]`https://archive.ics.uci.edu/ml/datasets/SGEMM+GPU+kernel+performance`

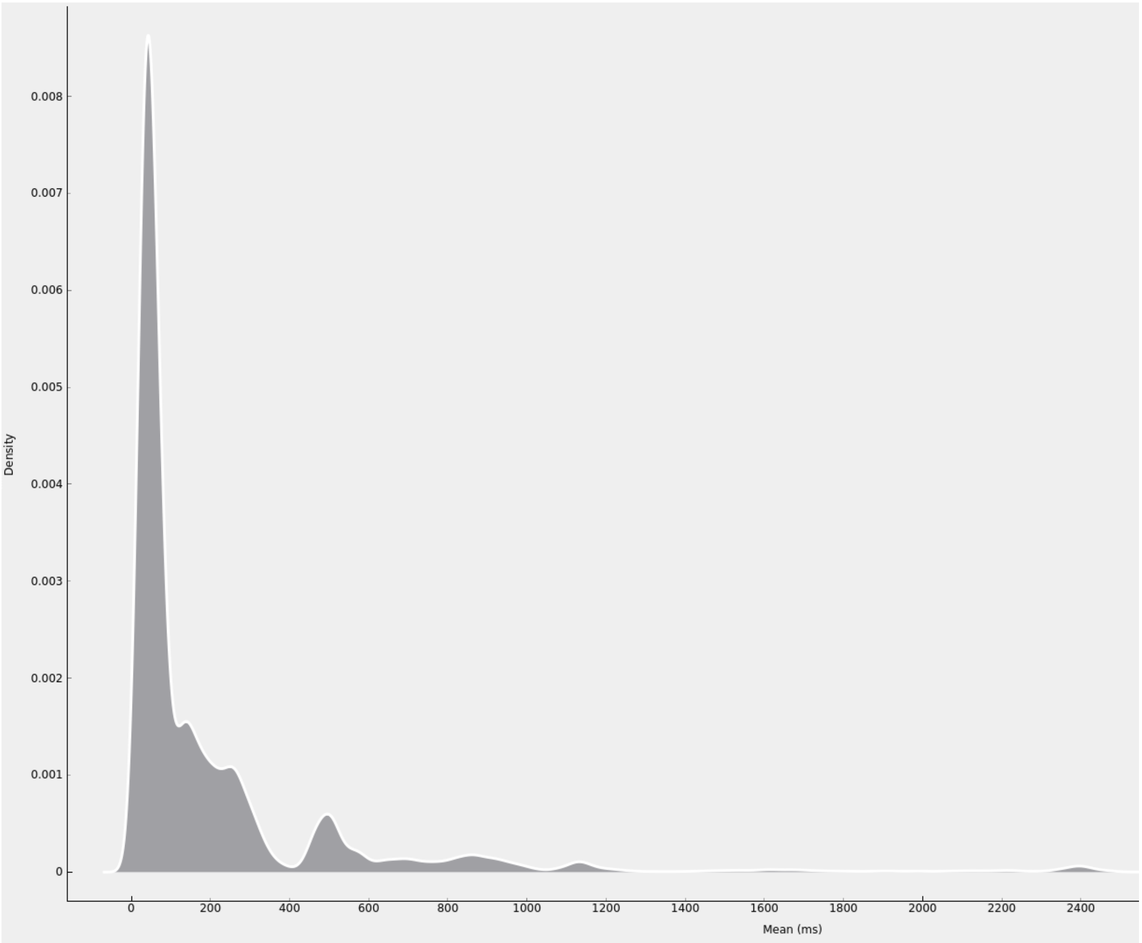

Figure 1. Distribution of the average of four samples per parameter.

Table 1. Combination of parameters that yielded the 5# best results

MWG	NWG	KWG	MDIMC	NDIMC	MDIMA	NDIMB	KWI	VWM	VWN	STRM	STRN	SA	SB	Mean (ms)
16	16	16	8	8	8	8	2	1	1	0	0	0	0	116,37
16	16	16	8	8	8	8	2	1	1	0	0	0	1	78,705
16	16	16	8	8	8	8	2	1	1	0	0	1	0	80,565
16	16	16	8	8	8	8	2	1	1	0	0	1	1	86,6375
16	16	16	8	8	8	8	2	1	1	0	1	0	0	118,6625