The Title of Your Report

Cameron F Clark[†] and Kian S Frassek[‡]
EEE4120F Class of 2024
University of Cape Town
South Africa
[†]CLRCAM007 [‡]KIAFRS001

I. Introduction

Parallel computing is a power tool which enables code to run at phenomenal rates compared to single thread. There are many different ways to implement parallel programs on many different problems. The implementation that this report focuses on is using OpenCL and C++ to write a parallel program to efficiently multiply square matrices. Importantly, it is futile to design parallel programs that do not actually increase the efficiency and/or speed of the non parallelized programs, thus, this report will additionally evaluate the performance of the parallel program versus the single thread program with different matrix sizes and number of matrix multiplications.

II. Methodology

A. Hardware

The hardware used to conduct these experiments is a MacBook Pro 13" 2019 model. All of these parallel programs were conducted on the GPU which is a Intel Iris Plus Graphics 1536 MB. All of the single thread programs were conducted on the CPU which is 2 GHz Quad-Core Intel Core i5.

B. Implementation

1) Single Thread Program: The single thread program is a relative simple implementation in C code. It requires two for loops which iterate over the row and column of the result matrix and a third which calculates the dot product of the input matrices to find the value of the cell of the output matrix. There is one additional for loop which allows for multiplying a given number of matrices together

```
int matrix_output[matrix_size], matrixA[matrix_size], cell;
for (int c = 0; c < matrix_size; c++) matrixA[c] = matrices[0][c];

//Matrix multiplication
for (int m = 1; m < MATRIX_COUNT; m++) {
 for (int row = 0; row < Size; row++) {
  for (int col = 0; col < Size; col++) {
   cell = 0;
   for (int dot = 0; dot < Size; dot++) {
    cell += matrixA[row * Size + dot] * matrices[m][col + dot * Size];
   }
   matrix_output[row * Size + col] = cell;
  }
 }
 //rewrite matrix A with output so loop can re-iterate
 for (int c = 0; c < matrix_size; c++) matrixA[c] = matrix_output[c];
}
```

2) Parallel Program: The parallel program involves setting up OpenCL to run instances of a kernel program. The kernel program itself runs a very similar instance of the cell calculation for the single thread program above where the row is equivalent to the workGroupNumber and the column is equivalent to the localGroupID. A for loop is used to calculate the dot product as in the single thread program. Lastly, there is an additional for loop in the OpenCL setup code which allows for successive matrix multiplication.

```
__kernel void matrixMultiplication(
 __global int* matrixA_buffer,
 __global int* matrixB_buffer,
 __global int* output_buffer
){
 int workItemNum = get_global_id(0); //Work item ID
 int workGroupNum = get_group_id(0); //Work group ID
 int localGroupID = get_local_id(0); //Work items ID within each work group
 int localSize = get_local_size(0); //Get the work number of items per group

 //Row <=> workGroupNum; Column <=> localGroupID
 //workItemNum <=> workGroup * localSize + localGroupID
 int cell = 0;
 for (int i = 0; i < localSize; i++) {
  cell = cell + *(matrixA_buffer + workGroupNum * localSize + i) *
   *(matrixB_buffer + localGroupID + i * localSize);
 }
 *(output_buffer + workItemNum) = cell;
}
```

C. Experiment Procedure

To evaluate the performance of the single thread versus parallel program two experiments will be conducted and evaluate the difference in time taken and the speed up of the parallel program. The first will be multiplying two matrices with different sizes, and the second will be to multiply different numbers of successive matrices of the same size. Both methods of testing have different amounts of parallelization and different amounts of overhead and should give good understanding of the effectiveness of parallelizing matrix multiplication code.

III. Results

The results section is for presenting and discussing your findings. You can split it into subsections if the experiment has multiple sections or stages.

A. Figures

Include good quality graphs. These were produced by the Octave code presented in listings 2 and 3. You can play around with the `PaperSize` and `PaperPosition` variables to change the aspect ratio. An easy way to obtain more space on a paper is to use wide, flat figures, such as Fig.

Always remember to include axes text, units and a meaningful caption in your graphs. When typing units, a µ sign has a tail! The letter "u" is not a valid unit prefix. When typing resistor values, use the Ω symbol.

B. Tables

Tables are often a convenient means by which to specify lists of parameters. An example table is presented in table I. You can use Tablesgenerator to make your LaTeXtables.

```
function FormatFig(X, Y, File);
 set(gcf, 'PaperUnits'      , 'inches');
 set(gcf, 'PaperOrientation', 'landscape');
 set(gcf, 'PaperSize'       ,       [8, 4]);
 set(gcf, 'PaperPosition'   , [0, 0, 8, 4]);

 set(gca, 'FontName', 'Times New Roman');
 set(gca, 'Position', [0.1 0.2 0.85 0.75]);

 xlabel(["\n" X]);
 ylabel([Y "\n\n"]);

 setenv("GSC", "GSC"); # Eliminates stupid warning
 print(...
  [File '.pdf'],...
  '-dpdf'...
 );
end
```

Listing 1. Octave function to format a figure and save it to a high quality PDF graph

```
figure;                                   # Create a new figure
# Some code to calculate the various variables to plot...
plot(N, r, 'k', 'linewidth', 4); grid on; # Plot the data
xlim([0 360]);                            # Limit the x range
ylim([-1 1]);                             # Limit the y range
set(gca, 'xtick', [0 90 180 270 360]);    # Set the x labels

FormatFig(...                             # Call the function with:
 'Phase shift [\circ]',...                      # The x title
 'Correlation coefficient',...                  # The y title
 ['r_vs_N;_f=' num2str(f) ';_P=' num2str(P)]... # Format the file name
);
close all;                                # Close all open figures
```

Listing 2. Example of how to use the FormatFig function

TABLE I
MY INFORMATIVE TABLE

Heading 1	Heading 2	Heading 3
Data	123	321
Data	456	654
Data	789	987

C. *Pictures and Screen-shots*

When you include screen-shots, pdfLaTeX supports JPG and PNG file formats. PNG is preferred for screen-shots, as it is a loss-less format. JPG is preferred for photos, as it results in a smaller file size. It's generally a good idea to resize photos (not screen-shots) to be no more that 300 dpi, in order to reduce file size. For 2-column article format papers, this translates to a maximum width of 1024. **Never change the aspect ratio of screen-shots and pictures!**

The source used to import a picture in an exact spot, with a caption and labels

University of Cape Town • iYunivesithi yaseKapa • Universiteit van Kaapstad
SPES BONA
Fig. 1: An example image

D. *Maths*

LaTeX has a very sophisticated maths rendering engine, as illustrated by equation 1. When talking about approximate answers, never use ± 54 V, as this implies "positive or negative 54 V". Use ≈ 54 V or ~ 54 V instead.

$$y = \int_0^\infty e^{x^2} \mathrm{dx} \tag{1}$$

IV. CONCLUSION

The conclusion should provide a summary of your findings. Many people only read the introduction and conclusion of a paper. They sometimes scan the tables and figures. If the conclusion hints at interesting findings, only then will they bother to read the whole paper.

You can also include work that you intend to do in future, such as ideas for further improvements, or to make the solution more accessible to the general user-base, etc.

Publishers often charge "overlength article charges" [7], so keep within the page limit. In EEE4084F we will simulate overlength fees by means of a mark reduction at 10% per page. Late submissions will be charged at 10% per day, or part thereof.

REFERENCES

[1] T. Oetiker, H. Partl, I. Hyna, and E. Schlegl, "The Not So Short Introduction to LaTeX 2_ε," https://tobi.oetiker.ch/lshort/lshort.pdf, Jul. 2015, version 5.05.
[2] "IEEE Conference Paper Templates," http://www.ieee.org/conferences_events/conferences/publishing/templates.html.
[3] A. Baboon, B. Charles, D. Ester, and F. Generalson, "An Amazing Title," Their Not-so-awesome University, Technical Report, Apr. 1492.
[4] B. van der Zander, J. Sundermeyer, and T. Hoffmann, "TeXstudio – A LaTeX Editor," https://sourceforge.net/projects/texstudio/.
[5] "InkScape Website," http://www.inkscape.org/.
[6] J. Taylor and J. G. Hoole, "Robust Protocol for Sending Synchronisation Pulse and RS-232 Communication over Single Low Quality Twisted Pair Cable," in *Proceeding of ICIT*. Taiwan: IEEE, Mar. 2016.
[7] "Voluntary Page and Overlength Article Charges," http://www.ieee.org/advertisement/2012vpcopc.pdf, 2014.