Practical 2 - Python vs. C

Ronak Mehta[†] and Vikyle Naidoo[‡]
EEE3096S Class of 2019
University of Cape Town
South Africa
[†]MHTRON001 [‡]NDXVIK005

***Abstract*—Performance and optimization of the C code in comparison to the python code using a combination of parallelization, compiler flags, bit widths, and hardware level features**

I. Introduction

This practical will investigate and compare the performance between C and python code in the context of embedded systems development. It will start by running a program in Python, and comparing it to the same program but written in C. Thereafter, it will try and optimize the C code, by using different bit widths, compiler flags, particular hardware available in the ARM Processor, as well as implementing threading to improve performance. The quest for optimization can be a long, unending spiral but through benchmarking the results, this report will try to find a suitable performance comparison between the two languages which will be vital in deducing the importance of using a particular language over the other.

II. Methodology

In this section you should describe the method of the experiment.

A. Hardware

Include detail such as the hardware used. It's generally a good idea to include a block diagram at this point. This figure was drawn in InkScape [1]. When you want to import an InkScape figure (SVG format) into LaTeX, simply save it to PDF (use the drawing extents as the media box area) and include the figure.

B. Implementation

Also mention the implementation source code:

```
# You can include inline Matlab / Octave code
x = linspace(0, 2*pi, 1000);
y = sin(x);
plot(x, y); grid on;
```

or you could turn it into a float: see listing 1. Floats are tables, figures and listings that appear at a different place than in the source code. This template is set up to put floats at the top of the next column, as prescribed by the IEEE article specification.

Only list what is relevant. Don't give too much detail - just enough to show what you've done. This template supports the following languages:

- Matlab (Octave)

```
__kernel void Multiply(
 __global float* A, // Global input buffer
 __global float* B, // Global input buffer
 __global float* Y, // Global output buffer
   const  int    N  // Global uniform
){
 const int i = get_global_id(0); // 1st dimension index
 const int j = get_global_id(1); // 2nd dimension index

 // Private variables
 int   k;
 float f = 0.0;

 // Kernel body
 for(k = 0; k < N; k++) f += A[i*N + k] * B[k*N + j];
 Y[i*N + j] = f;
}
```

Listing 1. OpenCL kernel to perform matrix multiplication

- GLSL
- OpenCL
- Verilog
- C++ (use the name "Cpp")

C. Experiment Procedure

Furthermore, include detail relating to the experiment itself: what did you do, in what order was this done, why was this done, etc. What are you trying to prove / disprove?

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.

```
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 2. 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 3. 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

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" [2], 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] "InkScape Website," http://www.inkscape.org/.
[2] "Voluntary Page and Overlength Article Charges," http://www.ieee.org/advertisement/2012vpcopc.pdf, 2014.