High Performance Computing

National University of Computer and Emerging Sciences

Deliverable # 01

Complex Computing Problem

By

Hamza Kaleem — 23i-0783
Umaima — 23i-0790
Warisha Shaukat — 23i-0809

Contents

1 Introduction **2**

2 Profiling **2**

2.1 Makefile 2

2.2 gprof Usage 2

3 Compute Heavy Functions **3**

3.1 Identified Functions 3

3.2 Profiling Insights 3

4 Compilation and Configuration **3**

4.1 Procedure 3

4.2 Terminal Commands 3

1 Introduction

This report details the implementation of the provided KLT algorithm. The report will detail the following:

- Profiling
- Identifying Compute Heavy Functions
- Steps to run and profile the application

2 Profiling

2.1 Makefile

The makefile was altered to contain the -pg flag, enabling profiling on the generated executable. The executable was generated using the following commands:

```
make all
```

To generate an executable for all 5 examples, Or:

```
make lib
make example3
```

To generate our target executable example.
Additional examples were generated to gain a thorough understanding of the application's implementation when tested on various data.

2.2 gprof Usage

The executable was first run to generate a gmon.out file; the output of the gmon file was piped to a profile.txt which was fed to a gprof shell and python script that generated our desired pdf detailing function weight relative to the total runtime of the application.
The following commands outline this procedure:

```
./example3
gprof ./example3 gmon.out > profile.txt
./gprof2pdf.sh profile.txt profile.pdf
```

This code generates a pdf visualizing the function calls in a tree structure, along with the relative runtime of each function as a part of the total runtime.

3 Compute Heavy Functions

3.1 Identified Functions

Based on the profile generated for example3 (see threshold_profile_ex3.pdf), the priority functions to improve runtime are:

- _convolveImageVert()
- _interpolate()
- _computeIntensityDifference()
- _KLTSelectGoodFeatures()

3.2 Profiling Insights

These functions were identified by filtering downward from the top-level hotspots and identifying where the main load was coming from in those hotspots. For example: KLTTrackFeatures contributes 85.71% of the main program, however, nearly all of its runtime weight comes from the lower level, embarrassingly parallel functions like _convolveImageVert() and _interpolate().

4 Compilation and Configuration

4.1 Procedure

(i) Alter makefile → Add -pg flag

(ii) Compile example3 using makefile

(iii) Run example3 to generate gmon.out

(iv) Redirect gmon.out contents to profile.txt using gprof

(v) Generate profile pdf using gprof

4.2 Terminal Commands

The above steps and terminal commands for the complete procedure are shown below.

```
make lib
make example3
./example3
gprof ./example3 gmon.out > profile.txt
./gprof2pdf.sh profile.txt profile.pdf
```