Project Report for CS573-Optimization Compilers
Resuse Distance Analysis of the Mediabench Suite

Athanasios E. Papathanasiou,
Graduate Student
Computer Science Dept.
University of Rochester – Rochester, NY
papathan@cs.rochester.edu
http://cs.rochester.edu/ papathan

Abstract

This is the project report for the Course CS573 - Optimization Compilers. The goal of the project is to study the reuse distance of several of the applications included in the MediaBench Suite.

1 Methodology

The reuse distance of a memory location or variable of a program is the number of individual memory references between two consecutive references to the same memory location or variable. The importance of the reuse distance analysis comes from the fact that it gives insight to the cache performance of a program for any given cache size without the necessity to conduct detailed cache simulations. A more extensive description of the reuse distance analysis and its benefits is given in [1].

In order to study the reuse distance of the benchmarks, the ATOM simulator in combination with the code provided by C. Ding for monitoring *load* and *store* was used. For each program tested a reuse distance histogram is provided The histogram presents the frequency with which a specific reuse distance re-appears in a program.

In addition to the above, the C language instrumentor provided by C. Ding was used with one of the benchmarks in order to get source-level information related to the program's access patterns. However, the instrumentor failed to compile successfully the progrmas of the MediaBench suite. Thus, insight on

the reasons for which the observed reuse distances appeared was not obtained and will not be presented in this report.

1.1 Media Benchmarks

This section describes the applications that are included in the UCLA Media Benchmark Suite [2].

The Media Benchmark Suit consists of the following applications:

- **JPEG**: It is a package used to implement JPEG image compression and decompression. It consists of two separate programs *jpegencode* and *jpegdecode*.
- **MPEG2**: It is a package used to implement MPEG encoding and decoding of video streams. It consists of two separate programs *mpeg2encode* and *mpeg2decode*.
- **GSM**: It is a package used to implement speech encoding and decoding. It consists of two separate programs *gsmencode* and *gsmdecose*.
- **ADPCM**: ADPCM stands for Adaptive Differential Pulse Code Modulation. It is a family of speech compression and decompression algorithms. It consists of the programs *adpcmencode* and *adpcmdecode*.
- **G.721**: It is a package for voice compression. It consists of the programs *g721encode* and *g721decode*.
- **PGP**: It is package that uses "message digests" to form signatures. It includes several assembly source files, which might make the study of this program more difficult.
- **PEGWIT:** It is a program for performing public key encryption and authentication. It provides three distinct operations: public key generation from a secret key, encryption using the public key and decryption using the secret key.
- **Ghostscript**: It is a package that provides an interpreter for the PostScript language, a set of C procedures (the Ghostscript library) that implement the graphics capabilities that appear as primitive operations in the PostScript language, and an interpreter for Portable Document Format (PDF) files.
- **Mesa**: Mesa is a 3-D graphics library with an API which is very similar to that of OpenGL. Three graphics applications were used to experiment with the library: *mesa_mipmap*, *mesa_osdemo* and *mesa_texgen*.
- **RASTA**: It is a speech processing package.

- **EPIC:**: It is an experimental image data compression utility. It consists of *epic* and *unepic*; used for compression and decompression respectively.

2 Experimental Results

This section describes the results that were obtained by analyzing five of the programs included in the MediaBench Suite. These programs were:

1. *texgen*, *mipmap*, and *osdemo*: These three programs use the MESA graphics library to draw three different images and save them to a file. The distances for these programs are shown in Figures 1 and 2. Figure 1 presents abolute values, while Figure 2 provides the percentages with which each reused ditance was observed. Figure 2 leads to othe conclusion that depnding on the object that is being drawn, the MESA library results in very different reuse distance patterns.

2. *cjpeg*, and *djpeg*: These are the JPEG encode and JPEG decode programs respectively. JPEG Encode (*cjpeg*) takes as input a PPM image and converts it into a JPEG image. JPEG decode (*djpeg*) takes as an argument a PPM image and converts it into a PPM image. Four different images were used in this experiment. The JPEG versions of the images had sizes 6KB, 22KB, 79KB, 120KB. The sizes of the corresponding PPM images were 112KB, 464KB, 1496KB and 2376KB respectively. Figure 3 shows the absolute reuse distances for the *cjpeg* program and Figure 4 provides the percentages with which each reused ditance was observed. Figures 5 and 6 show the reuse deistances (absolute values and percentages) for the *djpeg* program.

   Figure 4 leads to the conclusion that jpeg encoding results in the same reuse distance patterns, regardless of the image size. The graph has two peaks (for all image sizes). The first peak is observed at a reuse ditance of 2^3, which appears with a frequency of 21%, while the second one is observed at a reuse distance of 2^8, which appears with a frequency of 16%. The tails of the graphs have two more smaller peaks. The reuse distance at which these peaks are observed is different for each image size. Larger images create the two smaller peaks at larger distances.

   The reuse rdistance patterns of jpeg decoding are not as uniform as those of jpeg encoding. However, independent of the image size all graphs have three peaks at distances 2^4, 2^{11}, and 2^{17}. However, the frequences with which these peaks appear are not the same for all image sizes. In fact **larger** image sizes lead to *higher* frequencies for these peaks. For example for the image with size of 1496KB the first peak at a distance of 2^4 appear with a frequency of 14%, while the same peak appears for the image with size of 112KB with a frequency of 10%. Finally, in all cases the graphs

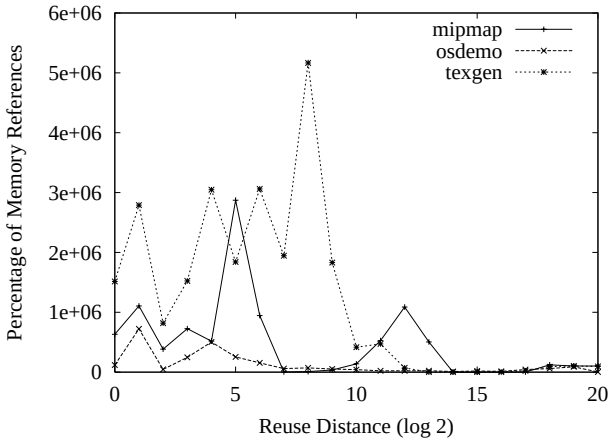

Figure 1: Absolute values of the reuse distance frequency for the three programs that use the MESA Graphics Library.

have heavy tails with different peaks. It is apparent that smaller images result in higher peaks in the tails of the graphs (at large reuse distances).

All data files that were used to obtain these graphs, and all postscript files of the figures may be found in;

$$/u/papathan/COURSES/CS573 - Compilers/Project/dist_data/$$

References

[1] Ding, C., and Zhong, Y. Reuse distance analysis. Tech. Rep. UR-CS-TR-741, Department of Computer Science, University of Rochester, Feb. 2001.

[2] Lee, C., Potkonjak, M., and Mangione-Smith, W. Mediabench: A tool for evaluating multimedia and communications systems. In *Proc. of 13th International Parallel Processing Symposium* (Nov. 1997).

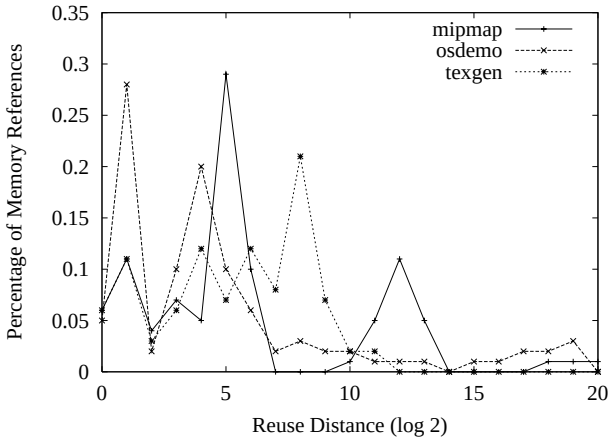

Figure 2: Percentages of the reuse distance frequency for the three programs that use the MESA Graphics Library.

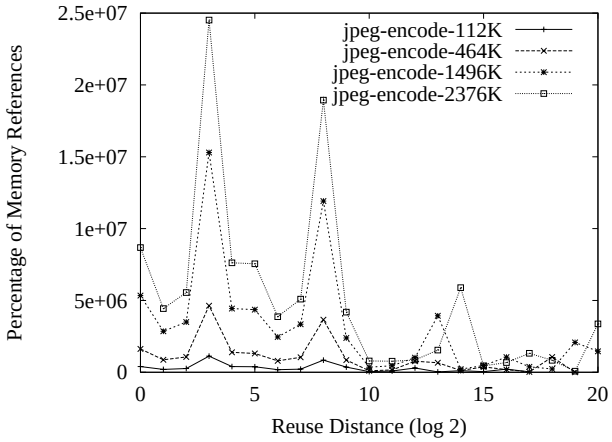

Figure 3: Absolute values of the reuse distance frequency for the JPEG encode program (*cjpeg)*. Four different PPM images have been used as input. Their sizes were: 112KB, 464KB, 1496KB and 2376KB.

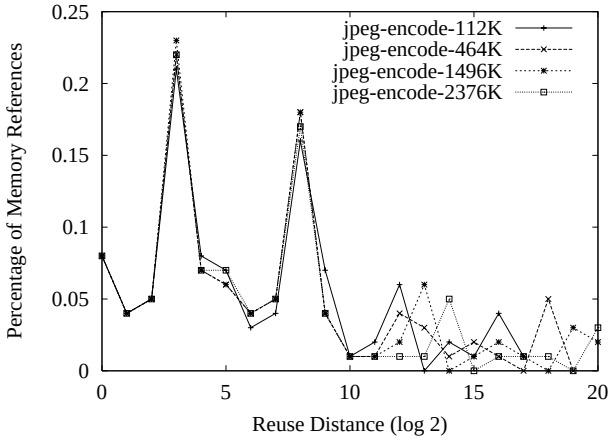

Figure 4: Percentages of the reuse distance frequency for the JPEG encode program (*cjpeg)*. Four different PPM images have been used as input. Their sizes were: 112KB, 464KB, 1496KB and 2376KB.

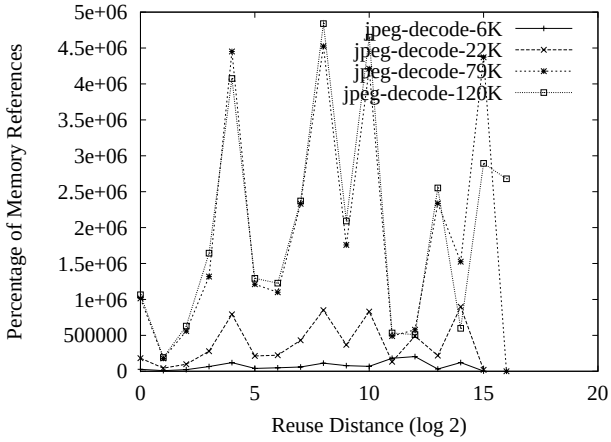

Figure 5: Absolute values of the reuse distance frequency for the JPEG decode program (*djpeg)*. Four different JPEG images have been used as input, and they were transformed to PPM images. Their sizes were: 6KB, 22KB, 79KB, 120KB. These JPEG images correspond to the PPM images that were used as input in the JPEG encode program.

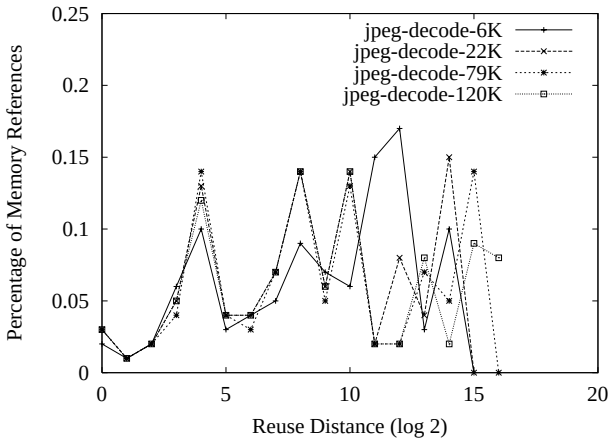

Figure 6: Percentages of the reuse distance frequency for the JPEG decode program (*djpeg)*. Four different JPEG images have been used as input, and they were transformed to PPM images. Their sizes were: 6KB, 22KB, 79KB, 120KB. These JPEG images correspond to the PPM images that were used as input in the JPEG encode program.