Review TACLeBench

Heiko Falk

September 29, 2015

Abstract

This document will contain the first review of the TACLeBench code regarding the topic image coding/encoding algorithms.

1 Image Coding/Encoding

- MediaBench/cjpeg_jpe6b_transupp
- MediaBench/cjpeg_jpeg6b_wrbmp
- MediaBench/epic
- ~~MediaBench/h264dec_ldecode_block~~
- MediaBench/h264dec_ldecode_macroblock
- MediaBench/mpeg2
- MiBench/susan

1.1 MediaBench/cjpeg_jpe6b_transupp

KEEP: Lots of true image processing, rotating and mirroring. Looks unique and useful.
TODO: At end of C source code, quite some 100's of lines are commented out. Is this also the case in the original MediaBench code? If not, why is it commented out here. . . ?

1.2 MediaBench/cjpeg_jpeg6b_wrbmp

KEEP: Routines to write JPEG images into some buffer in memory. Looks unique and useful, is definitely not redundant with MediaBench/cjpeg_jpe6b_transupp.
TODO: Again, lots of code are commented out at the end. Is this intentional?

1.3 MediaBench/epic

KEEP: Pretty complex and unique code (many original C files merged into a single one, input data defined as `extern` and put in separate compilation unit, size of input arrays left open, deeply-nested and regular loops, lots of DSP code. epic is known to be a stress-test for timing analyzers and compiler optimizations.
TODO: Includes matrix transpose kernels, convolution filter and DSP code. We

need to check whether we can kick out some of the trivial matrix and DSPStone kernels in favor of this one.

1.4 MediaBench/h264dec_ldecode_block

REMOVE: Code of rather low complexity (loop nesting level of approx. 3 only, not too complex code with many array references, only a little bit of arithmetic – mostly addition/subtraction, a few divisions or modulos; again lots of code commented out, contains lots of pre-initialized global arrays directly in the C source code), by far not as complex as epic above.

1.5 MediaBench/h264dec_ldecode_macroblock

KEEP: Much more complex code and completely different code as compared to the previous h264 benchmark, more control flow (if-else), more deeply nested loops; basically, 1 fat C function doing the whole work.

1.6 MediaBench/mpeg2

KEEP: Completely different algorithm than anything else in this category, should definitely be kept; code structured in various functions, with complex control flow and quite deeply nested loops inside.
TODO: Input data (fat!!! arrays) directly encoded in C source file, which makes $>$ 67,000 lines of the 69,000 lines of the entire code.

1.7 MiBench/susan

KEEP: Unique algorithm that is not comparable with anything else in this category (complex medical image processing algorithm, including inlined library calls for input/output handling and `memcpy`; quite some sequences with straight-line code)