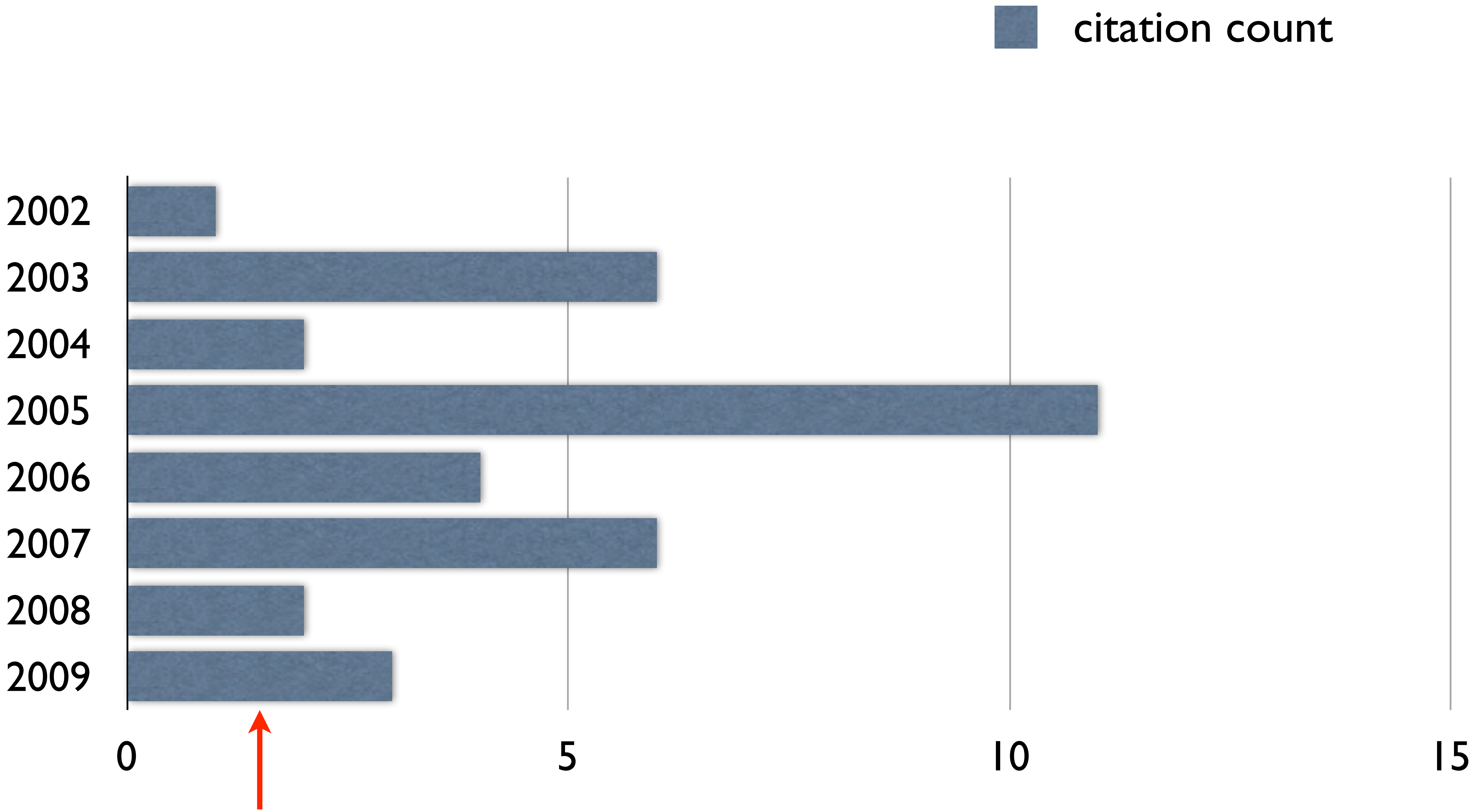
Profile-Guided Code Compression *

Saumya Debray
Department of Computer Science
University of Arizona
Tucson, AZ 85721.
debray@cs.arizona.edu

William Evans
Department of Computer Science
University of British Columbia
Vancouver B.C. Canada, V6T 1Z4.
will@cs.ubc.ca

2010.05.17
読んだ人: みよしたけふみ

Citation Count



DAC, ASPDAC,
IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems

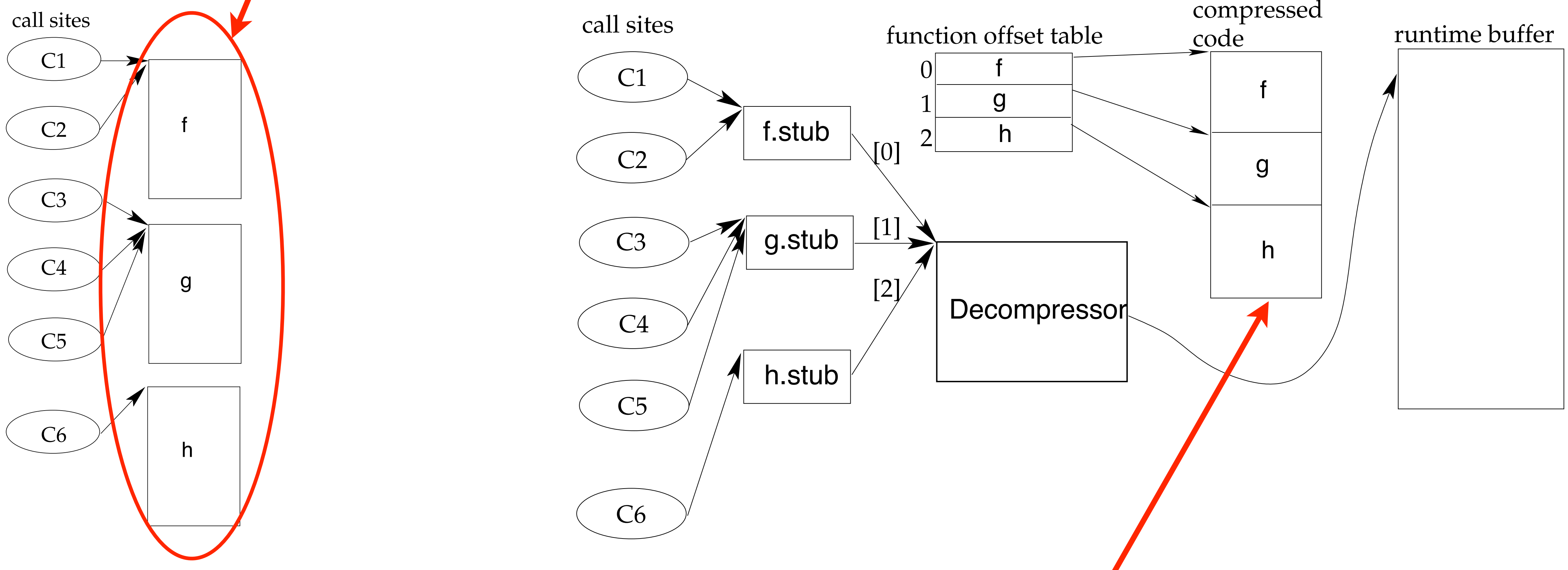
概要

- 組込向けのコードの削減
  - profile-directed optimization
  - runtime code generation/modification
  - program compression
- 削減量13.7%(θ=0.0) - 18.8%(θ=0.00005)
- 実行時間 +Δ(θ=0.0) - -27%(θ=0.00005)

The basic orgnization

infrequently executed functions

frequently codeとstubは*never-compressed* partに配置される



(a) Original

(b) Compressed

Figure 1: Code Organization: Before and After Compression

f中にgの呼び出しがあるときが問題

(1) 関数呼び出しを含む関数は圧縮しない

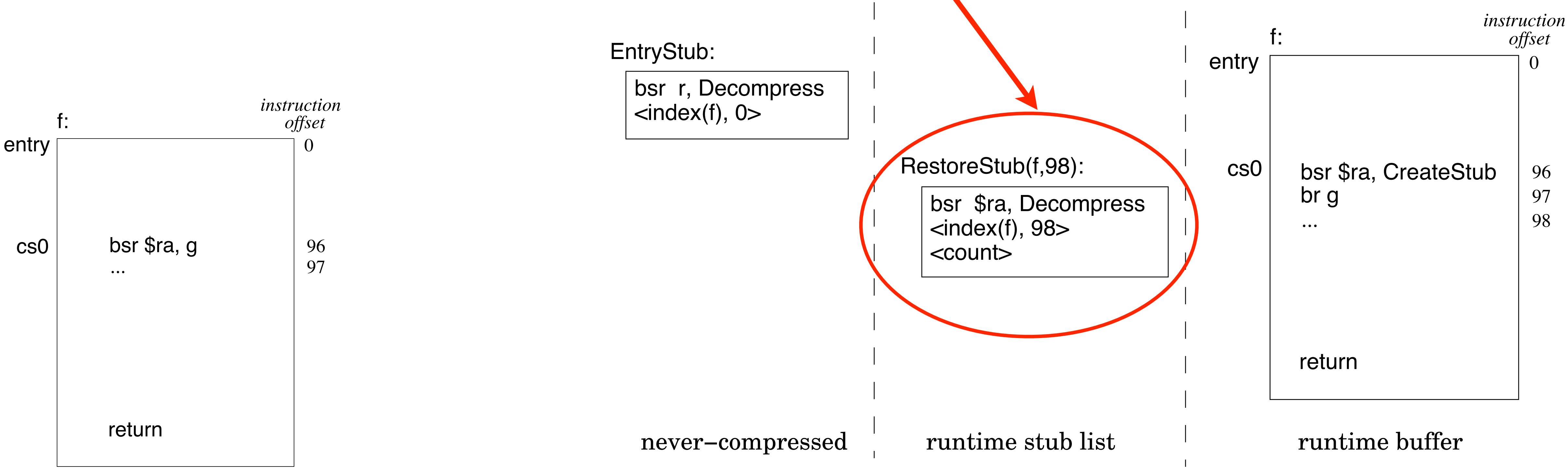
(2) 上書き/廃棄しない(JITっぽい)

(3) fをrestoreできるようにしてgを解凍/実行

関数呼び出し管理

gから戻るためだけのstubを実行時に生成する

never-compressed partに配置



(a) Original

(b) Transformed, during runtime after `CreateStub` has created `RestoreStub(f,98)`

Figure 2: Managing Function Calls Out of the Runtime Buffer.

Compression & Decompression

- splitting streams approach [9]
- by encoding each field using Huffman code
  - canonical Huffman encoding

Compressible Region

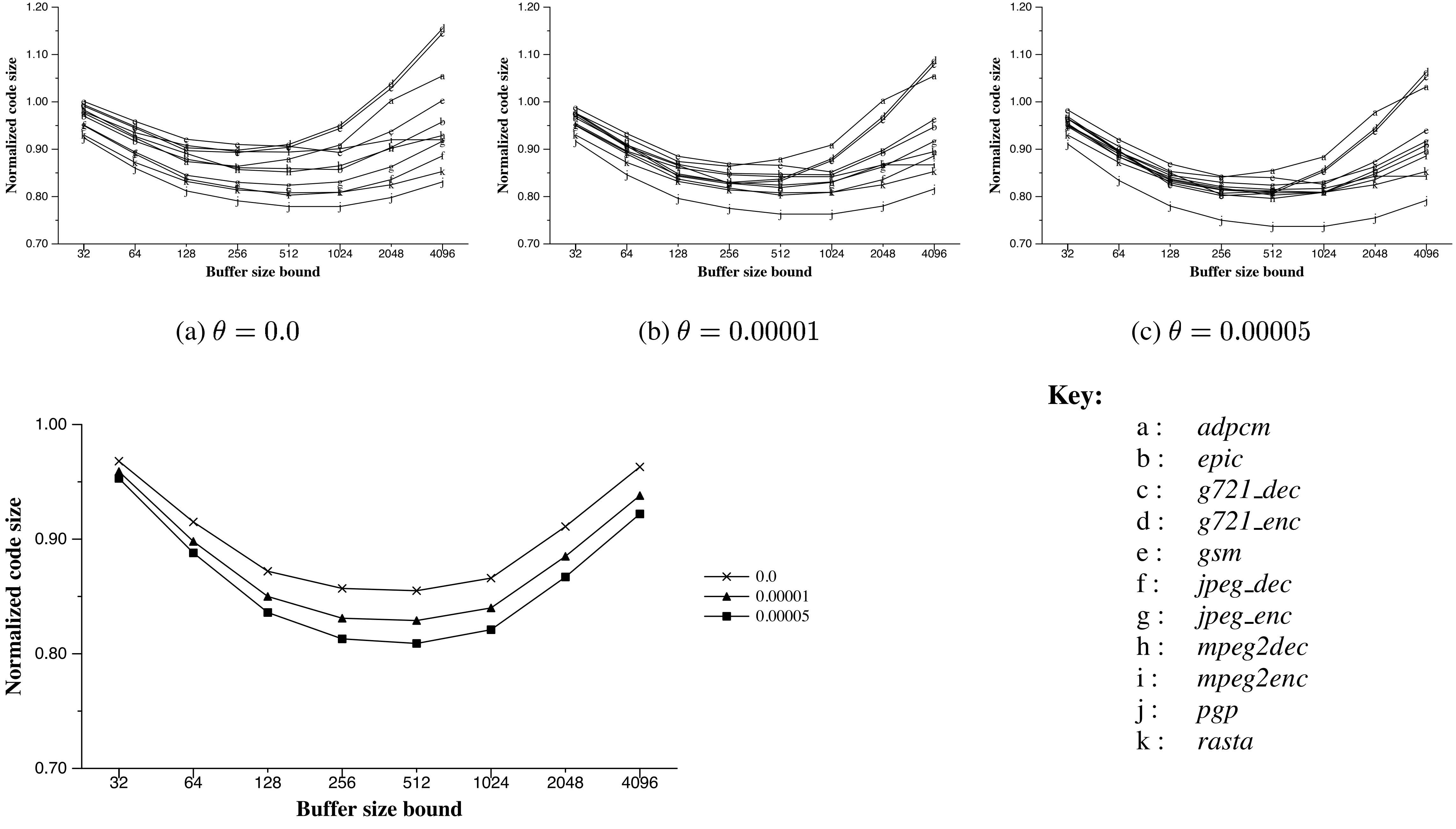
This is an optimization problem. The input is a control flow graph $G = (V, E)$ for a program in which a vertex b represents a basic block and has size $|b|$ equal to the number of instructions in the block, and an edge (a, b) represents a control transfer from a to b. In addition, the input specifies a subset U of the vertices that can be compressed. The output is a partition of a subset S of the compressible vertices U into regions $R_1, R_2, \ldots, R_k$ so that the following cost is minimized:

$$\begin{array}{ll}
\displaystyle\sum_{b \in V \setminus S} |b| & \text{never-compressed code} \\[2ex]
\displaystyle + \sum_{i=1}^{k} s(R_i) & \text{compressed code} \\[2ex]
+ \, k & \text{function offset table} \\[2ex]
+ \, 2|Y| & \text{entry stubs} \\[2ex]
\displaystyle + \max_i \{c_i + \sum_{b \in R_i} |b|\} & \text{runtime buffer}
\end{array}$$

where $s(R_i)$ is the size of the region R_i after compression, Y is the set of blocks requiring an entry stub, i.e.,

$$Y = \{b : (a, b) \in E, b \in R_i, \text{ and } a \notin R_i \text{ for some } i\},$$

Compressible Regions



(a) $\theta = 0.0$

(b) $\theta = 0.00001$

(c) $\theta = 0.00005$

(d) mean

Key:

a :	*adpcm*
b :	*epic*
c :	*g721_dec*
d :	*g721_enc*
e :	*gsm*
f :	*jpeg_dec*
g :	*jpeg_enc*
h :	*mpeg2dec*
i :	*mpeg2enc*
j :	*pgp*
k :	*rasta*

upper bound of runtime buffer K= 512

Figure 3: Effect of Buffer Size Bound on Code Size

Cold Codeと圧縮後サイズ

(the geometric mean of) the relative amount of cold and compressible code in our programs

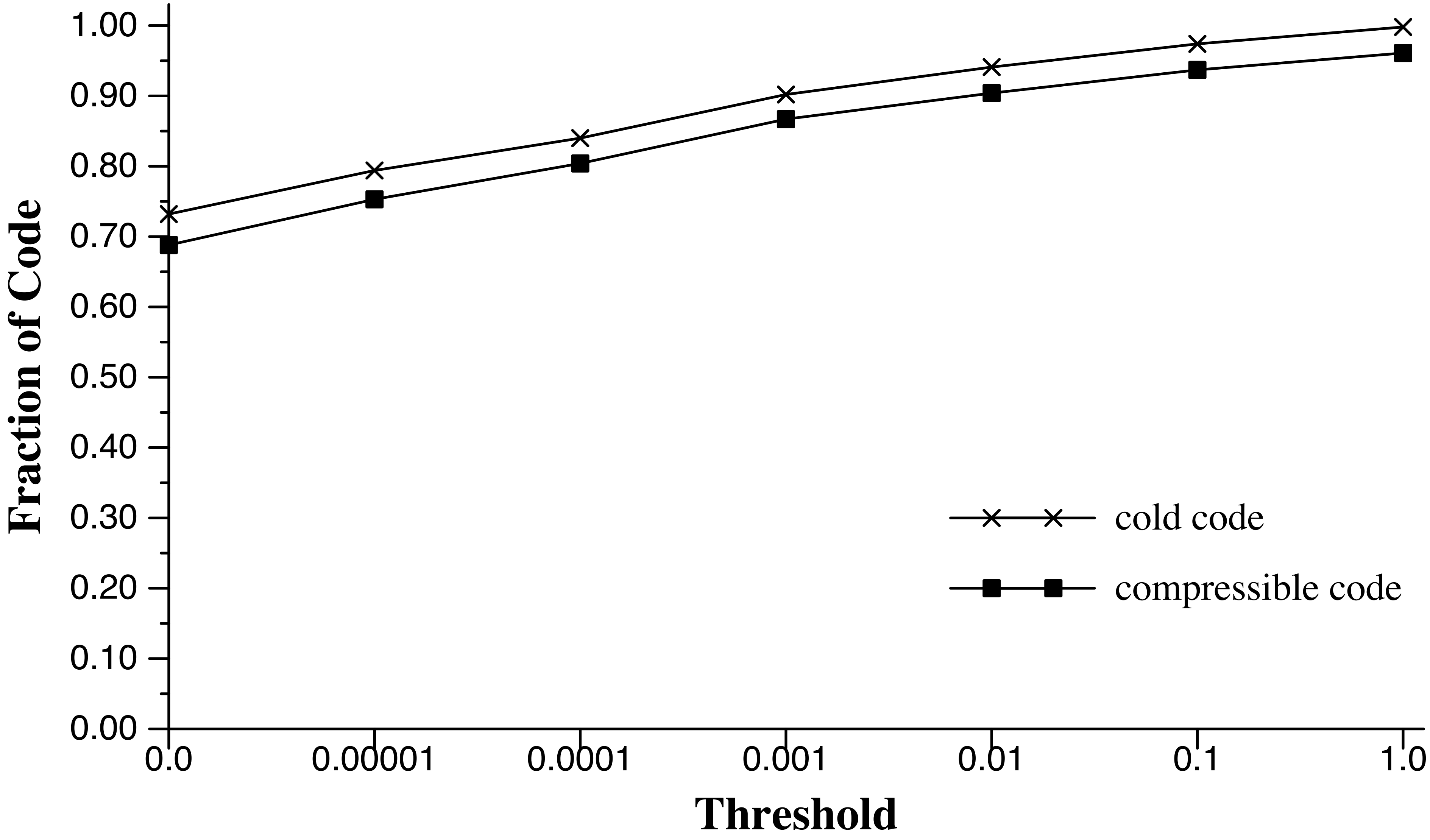
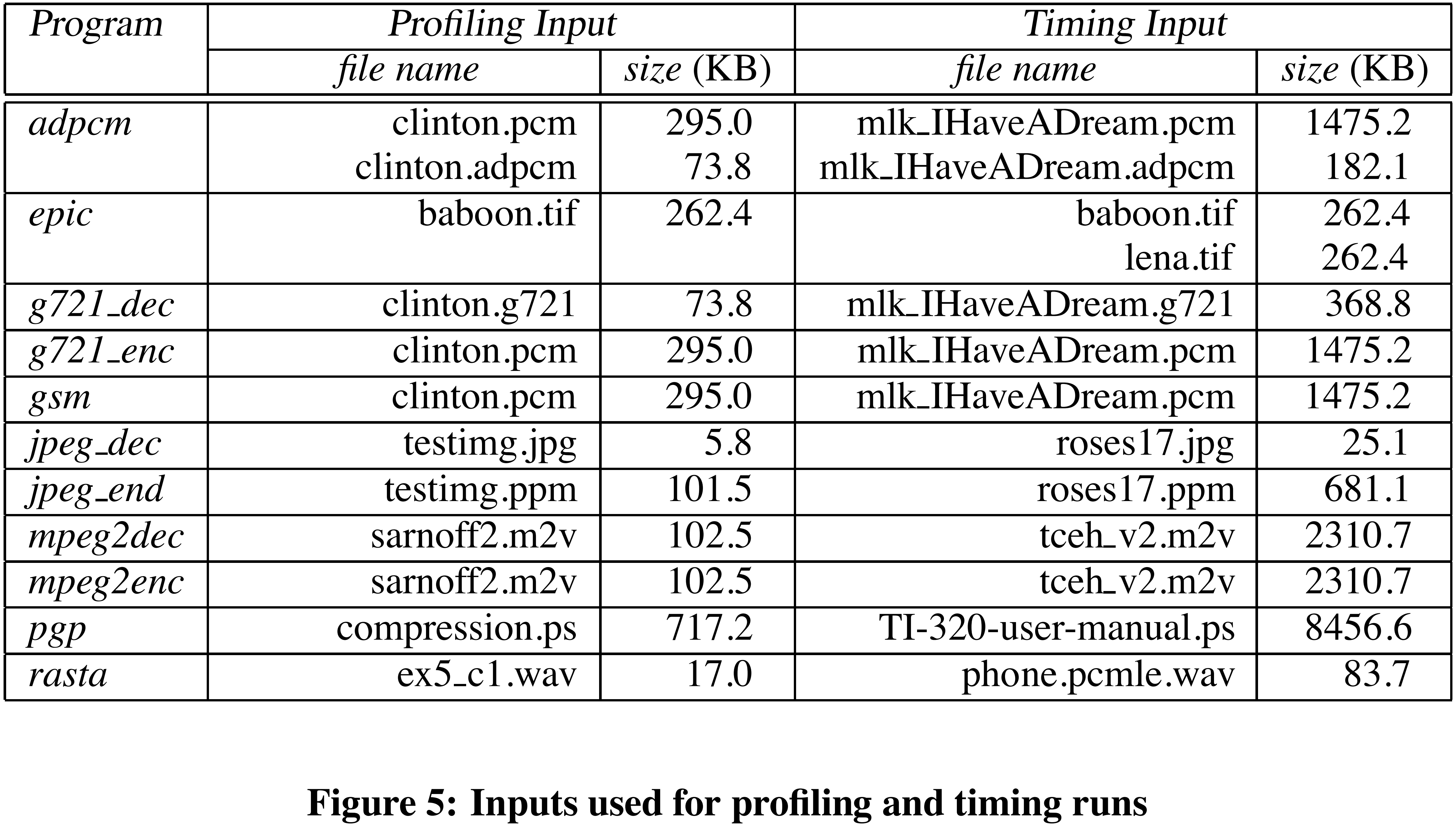


Figure 4: Amount of Cold and Compressible Code (Normalized)

Optimizations

- Buffer-Safe Functions
  - 圧縮コードから非圧縮コード呼出し
- Unswitching
  - indirect jump

プロファイル　　　　　　評価



Program	Profiling Input		Timing Input	
	file name	*size* (KB)	*file name*	*size* (KB)
adpcm	clinton.pcm clinton.adpcm	295.0 73.8	mlk_IHaveADream.pcm mlk_IHaveADream.adpcm	1475.2 182.1
epic	baboon.tif	262.4	baboon.tif lena.tif	262.4 262.4
g721_dec	clinton.g721	73.8	mlk_IHaveADream.g721	368.8
g721_enc	clinton.pcm	295.0	mlk_IHaveADream.pcm	1475.2
gsm	clinton.pcm	295.0	mlk_IHaveADream.pcm	1475.2
jpeg_dec	testimg.jpg	5.8	roses17.jpg	25.1
jpeg_end	testimg.ppm	101.5	roses17.ppm	681.1
mpeg2dec	sarnoff2.m2v	102.5	tceh_v2.m2v	2310.7
mpeg2enc	sarnoff2.m2v	102.5	tceh_v2.m2v	2310.7
pgp	compression.ps	717.2	TI-320-user-manual.ps	8456.6
rasta	ex5_c1.wav	17.0	phone.pcmle.wav	83.7

Figure 5: Inputs used for profiling and timing runs

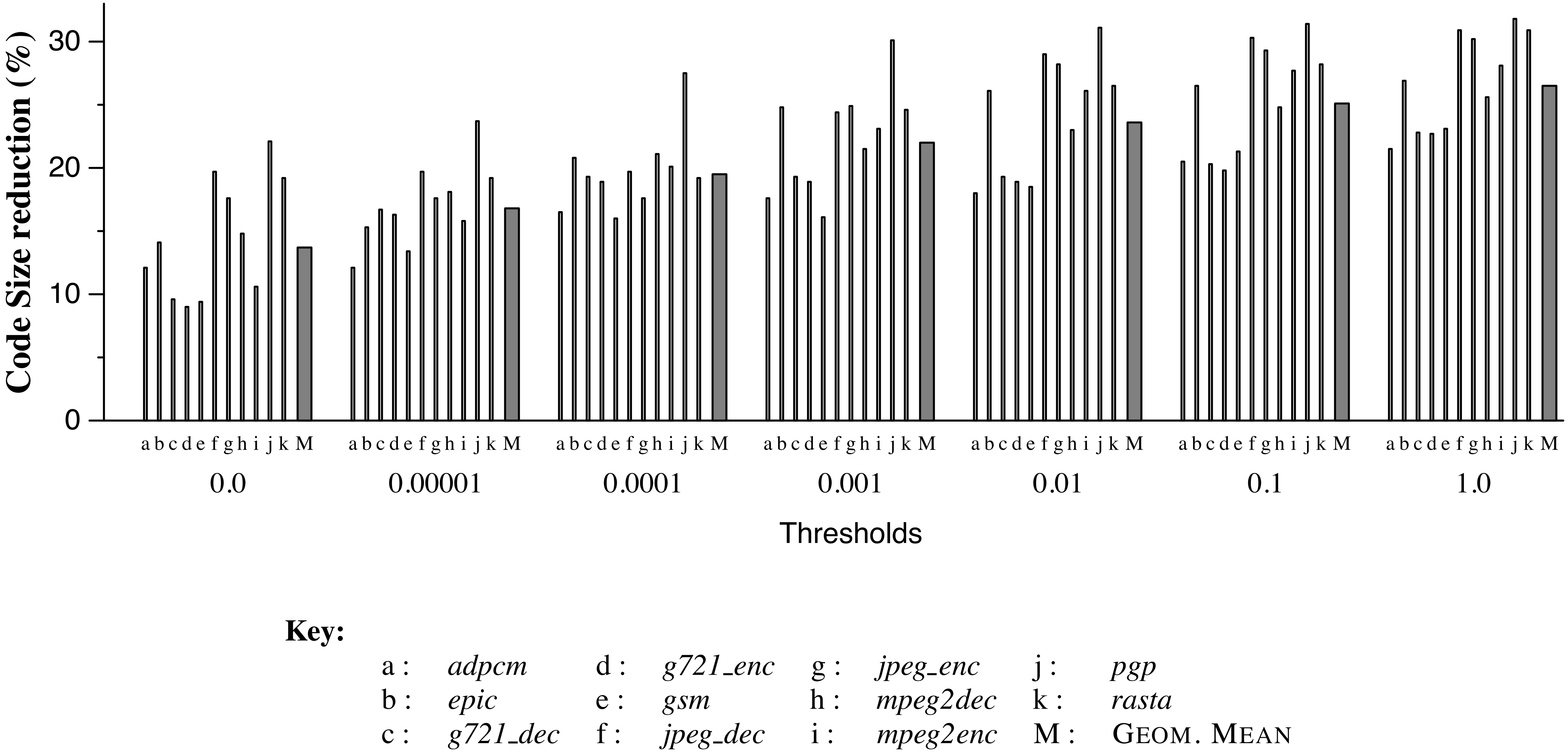
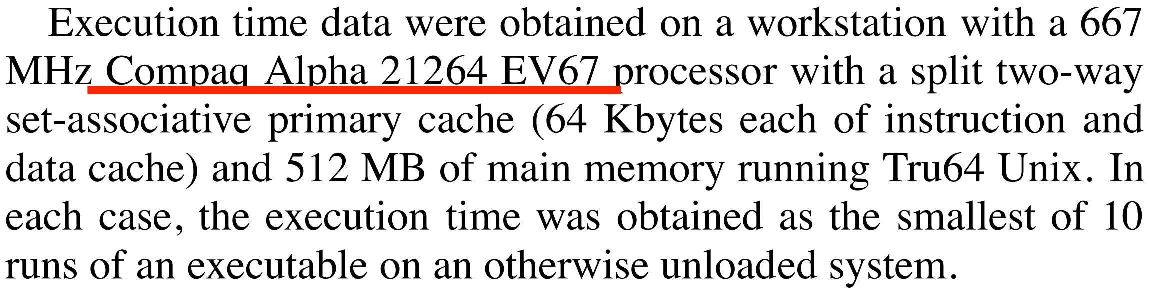
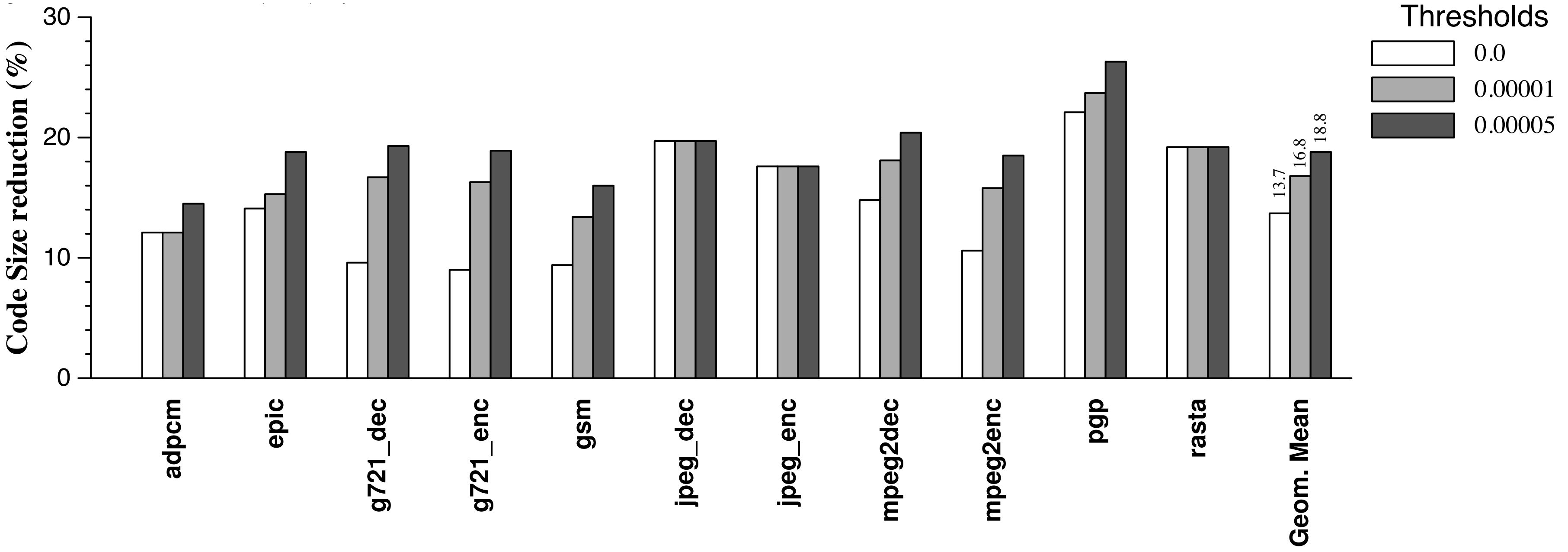


Figure 6: Code Size Reduction due to Profile-Guided Code Compression at Different Thresholds

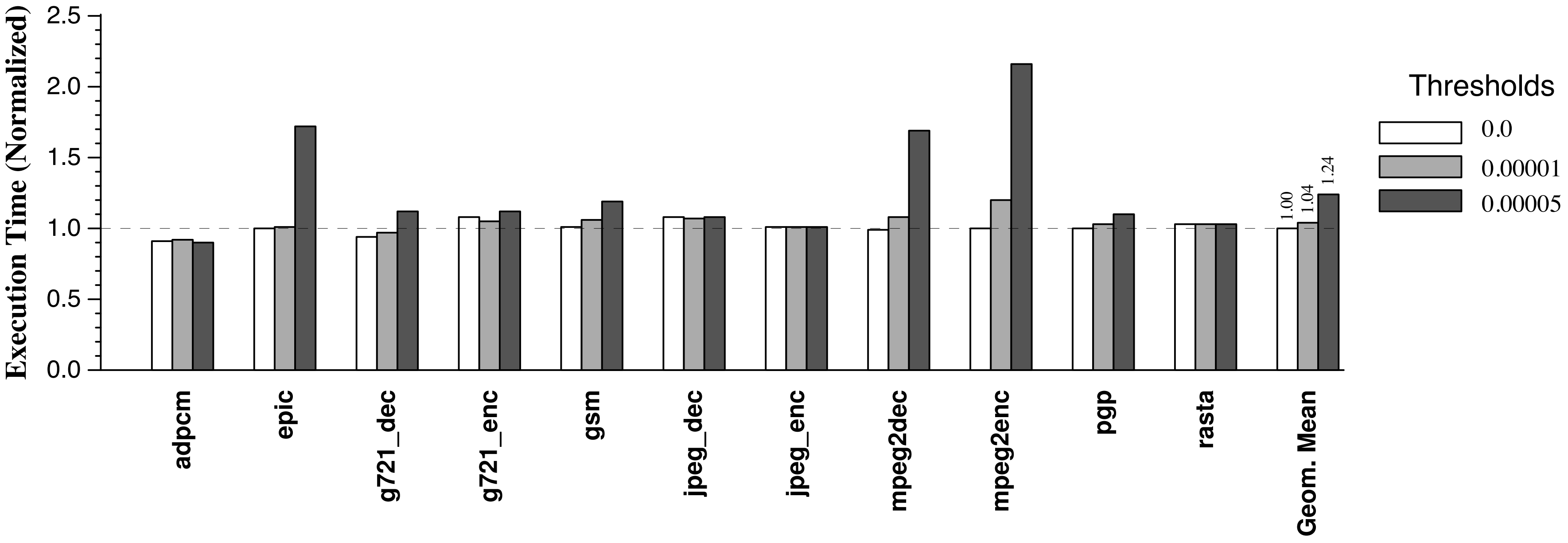
評価結果



Execution time data were obtained on a workstation with a 667 MHz Compaq Alpha 21264 EV67 processor with a split two-way set-associative primary cache (64 Kbytes each of instruction and data cache) and 512 MB of main memory running Tru64 Unix. In each case, the execution time was obtained as the smallest of 10 runs of an executable on an otherwise unloaded system.



(a) Code Size



(b) Execution Time