Inlining-Benefit Prediction with Interprocedural Partial Escape Analysis (Appendix)

Matthew Edwin Weingarten*
Oracle Labs
Zurich, Switzerland
matthew.weingarten@inf.ethz.ch

Theodoros Theodoridis
Swiss Federal Institute of Technology
Zurich, Switzerland
theodoros.theodoridis@inf.ethz.ch

Aleksandar Prokopec
Oracle Labs
Zurich, Switzerland
aleksandar.prokopec@oracle.com

Abstract

Inlining is the primary facilitating mechanism for *intraprocedural* Partial Escape Analysis (PEA), which allows for the removal of object allocations on a branch-by-branch basis and is critical for performance in object-oriented languages. Prior work used *interprocedural* Escape Analysis to make inlining decisions, but it discarded control-flow-sensitivity when crossing procedure boundaries, and did not weigh other metrics to model the cost-benefit of inlining, resulting in unpredictable inlining decisions and suboptimal performance. Our work addresses these issues and introduces a novel Interprocedural Partial Escape Analysis algorithm (IPEA) to predict the inlining benefits, and improve the cost-benefit model of an existing optimization-driven inliner. We evaluate the implementation of IPEA in GraalVM Native Image, on industry-standard benchmark suites Dacapo, ScalaBench, and Renaissance. Out of 36 benchmarks with a geometric mean runtime improvement of 1.79%, 6 benchmarks achieve an improvement of over 5% with a geomean of 9.10% and up to 24.62%, while also reducing code size and compilation times compared to existing approaches.

CCS Concepts: • **Software and its engineering → Source code generation**; Runtime environments; Just-in-time compilers.

Keywords: escape analysis, inlining, compilers, graalvm

ACM Reference Format:
Matthew Edwin Weingarten, Theodoros Theodoridis, and Aleksandar Prokopec. 2022. Inlining-Benefit Prediction with Interprocedural Partial Escape Analysis (Appendix). In *Proceedings of the 14th ACM SIGPLAN International Workshop on Virtual Machines and Intermediate Languages (VMIL '22), December 05, 2022, Auckland, New Zealand*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3563838.3567677

*Also with Swiss Federal Institute of Technology.

VMIL '22, December 05, 2022, Auckland, New Zealand

ACM ISBN 978-1-4503-9912-8/22/12.
https://doi.org/10.1145/3563838.3567677

A Appendix

This document is a companion document to the original publication. We show the complete results for allocation profiling in Table 1 and heuristic tuning in Figure 1.

Table 1. Allocation profiling results for entire bench suite. Threshold for profiling an allocation site is > 1MB over the entire runtime (multiple runs). Some cases show an increase in allocated bytes for PESI or IPEA and can be caused by grouping allocation nodes after inlining, increasing likelihood of passing the threshold.

	GB allocated			$\frac{IPEA-PESI}{PESI}$		
Benchmark	Base	PESI	IPEA	ΔAlloc	ΔRuntime	
avrora	2.94	1.49	2.76	+85.6%	-16.07%	Dacapo
fop	7.92	7.37	7.41	+0.41%	-2.66%	
h2	111.93	111.12	110.47	-0.58%	-0.52%	
luindex	4.32	4.31	4.08	-5.26%	-0.77%	
lusearch	233.99	222.69	201.63	-9.46%	-1.12%	
pmd	125.34	120.49	114.06	-5.34%	-2.23%	
sunflow	71.41	69.19	82.84	+19.73%	-0.81%	
xalan	72.6	72.48	70.81	-2.3%	+16.17%	
geomean				+6.5%	-1.33%	
akka-uct	803.12	795.99	797.34	+0.17%	-0.23%	Renaissance
als	191.08	189.54	187.05	-1.31%	-0.35%	
db-shootout	710.24	694.21	688.74	-0.79%	+5.07%	
dotty	78.78	77.73	74.91	-3.63%	-2.56%	
finagle-chirper	1584.62	1527.25	1467.49	-3.91%	-7.74%	
finagle-http	210.75	200.94	189.49	-5.7%	-0.33%	
fj-kmeans	576.4	576.22	577.29	+0.19%	-2.4%	
future-genetic	75.35	76.19	114.29	+50.0%	+1.74%	
gauss-mix	110.24	108.14	108.47	+0.3%	+0.16%	
mnemonics	143.76	156.53	133.48	-14.73%	-24.62%	
movie-lens	324.5	323.24	319.18	-1.26%	-2.55%	
naive-bayes	59.34	54.5	54.46	-0.07%	+0.25%	
neo4j-analytics	136.77	129.85	115.69	-10.91%	-5.58%	
page-rank	294.67	282.43	266.37	-5.68%	+6.13%	
par-mnemonics	141.65	153.95	146.83	-4.63%	-0.81%	
philosophers	359.02	337.67	339.09	+0.42%	+0.37%	
reactors	65.07	63.15	58.85	-6.82%	-2.21%	
rx-scrabble	15.76	16.6	14.32	-13.72%	-0.05%	
scala-doku	6.27	5.6	5.55	-0.84%	+0.62%	
geomean				-1.9%	-2.07%	
apparat	22.95	22.41	22.39	-0.09%	-4.14%	ScalaBench
factorie	116.36	106.76	90.7	-15.05%	+3.35%	
kiama	23.92	24.04	23.55	-2.04%	+0.91%	
scalac	20.98	20.76	19.78	-4.73%	-5.74%	
scaladoc	23.23	22.39	18.69	-16.52%	-2.48%	
scalap	6.66	6.49	6.57	+1.23%	-5.78%	
scalariform	8.27	8.09	7.77	-3.99%	+1.78%	
scalaxb	14.15	14.2	14.1	-0.74%	-1.81%	
tmt	251.32	136.03	250.33	+84.02%	-0.13%	
geomean				+2.04%	-1.61%	
total				**+1.02%**	**-1.79%**	

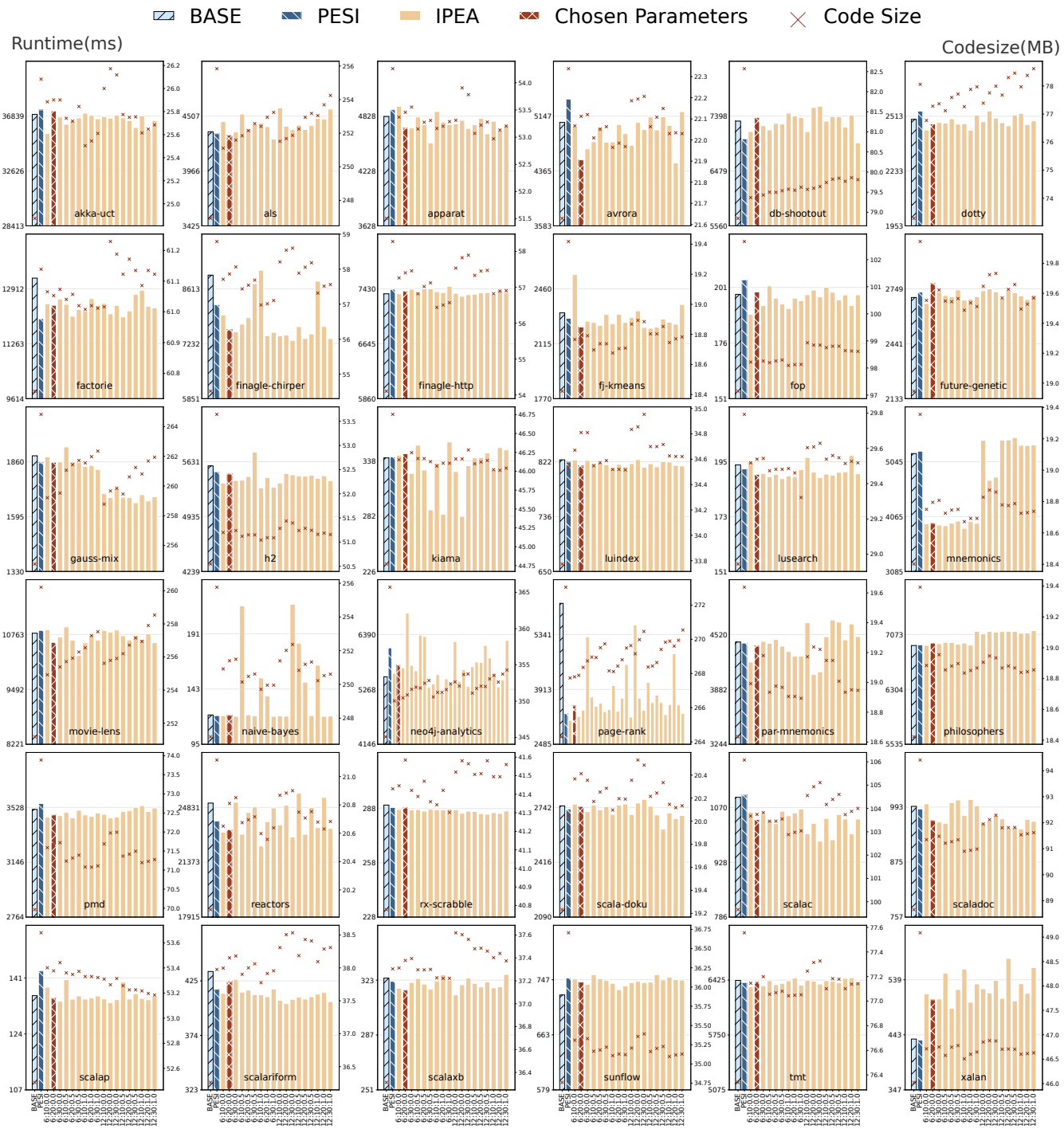

Figure 1. Grid search of hyper parameters, left axis is runtime, right axis is code size, lower is better. Labels on the x-axis show parameters ($\delta_{eb} : \delta_{mb} : \delta_{cw}$).