The DaCapo Benchmarks: Java Benchmarking Development and Analysis

Stephen M Blackburn[α β], Robin Garner[β], Chris Hoffmann[γ], Asjad M Khan[γ], Kathryn S McKinley[δ], Rotem Bentzur[ε], Amer Diwan[ζ], Daniel Feinberg[ε], Daniel Frampton[β], Samuel Z Guyer[η], Martin Hirzel[θ], Antony Hosking[ι], Maria Jump[δ], Han Lee[α], J Eliot B Moss[γ], Aashish Phansalkar[δ], Darko Stefanović[ε], Thomas VanDrunen[κ], Daniel von Dincklage[ζ], Ben Wiedermann[δ]

the DaCapo benchmarks are not definitive, and they may or may not be representative of workloads that vendors and clients care about most. Regardless, we believe this paper is a step towards a wider community discussion and eventual consensus on how to select, measure, and evaluate benchmarks, VMs, compilers, runtimes, and hardware for Java and other managed languages.

2. Related Work

We build on prior methodologies and metrics, and go further to recommend how to use them to select benchmarks and for best practices in performance evaluation.

2.1 Java Benchmark Suites

In addition to SPEC (discussed in Section 3), prior Java benchmarks suites include Java Grande [26], Jolden [11, 34], and Ashes [17]. The Java Grande Benchmarks include programs with large demands for memory, bandwidth, or processing power [26]. They focus on array intensive programs that solve scientific computing problems. The programs are sequential, parallel, and distributed. They also include microbenchmark tests for language and

variable workloads make performance hard to analyze and reason about. For example, the level and number of classes optimized and

(a) SPEC

	Classes	Methods	Methods & Bytecodes Compiled		I-Cache Misses
			All		

	Heap Volume (MB)		Heap Objects	Mean Object Size	4MB
			Alloc/		

objects are 12 bytes, but none stay live. In fact, 65% of live objects are 2 Kbytes, whereas they make up only 2% of allocated objects. How well these large objects are handled will thus in large part determine the performance of the collector on *xalan*.

Metric	Rank			
	PC1	PC2	PC3	PC4
Architecture				
Instruction mix – ALU	-9	-15	7	-3
Intruction mix – branches	-10	**-4**	16	1
Instruction mix – memory	**1**	13	e718. 1 29457.. 1 29457.. 1 29457.. 1 29457. 0 J 0.398 w 0.199 0 m 0.1/F952666 l S Q 1 0 1 -275.806 -681.9	

Acknowledgments

We thank Andrew Appel, Randy Chow, Frans Kaashoek, and Bill Pugh who encouraged this project at our three year ITR review. We thank Mark Wegman who initiated the public availability of Jikes RVM, and the developers of Jikes RVM. Fahad Gilani wrote the original version of the measurement infrastructure for his ANU Masters Thesis.

otparalleal rbragecollectors(,)4n

References

[1] B. Alpern, D. Attanasio, J. J. Barton, A. Cocchi, S. F. Hummel, D. Lieber, M. Mergen, T. Ngo, J. Shepherd, and S. Smith. Implementing Jalapeño in Java.

CneVal,37(encM)3170ionnommtinge(,)]TJ-aand Apbliatsioes

Benchmark Descriptioh and Origin

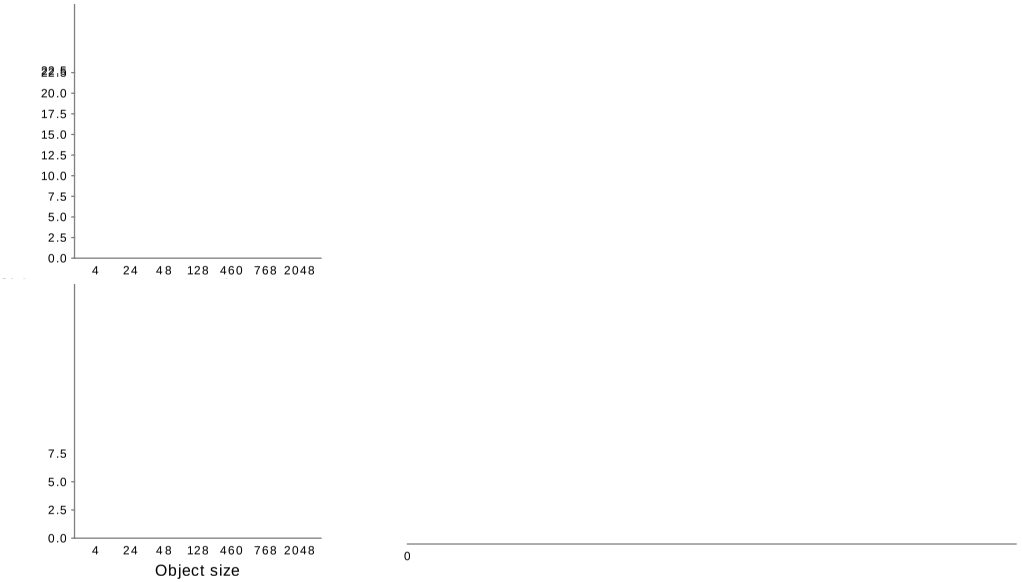
Short Descriptioh	A parser generator and translator gener-ator
Long Descriptioh	ANTLR parses one or more grammar files and generate a parser and lexical analyizer for each.
Threads	Sihgle threaded

Benchmark Description and Origin

Short Description	A Bytecode-level optimization and analysis tool for Java
Long Description	BLOAT analyzes and optimizes some of its own class files
Threads	Single threaded
Repeats	Single iteration, transitively optimizes classes referenced by a single root class
Version	1.0
Copyright	Copyright (c) 1997-2001 Purdue Research Foundation of Purdue University
Author	Nathaniel Nystrom and David Whitlock
License	BSD-style

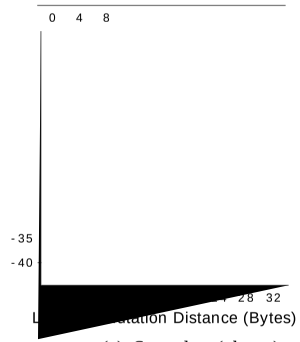
Benchmark Characteristics

Total Allocation (MB)	1,222.5
(Obj)	33,487,434
Maximum Live (MB)	6.2
(Obj)	149,395
Pointer Mutations (M)	257.84
Classes Loaded	281



(a) Allocated (above) and Live (below) Object Size Histograms and Time-series

(b) Heap Composition Time-series, in Allocations (above) and Mutations (below)



(c) Snapshot (above) and Mutation (below) Pointer Distance Histograms and Time-series

Figure 5. Benchmark Characteristics: bloat

Benchmark Description and Origin

Short Description	A graph plotting toolkit and pdf renderer

Benchmark Description and Origin

Short Description	An integrated development environment
Long Description	Run a series of eclipse jdt (non-gui) performance tests
Threads	Workload is single threaded, but Eclipse uses multiple threads internally
Repeats	Single iteration, performs multiple distinct Eclipse tasks
Version	3.1.2
Copyright	Eclipse Foundation

Benchmark Description and Origin

Short Description	An output-independant print formatter
Long Description	fop takes an XSL-FO file, parses it and formats it, generating an encrypted pdf file
Threads	Single threaded
Repeats	Single iteration, renders a single XSL-FO file

Benchmark Description and Origin

Short Description	An SQL relational database engin7 written in Java
Long Description	HSQLDB executes a JDBC-like in-memory benchmark, executing a number of transactions against a model of a banking application
License	The HSQLDB license.
Threads	10 client threads
Repeats	40 transactions per client
Version	1.8.0.4
Copyright	Copyright (c) 2001-2002, The HSQL Development Group
Author	The HSQLDB Development Group

Total Allocation (MB)	142.7

Benchmark Description and Origin

Short Description	A python interpreter written in Java
Long Description	jython executes (interprets) the pybench benchmark or a small python program
Threads	Single threaded
Repeats	Single iteration runs a single iteration of the pybench python benchmark
Version	2.1
Copyright	Copyright (c) Python Software Foundation
Author	Jim Hugunin and Barry Warsaw
License	Jython Software License.

Benchmark Characteristics

Total Allocation (MB)	1,183.4
(Obj)	25,940,819
Maximum Live (MB)	0.1
(Obj)	2,788
Pointer Mutations (M)	82.96
Classes Loaded	251

Benchmark Description and Origin

Short Description	A text indexing tool
Long Description	Indexes a set of documents, the works of Shakespeare and the King James Bible

Benchmark Description and Origin

Short Description	A source code analyzer for Java
Long Description	pmd analyzes a list of Java classes for a range of source code problems
Threads	Single threaded
Repeats	Single iteration checks a single large source file against 18 coding rules
Version	1.8