Stack Allocation of Objects in the Cacao Virtual Machine

Institut für Computersprachen
Technische Universtät Wien
Austria

Peter Molnar   Andreas Krall   Florian Brandner

Peter Molnar is affiliated with Theobroma Systems Design und Consulting GmbH.

Overview

CACAO Virtual Machine

- JIT-only research Java Virtual Machine
- ultra-fast basic compiler
- higher optimizing compiler under development
- recompilation with optimizations
- on-stack replacement
- deoptimization when assumptions become invalid

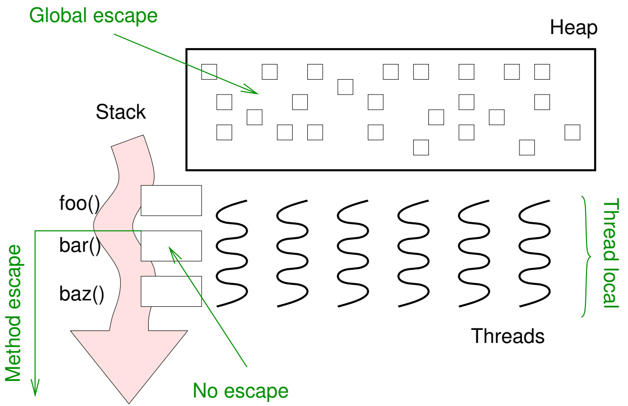
Memory Management

- object allocation
- garbage collection
- stack allocation more efficient
- scalar replacement

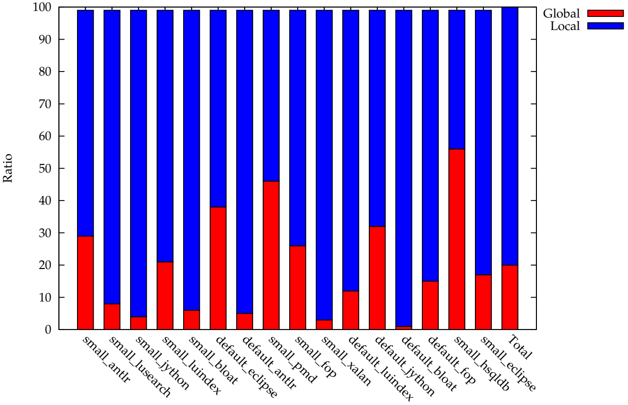
Potential of Stack Allocation

- instrumentation by JIT compiler
- dynamic not static evaluation
- grouping of objects into regions
  global (heap), local (thread), stack (number of frames)
- only executed path are taken into account

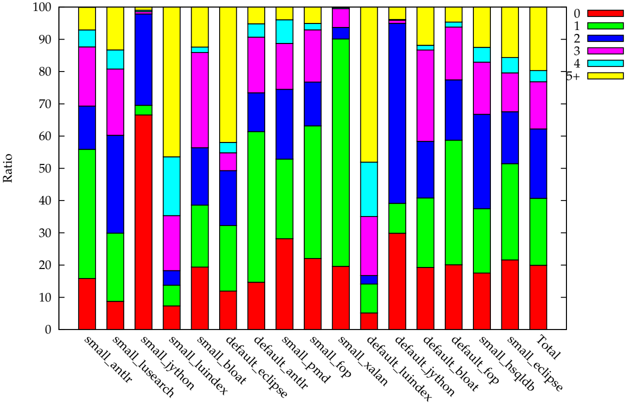
Escape States



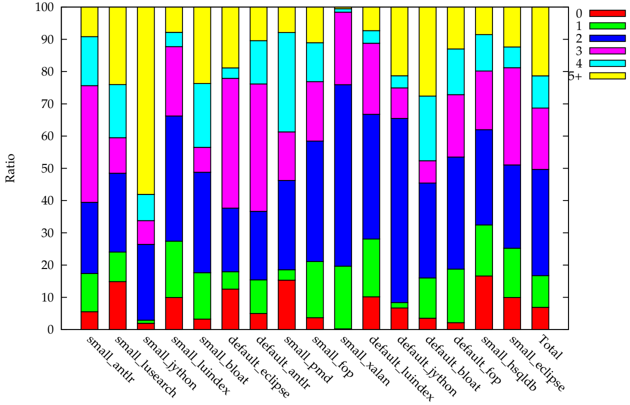
Thread Local Objects



Depth of Objects passed towards Caller



Depth of Objects passed towards Callee



Analyis Algorithm

- factored control graph
- static single assignment form
- loop analysis
- native functions

Escape State

ESCAPE_NONE: the object is accessible only from its creating method.

ESCAPE_METHOD: the object escapes its creating method, but does not escape the creating thread.

ESCAPE_METHOD_RETURN: the object escapes its creating method via a return to the caller.

ESCAPE_GLOBAL: the object escapes its creating method and its creating thread.

Analysis Details

- Steensgard style analysis
- flow insensitive
- intraprocedural analysis
- interprocedural analysis

Native Methods

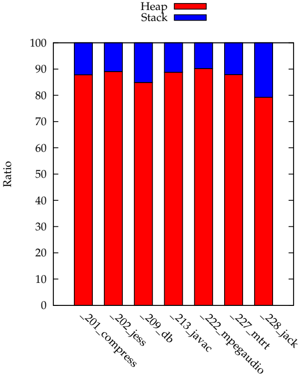
Benchmark	Pessimistic	Optimistic
_202_jess	0.07%	26.54%
_228_jack	48.16%	69.18%

SpecJCM98

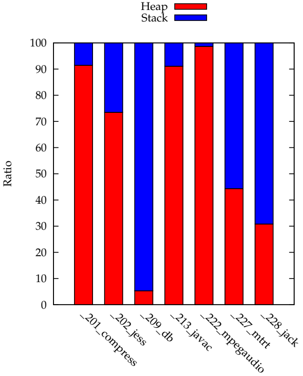
Benchmark	Pessimistic	Optimistic
eclipse	3.37%	3.62%
pmd	0.21%	11.87%
xalan	2.47%	3.73%

dacapo

Ratio Stack Allocated Objects SpecJVM98

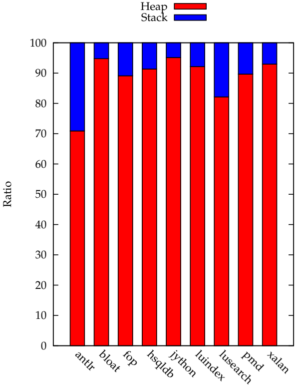


Static

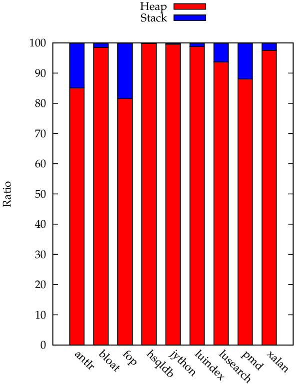


Dynamic

Ratio Stack Allocated Objects dacapo



Static



Dynamic

Effectiveness SpecJVM98

Benchmark	Inlining 1	Inlining 2	Stack allocated
_201_compress	30.87 %	44.84 %	8.56 %
_202_jess	64.27 %	70.09 %	26.54 %
_209_db	94.73 %	99.32 %	94.68 %
_213_javac	40.13 %	41.45 %	8.93 %
_222_mpegaudio	16.87 %	24.44 %	1.33 %
_227_mtrt	93.30 %	94.18 %	55.63 %
_228_jack	64.88 %	90.59 %	69.18 %

Effectiveness dacapo

Benchmark	Inlining 1	Inlining 2	Stack allocated
antlr	60.53 %	78.58 %	14.84 %
bloat	61.05 %	78.05 %	1.50 %
fop	59.77 %	79.96 %	18.42 %
hsqldb	21.48 %	31.80 %	0.22 %
jython	7.59 %	60.09 %	0.40 %
luindex	16.23 %	18.33 %	1.22 %
lusearch	36.34 %	66.74 %	6.25 %
pmd	56.21 %	74.79 %	11.87 %
xalan	32.21 %	63.56 %	2.46 %

Execution Times SpecJVM98

Benchmark	With EA	Without EA	Speedup
_201_compress	8.51 s	8.50 s	-0.12 %
_202_jess	46.01 s	55.81 s	21.29 %
_209_db	28.52 s	42.71 s	49.75 %
_213_javac	50.91 s	53.56 s	5.20 %
_222_mpegaudio	13.02 s	13.12 s	0.77 %
_227_mtrt	20.82 s	35.23 s	69.21 %
_228_jack	43.69 s	64.06 s	64.62 %

Execution Times dacapo

Benchmark	With EA	Without EA	Speedup
antlr	34.16 s	37.52 s	9.84 %
bloat	219.34 s	216.95 s	-1.09 %
fop	11.95 s	12.64 s	5.77 %
hsqldb	2.77 s	2.90 s	4.69 %
jython	274.30 s	274.54 s	0.09 %
luindex	96.78 s	96.80 s	0.02 %
lusearch	247.07 s	261.70 s	5.92 %
pmd	178.63 s	197.12 s	10.35 %
xalan	73.19 s	73.24 s	0.07 %

Conclusion

- stack allocation is very effective in reducing cost of memory management
- CacaoVM dedects up 95% of potentially stack allocatable objects
- CacaoVM gets speedups up to 69%
- further evaluate more precise analysis algorithms