

Intelligent Selection of Application Specific Garbage Collectors

Jeremy Singer¹

Gavin Brown¹
John Cavazos²

Ian Watson¹

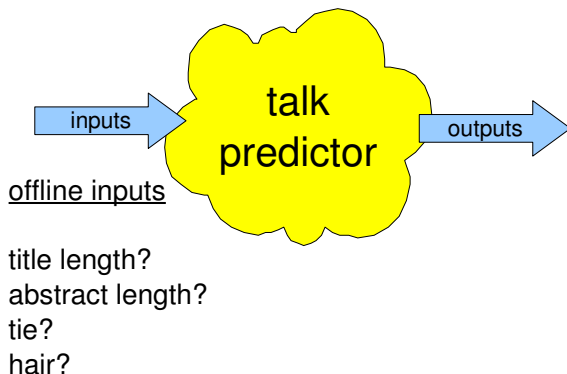
University of Manchester¹, UK

University of Edinburgh², UK

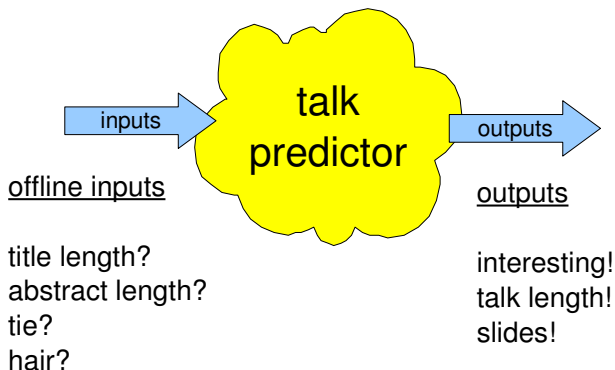
What is Machine Learning?



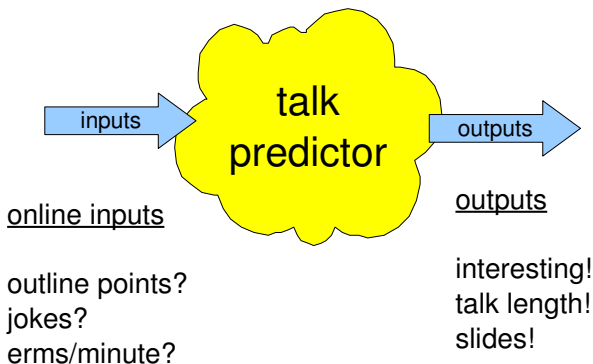
What is Machine Learning?



What is Machine Learning?



What is Machine Learning?



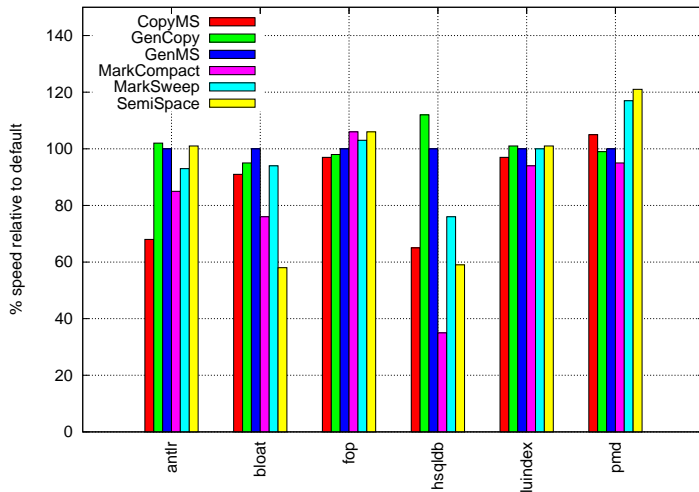
Garbage Collection Research

- for 50 years
- efficient ways to manage heap
- changes with programming style (generational)
- plenty of algorithms
 - ▶ mark and sweep
 - ▶ compacting
 - ▶ copying
 - ▶ generational

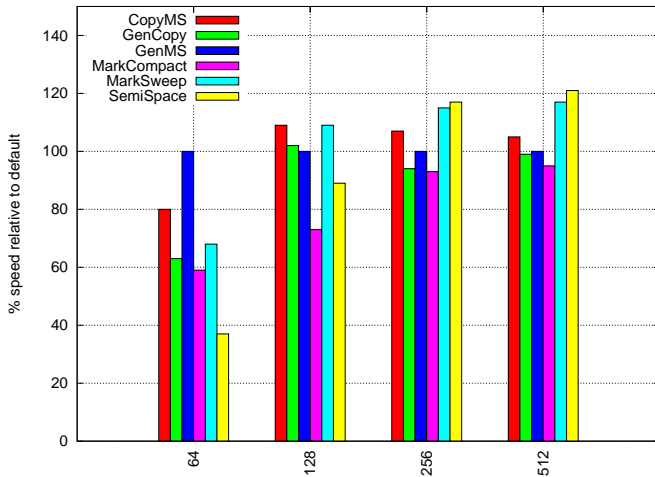
Why Application-Specific GC?

- different GC algorithms
 - ▶ better for different apps
 - ▶ better for different heap sizes

Per-Benchmark Variations



Per-Heapsize Variations



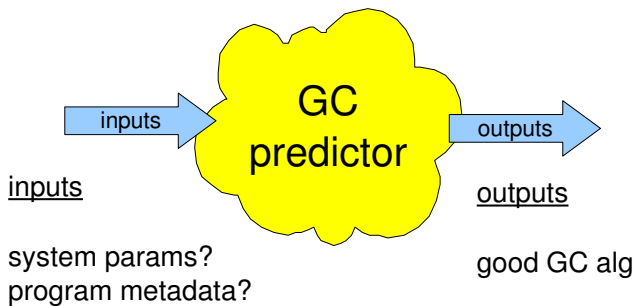
How to find best GC

- for each program there is an optimal GC algorithm
- but how can we find the best combinations?
- exhaustive profiling—run all (program,gc) pairs
- a better way . . .

Artificial Intelligence for Dummies

- choose *features*
- run *training* experiments
- build prediction model
- *test* on previously unseen expt

What is Intelligent GC?



Simple Example (train)

program	# calls of <code>System.gc</code>	best GC alg
X	4	foo
Y	1	bar

Simple Example (test)

- `if (num calls of System.gc > 2)`
`then foo else bar`
- **Now predict for program Z with 3 calls of**
`System.gc`

Static Metrics

- weighted methods per class (WMC)
- depth of inheritance tree (DIT)
- number of children (NOC)
- coupling between object classes (CBO)
- response for a class (RFC)
- lack of cohesion of methods (LCOM)

Object Demographics

- number of allocated objects
- mean size of allocated objects
- number of allocated arrays
- mean size of allocated arrays
- number of allocated bytes
- number of nursery-allocated bytes
- proportion of `java.lang.String` objects
- bins for different ranges of array sizes

GC Profile Stats

- nursery survival rate
- number of major GCs
- number of minor GCs

Timing Stats

- proportion of time spent in mutator
- proportion of time spent in GC

Heap Memory Stats

- current JVM fixed heap size
- ratio of current heap size to min

Benchmark Programs

- for training and testing (LOOCV)
- Three suites
 - ▶ SPECjvm98
 - ▶ JOlden
 - ▶ DaCapo

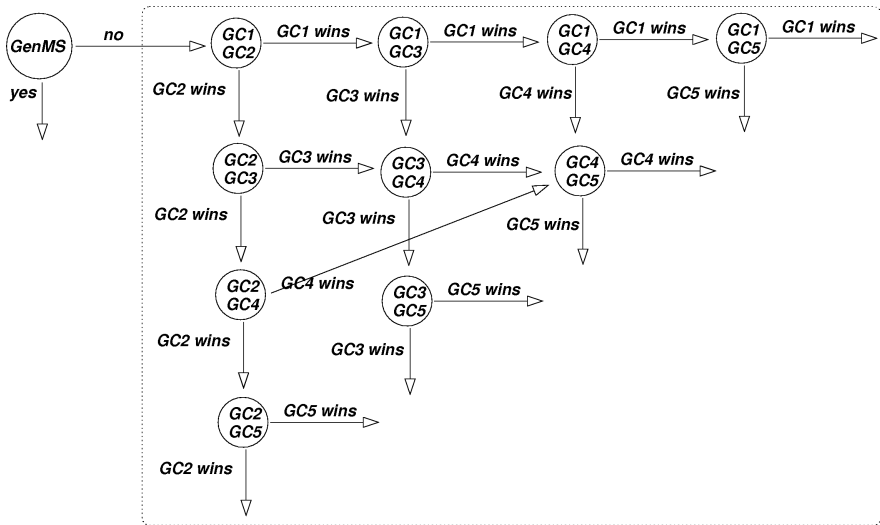
GC Algorithms

- We used Jikes RVM / MMTk
 - ▶ GenMS
 - ▶ GenCopy
 - ▶ SemiSpace
 - ▶ MarkSweep
 - ▶ MarkCompact
 - ▶ CopyMS
 - ▶ others broken?

Prediction Model

- 6-class prediction (difficult)
- reduce to a set of binary predictors (easier)
- yes/no for a GC algorithm
- A/B better for two GC algorithms
- use weka tool to generate C4.5 decision trees for binary predictors
- interface binary predictors with custom scripts

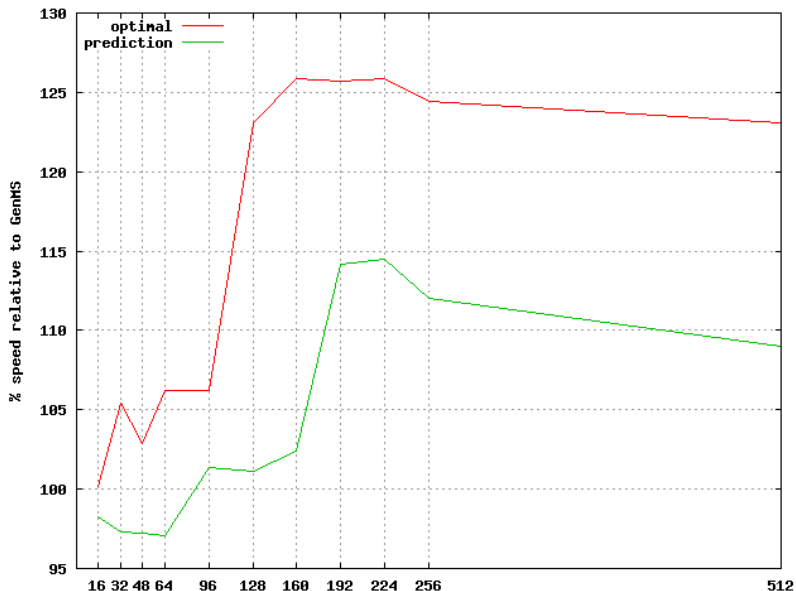
Predictor Structure



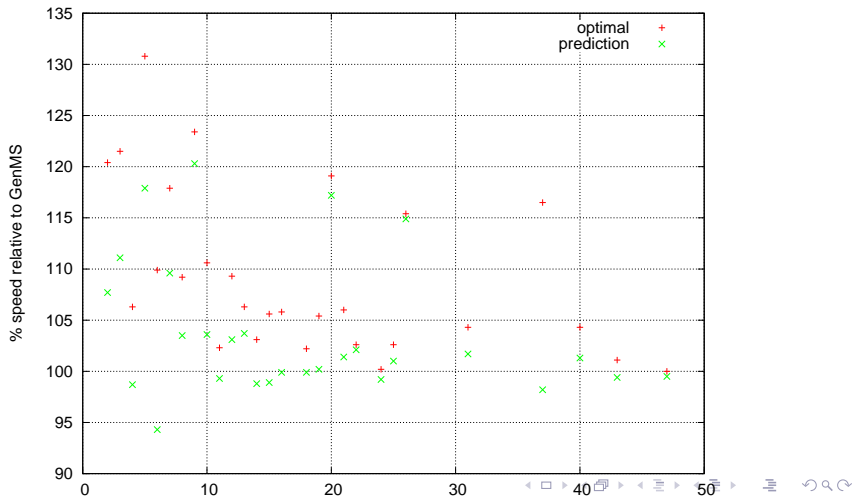
Evaluation

- generate feature vectors for all benchmarks
- run all benchmarks at diff heap sizes to get times
- use this info to
 - ▶ train predictors
 - ▶ calculate optimal speedups
 - ▶ perform LOOCV to evaluate predictor performance

Predictor Performance



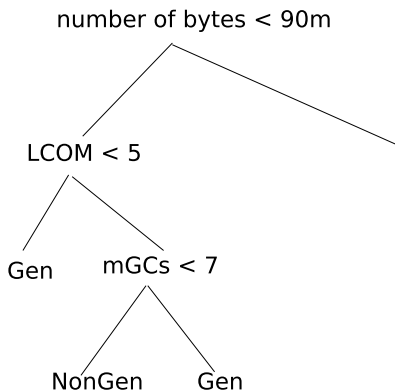
Predictor Performance (relative heap sizes)



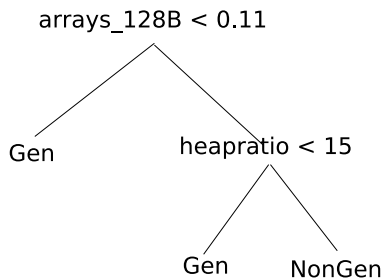
Opening the Black Box

- do the decision tree heuristics make sense to a domain expert?
- can we learn anything from them?

Example Decision Tree



Example Decision Tree (cont)



Conclusions

- Our prediction model gets 5% speedup,
- optimal is 17%
- $O(1)$ profiling rather than $O(N)$ for N GC algs
- better feature selection, prediction models?
- larger training corpus

Future Work

- not just selecting optimal algorithm
- can also tweak params within algorithm
- apply to another JVM (HotSpot)

Examples of Intelligent GC

- when to collect?
 - ▶ Andreasson et al, *JVM '02*
- what to collect?
 - ▶ Marion et al, *ISMM '07*
- *how* to collect?
 - ▶ Singer et al, *ISMM '07*

Closing Thought...

GC implementations generally have a waggonload of knobs and levers which impact performance, but tuning is difficult since the right settings depend on the characteristics of the executed program.'
Dieckmann and Hölzle, *ECOOP* '98