# Automatic Application Performance Improvements Through VM Parameter Modification after Runtime Analysis

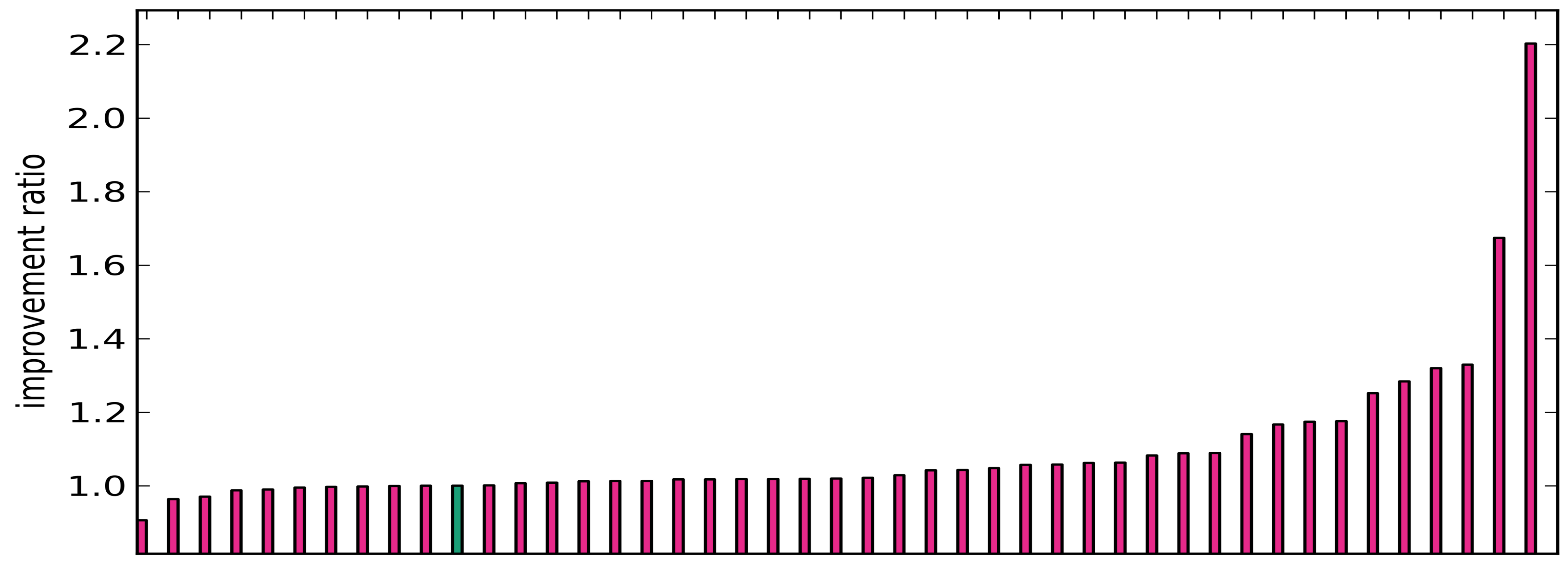
**Nicolas Neu, Charlie Gracie, Prof. Dr. André Hinkenjann,**
**Prof. Dr. Kenneth Kent**
University of New Brunswick, Bonn-Rhein-Sieg University, IBM Canada
Faculty of Computer Science
nicolas.neu@unb.ca, charlie_gracie@ca.ibm.com, andre.hinkenjann@h-brs.de, ken@unb.ca

## Problem Description

The Java Virtual Machine, or JVM, is needed to execute Java bytecode. Current versions serve as a powerful runtime environment not only for Java, but also for other languages compiling to Java bytecode like Scala or Clojure. Features like automatic Garbage Collection make programming easier for the developer but also causes performance hits compared to programming methods with manual memory management. The VM provides powerful mechanisms to adjust itself to a given application by changing the VM parameters. The garbage collection policy or the memory layout can for example be specified like this. **For an unknown application, what are the best parameters that guarantee an optimal performance?**

## Project Goal

The JVM is treated as a blackbox and there is no prior knowledge about the application. An approach was developed that maps patterns detected in the VM log output to rules that define an adjusted VM parameter set: If for example the VM memory is too full most of the time, the available memory is increased for the next run.

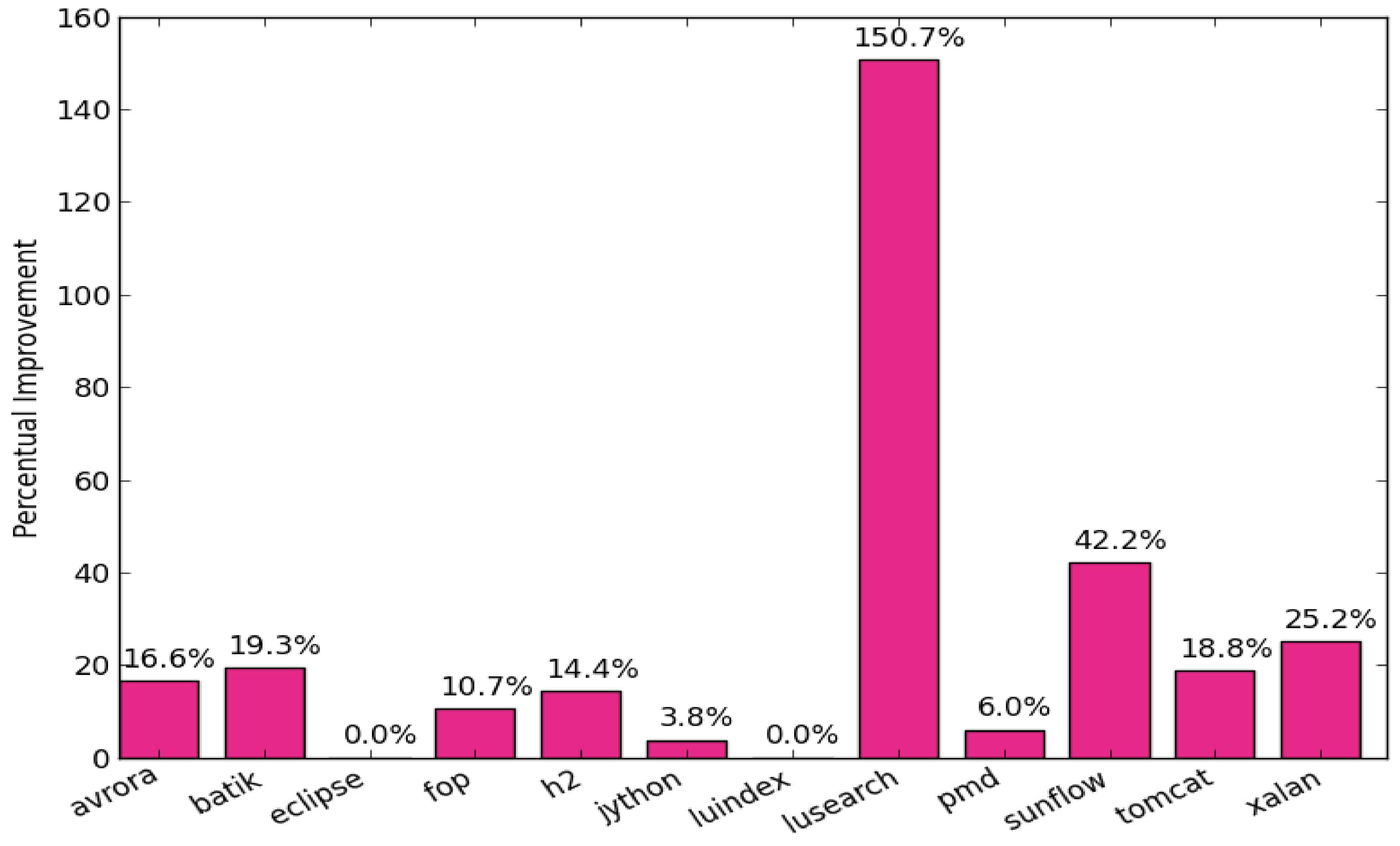


The Graph shows the performance improvement for the heap size adjustment using the balanced GC policy. The green bar is the reference value, red bars represent the ratio of the performance change for DaCapo or SpecJVM benchmarks.

## Automating the Process

The amount of data that has to be reviewed in order to decide what rules to apply is huge and can not be done by hand. An application was developed that analyzes a logfile produced by the VM and outputs possible parameter changes. To make this process as straight forward to use as possible, this principle was taken one step further: Running the entire application, analyzing the logfile and verifying the result is done in one automated process. This process is repeated multiple times, so that several options can be evaluated. Optimized parameters are taken as a basis for further improvements. A Graphical User Interface lowers the entry barrier and makes it possible to attune the VM and the application to each other without requiring advanced knowledge about the VM or the Application.

## Verification



The barchart shows the performance changes for benchmarks taken from DaCapo Benchmark Suite. The benchmarks were run with settings gathered by the automated parameter optimizer. The results are compared to a reference run with the default VM setting.

## Future Work

The toolkit was designed in a way, to make it as easy as possible to extend the existing functionality including the rules. If in the course of the studies on the VM new rules and interdependencies are discovered, they can be incorporated. An extension to other languages run on the VM or even other VMs is possible as well.