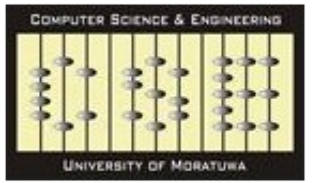
Auto-tuning HotSpot JVM using OpenTuner

OpenTuner Workshop
International Symposium on Code Generation and Optimization 2015

Milinda Fernando (CSE,Univ. of Moratuwa)
Tharindu Rusira (CSE, Univ. of Moratuwa)
Chalitha Perera (CSE, Univ. of Moratuwa)
Chamara Philips (CSE, Univ. of Moratuwa)

Prof. Sanath Jayasena (CSE, Univ. of Moratuwa)
Prof. Saman Amarasinghe (CSAIL, MIT)



Motivation

- Java Virtual machine is a complex piece of Software
- Responsible for providing execution environment for Java programs
- What if the JVM can execute Java applications better (faster?)

JVM and Complexity

- HotSpot JVM
- More than 600 tunable flags and parameters
- How to handle a configuration space of this scale?

OpenTuner[1]

[1] *J. Ansel, S. Kamil, K. Veeramachaneni, U.-M. O'Reilly and S. Amarasinghe, "OpenTuner: An Extensible Framework for Program Autotuning," in MIT CSAIL Technical Report MIT-CSAIL-TR-2013-026, November 1, 2013.*

- [1] provides results for a number of successful case studies
- GCC/G++ auto-tuner inspired a solution for JVM auto-tuning
- Multiple search techniques
- Evolutionary algorithms allow to reach optima aggressively in non-trivial configuration landscapes
- works best with massive search spaces and manages computational complexity really well

Configuration Manipulator

Used to define the configuration space

```
def manipulator(self):
    m = manipulator.ConfigurationManipulator()
    for flag_set in self.bool_flags:
      for flag in flag_set:
        m.add_parameter(manipulator.EnumParameter(flag,['on','off']))
    for flag_set in self.param_flags:
      for flag in flag_set:
        value = flag_set[flag]
        if(value['min']>= value['max']):
                m.add_parameter(manipulator.IntegerParameter(value['flagname'],value['max'],value['min']))
        else:
                m.add_parameter(manipulator.IntegerParameter(value['flagname'],value['min'],value['max'])
       return m
```

Run function

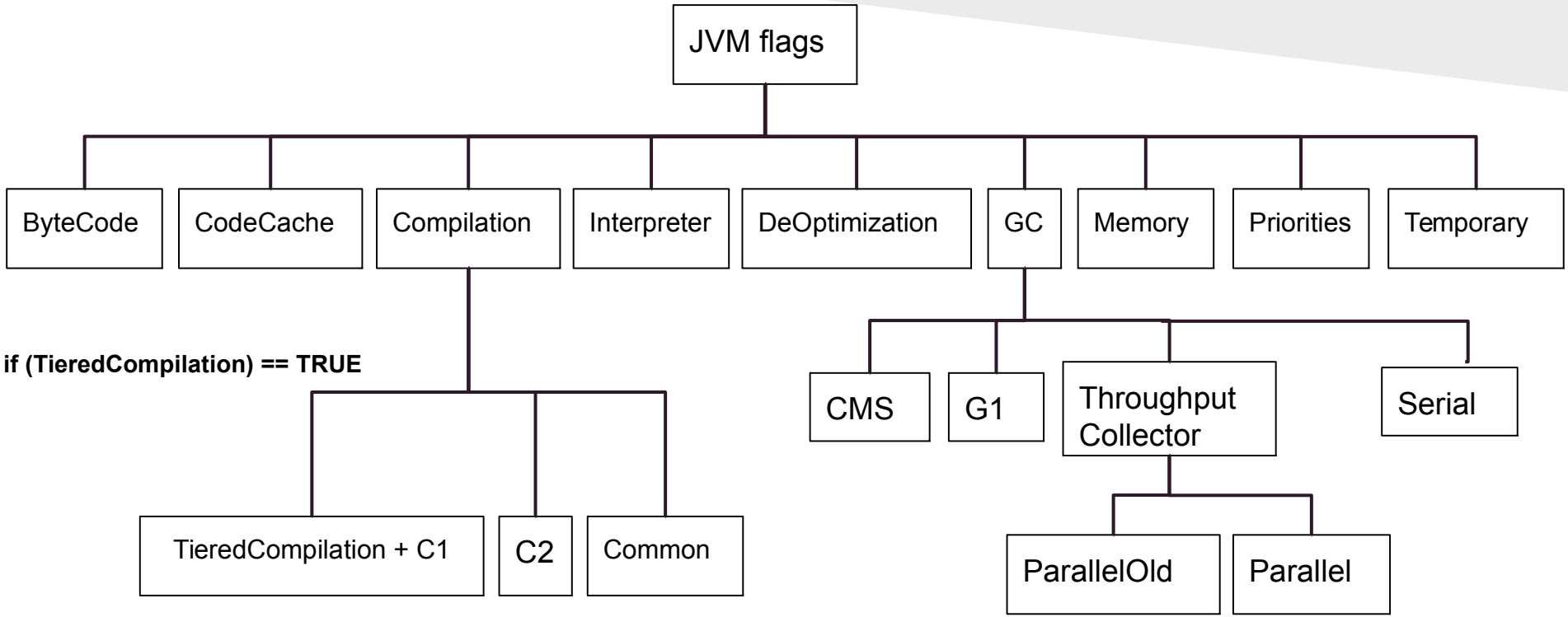
Measures the quality (fitness) of a given configuration.

Eg.

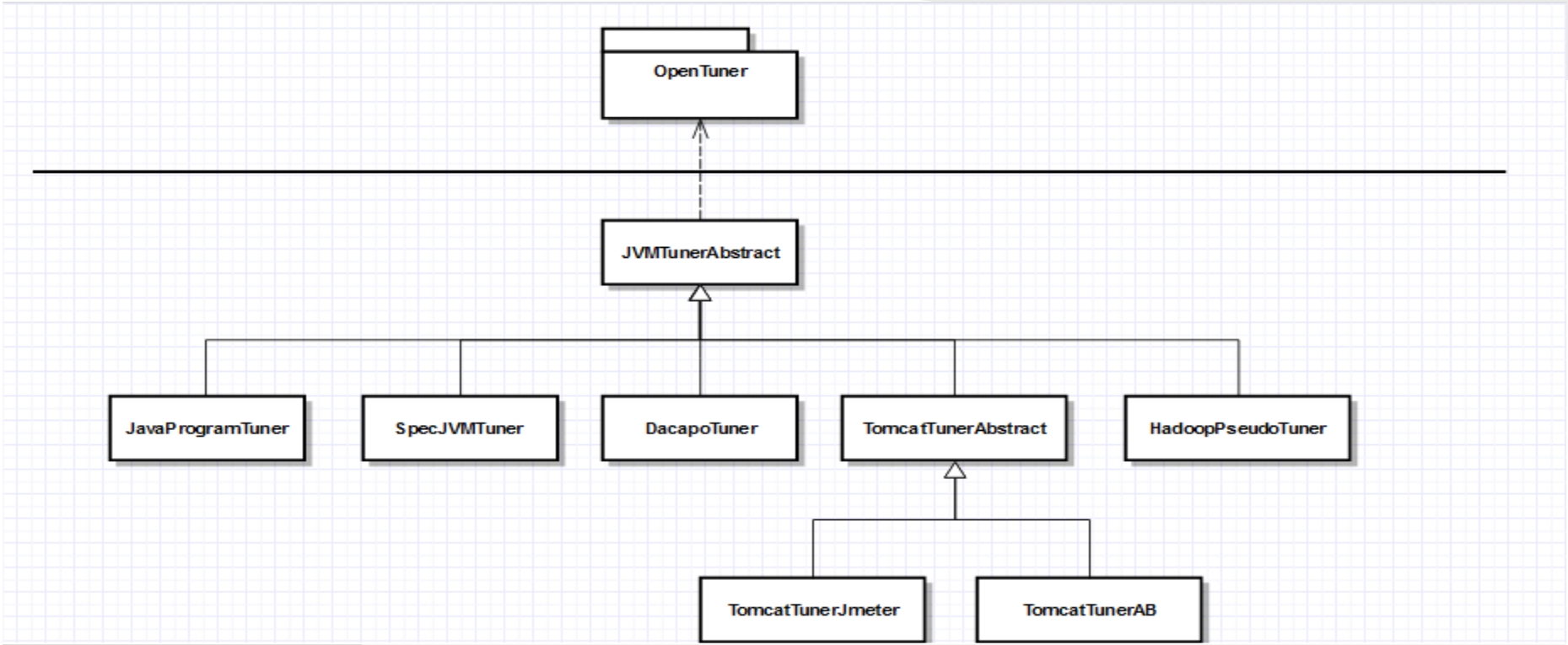
- For SPECjvm2008, operations per minute (ops/m)

- For DaCapo, execution time in ms

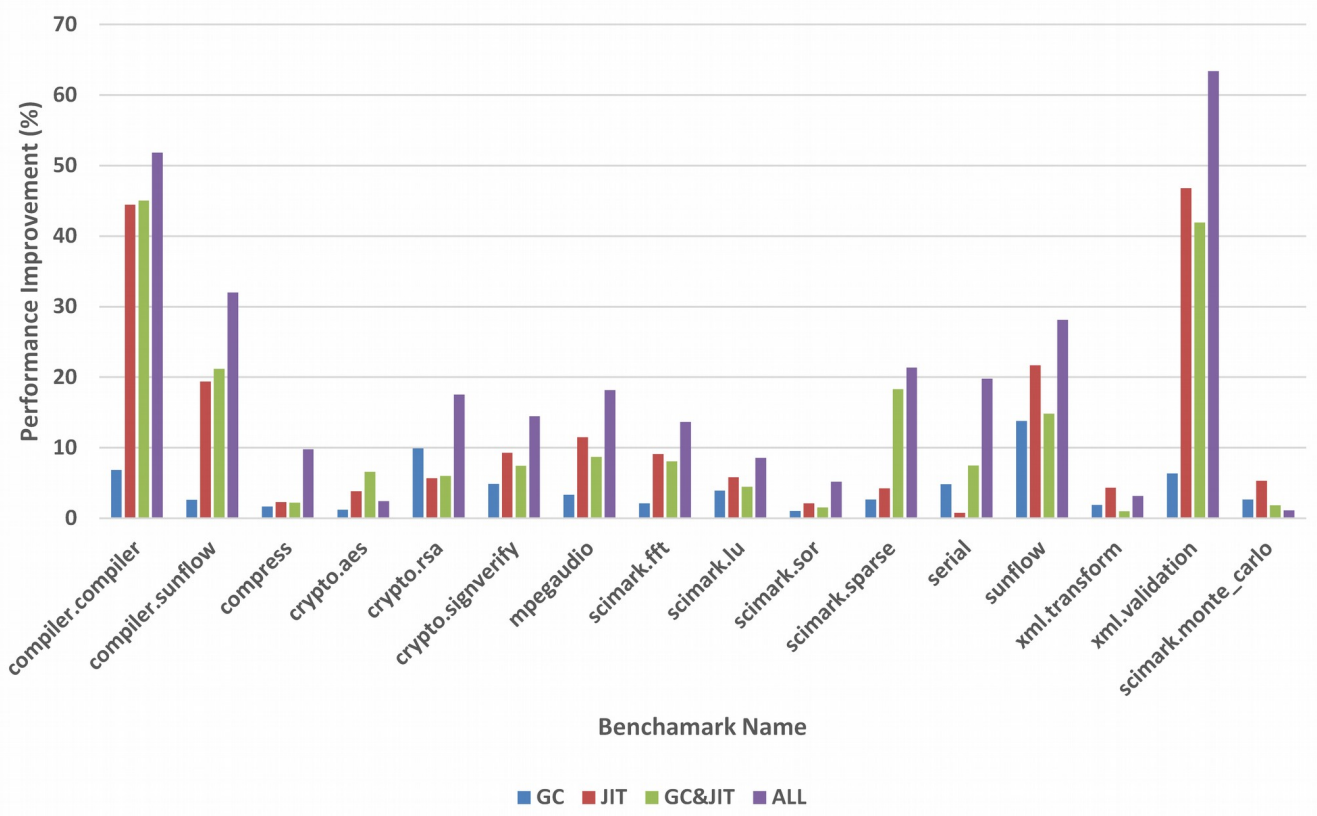
JVM Flag Hierarchy



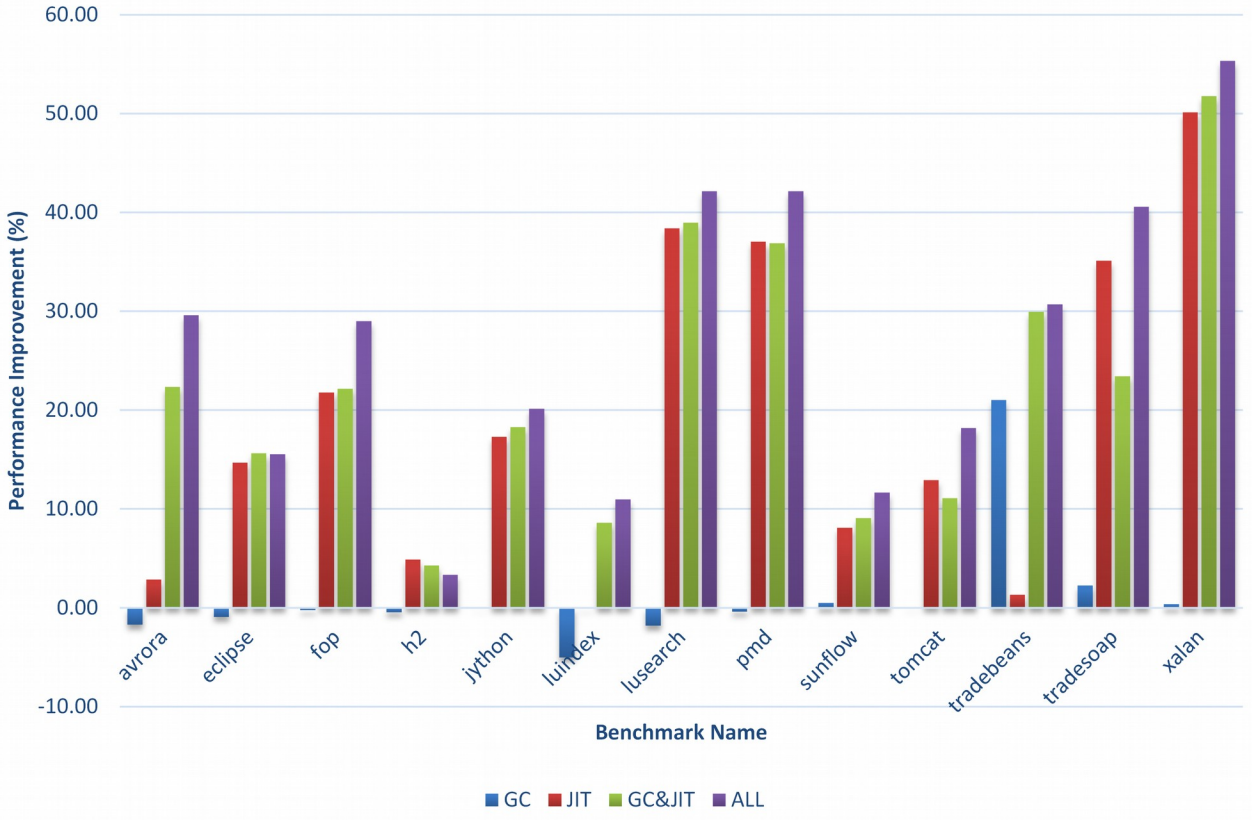
JVM Tuner


OpenTuner
JVMTunerAbstract
JavaProgramTuner
SpecJVMTuner
DacapoTuner
TomcatTunerAbstract
HadoopPseudoTuner
TomcatTunerJmeter
TomcatTunerAB

Performance Improvement of SPECjvm2008 Startup Benchmarks

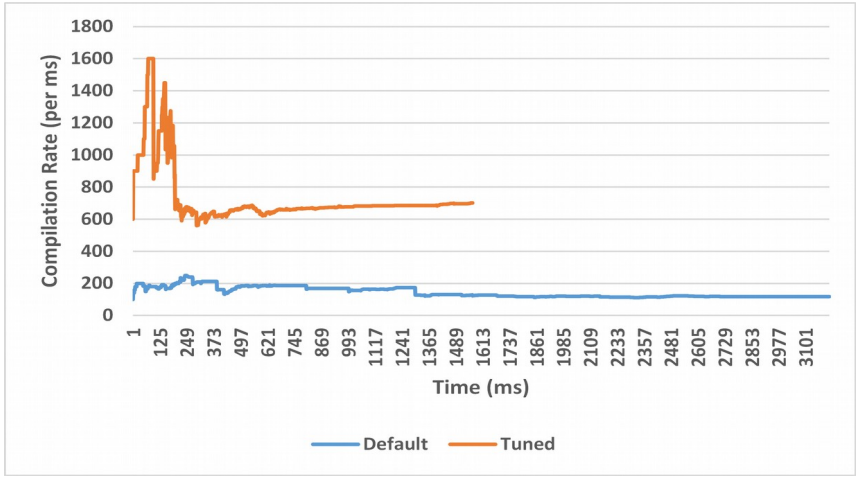


Performance Improvement of Dacapo Benchmarks

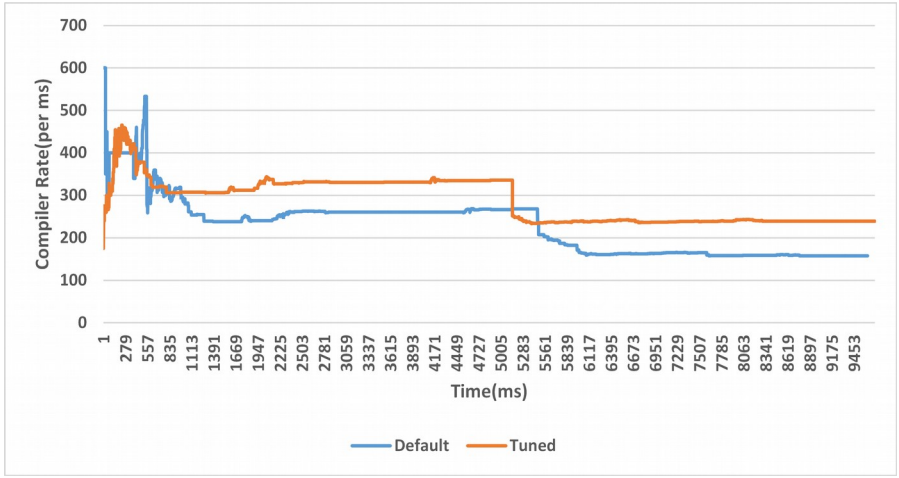


What happens to the JVM when auto-tuned?

- Observations on heap usage, compilation and class loading before and after tuning
- Compilation rate has a major impact on performance



DaCapo pmd benchmark CR (38.73%)



DaCapo h2 benchmark CR (5.76%)

Thank You