Measuring the Cost of Volatile in Java

Stefan Nüesch
Seminar Software Composition 2013

Overview

- Motivation
- Volatile Semantics
- Micro-Benchmark
- Displaying the data
- Results
- Questions

The Problem

- Performance impact of volatile variables often not clear

- -> How do resource contention, read / write ratio, architecture and spatial locality in memory influence the performance of accessing volatile variables in Java?

Scope

- In scope: compare performance for different levels of contention, different read / write ratios and other parameters

- Out of scope: Comparing `volatile` to `synchronized`, locking or other constructs

Methodology

- Varying parameter settings systematically
- Warm-up runs
- Mutliple run cycles
- Take averages for comparison
- Same number of accesses per run

Volatile Semantics (Java 1.5+)

- volatile writes are always written through to memory
- `volatile` established a "happens before" relationship
- reads / writes subsequent to a write cannot be reordered

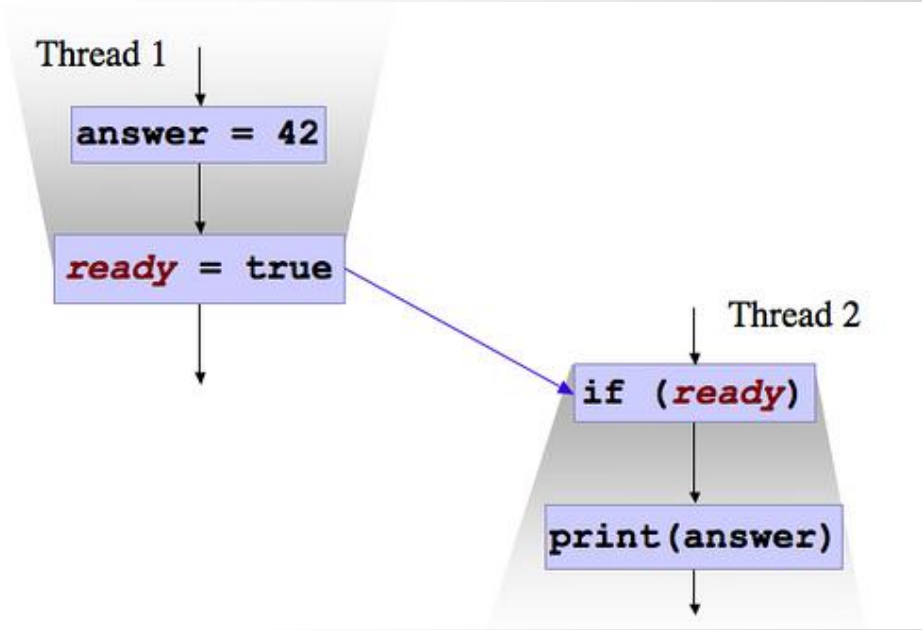
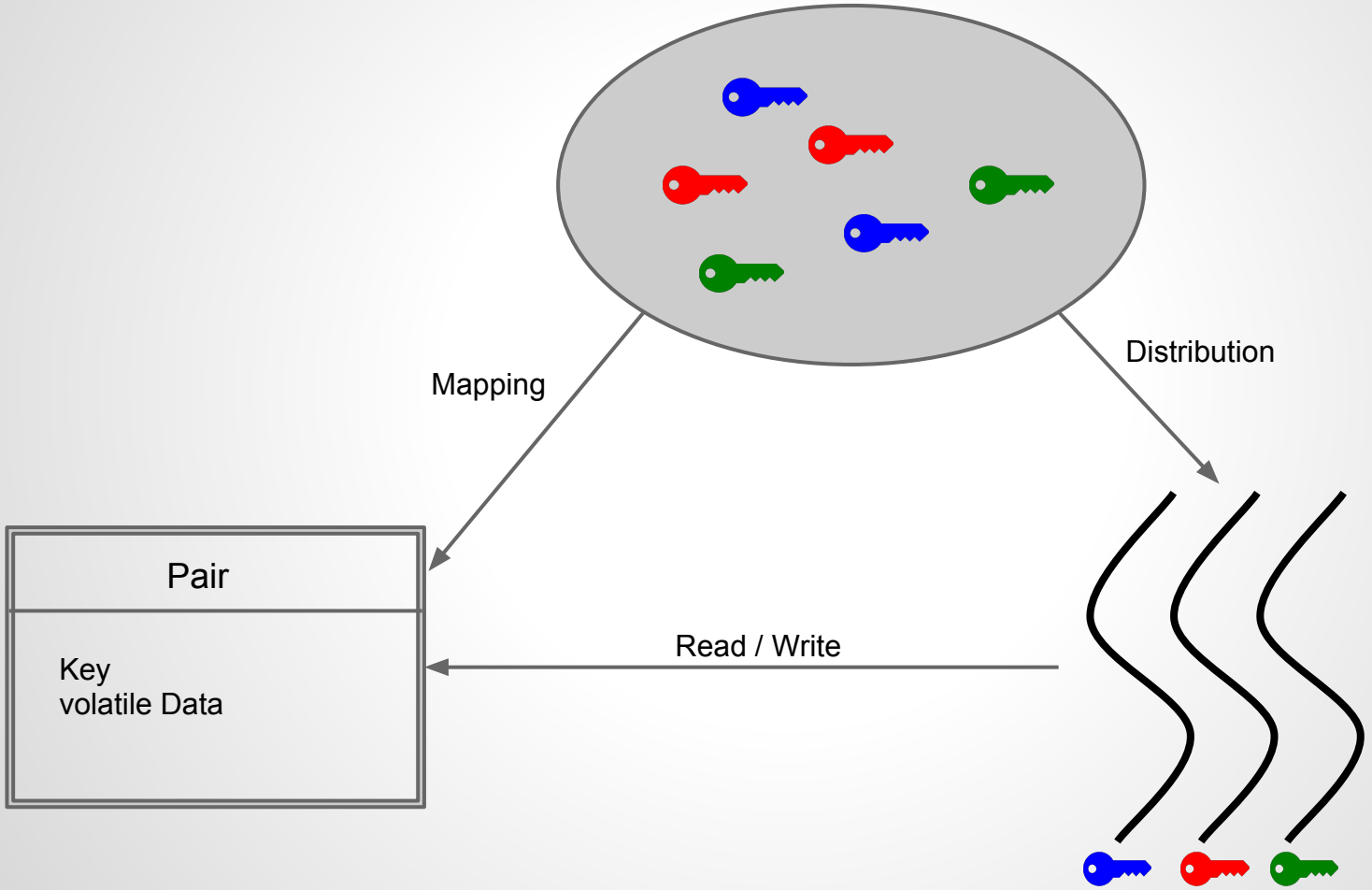


Image source: http://jeremymanson.blogspot.ch/2008/11/what-volatile-means-in-java.html

Micro-Benchmark



Micro-Benchmark (cont'd)

- Parameters:
  - Number of Keys
  - Key length
  - Number of Threads
  - Number of read / write cycles
  - Read / write ratio
  - Keyset overlapping
  - Number of run cycles

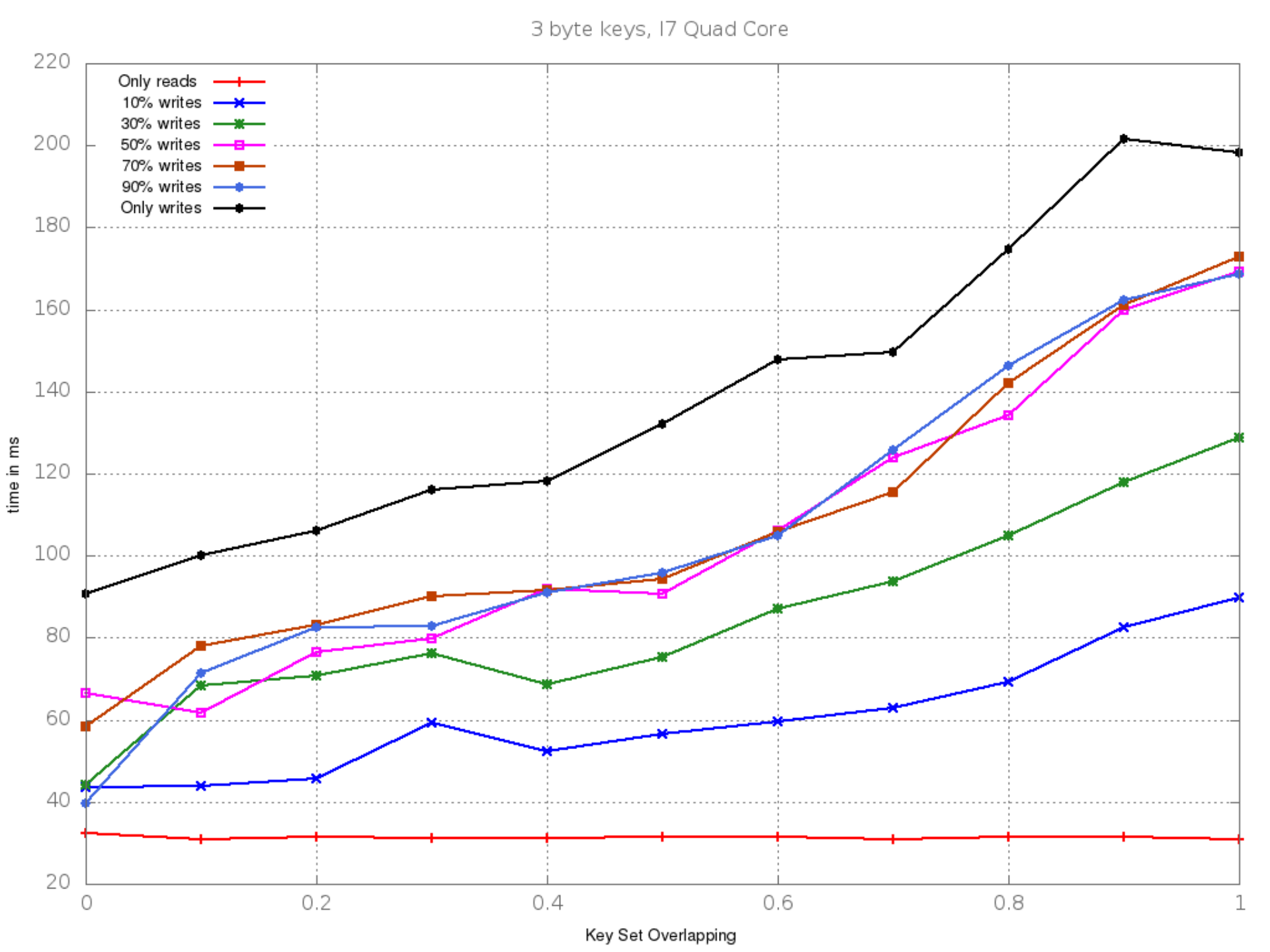
Plotting Results

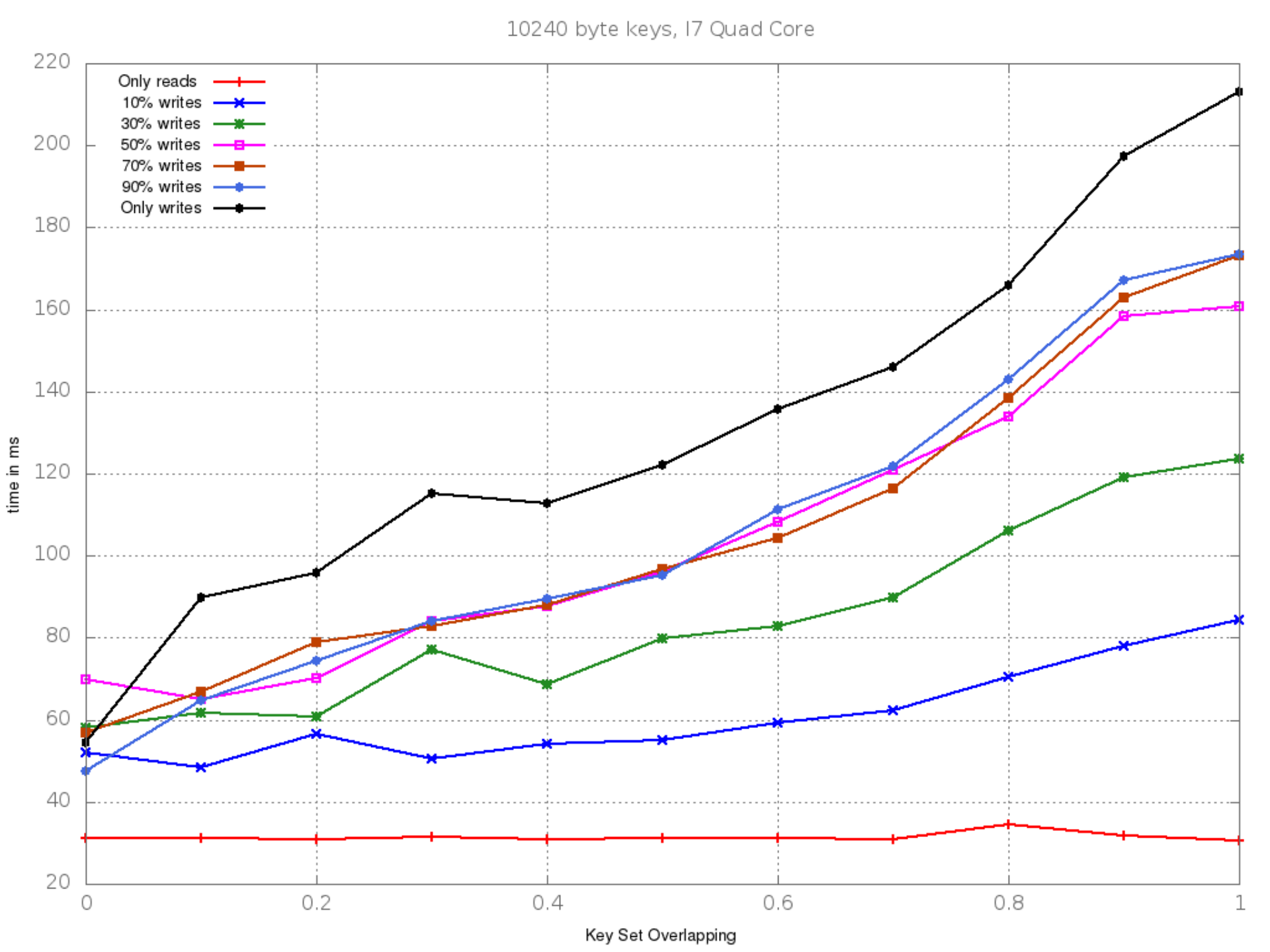
- Python script for collecting the data and preparing the plot
- gnuplot (with python-gnuplot) used for drawing the plot
- single plots created automatically

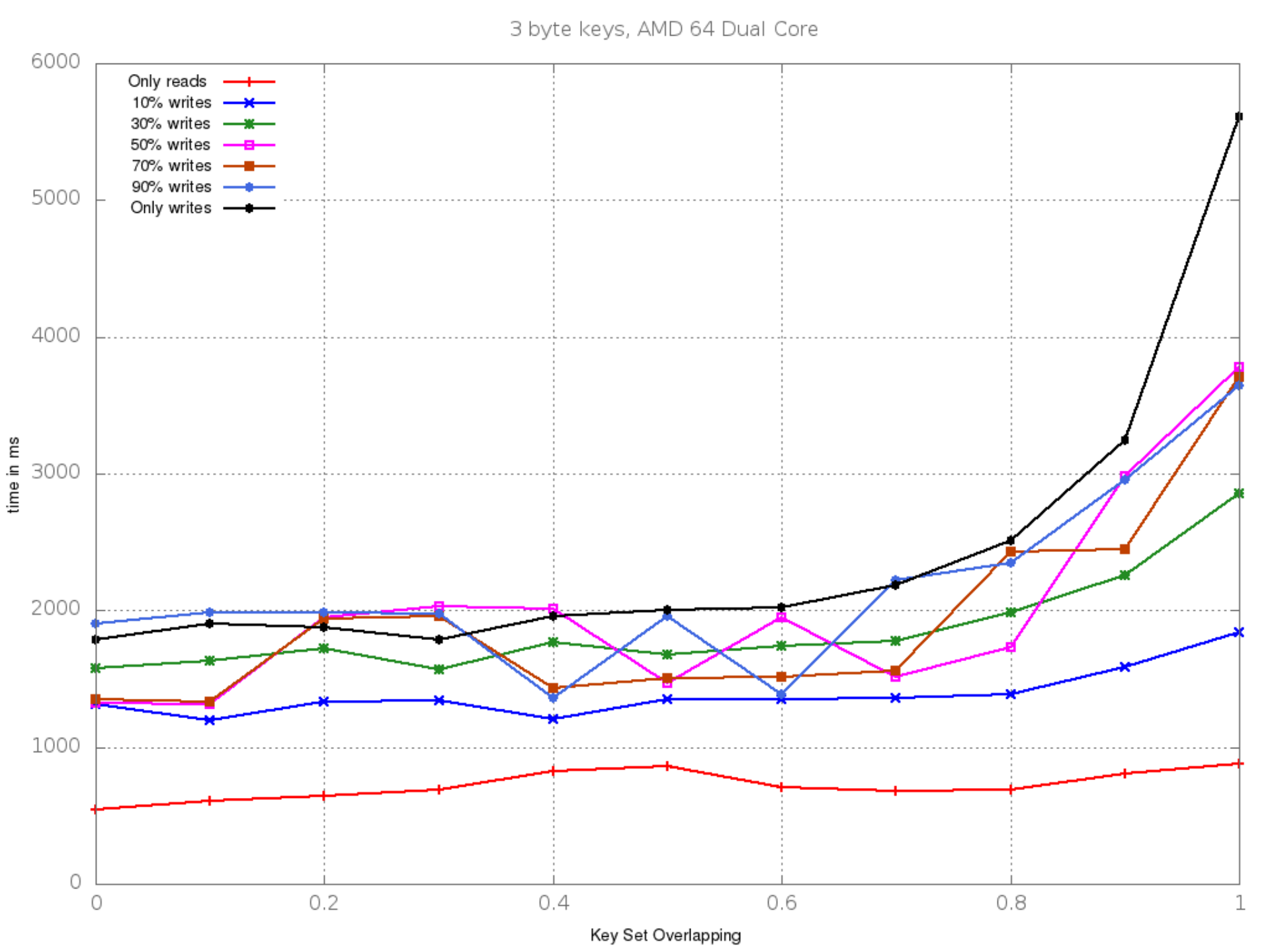
Plotting Results - Python Code

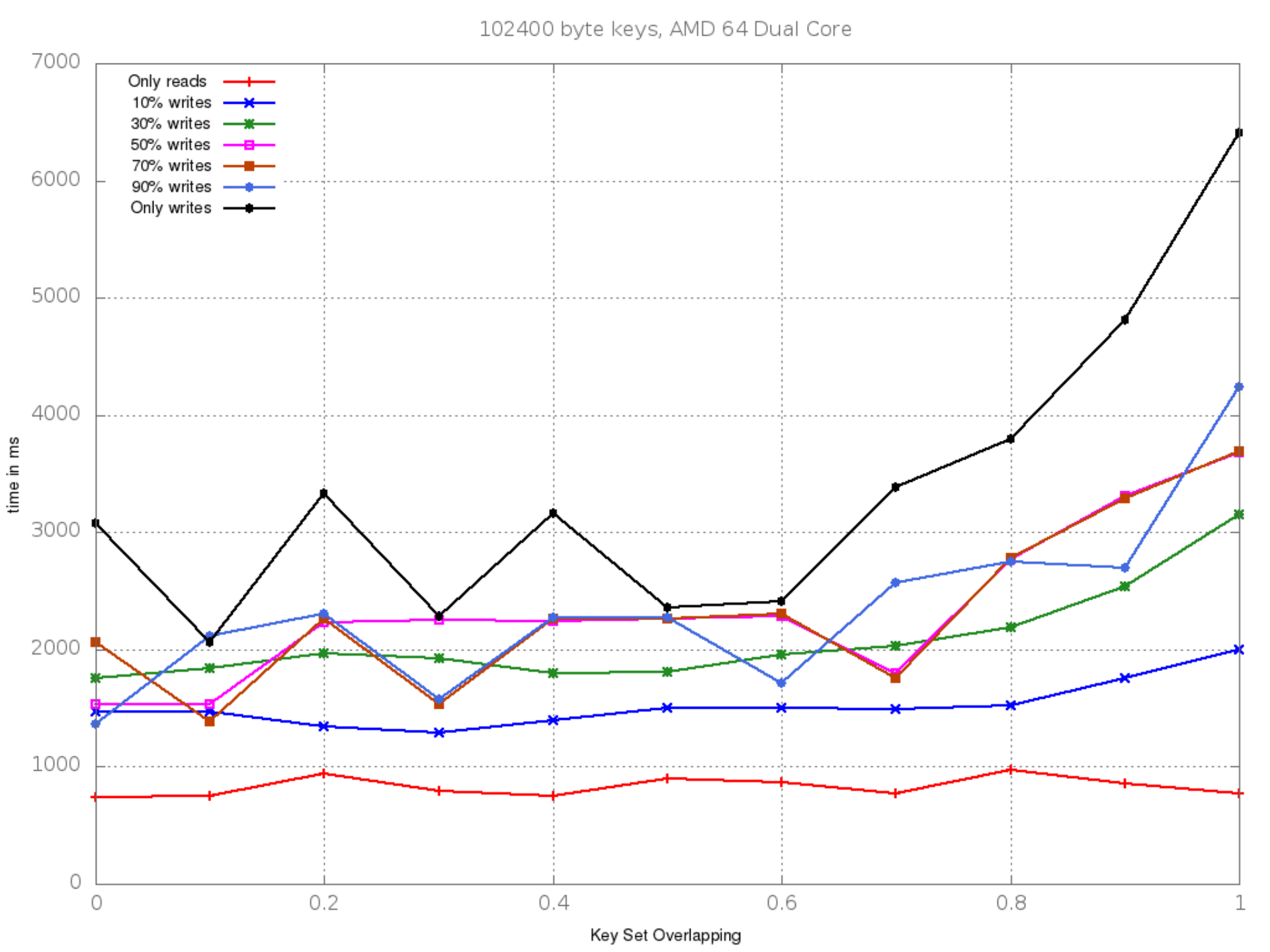
```
plot = Gnuplot.Gnuplot()
plot.title(set)
plot('set term pdf font "Helvetica, 10"')
plot('set style line 1 lt 1 lc rgb "#FF0000" lw 3 # red')
plot('set output "'+resultDir+set.replace(' ','_')+'.pdf";')
plot('set ylabel "'+yAxisLabel+'";')
plot('set xlabel "'+xAxisLabel[varPosition]+'";')
plot('set key top left;')
plot('set yrange [:]')
plot('set xrange [:]')
plot('plot "'+resultDir+dataFile+'" using ($1):($2*0.000001):($3*0.000001):($4*0.000001)
with yerrorlines title "'+xAxisLabel[varPosition]+'";')
```

Results


3 byte keys, I7 Quad Core
Only reads
10% writes
30% writes
50% writes
70% writes
90% writes
Only writes
time in ms
Key Set Overlapping
220
200
180
160
140
120
100
80
60
40
20
0
0.2
0.4
0.6
0.8
1


10240 byte keys, I7 Quad Core
Only reads
10% writes
30% writes
50% writes
70% writes
90% writes
Only writes
time in ms
Key Set Overlapping
220
200
180
160
140
120
100
80
60
40
20
0
0.2
0.4
0.6
0.8
1


3 byte keys, AMD 64 Dual Core
Only reads
10% writes
30% writes
50% writes
70% writes
90% writes
Only writes
time in ms
Key Set Overlapping
0
1000
2000
3000
4000
5000
6000
0
0.2
0.4
0.6
0.8
1


102400 byte keys, AMD 64 Dual Core
Only reads
10% writes
30% writes
50% writes
70% writes
90% writes
Only writes
time in ms
Key Set Overlapping
0
1000
2000
3000
4000
5000
6000
7000
0
0.2
0.4
0.6
0.8
1

Results

- No impact of read contention
- Considerable performance impact with little writing -> factor 3 on AMD 4
- Performance of writing is strongly affected by contention -> factor 3

Future Work

- Assess impact of memory layout
- Compare several architectures
- In depth analysis of certain behaviour