Java Performance and Replay Compilation

Java app performance affected by:

1. Input
2. Hardware and OS (host platform)
3. Language, compiler, libraries, runtime
   - includes GC algorithm
4. Heap size
5. Nondeterminism
   - timer-based sampling
   - derived JIT compilation
6. Warmup
   - with more runs, app is more optimized
To manage nondeterminism of JIT optimizations,

a current idea is to use REPLAY COMPILATION

① execute app \( q \) times
collect a profile run
list of optimization level for each fune.

② restart JVM.

③ repeat step ① \( n \) times.

now you have \( n \) profile runs.
4. From the n profile runs, build 1 COMPILATION PLAN, which is a fixed list of optimization levels for each function.

    compilation plan can be the optimal profile run or some merged version of all n profile runs.

5. run experiments by replaying fixed compilation plan
Java Performance Evaluation
through Rigorous Replay Compilation

Andy Georges et al,
Ghent Univ., OOPSLA 2008

Jikes Research VM
- Supports replay compilation
- Compilation plan called an "advice file"

Using Jikes RVM, they show

Performance Results for one compilation plan may not be representative of other compilation plans
using specjvm98 and DaCapo,

They found
- little overlap between
  10 compilation plans
  each computed from
  a 10-iteration profile run

- different compilation plans may not agree
  on which GC runs fastest

- and of course, even running
  the same compilation plan multiple times will give you
  different execution times

BOTTOM LINE:
You need multiple compilation plans.
Statistical Analysis

with innovation

compilation plan 1

without innovation

comp. plan 1

with innov.

comp. plan n

w/o innov.

comp. plan n

do a within-subjects t-test