Compiler Optimisation

13 – Adaptive and Profile Directed Compilation

Hugh Leather
IF 1.18a
hleather@inf.ed.ac.uk

Institute for Computing Systems Architecture
School of Informatics
University of Edinburgh

2019

Introduction

- Why we fail to optimise
- Profile directed compilation
- Iterative compilation
- A bit of stats

Why we fail

- Optimisation space is big. [1]
  - Compiler options 10^{400+} per file alone!
  - Consider choices made per function, block, instruction
  - Some choices make more choices - e.g. inlining, unrolling

[1] You just won't believe how vastly, hugely, mind-bogglingly big it is. I mean, you may think it's a long way down the road to the chemist's, but that's just peanuts to space.

Why we fail

- Modern architectures very complicated
  - Huge number of components
  - Non deterministic cache and O-O
  - Different one every few weeks

Why we fail

- Runtime data not known
    - Can't tell what code paths executed
    - Can't tell cache miss frequencies
    - Can't tell lots of stuff

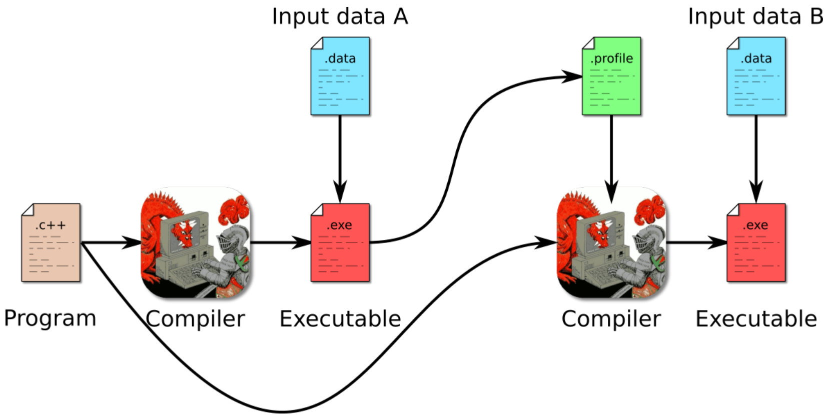
Profile Directed Compilation

Profile Guided Optimisation

- Run program with *representative* inputs
- Collect interesting info
- Recompile using interesting info

- Costly
- What if not representative inputs?

Profile Directed Compilation

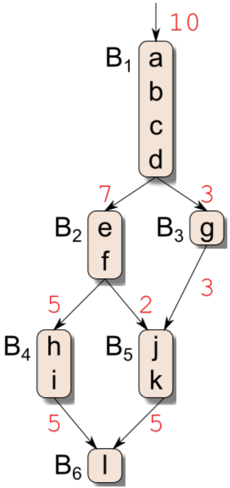


Profile Directed Compilation

Typically record CFG edge frequencies

- Already seen in insn scheduling
- Also for spill costs in reg alloc
- Also BB layout
- Also inlining costs
- Many others potentially

- But, most compilers do very little

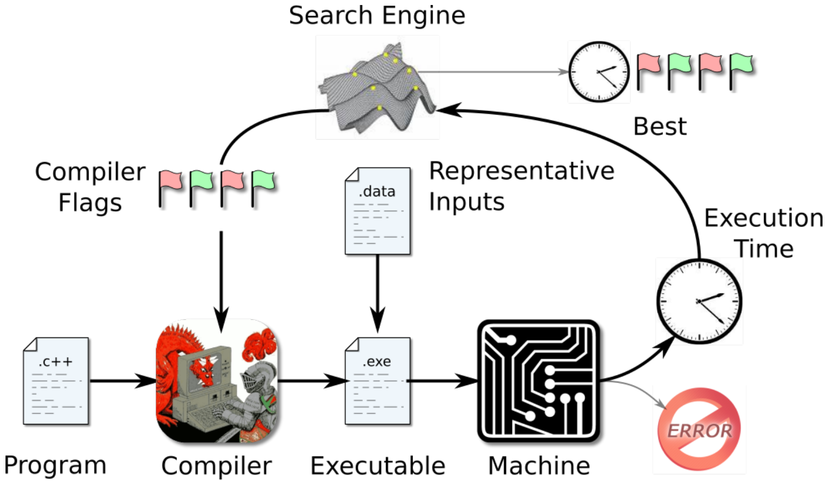


Profile Directed Compilation

Beyond edge frequencies

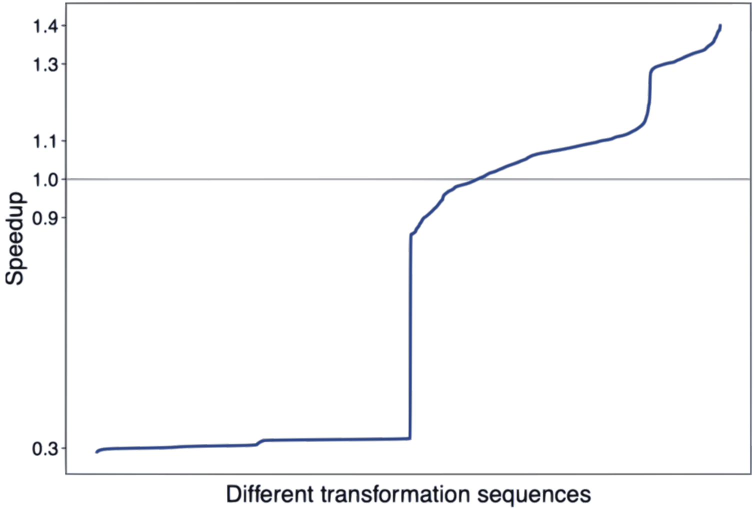
- Typically gains small
- Challenge of undecidability and processor behaviour not addressed
- What happens if data changes on the second run?
- Really focuses on persistent control-flow behaviour
- All other information e.g. runtime values, memory locations accessed ignored
- Can we get more out of knowing data and its impact on program behaviour?

Iterative Compilation
Adaptive Compilation

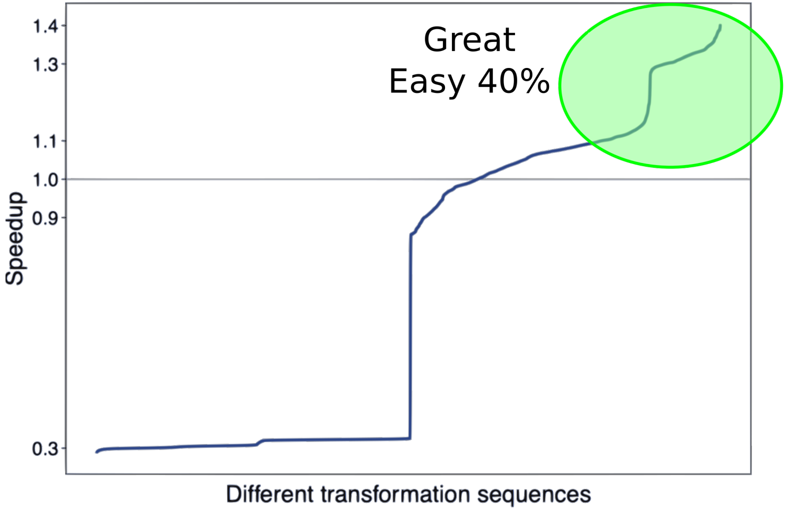


- Avoids thinking about right optimisation
- Search space can potentially include every choice
- Architecture, memory behaviour, etc all handled
- Performance gains substantial

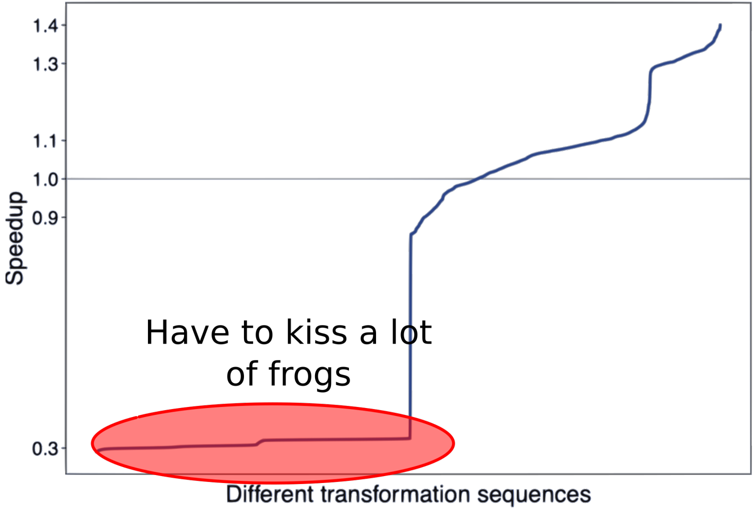
Iterative Compilation



Iterative Compilation



Iterative Compilation



Iterative Compilation

- Can be very costly - thousands of compile/run cycles
- Search techniques can have significant impact on cost
  - Typically Random or Genetic Algorithm
  - But remember No Free Lunch Theorem[2]
- Only iterate over hot code and use minimal inputs
- Check compiler strategies actually change code

[2] To paraphrase: No one search technique is better than any other over all problems

A Bit of Statistics

How to deal with noise

- Most program measurements are noisy (e.g. energy/performance)
  - Other programs
  - OS interaction
  - Small changes in initial state
  - Temperature
  - etc
- Comparisons between measurements not straightforward

A Bit of Statistics

Random Variable

Variable whose value is subject to chance - e.g. runtime

Probability Distribution

Assigns a probability to each value that a random variable may take

Observation

A particular 'read' of a random variable

Sample

A collection of observations.

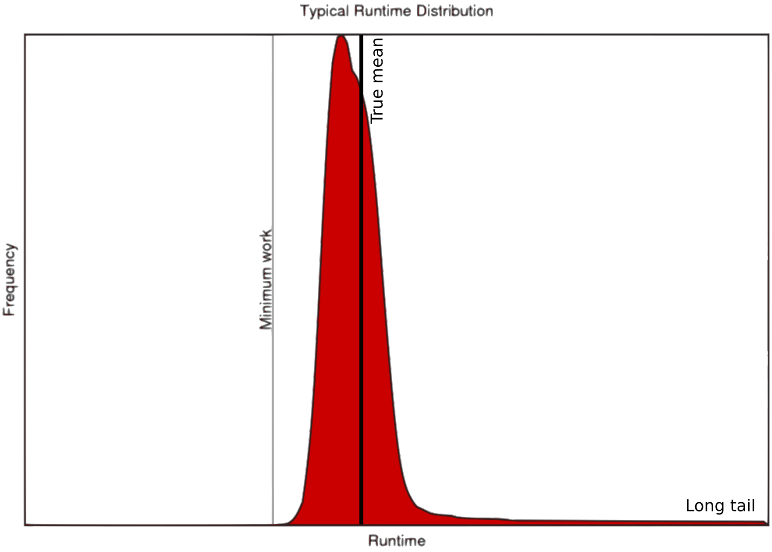
True vs Sample Mean

True mean is mean of the underlying distribution

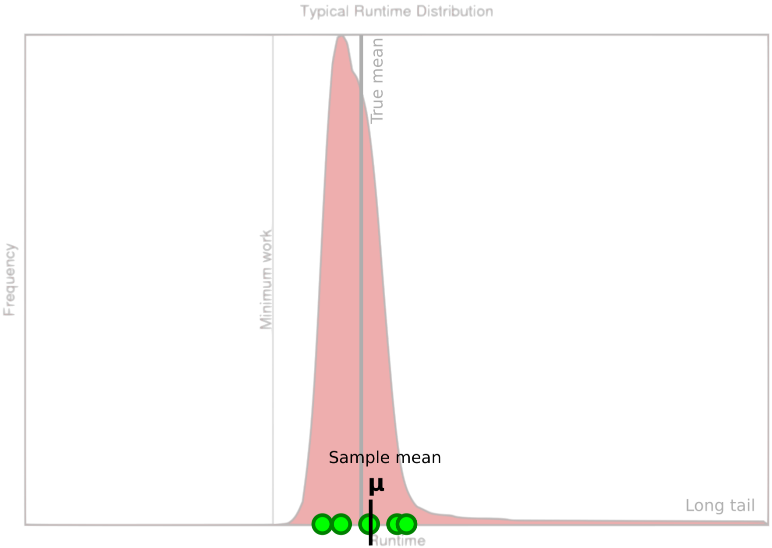
Sample mean is mean of a particular sample

As $|\text{Sample}| \to \infty$, sample mean $\to$ true mean

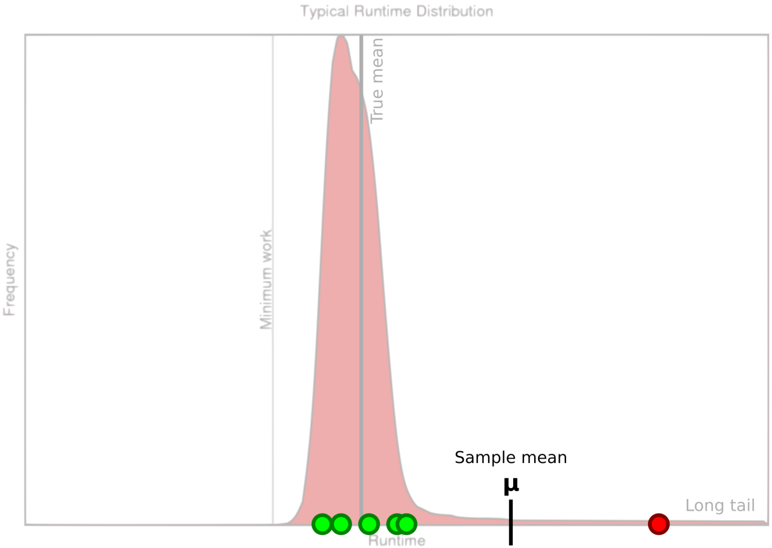
A Bit of Statistics



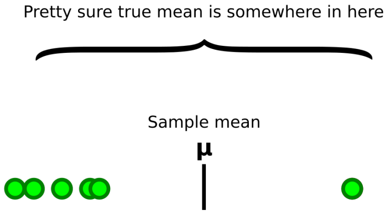
A Bit of Statistics



A Bit of Statistics



A Bit of Statistics



A Bit of Statistics

Confidence Interval

An interval estimate of a population parameter
CI usually has a confidence level, e.g. 95%
Converse is significance, i.e. 1 - level

- Typically confidence intervals applied to mean
- Interval does not say "True mean is 95% likely in here"
  Interpret as "How much do I like this estimate?"
- The **more confident** want to be about an estimate, the **wider** the interval
- **Large sample size** generally gives **smaller** intervals [3]

[3] NB: Same is not true for standard deviation

A Bit of Statistics

- How do we know if sample is big enough?
- If not comparing distributions then use mean / CI[4]
- If comparing two+ distributions then use statistical tests, e.g. Student's t-test, Anova[5]

[4] Strictly speaking, some care must be taken here as this type of sequential sampling plan is not rigorously correct

[5] Also take care about this. May need Bonferroni adjustment or otherwise

Summary

- Why we fail to optimise
- Profile directed compilation
- Iterative compilation
- A bit of stats

PPar CDT Advert