CS 326
Performance Tuning
Stephen Freund

The Goal
• Ensure system has acceptable
  – throughput
  – latency
  – memory footprint
  – network usage
  – ...
• For anticipated workload

When To Do Performance Tuning
• During design? Nope
  – But understand algorithm running times
  – Identify design tradeoffs that may impact performance (sorted vs. unsorted list, etc.)
  – Document choices you make, alternatives, assumptions about the workload
  – Always favor simple solutions until proven they are insufficient.
• It is much easier to optimize a well-designed system than a poorly-designed system.

When To Do Performance Tuning
• During initial implementation? Nope
  – But following good programming discipline
  – Abstractions so you can change reps later
  – Basic programming practices
    • don’t recompute same value unnecessarily
    • don’t do more work than necessary
    • ...
• It is much easier to optimize a well-written system than a poorly-written system.
When To Do Performance Tuning

• When writing correctness tests? Nope
  — Focus on correctness tests first

• It is much easier to optimize a correct system than an incorrect system.

Performance Tuning

• Premature optimization is the root of all evil.
  — Donald Knuth

• Only when you recognize that the system fails to meet desired performance goals

• Methodology is similar to debugging once a failure has occurred.

Experiment-Driven Methodology

• Measure the performance of the system before modification.
• Identify the bottleneck
  — part of the system that is critical for improving the performance.
• Modify the system to remove the bottleneck.
• Measure the performance of the system after modification.
• If the modification:
  — improves performance better, adopt it
  — does not improve performance, revert to origina

Reproducible Test Inputs

• Real Data (eg: Marvel Comics)
• Synthetic Data (eg: my "synthetic" graph)
  — Write a program to generate "fake data"
  — Mimics all relevant characteristics of real world
  — Better understanding of structure of data
  — Easier to create different size inputs, inputs w/ different features, etc.

• Typically a combination, but being able to quickly write data generators is a handy skill.
Identifying Bottlenecks

• "90-10 Rule"
  — 90% of the time is spent in 10% of the code.
• How to find that 10%
  — Log messages
  — Timers/counters in code
  — Profiler
    • dynamic analysis to measure properties of program behavior at run time
    • where does the program spend time?
    • how is memory being used?
    • ...

Algorithmic Bottlenecks

• Implementation contains algorithms with inadequate Big-O run times.
• Example:
  — originally kept an array unsorted with O(n) search
  — change to a sorted array with O(log n) search
    • but at the cost of slower insertions...
• Requires you to know algorithm design and analysis basics.
• May need to change ADT internal reps, etc.

Implementation Bugs

• Defect leads to unnecessary computation.
• Example:
  — func inTree(root: Tree, v: Int) -> Bool {
    if (v == root.value) return true
    let inLeft = inTree(root.left, v)
    let inRight = inTree(root.right, v)
    return inLeft || inRight
  }

• Standard debugging techniques often come in to play.

Bad Assumptions or Limitations of Libraries

• Code you use may have different performance characteristics than you assumed.
  — Particular library call creates a new http connection every time it is called rather than reusing an existing connection
  — Creating UIBezierPaths can become a bottleneck if you try to render a few thousand nodes in your GraphView
• Primitive ADTs may have limitations
  — Copy-on-Write Sets/Dicts/Arrays – how much are you paying for the potential copying???
Expensive Run-time Checks

- precondition/invariant assertions
- checkRep() calls

- Never delete checks
  - Add flag to selectively turn them off.
  - You will likely need to do this today...

Running Profiler in XCode

- Ctrl-Click Play Button next to Unit Test
- Run Timer
- Track (CPU Usage)
- Heaviest Stack Trace
- Detail Pan
- Top of Trace/Detail

Recording Controls

- Time Profiler performs low-overhead time-based sampling of processes running on the system's CPUs.
Modify checkRep(...)
Change From Set To MutableSet

Running App With Profiler