# Performance Monitoring with Repeatability Tips & Techniques

#DevLin2016

# About Gareth

- Worked as a performance analyst at IBM since 2012 until end of September.
- Originally looking at Java (IBM SDK for Java), later also Node.js
- Member of Node.js Community Benchmarking Workgroup
- Moved to Linköping three weeks ago!

Contact me:

twitter.com/gse1986
github.com/gareth-ellis

# Agenda

- Introduction to performance testing
- Biggest challenges of performance testing
- Ways to overcome challenges
- Tools to use

# Introduction to performance testing

- Quite different to functional & system testing
- Change one thing at a time
  - Application code / benchmark
  - Machine / machine configuration
  - Runtime
- Can’t necessarily rely on previous measures
  - Most investigations involve a run of a “good” build and a build under test

# Designing a performance test

- What do you want to test?
- Ideal state?
  - Has the system warmed up?
  - Are you going to be measuring something useful?
- What else is needed to ensure a fair test?
  - Saturating server for a throughput test
  - Ensuring that system has adequate resource
- What logs/data should we store?
- How do you tidy up afterwards, so future tests are unaffected?

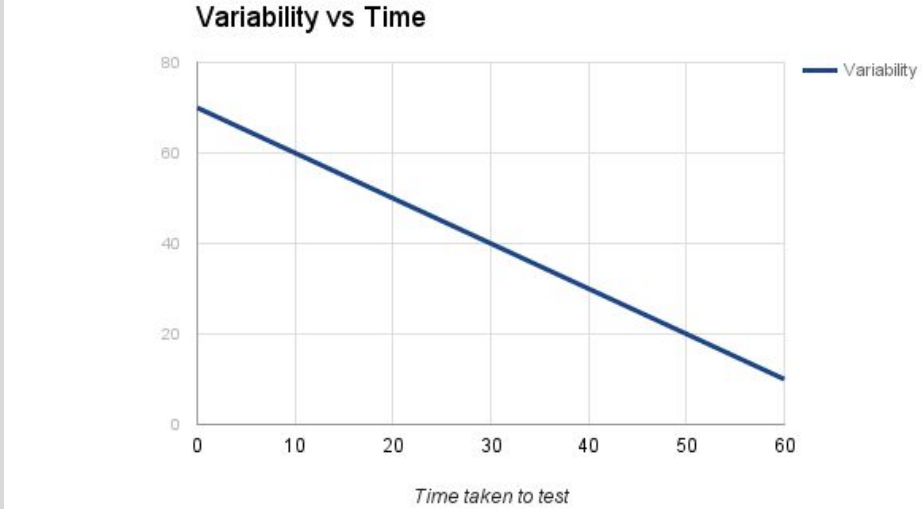
# Methods of benchmarking

- Micro-Benchmarks
  - Measure a specific function/API
  - Ex: Buffer.new()
  - Compare key characteristics
  - Micro-benchmark improvements may not mean real world improvements
  - Risk of not measuring exactly what you expect – especially where a JIT is involved
- Whole System
  - Benchmark expected customer use case
  - Ex: AcmeAir - http://github.com/acmeair/acmeair-nodejs
  - The more you test, the more chance for variance

# Biggest Challenges

- Run-to-run variability
  - False positives
  - Collecting enough samples to be confident in the result
  - Documenting expected variability
  - Repeatability
- Consistent environment
  - Known initial machine state
  - Machine isolation
  - Interleaving runs
- Jumping to conclusions

# Reliable data is typically expensive....



# Approaches to overcoming issues

- Setting up the machine for performance testing
  - Turn off power saving
  - Ensure RAM exceeds requirement - paging is bad
  - Ensure the only “bottleneck” is what we’re testing.
- Ensure test approach is repeated to the letter
  - Machine state
  - Software stack state
  - VM state - shared classes

# You’ve found a regression, now what?

- Double check data, is it genuine?
- If so, what changed?
  - Machine?
  - Application code?
  - Software stack?
- Compare between good and bad case
  - Binary search
  - Tooling

# Tooling

- Perf - Unix
  - https://perf.wiki.kernel.org
- WPT - Windows (as part of Windows SDK)

# Thank you - Questions?

http://github.com/gareth-ellis/slides