LVC20-311 Benchmarking Android Runtime: Past, Present and Future

Evgeny Astigeevich
Arm
September 2020

Linaro Connect VIRTUAL 2020

Agenda

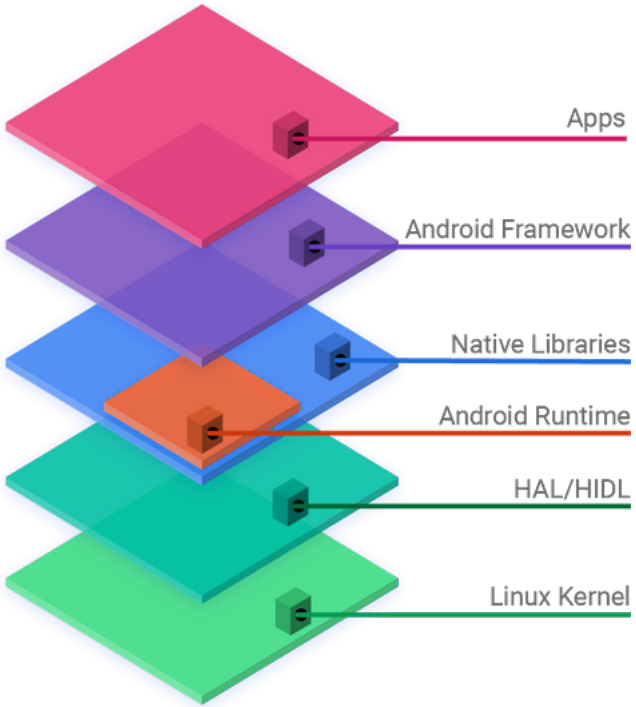
- Android Runtime (ART) and LCG ART Team
- Benchmarking Overview
- Benchmarking: Past
- Benchmarking: Present
- Benchmarking: Future
- Conclusion

Android Runtime (ART)

The managed runtime for running Dex bytecode.

Major features include:

- Ahead-Of-Time (AOT) compilation
- Just-In-Time (JIT) compilation
- Bytecode interpretation
- Garbage collection
- Core libraries



(Source: https://source.android.com/)

LCG ART Team

- Part of Linaro Consumer Group.
- Consisting of three member engineers from Arm.
- Focusing on optimizing ART for mobile ARM architectures (ARM/ARM64), mostly code generation and compiler optimizations.
- Contributing to the Android Open Source Project (AOSP).
- Reviewing patches from other contributors, like Google, Intel.

("LVC20-307 Android Runtime Development Update 2020" for more information)

Agenda

- Android Runtime (ART) and LCG ART Team
- Benchmarking Overview
- Benchmarking: Past
- Benchmarking: Present
- Benchmarking: Future
- Conclusion

Benchmarking Overview

- Main repository: https://android-git.linaro.org/linaro/art-testing.git/about/
  - Benchmarks and a benchmarking framework.
- Helper scripts: https://android-git.linaro.org/linaro-art/art-build-scripts.git/about/
  - To run benchmarks on devices, to collect performance data, to generate reports from performance data.

Benchmarking Overview

- Targets
  - The optimizing compiler, not the whole Android system.
  - Code from the AOT compilation.
  - Performance on Little Cores.

Benchmarking Overview

- The challenge to benchmark/test changes in ART
  - As ART is part of the Android system, the whole system had to be rebuilt and to be flashed to a device.
- Solution: a special art-testing mode
  - Build only ART and its dependencies. Use a device with a stable AOSP version and reconfigure it to use the new built ART.

Agenda

- Android Runtime (ART) and LCG ART Team
- Benchmarking Overview
- Benchmarking: Past
- Benchmarking: Present
- Benchmarking: Future
- Conclusion

Benchmarking: Past

- 31 micro-benchmarks.
- Ported benchmarks
  - Dhrystone.
  - Linpack.
  - Sorting algorithms: bubble, insertion, merge.
  - The DeltaBlue algorithm: an Incremental Constraint Hierarchy Solver.
  - The Richards benchmark: simulation of the task dispatcher of an operating system.
  - The Reversi Game.

Benchmarking: Past

- Ported benchmarks (cont.)
  - 6 from Caffeinemark.
  - 9 from WebKit SunSpider.
  - 10 LLVM Stanford benchmarks.
  - 15 from Benchmarks Game.

Benchmarking: Past

- The special art-testing mode was based on the “hacked” system linker which could use new ART binaries instead of the old ones.

Benchmarking: Past

- On a device linux perf was used for collecting performance data.

Agenda

- Android Runtime (ART) and LCG ART Team
- Benchmarking Overview
- Benchmarking: Past
- Benchmarking: Present
- Benchmarking: Future
- Conclusion

Benchmarking: Present

- *Big changes happened*
  - The JIT compiler.
  - The special art-testing mode has been changed to a virtualization mode by using the chroot environment.
  - Simpleperf replaced Linux perf.
  - The low-hanging fruit has been picked.

Benchmarking: Present

- *Problem: “The JIT compiler.”*
  - The AOT compilation mode is explicitly set and checked during benchmarking.

Benchmarking: Present

- *Problem: "The special art-testing mode has been changed to a virtualization mode by using the chroot environment."*
  - We have rewritten scripts to support the new mode.
  - We have been working to make the benchmarking framework independent on art-testing modes.

Benchmarking: Present

- *Problem: "Simpleperf replaced Linux perf."*
  - We have switched to simpleperf and use it for runs in the new art-testing mode.
  - We use the simpleperf framework to generate reports from performance data.
  - We have created a new tool: perf2cfg, which annotates a control flow graph of a compiled program with simpleperf data. The tool simplifies performance analysis of compiler optimizations and generated code.
    - https://android.googlesource.com/platform/system/extras/+/a96327f4086ed99e3e889bfcbd72b8920e757d3a

Benchmarking: Present

- *Problem: "All the low-hanging fruit has been picked."*
  - We have improved the stability of existing benchmarks.
  - We have researched existing Java benchmarks.
    - DaCapo (http://dacapobench.org): a set of open source applications with non-trivial memory loads.
    - SPECjvm2008 (https://www.spec.org/jvm2008/): a set of applications and benchmarks focusing on core java functionality.
    - Math benchmarks:
      - JTransforms benchmark (https://sites.google.com/site/piotrwendykier/software/jtransforms): a part of an open source multithreaded FFT library written in pure Java.
      - Java Matrix Benchmark (http://lessthanoptimal.github.io/Java-Matrix-Benchmark/): an open source benchmark for evaluating Java linear algebra libraries for speed, stability, and memory usage.

Benchmarking: Present

- *Issues with Java benchmarks*
  - Too much use of class loaders and reflection.
  - Low quality of code.
  - Dependencies on Java API missing in Android.

Benchmarking: Present

- *Java benchmarks we managed to run on Android*
  - DaCapo (4 out of 14)
    - Xalan: an XSLT processor for transforming XML documents.
    - Avrora: a set of simulation and analysis tools in a framework for AVR micro-controllers. The benchmark exhibits a great deal of fine-grained concurrency.
    - H2: an in-memory database benchmark.
    - Luindex: a text indexing tool.
  - SPECjvm2008 (5 out of 9)
    - Compress: the benchmark compresses data, using a modified Lempel-Ziv method (LZW).
    - Crypto: the benchmark focuses on different areas of crypto.
    - MPEGAudio: a floating-point heavy benchmark that tests of mp3 decoding.
    - Scimark: the benchmark was developed by NIST and is widely used by the industry as a floating-point benchmark.
    - Serial: the benchmark serializes and deserializes primitives and objects.

Benchmarking: Present

- *Java benchmarks we managed to run on Android (cont.)*
  - Math benchmarks:
    - JTransforms benchmark
    - Java Matrix Benchmark and the math libraries
      - Commons Math
      - Efficient Java Matrix Library
      - Jama
      - Elegant Linear Algebra for Java
      - Colt

Benchmarking: Present

- *Optimization opportunities found in SPECjvm2008 workloads*
  - Array bounds checks eliminations
  - Suspend checks elimination
  - Jump threading optimization
  - Inlining tuning
  - Loop unrolling
  - Loop unswitching
  - Accesses to object fields
  - Auto-vectorization
  - Strength reduction of div/rem operations

Agenda

- Android Runtime (ART) and LCG ART Team
- Benchmarking Overview
- Benchmarking: Past
- Benchmarking: Present
- Benchmarking: Future
- Conclusion

Benchmarking: Future

- Make the benchmarking repository part of AOSP.
    - Being part of AOSP helps to reduce a risk of broken scripts/functionality.

Benchmarking: Future

- Look at the latest version of the DaCapo benchmarking suite which has
  - New benchmarks:
    - BioJava
    - Cassandra
    - GraphChi
    - H2O
    - JMonkeyEngine
    - Kafka
    - ZXing
  - Updated existing benchmarks.

Benchmarking: Future

- Look at the Renaissance benchmarking suite (https://renaissance.dev/)

  "... is a modern, open, and diversified benchmark suite for the JVM, aimed at testing JIT compilers, garbage collectors, profilers, analyzers and other tools."

  Benchmarks groups

  - Actors
  - Apache-spark
  - Database
  - Jdk-concurrent
  - Jdk-streams
  - Neo4j
  - Rx
  - Scala-*
  - Twitter-finagle

Benchmarking: Future

- Create an Android specific Java/Kotlin benchmarking suite.
  - Android specifics: application startup time, code size, performance on Little Cores.
  - F-Droid (https://www.f-droid.org/) and GitHub contain a lot of open source applications for the Android platform.

Agenda

- Android Runtime (ART) and LCG ART Team
- Benchmarking Overview
- Benchmarking: Past
- Benchmarking: Present
- Benchmarking: Future
- Conclusion

Conclusion

- ART continues evolving. There are still opportunities for optimizations.
- The LCG ART Team has an up-to-date solution for performance analysis.
- The LCG ART Team has found Java benchmarks which are runnable on Android.
- Not being part of AOSP complicates maintenance of the LCG ART team benchmarking repository.
- There are no good Android specific Java/Kotlin benchmarking suites.
- Existing Java benchmarks are not always suitable for Android or difficult to port.
- The LCG ART Team wants to have an Android specific Java/Kotlin benchmarking suite. (Help is welcome.)

Thank you

Accelerating deployment in the Arm Ecosystem

Linaro Connect VIRTUAL 2020