Real-time Ada applications on Android

Alejandro Pérez Ruiz, Mario Aldea Rivas and Michael González Harbour

Software Engineering and Real-Time Group
University of Cantabria
Introduction (I)

- Android is the most extended operating system in smartphones.
  - Steadily improving its features and hardware support.
  - Extended to new types of devices.

- There is a great interest in using real-time apps on Android.
  - Low cost mobile devices.
  - Energy efficient devices.
  - Interaction between real-time software and large base of Android apps.
Introduction (II)

- It is possible to run applications written in any language using a proper compiler for Android.

- Android devices tend to have multiple cores in their CPUs.

- Open-source licenses in Android allow us to analyze and understand the functionality of this OS.

- Android has shortcomings to run real-time Ada applications without applying additional mechanisms.
Android Architecture

- The Java virtual machine executes most of the Android apps.

- The libc library has been re-implemented by Google and is called Bionic.

- Linux kernel provides the basic system functionality.
Shortcomings of Android for Real-Time

- Well-known limitations of a general purpose operating system like Linux.
  - Response times of the Linux kernel.
  - Completely Fair Scheduler (CFS).
- The bionic library does not have priority inheritance mutexes.
- The Java virtual machine has no bounded response times.
Related Work – Possible modifications to have real-time behaviour in Android

- Patched Linux kernel with real-time features.
  - Native (C/C++) real-time applications.
- Add a real-time Java virtual machine.
- Extend ART (Android RunTime) with real-time features.
- Real-time hypervisor.
  - Real-time applications (C/C++) run on a simple RTOS.
Related Work – Possible modifications to have real-time behaviour in Android

- Patched Linux kernel with real-time features.
  - Native (C/C++) real-time applications.
- Add a real-time Java virtual machine.
- Extend ART (Android RunTime) with real-time features.
- Real-time hypervisor.
  - Real-time applications (C/C++) run on a simple RTOS.

Great effort to apply these modifications through different Android versions
Our solution (I)

- Isolated one CPU core
- Execute real-time applications in the isolated core
Our solution (II)

- **Using the mechanisms** provided by the Linux kernel to isolate a CPU, we can **run soft real-time apps**.
  - CPUSets
  - Interrupt request affinity (proc/irq/int_n/smp_affinity)
  - Static CPU frequency

- **Our solution is very portable** among all kinds of Android devices.

- There is a **substantial improvement in response times** when using the isolation mechanisms.
Worst-case response times (µs)

- **Isolated core**
- **Non isolated core**

- Tests 1 to 4:
Ada on Android?

• It is possible to run C/C++ applications
  - NDK (Native Development Kit)
  - By default, Bionic library

• Google does not offer any compiler for Ada

• There is interest in having Ada in Android
The **Bionic** library is **developed** by Google **under the BSD license**.

- The BSD license allows you to create **proprietary apps**.

**Bionic is smaller** than the traditional glibc library.

- It has some **limitations** to execute **real time applications**.
Bionic library and Ada (II)

• Functions and symbols that are not available in the Bionic library:

  • **Mutexes**:
    • `pthread_mutexattr_setprotocol`
    • `pthread_mutexattr_setprioceiling`
    • `pthread_mutexattr_getprioceiling`

  • **Priorities**:
    • `pthread_setschedprio`
    • `PTHREAD_EXPLICIT_SCHED` (symbol)

  • **Signals**:
    • `sigwaitinfo`
    • `sigqueue`
    • `sigtimedwait`
Building GNAT for Android (I)

- There are several distributions of GNAT. The three main ones are:
  - **GNAT GPL Edition**: Developed by AdaCore and not oriented to professional software.
  - **GNAT Pro**: Supported professional version of GNAT from AdaCore that uses a commercial software license.
  - **GNAT FSF**: Not distributed in binary format and included in the main GCC sources. It has a license (modified GPL) that allows the development of proprietary software.
Building GNAT for Android (I)

- There are several distributions of GNAT. The three main ones are:
  - **GNAT GPL Edition**: Developed by AdaCore and not oriented to professional software.
  - **GNAT Pro**: Supported professional version of GNAT from AdaCore that uses a commercial software license.
  - **GNAT FSF**: Not distributed in binary format and included in the main GCC sources. It has a license (modified GPL) that allows the development of proprietary software.
Building GNAT for Android (II)

- Build a GNAT FSF cross compiler:
  - Target: arm-unknown-linux-gnu
  - Host: linux x86_64

- Steps:
  - Download and build the next sources:
    - binutils
    - gcc
    - glibc and Linux kernel headers
    - mpfr, gmp and mpc (support for floating-point)
GNAT cross compiler

Host
(Linux x86_64)

- C Cross-Compiler
  built from GCC
- Ada Cross-Compiler
  built from GNAT FSF
- Cross-Assembler
  Cross-Linker
  built from Binutils

Target
(Linux arm)

- a.out
  (Ada program)
- RunTime Ada
  built from GNAT FSF
- Standard C Library
  built from Glibc
- Linux Kernel
Replace Bionic by glibc

- The previous cross compiler is not possible to be used with the Bionic library.
  - Important functions for real-time not available.

- It *is possible* to use the traditional glibc library for ARM/Linux on Android:
  - Copy the standard glibc in our target device.
  - Indicate the dynamic linker to be used and the path for the dynamic libraries in the Android device.

```
arm-unknown-linux-gnueabi-gnatmake hello_world.adb -fPIE -pie -largs -Wl,--dynamic-linker=/data/local/libs/ld-linux.so.3 -Wl,--rpath=/data/local/libs
```
Is Glibc suitable for Android?

- We have adapted the functional tests called “OPEN POSIX Test suite”.
  - Threads.
  - Semaphores.
  - Timers.
  - Conditional variables.
  - Message queues.
  - Priority inheritance protocols with mutexes.

- All these tests have been successfully passed when they were run on Android using the glibc library.
Is GNAT FSF suitable for Android?

- The Ada Conformity Assessment Test Suite (ACATS) is a test suite used to verify compilers for conformance with the Ada standard.

- We have adapted an existing script in the GCC sources to execute all ACATS in an Android device.

- The ACATS tests are composed of more than 2300 individual tests. All these tests passed successfully.
Ada and real time on Android

- CPU 1
  - CPU 2
  - CPU 3
    - Applications
    - Application Framework
    - Libraries
    - Runtime
  - Isolation
    - CPU 4
      - Real-Time Ada Applications
      - RunTime Ada
      - Native libraries
      - glibc

Android/Linux Kernel
Ada and real time on Android

- CPU 1
  - Applications
  - Application Framework
  - Libraries
  - Runtime

- CPU 2

- CPU 3

- CPU 4
  - Real-Time Ada Applications
    - RunTime Ada
    - Native libraries
      - glibc

- Android/Linux Kernel

Isolation

Sharing data
Sharing data between applications on Android

- **Anonymous Shared Memory (Ashmem)**
  - Exclusively for Android.
  - Access by file descriptor.
  - Poor documentation.

- **Memory-mapped files**
  - Every 4096 bytes written in the mapped memory segment there is a dump of the data to the file on disk.

- **POSIX shared memory**
  - It is necessary to disable SELinux (security layer).

- **tmpfs**
  - File system that keeps files in virtual memory
  - Supported natively in Android
Sharing data between applications on Android

- **Anonymous Shared Memory (Ashmem)**
  - Exclusively for Android.
  - Access by file descriptor.
  - Poor documentation.

- **Memory-mapped files**
  - Every 4096 bytes written in the mapped memory segment there is a dump of the data to the file on disk.

- **POSIX shared memory**
  - It is necessary to disable SELinux (security layer).

- **tmpfs**
  - File system that keeps files in virtual memory
  - Supported natively in Android
Conclusions and future work

• A solution for the execution of Ada applications in Android has been presented.
  - We have replaced the Bionic library by the traditional libc.

• Using the mechanisms provided by the Linux kernel to isolate a CPU, we can run Ada soft real-time apps.

• Our solution is very portable among all kinds of Android devices.

• We have analyzed which is the most suitable mechanism to share data between real-time apps and non real-time apps.
  • tmpfs.

• Future work: develop non-blocking synchronization mechanisms for communication between real-time and native applications.
Thank you for your attention