Integrating Linux and the real-time ERIKA OS through the Xen hypervisor

Arianna Avanzini

Evidence Srl — University of Modena and Reggio Emilia

October 15, 2014
Motivations: use cases, Evidence Srl’s previous solution

ERIKA Enterprise RTOS as a Xen-on-ARM domU
- Status of the port
- Communication between the ERIKA domU and a Linux dom0

Ongoing and future work
Use cases

- Systems where **infotainment** services are provided as much as **safety-critical** ones
- Trend: complement **general-purpose** operating systems with **real-time** ones
  - Interacting with each other
  - Automotive, avionics
Requirements

- Guarantees on execution time for safety-critical tasks
  - Guarantees on boot time of the operating system
- Protection of the integrity of the real-time OS
  - due to the general-purpose OS malfunctioning
- High performance: multi-core platforms
- Communication between the two operating systems
Evidence Srl’s existing solution: outline

- Dual-OS design implemented on a dual-core 1.2 GHz Freescale i.mx6 board
- Each operating system is assigned exclusively a core
- Operating systems share memory

Existing solution: real-time OS

ERIKA

http://erika.tuxfamily.org

- Small-footprint OS (1-4 Kb)
- With hard real-time support
- Supports multi-core platforms and stack sharing between tasks
  - OSEK/VDX-certified
  - Open source (GPLv2 + linking exception)
Existing solution: general-purpose OS

- Fully-featured Linux operating system
- Extended with a driver implementing interaction with ERIKA
- Not certified, but responsible only for non-safety-critical tasks
Existing solution: pros

- Each OS runs on a dedicated core
  - No temporal interference, allowing for guarantees on response time in safety-critical tasks
  - Able to provide guarantees on boot time of the real-time OS
- Shared memory allows for very efficient communication
Existing solution: cons

- Limited support to isolation
- Access to shared memory is implemented with mutual exclusion mechanisms...
- ... but a malfunctioning general-purpose OS could **overwrite the memory area assigned to the safety-critical OS**
  - Conversely, a failure in the real-time OS could pollute the memory area assigned to the non-safety-critical OS
Introducing Xen

- Main idea: **use the Xen hypervisor to guarantee isolation**
  - The hypervisor guarantees protected access to memory
- Further ideas:
  - avoid temporal interference even if OSes run on virtual CPUs
  - exploit mechanisms provided by Xen to implement communication
Goal: proof-of-concept dual-OS system running on top of Xen

→ Selected platform: SUNXI Allwinner A20 (cubieboard2)

1. Have Linux as a dom0, ERIKA as a domU
   - Port ERIKA as a Xen-on-ARM domU

2. Allow ERIKA to access I/O memory of devices

3. Have the hypervisor assign statically a core to each domain
   - Can be achieved with system configuration

4. Implement communication between dom0 and domU
   - Exploit grant table references, event channels
Sub-goal: have ERIKA actually running as Xen-on-ARM domU

- Starting point: pilot MiniOS ARM port, Linux ARM
- Perform changes on ERIKA core and build system
  - zImage preamble, image start address, work around instructions not allowed for domUs, ...
- Add basic debugging framework
- Add Generic Interrupt Controller driver
Step 1/4: working domU (b)
Step 2/4: I/O-memory access (a)

**Sub-goal:** enable ERIKA to use I/O memory of peripherals

- Chosen peripheral for the proof of concept: GPIO controller
- Starting point: SUNXI GPIO driver, ported to ERIKA
- Use of the memory mapping Xen DOMCTL to allow to the ERIKA domU access to the needed I/O-memory ranges
Step 2/4: I/O-memory access (b)
Step 3/4: static assignment of CPU cores (a)

Sub-goal: provide ERIKA with exclusive use of a CPU core
- Easily achievable via dom0 and domU configuration, by assigning statically one physical CPU to dom0 and mapping the domU’s vCPU to the remaining core
Step 3/4: static assignment of CPU cores (b)
Step 4/4: implement inter-domain communication (a)

**Sub-goal:** enable Linux to trigger the execution of an ERIKA task

- Communication must be:
  - synchronous for the Linux dom0
  - asynchronous and non-preemptive for the ERIKA domU

- Exploit the event channel inter-domain notification mechanism provided by Xen
  - event channels can be masked, ensuring that high-priority tasks are not preempted
Step 4/4: implement inter-domain communication (b)

**Sub-goal**: enable Linux to trigger the execution of an ERIKA task
- Communication must be as efficient as possible
  - share memory
- A dedicated set of memory pages is explicitly shared by the ERIKA domU
  - exploiting the memory granting mechanism provided by Xen
  - access permissions are granted only to dom0
Step 4/4: inter-domain communication setup (a)

Arianna Avanzini

Integrating Linux and ERIKA through Xen
Step 4/4: inter-domain communication setup (b)
Step 4/4: inter-domain communication setup (c)
Step 4/4: implement inter-domain communication (a)

- Basic driver implemented in Linux
- Allows Linux to trigger the execution of an ERIKA task
- Exposes two tunables in the dom0’s sysfs
  - pin number and pin value
- Triggers the assignment of a value to a certain GPIO pin through ERIKA
Step 4/4: implement inter-domain communication (b)
Step 4/4: implement inter-domain communication (c)
Step 4/4: inter-domain communication protocol (a)

- The shared memory area is used as a container for a message
- The dom0 writes a command to be executed by the domU
  - The command is a set of values: (pin number, pin value)
- The domU writes a return value for the operation as soon as it’s completed
Step 4/4: inter-domain communication protocol (b)
Step 4/4: inter-domain communication protocol (c)
Xen-based design

- The proposed solution matches Evidence Srl’s dual-OS design
  - adding the Xen hypervisor as an extra layer
- Pros and cons ahead
Xen-based design: pros

- The Xen-based design guarantees the isolation of the operating systems
- Communication is still possible
  - with safe use of shared memory, mediated by Xen
  - and Xen’s synchronous, maskable inter-domain interrupts (events)
Xen-based design: cons

- ERIKA runs as an unprivileged domain
  - must wait for the Linux dom0 to boot
- No guarantees about its boot times!
  - Actually, no guarantees that it boots at all...
- Xen is not certified
  (and apparently not easily certifiable above DAL-E/D)
Further steps

- Port ERIKA Enterprise as a Xen-on-ARM dom0
  - Consequently, port Xen’s toolstack (or part of it) to ERIKA
- Investigate the possibility of an ASIL or DAL-B/A certification for Xen
  - There is in-progress work to certify the core subset of the Xen codebase
Thank you

Questions are welcome

Also offline:

- ask Paolo Valente conceptual questions here,
- mail me at avanzini.arianna@gmail.com for low level details.