COL862

Automated OS-level Device Runtime Power Management

01-10-2016

Rajesh Kedia
Shailja Pandey

PowerAdvisor for PM code insertion

- PowerAsvisor: A tool that suggests location of PM calls.
- Flow:
  - Instrument the driver with the tool.
  - Run the instrumented driver with test runs and generate traces.
  - Analyze the traces and generate a set locations for pm_get() and pm_put() calls.

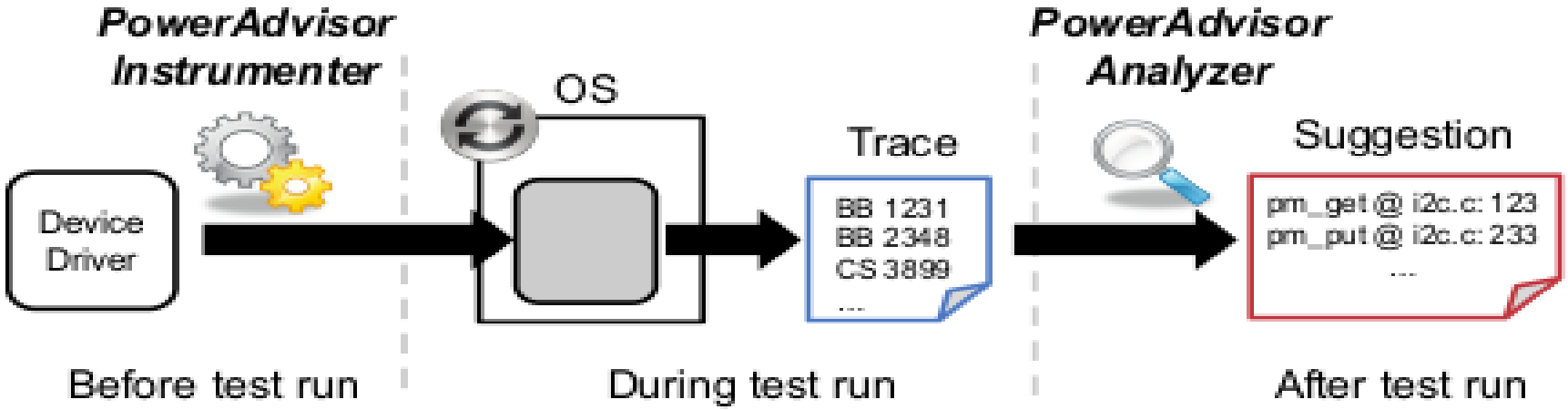

Figure 8: The workflow of PowerAdvisor

Is user involvement needed?

- PowerAdvisor provides 2 types of guarantees:
  - G1: During any no-pending-task period with time greater than Threshold time, the reference counter will remain as zero.
  - G2: During any active time, the reference counter will remain greater than zero.
- Where exactly user involvement is needed?
  - PowerAdvisor may not understand intricacies of such calls and there effects on applications.
    - Since number of calls are not too many, user can help analyze if any functionality would be broken.
  - Limited to comprehensiveness of the trace and user needs to check if all paths are exercised.
    - Some static analysis tools can help generate high coverage test runs.

Internals of PowerAdvisor

- Candidate locations:
  - pm_get(): start of any basic block (BB) that contains device register access.
  - pm_put(): end of any basic block that contains device register access.
- These locations can help meet G1 and G2.
- Tradeoff analysis:
  - Number of locations to insert PM calls.
  - Insert calls only in a subset of paths in BB shared across different execution paths.

Components

- Inserting Tracepoints:
  - Devices can be accessed using:
    - Device register access.
    - Function calls to perform device register access.
  - During instrumentation, following locations are added as tracepoints:
    - Start and end of each marked basic block.
    - Locations before and after each marked function call sites.
  - Each tracepoint has a unique identifier.

Components

- Collecting Traces:
  - Whenever a tracepoint is reached, it appends its entry into a global buffer.
  - A no-pending-task entry is created if the interval between consecutive tracepoints is more than threshold time – done by a small piece of code added to kernel.
  - Resulting trace is a sequence of identifiers delimited by no-pending-task periods.

Components

- Analyzing Trace:
  - Formulates the problem into a SMT (Satisfiability Modulo Theory) problem.
  - Establishes the SMT constraints based on trace to satisfy G1 and G2.
  - Generates subsequences from trace based on no-pending-task periods.
  - Constraints:
    - G1: All calls to pm_get() and pm_put() are balanced.
    - G2: Whenever device is accessed, pm_get() must be more than pm_put().
  - Optimization: Minimize the number of calls.

Components

- SMT formulation:
  - Add SMT variables for the start (S_{BB}GET), and the end (S_{BB}PUT) of the marked basic blocks, Before (S_{CS}GET) and after (S_{CS}PUT) the marked call sites.

$$\forall v \in S_{BB\_GET} \cup S_{CS\_GET}, v \in \{0, 1\}$$
$$\forall v \in S_{BB\_PUT} \cup S_{CS\_PUT}, v \in \{0, -1\}$$

- G1: Calls to pm_get and pm_put are balanced.

$$\sum_{q_i \in Q} q_i = 0$$

- G2: Reference counter larger than 0 when device access.

$$\forall j \in \{1...N\}, \text{if } q_j \in S_{BB\_GET}, \text{then } \sum_{i=1}^{j} q_i > 0$$

Components

- Minimize number of PM calls.
- The analyzer explores values of C from 2 and upwards until a solution is found.

$$C = \sum_{v_i \in V} |v_i|.$$

- SMT variables having non-zero values are candidates for PM code insertion.
-
- Though ILP could be used and should provide better efficiency, authors are more familiar with SMT and hence used it.

Evaluation of PowerAdvisor

- Study the driver characteristics for the 4 modules used.
- For I2C and MMC, PowerAdvisor suggests PM code similar to hand-tuned code.
- SDIO and DISPC have no PM code, but PowerAdvisor suggest PM code.
- Tested for 48 hours of continuous animation, no break in the application.

Table 3: Driver traces used in evaluating PowerAdvisor. *#NPT stands for the number of no-pending-task periods.

Device	Trace			SMT Problem	
	Time (s)	#Entries	#NPT*	#Vars	#Constraints
MMC ctrl.	50.2	20,000	19	58	4,930
I2C ctrl.	33.9	5,000	13	32	1,480
SDIO ctrl.	24.4	5,000	78	14	1,708
DISPC	183.2	679,915	275	294	126,274

Related Work

- System Suspension Management
- Runtime Power Management
- Formal Methods and Synthesis Tools

Related Work: System Suspension Management

- Suspend the SoC if no user interaction is seen in recent past.
- Android does this but applications can prevent using wakelocks.
- Incorrect wakelock placement leads to multiple power related bugs in the system.
- Techniques to detect wakelock related bugs are limited to code with single entry/single exit paths.
- Static analysis can handle wakelocks that are not reference counted.

Related Work: Runtime Power Management

- Performance setting for each of the computational units (e.g. DVFS) depending on workload.
- Needs to be done for uncore part of the chip also for gaming applications.
- Device runtime PM requires individual devices to be turned off when not in use.
- However, runtime PM for device has a challenge to timely turn on/off the device.
- ICEM integrates PM code into locks for drivers. But works only for shared drivers, which is not common in Linux.
- I/O API for applications to expose hints for better PM. But are generally restricted to certain devices.

Related Work: Formal Methods and Synthesis Tools

- PowerAdvisor is closely related to formal methods.
- Most of the model checking and formal analysis tools can help detect bugs in code, but do not suggest missing PM calls.
- Generally PM calls are inserted by designer.
- Some synthesis tools can generate PM code but need detailed specifications or RTL code.
- Authors claim that generating such specifications could involve equal amount of work as writing PM in drivers.

Conclusion

- How far can pure HW PM support go?
  - Complete HW based PM may be infeasible due to its complexity.
    - Storage overhead – Retention of state, etc.
    - Managing QoS in HW can be difficult.
- How general is the central PM agent?
  - Works under certain conditions:
    - Memory mapped devices
    - Bounded interval for register access to device
    - Interruptible slave modules (e.g. I2C) when in low power state.
  - All these conditions are mostly true for ARM based SoC.

Conclusion

- Burden of PM from Linux drivers can be offloaded to central PM agent.
- Hardware based and Software based techniques proposed.
- A best-effort tool named PowerAdvisor proposed to analyze and suggest where to put PM calls.