Kernel Level DVFS Implementation
Progress Report
February 17, 2015

Andreas Prodromou
David Lane

Motivation
Since power management is one of the most limiting problems with embedded systems, there are plenty of reasons to develop a more effective power governor. Excess power consumption generates more heat, discharges batteries more quickly, and can lead to shorter lifetime of components.

Project History
Our project has changed a couple of times during these first couple of weeks. Our initial project idea was a “digital handshake” that would use your smartphone to detect when you were approaching or being approached by someone else that had a digital handshake enabled on their phone. When the phone did detect a potential contact, then the application would display the contact's picture and basic information on your phone so that you could be given a brief introduction or summary of the person before actually meeting.

This project idea was stopped because it was determined to be infeasible with the technology of the phone. The intent was to use bluetooth to monitor the area for potential contacts, and to alert / report to a user when a contact was approaching. The intended implementation was to monitor the received signal strength indication (RSSI) as a metric for distance, and to label an approaching contact as a marked increase in the RSSI on a particular bluetooth frequency. Upon further examination however, the idea was scrapped on account of a poor correlation between distance and RSSI – therefore not allowing our implementation to accurately distinguish between people approaching and merely being in the same general area.

The new project is to design and implement a kernel level governor on the Qualcom 8960 smartphone that was used in the first project. Since this implementation will have access to the kernel, we expect good results for general usage patterns. Like the first project, we plan to test our governor with a mixture of SPEC tests and, more importantly, tests that simulate a more interactive work flow.

Progress Summary
So far we have read through the ARM Cortex A-8 manual and have identified the main hardware counters of interest. We have read a few related research papers and used those to define our governor policy according to the available counters, but have not yet implemented or tested the policy. We familiarized ourselves with the process of compiling and injecting a module that interacts with the kernel - however have yet to fully succeed in getting a working policy.

Related Work

a) DVFS is not a new topic and therefore there is plenty of work that has already been done. Previous research shows that overall performance is not degraded by by decreasing the CPU frequency during memory-bound sections of code. Various governor policies have been proposed in research papers - however many of these use a priori knowledge of process runtimes and deadlines in an experimental environment.

b) Our project aims to create general purpose governor policies that will use dynamic events that are represented in hardware counters to determine our decisions. The first step will be to monitor each performance counter during execution of workloads. Based on our measurements, we will focus on correlating each counter with power consumption. We will be essentially assigning weights to each counter. These weights will be used by our DVFS policy later on.

At the end of specified intervals, our governor will check the difference at each counter's value since the last interval. According to the individual counter weights, the governor will "decide" on the best frequency to be used in order to optimize the EDP for the next interval.

We will implement three different DVFS policies which will react with a different decision under the same measurements. We will create a conservative, a neutral and an aggressive policy. Conservative will be the most power efficient one, without setting the frequency at the lowest level at all times. Essentially, this policy will make it harder to jump to a higher frequency, but if the demand seems to require such a decision, it will be made. The aggressive policy will work in the opposite way i.e. make it harder to drop to lower frequencies. For the neutral policy, we will try to balance the two other policies, hopefully getting the best of both worlds.

Development Progress

a) The Cortex -A8 has a number of different hardware counters that are accessible when in kernel mode. However only 4 can be interfaced with at a single time. These are the counters which we plan to focus on:

- 0x01 or 0x03 or both: These count the number of memory accesses that cause a cache refill at the lowest level in the instruction and data caches respectively. In general, data cache misses are much more frequent than instruction cache misses so potentially there is less opportunity to gain much power savings. However if we find that the data cache misses are too frequent or do not indicate strongly enough that there is a memory bound section ahead, we could use the instruction cache misses instead because they are indicative of a jump in the instruction flow (branch mispredicted, or exiting a for loop) which is likely to be a strong indicator of future cache misses.
- 0x02 or 0x05 or both: TLB misses for instructions and data. Similar to above, but less frequent.
- 0x06 and 0x07: Counters for each data read or write respectively.

All of these counters are clearly tied to memory accesses and we feel that they will be good indicators of the types of code that would benefit from a lower frequency.

b) The software component of the project will be the coding of the DVFS policy, as well as the testing to collect metrics used to fine-tune the policy with a number of different work loads. Thus far we have focused on getting a basic kernel governor to compile and interact with the kernel.

Experimental Section

a) Metrics of Success

- Average power consumed in terms of energy delay product divided by total run time of a test being run.
- Adequate performance in terms of frames per second

As a baseline, we can compare against the same tests being run with other governor policies. In particular, the 'ondemand', 'powersave', and 'performance governors should provide a good overview of the expected ranges we should anticipate.

b) We have not performed any of the experiments yet, but the testing should be pretty straight-forward. With any given DVFS policy and a set of parameters we will have a variety of interactive tests similar to project one. The governor will run and collect the average power consumption. By watching the test run, we will see if any policy is too aggressive and makes the tests jumpy.

Deliverables for End of Project

By the end of the project we will have an effective governor that adapts to a general purpose load that it typical of common use. This will be somewhat difficult to show with an in-class demonstration. We can briefly show the subjective metric of whether the performance (frames per second) stays adequate under a variety of different loads, however the presentation will likely focus on the design decisions made while making the governor. We can explain and hopefully quantify how important different hardware counters are in respects to the DVFS policy, and to show how and why various weighting parameters were applied.

References

- Choi, Soma, Pedram, "Fine-Grained Dynamic Voltage Frequency Scaling for Precise Energy and Performance Trade-off based on the Ratio of Off-chip Access to On-Chip Computation Times" (http://sportlab.usc.edu/~massoud/Papers/dvfs-tcad-journal.pdf)
- Chung-Hsing Hsu, Wu-Chun Feng, "Effective Dynamic Voltage Scaling through CPU-Boundedness Detection" (http://public.lanl.gov/radiant/pubs/sss/pacs04-lncs.pdf)