Project Part 2: DVFS on CPU and GPU

Due date: 2/3/2015

Hardware checkout

Each team should print out the release form, fill it out and bring it to CSE 2148 on Friday 10-12p or 1-4:30p to pick up a Qualcomm development phone.
Release form: http://cseweb.ucsd.edu/classes/wi15/cse237A-a/project/proj2/PhoneReleaseForm.pdf

- Connect the phone to your laptop by USB.
- If running Ubuntu on VirtualBox, there is an extra step so the VM recognizes your phone consistently:
  ```
  Machine > Settings > Ports > USB > [+] button > Android [0228]
  Devices > USB Devices > Android [0228]
  ```
- Use adb to see its status:
  ```
  $ adb devices
  List of devices attached
  b176272d        device
  ```
- Send files with adb
  ```
  $ adb push somefile.out /path/on/target
  ```
- Issue commands with adb
  ```
  $ adb shell run_something.sh
  ```

Metrics collection

Here are some known userspace metrics and locations in the Android filesystem where you can search for relevant statistics:

- CPU frequencies
  - `/sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_*`
  - `/sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_*`
  - `/sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq`
- CPU governors
  - `/sys/devices/system/cpu/cpu0/cpufreq/scaling_*`
- CPU utilization (reference: http://www.mjmwired.net/kernel/Documentation/filesystems/proc.txt#1231)
  - `/proc/stat`
- GPU frequencies
  - `/sys/class/kgsl/`
- GPU utilizations
  - `/sys/class/kgsl/kgsl-3d0/gpubusy` reports two integers - `g0` and `g1`. The actual load as a percentage can be calculated as `(g0/g1*100)`
- GPU governors
  - `/sys/class/kgsl/kgsl-3d0/pwrscale`
- Battery voltage, power and capacity
  - `/sys/class/power_supply/battery`

You are provided a skeleton of the code needed to report metrics above. Expand the code and use it to analyze workloads provided so that you can determine what would make for a good DVFS policy first for just CPUs, and then include also GPU.

To download, compile, and run the skeleton code on your target (# on the last line indicates you are running as root in the target shell):

```
$ curl -O http://cseweb.ucsd.edu/classes/wi15/cse237A-a/project/proj2/m.c
$ arm-linux-gnueabi-gcc -static -o m.out m.c
$ adb push m.out /data/local/.
$ adb shell
# ./data/local/m.out
```

CPU-only workloads

The SPEC_benchmarks_wi15 package provided contains 5 sample benchmarks from the SPEC CPU suite - bzip2, gobmk, hmmer, mcf and sjeng. These workloads are single-threaded, so they will only exercise one of the phone's CPU cores. Each benchmark directory contains a binary `*.arm-gcc42-05` and input files. As a habit, use the command `pwd` to verify that you are in the correct directory, and `ls` to check files residing in the current directory.

Download the benchmark suite

```
$ mkdir ~/project2
$ cd ~/project2
$ curl -O
http://cseweb.ucsd.edu/classes/wi15/cse237A-a/project/proj2/SPEC_benchmarks_wi15.tar.gz
$ tar xvf SPEC_benchmarks_wi15.tar.gz
$ cd SPEC_benchmarks_wi15
```

Use `adb shell` to access the target shell. On your device, create a directory for each benchmark.

```
# cd /data/local
# mkdir bzip2
# mkdir gobmk
# mkdir hmmer
# mkdir mcf
# mkdir sjeng
# exit
```

Back on the host machine, send the benchmark files to the target

```
$ adb push bzip2/ /data/local/bzip2
$ adb push gobmk/ /data/local/gobmk
$ adb push hmmer/ /data/local/hmmer
$ adb push mcf/ /data/local/mcf
$ adb push sjeng/ /data/local/sjeng
```

Go to your device again using adb shell, and grant execution privileges to each binary file. Execute these commands in the appropriate benchmark directory (`cd /data/local/<benchmark>` first)

```
# chmod 755 bzip2_base.arm-gcc42-05
# chmod 755 gobmk_base.arm-gcc42-05
# chmod 755 hmmer_base.arm-gcc42-05
# chmod 755 mcf_base.arm-gcc42-05
# chmod 755 sjeng_base.arm-gcc42-05
```

Run the benchmarks (again, in the appropriate `/data/local/<benchmark>` directory). `>` redirects `stdout` to a file and `2>>` redirects `stderr` to a file

```
# ./bzip2_base.arm-gcc42-05 dryer.jpg 2 > dryer.jpg.out 2>> dryer.jpg.err
# ./gobmk_base.arm-gcc42-05 --quiet --mode gtp < dniwog.tst > dniwog.out 2>> dniwog.err
# ./hmmer_base.arm-gcc42-05 --fixed 0 --mean 325 --num 45000 --sd 200 --seed 0
bombesin.hmm > bombesin.out 2>> bombesin.err
# ./mcf_base.arm-gcc42-05 inp_test.in > inp.out 2>> inp.err
# ./sjeng_base.arm-gcc42-05 test.txt > test.out 2>> test.err
```

Interactive apps

MSM8960 users: Refer to SmallProjectPart2InteractiveMSM8960.pdf
MSM8660 users: Refer to SmallProjectPart2InteractiveMSM8660.pdf

Project components:

1. **Design a userspace governor for CPU:** Take CPU-only benchmarks and design a userspace governor for CPU only. Using the metrics as input parameters, expand the example given in m.c to design an intelligent, energy-efficient governor. Obtain estimates of power, energy, performance, and energy delay product. Report how you designed your governor, why you made choices you made, and provide results for the workloads provided. Make sure to also test your governor with mixes of workloads. You may refer to the following table to estimate how much energy a workload would consume.

CPU Frequency (Mhz)	Active power (mW)	Idle Power (mW)
1512	450	35
1458	400	32
1350	350	28
1242	304	25
1188	262	23
1134	252	21
1080	231	19
1026	210	18
972	199	16
918	184	16
864	168	15
810	157	15
756	136	14
702	126	14

648	115	13
594	105	12
540	94	12
486	84	10
432	73	10
384	63	10
192	37	9

2. **Design a userspace governor for CPUs & GPU:** Play around with installed apps (not CPU-only benchmarks) on the phone while your metrics collector is running. Observe how application power profiles differ. Note that gaming and video apps are not the only ones that exercise the GPU significantly (try the Browser and Gallery apps as well). Use your insights to design a DVFS governor for CPU and GPU. Describe your design, justify your design choices and discuss how you quantified the tradeoff between minimizing power consumption vs. user satisfaction.

MSM8660

GPU-3D Frequency (MHz)	Active power (mW)	Idle Power (mW)
266.667	747	333
228.571	653	280
200.000	620	250
177.778	614	201

MSM8960

GPU-3D Frequency (MHz)	Active power (mW)	Idle Power (mW)
400	1000	350
300	450	280
200	300	250
128	270	180

You may get some ideas from these reference papers (remember to cite them in your report). All links are accessible from UCSD campus.

- Mittal, Sparsh. "A Survey of Techniques For Improving Energy Efficiency in Embedded Computing Systems." *arXiv preprint arXiv:1401.0765* (2014). link
- Dhiman, Gaurav, Kishore Kumar Pusukuri, and Tajana Rosing. "Analysis of dynamic voltage scaling for system level energy management." *USENIX HotPower* 8 (2008). link
- Pathania, Anuj, et al. "Integrated CPU-GPU Power Management for 3D Mobile Games." *Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference*. ACM, 2014. link

- Dhiman, Gaurav, and Tajana Simunic Rosing. "System-level power management using online learning." *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 28.5 (2009): 676-689. link

Project deliverables:

The project grade will be evaluated with the following components:

- Code (*.c and *.h files only)
  - Governor policies: one for CPU-only benchmarks, the other DVFS policy for both CPU & GPU
- Written report (PDF & hard copy)
  - Justification and description of governor policies
  - Results for CPU-only benchmarks, and CPU/GPU benchmarks
- In-class demo
  - Power/performance results collected for benchmarks provided in class

Compress your source code and PDF report into an archive named proj2_<ad login name 1>_<ad login name 2>.zip (e.g. proj2_tajana_csc019.zip). Submit this on Ted by the beginning of class on 2/3/2015. Bring a hard copy of your report to the demo.