XtndIt: A Framework to Predict Extended Battery Life by Closing an Application

Sanmukh R. Kuppannagari

December 11, 2014

1 Introduction

The objective of this project is to understand the tools and methods for energy and performance related statistics collection. Using the collected data and a simple power model of the computer, the affect of a candidate process on the battery life is predicted.

2 Power Model

We use a very simple power model for this project. The power model is stated below:

$$E_{ba} - E_p = X * E_{ca} + C \tag{1}$$

Here X is the number of Last level cache accesses. The energy dissipated from the battery E_{ba} and the energy consumed by the processor E_p can be collected as detailed in Section 3. L1 cache is treated as a part of the cpu cores and so it not modeled separately. A shared Last Level cache is modeled and the energy consumed is assumed to be proportional to the number of accesses to the LLC. Energy consumed per access to LLC is denoted by E_{ca}. Any other energy dissipated from the system is assumed to be a constant and is denoted by C. The constants E_{ca} and C are calculated using polynomial curve fitting as detailed in Section 3.

The affect of a single application/process on the battery life is calculated using the following equation:

$$E_{pid} = E_p * R + X * E_{ca} + C \tag{2}$$

where, R is the ratio of jiffies of the PID to the jiffies system wide X_{pid} is the LLC accesses performed by PID and E_{pid} is the final energy consumed by process pid.

Extension in battery life if the process is killed is calculated by the equation:

$$Time = (E_{ba} - Epid)/discharge\_rate \tag{3}$$

Discharge rate is calculated by collecting the data for battery energy at various intervals.

3 Data Collection Methodology

The following data are collected for this project:

1. The power consumed by the processor for a given interval.(Section 3.1).
2. The ratio of the jiffies (time unit in linux) utilized by the application and the system.(Section 3.2)
3. The total energy discharged by battery in a given interval.(Section 3.2)
4. Total number of LLC accesses systemwide in a given interval.(Section 3.3)
5. Number of LLC accesses performed by the process in a given interval.(Section 3.3)

Due to limitations of the tools on the candidate platform some other relevant statistics such as LLC-misses, the number of hits to DRAM or the power consumed due to disk accesses could not be collected. These statistics could have improved the power model.

The methods of collection of each type of data is detailed in the following subsections.

3.1 Power Gadget

RAPL or Running Average Power Limit provides a set of counters which provide information related to the power consumption of CPU. These counters are exposed using the msr (Model Specific Registers) which are exposed to userspace via "/dev/cpu/$cpuid/msr" device file. Power Gadget is a wrapper around RAPL and extends the functionality to calculate power consumption for CPU.

In servers, RAPL supports counters for reporting DRAM power consumption, however that is not the case with laptops and so the information could not be obtained.

Power Gadget is invoked as follows:

```
sudo ./power_gadget/power_gadget -e 1000 -d $TIME
```

The arguments mean that for $TIME duration samples are collected at intervals of 1000ms.

3.2 Linux Proc Filesys

Proc Filesystem is a special filesystem in linux which provides information about the processes in the system. We use the following files to collect statistics:

1. **/proc/stat:** Provides system wide statistics. We use this file to calculate the total number of jiffies utilized by the system.

2. **/proc/$pid/stat:** Provides the statistics for a single process with process ID: pid. We use this file to calculate the jiffies utilized by the process.

3. **/proc/acpi/battery/BAT0/state:** Provide the battery status in terms of mAh and mV. We use this data to calculate the battery discharge rate by reading the file at different intervals.

3.3 Perf Tool

Perf is a collection of linux tools to collect statistics from various hardware counters. Perf stat is one of the tools from this collection which collects performance related counters system-wide or for a running process. We use perf to collect number of Last Level Cache accesses both sytem-wide and for a given pid. Counters for LLC hits and misses were not available on the candidate platform and so they could not be modeled into our power model.

The perf-stat tool is invoked as follows for system-wide data collection:

```
perf stat -e LLC-loads,LLC-stores -a
```

It is invoked as follows for collecting data for a process with pid = PID:

```
perf stat -e LLC-loads,LLC-stores -p $PID
```

4 Experimental Results

4.1 Experimental Setup

The framework implemented as a part of this project is written in python and bash. Data is collected on a System 76 laptop which has an *Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz* with 4 cores. The operating system is linux.

Prior to the invocation of the framework, the following commands need to be executed.

```
$ modprobe cpuid
$ modprobe msr
```

The usage of framework is as follows:

```
sudo xtndit.py [-h] [-e ESTIMATE] [-s SAMPLES] [-w WAIT] [-p PID]
```

The candidate process is allowed to run first. Then the framework is invoked with the parameters. The framework waits for WAIT time before collecting the statistics. This time is used to warmup the process. Then $SAMPLES$ samples of 10 second each are collected. Each sample collects the following statitics

1. Processor Energy Consumption

2. Total LLC cache accesses

3. Total energy discharged by battery

These 10 samples are used to perform regression analysis and come up with the constants E_{ca} and C as mentioned in equation [1]. Once the equation is generated, statistics are collected for $ESTIMATE$ time interval. The statistics collected are:

1. Processor Energy Consumption
2. LLC Cache accesses by PID
3. Ratio of jiffies utilized by PID and systemwide
4. Energy discharged by battery

The values are substituted into equation [2] and [3] to obtain the minutes by which the battery life can be extended if the process with PID is killed.

4.2 Results

Apart from some standard applications, we created a simple bash script called **test.sh** to perform a computation infinitely. The bash script is also attached in the submission. The results for various applications are enumerated in the Table 1.

Table 1: Estimated battery life extension due to some applications

Application Name	Predicted Battery Life	Battery Life Extension	Comments
test.sh	126 min	46 min	Compute intensive shell script
firefox	164 min	40 min	No user interaction
firefox	160 min	102 min	video streaming
texstudio	155 min	60 min	while creating report
evince	153 min	52 min	pdf viewer while reading report

5 Limitations and Future Work

The power model is very limited. It does not capture the affect of various input and output devices. It also does not capture network interface, DRAM and hard disk. Future work could be to extend the data collection to various other devices and improve the accuracy of the results.

6 References

(1) Power Gadget tool: https://software.intel.com/en-us/articles/intel-power-gadget-20
(2) Perf-stat: http://manpages.ubuntu.com/manpages/precise/man1/perf-stat.1.html
(3) Proc filesystem: https://www.kernel.org/doc/Documentation/filesystems/proc.txt

(4) RAPL: https://01.org/blogs/tlcounts/2014/running-average-power-limit-%E2%80%93-rapl