Australian Journal of Basic and Applied Sciences, 7(5): 45-48, 2013
ISSN 1991-8178

Improving embedded Linux CPU throttling mechanism

Shuhaizar Daud, R. Badlishah Ahmad, Abid Yahya, Ong Bi Lynn

Kluster Embedded Computing, Unit Kluster Penyelidikan Universiti Malaysia Perlis, Jalan Pangkalan Assam, 01000 Kangar, Perlis, MALAYSIA.

Abstract: CPU power consumption is a subject of high importance in embedded system implementation. Current industry trend of adopting multicore embedded systems paired with linux backend require that the CPU throttling mechanism be as optimized as possible to provide with the best possible efficiency in utilizing available power. In this paper we study the actual CPU power requirement of a modern embedded processor during idle and active state to have a better insight on how the available power is utilized. This study are hoped to provide a better understanding on ways to improve the power consumption of an embedded processor by development of a better CPU throttling mechanism.

Key words: embedded system, embedded linux, CPU power consumption, CPU governor, CPU throttling, Dynamic Voltage & Frequency Scaling

INTRODUCTION

Since its first introduction to the world in 1991, Linux have been making its way to becoming one of the most popular operating system in the world. With the advent of smartphones and the ever increasing popularity of embedded systems, Linux is becoming even more popular as the choice operating system for embedded implementation. The general flexibility of the Linux kernel which could be ported to an embedded platform while still maintain a relatively wide support for new and upcoming hardware makes Linux a great choice for developers. Linux ability to work on older, slower platform and not requiring a large memory footprint for operation adds to the attraction as embedded systems in general have never came with a speedy CPU and large amount of memory.

While the general Linux operating system could be adapted to the embedded platform, some of the components are built for general computer systems and are not specifically developed for an embedded platform. Components such as the CPU governor though could operate in an embedded environment, are not developed around an embedded processor thus posses a possibility for further optimization in an embedded environment.

Dynamic Voltage & Frequency Scaling (DVFS):

In order to reduce the power consumption of the processor during low load, a method to reduce the CPU voltage and frequency have been developed to scale the CPU frequency according to the workload. This reduces not only the power consumption of the CPU but also helps to prolong the lifespan of the CPU by reducing heat dissipation thus avoiding premature failure (Kadin and Reda, 2008 and Ayoub *et al*., 2011). Such a technology is particularly beneficial for embedded systems mainly because of the power constraints and heat dissipation reduction that allow for better use of the available power (Kadin and Reda, 2008).

In Linux the DVFS function is provided by the *cpufreq* interface for processors supporting such function in their hardware mechanism (Pallipadi & Straikovskiy, 2006 and Pallipadi, Intel.com).

Linux CPU Throttling Mechanism:

Currently the best CPU throttling in Linux exists in the form of the *ondemand* governor which is introduced in October 2004 together with the linux-2.6.9 kernel (Pallipadi & Straikovskiy, 2006 and Fedora Project Docs., 2013). While it still provide some basic functionality to throttle the CPU voltage and clock, it's been a while since any improvement made to the component and with significant improvement made to the CMOS technology there is a possibility that the component could be improved further especially for embedded implementation.

The technology to throttle CPU clock on the fly used for the development of the *ondemand* governor are based on the general assumption that the power requirement of the CPU scales according to the clock frequency of which the processor is operating at (Saha and Ravindran, 2012, Lawitzky *et al*., 2008 and Kimm *et al*., 2007). The development of the governor itself are made around processor technology available back in 2004 and since then numerous technological advancement have been made in CMOS and processor technology and processor power management solutions.

Corresponding Author: Shuhaizar Daud, Kluster Embedded Computing, Unit Kluster Penyelidikan Universiti Malaysia Perlis, Jalan Pangkalan Assam, 01000 Kangar, Perlis, MALAYSIA.
E-mail: shuhaizar@gmail.com

CMOS Power Usage Prediction:

In past research, it is found that the power consumed by a CPU during operation is linear to the CPU clock frequency (Saha and Ravindran, 2012, Lawitzky *et al*., 2008 and Kimm *et al*., 2007). Simulation based power measurement prediction assumes that the CPU power consumption is calculated as the cube of frequency, which do not take into account the processor's idle power requirement (Saha and Ravindran, 2012). Simulated power consumption assumes that power consumption of the CPU is calculated with the equation below:

$P_{active} = S_3 \bullet f^3$, where P is the active power consumption, S_3 is a constant and f is the frequency.

This will lead to a very different simulated power consumption and the actual power consumption of the processor. Failure to take into account the actual power requirement hinders the ability to accurately predict and develop a suitable CPU governor for modern processors. Other studies have also found that the CMOS power consumption could be derived from the following equation:

$P = CV^2 \bullet f$,where P is power consumed, C is capacitance, V is supply voltage and f is the frequency.

Both equations do not consider the idle power requirement which is usually much smaller compared to the active state. During idle state, modern CPU have significantly smaller power requirement compared to during heavy operation. Failure to take this into consideration leads to a very different between predicted and actual measured power consumption. While both equations provide sufficient precision in predicting the active state power consumption, this is usually not the case during for actual operation whereby the processor would be sitting idly while waiting for incoming jobs (Saha and Ravindran, 2012). This is particularly true for an embedded system where the processor would be waiting in idle mode most of the time. By developing a CPU governor tailored only for reduction of power consumption during an active state tends to lead to a less than ideal efficiency in actual system implementation.

CPU Power Measurement Method:

To really understand the effects of DVFS scaling algorithm on actual CPU power consumption, the instantaneous current draw and voltage supply have to be measured during operation. Since the power requirement of the processor varies with clock frequency, the best possible method of measurement has to be done just before the CPU voltage supply (V_{supply}). Such a process require that the CPU power supply rail be bypassed and ran through a meter to measure the current and power. Another method is to measure the supply current through the entire embedded system. This however, measures the entire system power consumption and while it might provide a global insight on the entire system, it is not particularly accurate in measuring actual CPU power since other components will ultimately affect the measurement (Pallipadi & Straikovskiy, 2006).

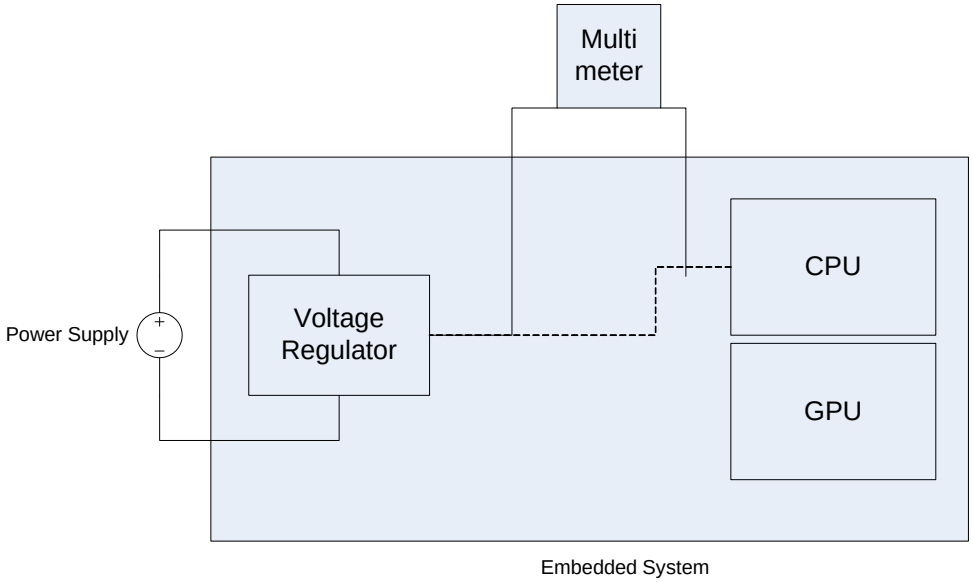

Fig. 1: CPU power measurement setup

To avoid other components power fluctuation from affecting the measurement process, we have taken the measurement on the CPU power supply rail as in Figure 1. This will avoid other components on the system from affecting the data and only provide clean and reliable measurement of the actual CPU power consumption.

A modern CPU developed particularly for embedded systems consisting of the Intel Atom N2800 multicore CPU have been chosen for implementation. The platform consists of Intel Desktop Board DN2800MT have been used. The board follows mini-ITX specification and run off a single DC power supply without any active cooling. The CPU supports 5 different frequency states (0.798, 1.064, 1.33, 1.596 and 1.862 Ghz) which could be manipulated by the *cpufreq* interface. The processor also has 2 different cores which could be manipulated differently from each other.

Idle & Active Power Consumption:

By using the CPU power measurement process as described in Figure 1, it is found that the idle and active power consumption of an embedded processor is significantly different. While power requirement increased linearly to the working frequency during active states, during idle state the actual measured power consumption drops significantly no matter what frequency the processor is idling at.

Table 1: Actual idle & active CPU power consumption

CPU state	cpu frequency	GHz				
		0.798	1.064	1.33	1.596	1.862
idle	voltage (V)	1.128	1.128	1.128	1.128	1.128
	current (A)	0.42	0.43	0.44	0.44	0.45
	power (W)	0.474	0.485	0.496	0.496	0.508
1 core active	voltage (V)	1.133	1.133	1.133	1.133	1.134
	current (A)	0.78	0.8	0.8	0.81	0.87
	power (W)	0.884	0.906	0.906	0.918	0.987
2 core active	voltage (V)	1.138	1.138	1.138	1.138	1.139
	current (A)	1.19	1.21	1.21	1.22	1.28
	power (W)	1.354	1.377	1.377	1.388	1.458

Data from actual measurement shows that during active power state, difference between highest clock frequency (1.862 GHz) and lowest clock frequency (0.798 GHz) is at 0.103W (an increase of 10.44%) with only a single active core and during both core active the difference is 0.104W, an increase of 7.13%. But during idle state, the CPU only consumes 0.508W of power at highest clock and 0.474W at the lowest clock. This measured to be only at 6.69% difference (0.034W). The difference between highest and lowest clock power requirement is larger when more cores are active simultaneously.

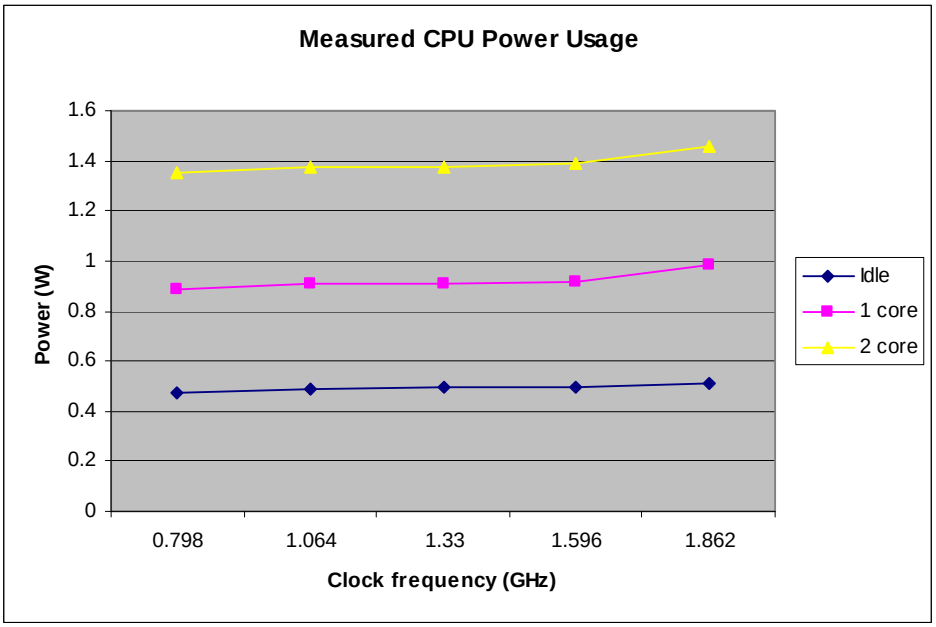

Fig. 2: Actual measured CPU power consumption during idle & active state.

From the graph above we can see that the power usage difference between highest and lowest frequency is much less than previously thought. The difference is significantly less during idle compared to an active state. This allows us to use a more relaxed frequency switching algorithm to reduce rapid frequency switching. By doing so we could avoid rapid frequency transition between low and high frequency which could impair system performance.

Since the power consumption difference between high and low frequency in idle state is much lower compared to power consumed during active state, it makes less sense to implement a strict switching rule to keep the processor speed as slow as possible as currently implemented in the general CPU governor.

Conclusion and Work in Progress:

In this paper we have found that actual CPU power consumption during idle and active state are very much different than previously thought. While there is a difference, during idle state the difference is not significant enough to demand a strict low frequency switching rule. By implementing a more relaxed and moderate switching parameter there is a possibility to reduce rapid frequency transition thus providing higher standby clock and faster run-to-idle for shorter, random tasks.

REFERENCES

Ayoub, R., U. Ogras, E. Gorbatov *et al*, 2011. OS-level Power Minimization Under Tight Performance Constraints in General Purpose Systems, IEEE Computer Society.

Fedora Project Documentation, 2013. Power Management Guide: Using CPUfreq Governors, Fedora Documentation at docs.fedoraproject.org.

Kadin, M. and S. Reda, 2008. Frequency and Voltage Planning for Multi-Core Processors Under Thermal Constraints, IEEE Computer Society.

Kimm, H., S. Shin and C.O. Sung, 2007. Evaluation of Interval-based Dynamic Voltage Scaling Algorithms on Mobile Linux System, In the Proceedings of SAC' 07, Seoul, Korea.

Lawitzky, M.P., D.C. Snowdon and S.M. Petters, 2008. Integrating Real Time and Power Management in a Real System. In the Proceedings of the Workshop on Operating Systems Platforms for Embedded Real-Time applications (OSPERT 2008), Czech Republic.

Pallipadi, V. and A. Straikovskiy, 2006. The Ondemand Governor: Past, Present and Future. In the Proceedings of the Linux Symposium, 2: 223-238.

Pallipadi, V., Enhanced Intel Speedstep Technology and Deman-Based Switching on Linux. Intel Developer Article, Intel.com

Saha, S. and B. Ravindran, 2012. An Experimental Evaluation of Real-Time DVFS Scheduling Algorithms, In the Proceedings of SYSTOR'12, Haifa, Israel.