Importance of the Size of the Period of the Duty Cycle in insertDelay.c for use in DR Control Loop

Joseph Hall

October 18, 2015

Abstract

This report shows the importance of the size of the duty cycle period relative to the size of the power measurement window used for calculating the power. If the period of the duty cycle is on the same order of magnitude as the power window, then the reported power has very large variance due to the fact that the computer will consume relatively large amounts of power during the pulse duration of the cycle. When the window over which we estimate the power is of similar length as the pulse duration, that power estimate will be very large compared to the power estimate during the idle duration.

1 Affect of the Duration of T_{work} on Variance of Power Measurements

Note: T_{work} = `work_interval` in `insertDelay.c`
`insertDelays.c` is program in C written by Josiah which takes an input $u[kh] \in (0, 1)$, where k is the time index and h is the nominal time period between samples. The ouput of `insertDelays.c` are Linux signals, SIGSTOP and SIGCONT, which are sent to a specified process (including all its children and threads).
Power is reported by two methods, `PKG` and `DAQ`:

`PKG[k]` $= \frac{E[kh]-E[kh-h}{h}$, where $E[kh]$ is the value of the energy counter MSR (via the RAPL interface) since the last time is was read and h is the time elapsed since it was last read. The register is cleared when it is read, therefore PKG is reported as an estimate of the average power consummed by the processor cores over the time period h. $n \in \mathbb{Z}$.

`DAQ[k]` $= \frac{1}{n_w} \sum_{n=n_k}^{n_k+n_w} I(nT_s)V(nT_s)$, where T_s is the sample rate, 10kSa/s, of the NI-DAQ, n_w is the number of samples used in the averaging window, and $I(t)$ and $V(t)$ are the AC current and voltage applied to the servers power supply from the outlet. Lastly, $n_k = \mathtt{k}n_w$ and $\mathtt{k}, n \in \mathbb{Z}$. Therefore `DAQ[k]` is reported as the measured average power consummed by the total server over the time period $n_w T_s$.

T_{work} is the time which `insertDelays.c` allows to pass between sending `SIGCONT` and `SIGSTOP` signals. To show why the maximum possible duration of T_{work} is important, let us assume that $n_w T_s = h$ so that `DAQ[k]` and `PKG[k]` measure power over the same time intervals. **NEED PLOT SHOWING IDLE CLOCK CYCLES AND NON-IDLE CYCLES WITH DIFFERENT DUTY-CYCLES AND DIFFERENT Tw RELATIVE TO NwTS AND H**

2 Affect of the Calculation of T_{idle} on Variance of Power Measurements

Note: $T_{idle} =$ `slen` in `insertDelay.c`
`insertDelays.c` is program in C written by Josiah which takes an input $u[kh] \in (0, 1)$, where k is the time index and h is the nominal time period between samples. The ouput of *insertDelays.c* are Linux signals, SIGSTOP and SIGCONT, which are sent to a specified process (including all its children and threads). The value of $u[kh] \in (0, 1)$ can be thought of as a psuedo duty cycle for the process. The actual duty cycle is $\frac{T_{work}}{T_{idle}+T_{work}}$. In *insertDelays.c*, T_{work}, is fixed and can be specified as a commandline argument. $u[kh]$ only changes T_{idle}. Several functions are possible.

$$T_{idle} = (1 - u[kh]) * X, \text{ where } X \text{ is some value} \tag{1}$$

In Josiah's original code, he used Eq. (1), where X was the time elapsed since that particular point in the code loop. This effectively subsummed the time required to update states `insertDelays.c` itself into T_{work} and therefore into T_{idle}. I have found that fixing X as a constant produces power output with less variance at the when the psuedo-duty $\approx 50\%$. I have yet to determine a clear rationale for this difference. A second alternative produces the true duty cycle from $u[kh]$ by Eq. (2):

$$\begin{aligned} D &= u[hk] \\ &= \frac{T_{work}}{T_{work} + T_{idle}} \\ \Rightarrow T_{idle} &= \frac{T_{work} - T_{work}D}{D} \\ \Rightarrow T_{idle} &= T_{work} * (\frac{1}{u[kh]} - 1) \end{aligned} \tag{2}$$

Best seems to be calculating T_{idle} using Eq. (2) with $T_w = 1ms$, because the $argmax_{u[kh]}\{T_{idle}\} = 0.999sec$ and $argmin_{u[kh]}\{T_{idle}\} = 1\mu$s. This is desireable because the maximum T_{idle} is the worst-case response time and response time

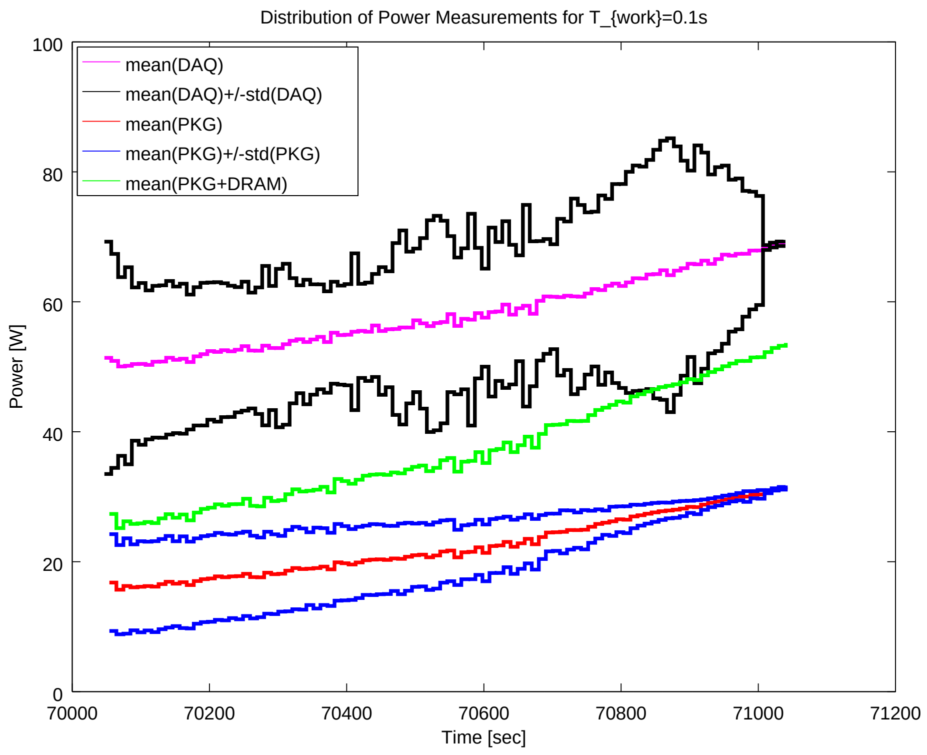

Figure 1: Simulation Results

of $< 1sec$ is acceptable. textbfNote: smaller values of $T_w < 1ms$ could be used, but the `usleep()` function used would have to be changed to `nanosleep()` to accommodate smaller idle times, $T_{idle} < 1\mu$s.

3 Conclusion

Write your conclusion here.