

Application of Variable-Voltage Dynamic Slack Reclamation Using a User-Level Thread Scheduler

1. Introduction

In this phase of the project, we designed and implemented a user-level thread scheduler in order to schedule a hard real-time frame-based periodic task set. We also gave the scheduler the ability to scale the processor voltage and speed to minimize the idle time within the frame while still guaranteeing that the deadlines will be met. We believe that because the user-level scheduler runs at the highest system priority (REAL-TIME 99) and since it is the only active task in the system, we believe this system closely approximates a system-level scheduler.

2. Task Set

The task set contains the following four tasks:

- *ATR frame processor*
The ATR frame processor is transformed into a periodic task by treating both the initialization section and the processing of each frame as a task invocation. The entire procedure is repeated in order to create a continuous series of ATR task invocations. Because the initialization segment has a relatively large execution time and the execution time for the processing of each frame is smaller but highly variable (based on number of detections), this task has runtimes that have a large variance.
- *Random Matrix Multiplier*
This task first generates two randomly sized matrices with dimensions 0 to 200 (with the same inner matrix dimension) and fills them with random double-precision floating-point numbers. Next, the matrices are multiplied using a standard algorithm. This is repeated for every task invocation. Because of the random nature of this task, the runtimes have a large variance.
- *Fast Fourier Transform*
This task utilizes a small FFT library to perform an FFT on arrays of random size. The real and imaginary input arrays are double-precision floating point and have a random size, between 1 and 65536 (using powers of 2). This task is repeated for every task invocation and as such, the runtimes have a high variance.
- *Time Recorder*
This task records the current time and statistical information to a file every 1 minute. Because the task is only written once a minute, the runtimes for this task have a high variance.

3. Task Profiling

Each task was also written as a standalone version that executes 10,000 times and records the longest execution time. This time is the time that is used as the worst-case execution time for each process. The worst-case execution times for each task are listed in the table below.

Task	WCET
ATR	750,000 us
Matrix Multiplication	700,000 us
FFT	55,000 us
Time Recorder	10,000 us

Table 1: Worst-case execution times

To determine the period (frame) length, the worst-case execution times for all tasks were summed and a conservative amount was added to the total. The period length chosen is 1800000 us.

4. Scheduler

The tasks are periodic and have the same periods. Because of this, the scheduler dispatches tasks in a first-in, first-out order. After all tasks have run within a period, the scheduler performs a busy-wait until the next period begins.

5. Power Management

Regardless of whether the scheduler is working in power management mode or not, the scheduler initialization and the scheduler itself run at full speed. Also regardless if power management is enabled, the idle process (busy wait) is run at minimum speed.

When power management is enabled, the scheduler takes advantage of static and dynamic slack to recalculate the processor speed after each task invocation is complete. In order to calculate a new speed for the processor, the following method is used. After each task invocation, the worst-case execution times for the remaining tasks are summed together and divided by the remaining time in the period. This value is used to set the new processor speed using the following manner.

$\frac{\text{remaining execution time}}{\text{remaining time in period}}$	setboost () speed	Clock rate	Clock ratio
< .5000	0	333 MHz	.4543
< .6000	16	400 MHz	.5457

< .7700	50	533 MHz	.7271
< .9300	83	667 MHz	.9100
otherwise	100	733 MHz	1

Table 2: Speed adjustment table

Adjusting the processor speed in this manner will minimize the idle time and still guarantee that the deadlines will be met.

6. Experimental Setup

The setup for each experiment was performed after fully charging the battery. The time is recorded before the experiment is started. The scheduler is started and the power cord is disconnected. Also, the laptop screen is closed so that no power is required to light the backlight (although several times the monitor is activated to check the status of the scheduler. The experiment is run in an identical manner with power management and without management, except for the power management setting. After the laptop's battery completely dies, the last time and statistics from the logfile are analyzed.

7. Experimental Results

	time	% @ 733	% @ 667	% @ 533	% @ 400	% @ 333	% idle	% sched
PM	284m	.0069	3.0348	.0160	1.1193	15.33	79.5674	.9217
w/o PM	232m	13.5329	0	0	0	0	85.7550	.7121

Table 3: Experimental Results

The results show a significant improvement with using power management over not using power management, although more improvement may be witnessed if the frame size was lower.