Optimización del Cálculo del Tiempo Ocioso en Planificadores DVS con Tiempos de Ejecución Variables

José M. Urriza, Javier D. Orozco y Ricardo Cayssials
Departamento de Ingeniería Eléctrica y Computadoras
Universidad Nacional del Sur / CONICET
8000 Bahía Blanca, Argentina
{jurriza, ieorozco, iecayss}@criba.edu.ar

and

Julius C. B. Leite
Instituto de Computação
Universidade Federal Fluminense
Niterói, Brasil
julius@ic.uff.br

Abstract

The consumption reduction of energy in mobile device is today a critical objective of design. This requirement not is limited an adequate realization of hardware, if not that, must be accompanied for a correct administration of available recourse for the software. The tasks scheduling in real time utilizing Dynamic Voltage Scheduling (DVS), permit minimize consume of energy, preserving the temporal restrictions and system functionality. This paper presents a task model for hard real time that permit the analysis and the implementations of methods types *Severs* or *Slack Stealing* with the purpose of permit the utilization the idle times of system by means of Dynamics Voltage Scheduling techniques. The model proposed include a tool that permit preserve the precision in the calculus of idle time even with variable execution times, with the finality the perform optimization of the DVS algorithms.

Keywords: DVS, Slack Stealing

Resumen

La reducción del consumo de energía en dispositivos móviles es hoy un objetivo crucial del diseño. Dicho requerimiento no se limita a una adecuada realización del hardware sino que debe ser acompañada por una correcta administración de los recursos disponibles por parte del software. La planificación de tareas de tiempo real utilizando planificación dinámica de voltaje (DVS), permite minimizar el consumo de energía preservando las restricciones temporales y funcionales del sistema. En este trabajo se presenta un modelo de tarea de tiempo real duro que permite el análisis y la implementación de métodos del tipo *Servidores* o *Slack Stealing* con el fin de permitir la utilización de los tiempos ociosos del sistema mediante técnicas de planificación dinámica de voltaje. El modelo propuesto incluye una herramienta que permite mantener la exactitud en el cálculo del tiempo ocioso aún con tiempos de ejecución variables, a fin de optimizar el desempeño de los algoritmos de DVS.

Palabras Claves: DVS, Slack Stealing.

1 INTRODUCCIÓN

Los requerimientos funcionales sobre los dispositivos móviles se incrementan continuamente. Dicho incremento en la disponibilidad de servicios requiere de la incorporación de dispositivos físicos: interfases de comunicación inalámbrica, pantallas de mayor resolución y más memoria, entre otros. Esto requiere de una capacidad de cómputo que debe crecer proporcionalmente a dicha demanda. En general, en los próximos años, no se espera un aumento sustancial en la capacidad de almacenamiento de energía de las baterías comerciales de pequeño tamaño por lo que, resulta imprescindible, un manejo eficiente de la energía. Actualmente, tanto los investigadores como los fabricantes de microprocesadores (Intel, AMD, Transmeta, ARM) tienen como objetivo, obtener el menor consumo posible en sus productos, conjuntamente con un desempeño acorde a las demandas del mercado.

En términos del producto final, la reducción en el consumo de energía compromete diversos requerimientos que comprenden: la planificación de tareas, la tecnología de baterías, las arquitecturas de microprocesadores, la optimización de código en el proceso de compilación con el objeto de minimizar los requerimientos de memoria y de carga computacional y, por último, optimizar los protocolos de comunicaciones para conservar una adecuada relación de seguridad, velocidad y consumo. Algunos motivos para ahorrar energía son originados en la necesidad de disminuir el tamaño del dispositivo, minimizar la disipación de energía, entre otros.

Muchos de los microprocesadores actuales, implementados sobre la tecnología CMOS incorporan la facilidad de poder cambiar su tensión de alimentación y frecuencia de funcionamiento. Esta característica permite una significativa reducción del consumo al disminuir la frecuencia y tensión de operación a costa de una lógica reducción en la velocidad de cómputo. Es por ello que, para poder explotar dicha propiedad en un sistema de tiempo real, es necesaria una correcta planificación de tareas teniendo como objetivo la preservación de las restricciones temporales y funcionales del sistema.

La utilización de planificadores que exploten las características de frecuencia y tensión variables (DVS, Dinamic Voltaje Scheduling) permite minimizar el consumo de energía preservando integridad del sistema de tiempo real.

El primer trabajo donde se intenta reducir el consumo de energía en microprocesadores mediante la utilización de relaciones voltaje-frecuencia variables fue propuesto por Weiser [1] en 1994 y extendido en [2] por Govil, donde se proponen y analizan diferentes mecanismos predictivos de la demanda a fin de permitir un a reducción del consumo de energía de un sistema genérico y no de tiempo real.

En la actualidad, los métodos de planificación por reducción de voltaje utilizan diversas disciplinas de prioridades para planificar sus tareas de tiempo real. En [3, 4, 5, 6, 7, 8, 9] se utilizan prioridades fijas (*RM* y *DM*), en [6, 9, 10, 11, 12] utilizan prioridades dinámicas (*EDF*) y en [13] prioridades mixtas (*Dual Priority*). En [10] se utiliza un ejecutivo cíclico y en [14, 15, 16] otros mecanismo de planificación estática fuera de línea.

Las técnicas de cálculo del tiempo ocioso del sistema (Slack) para su utilización con alguna finalidad particular (Servidores o Slack Stealing) han sido tratada en la literatura en [17, 18, 19, 20, 21, 22, 23, 24, 25, 26]. El objetivo en el caso que nos ocupa es reducir la velocidad de ejecución de las tareas en la medida en que resulte compatible con los requerimientos temporales del sistema. El tiempo ocioso del sistema a una determinada frecuencia de operación, es un indicador de cuan subutilizado está el procesador y en qué medida podrían reducirse sus prestaciones sin que surjan efectos no deseados.

En este trabajo se propone un modelo de tareas de ejecución determinística bajo *prioridades fijas*. El método propuesto permite recuperar el incremento del *Slack Disponible* (*SD*) producido por la variación del tiempo de ejecución de las tareas por debajo de su estimación de peor caso y para cualquier nivel voltaje-frecuencia en que se encuentre el microprocesador.

En la sección 2 se presenta el modelo del sistema. En la sección 3 se presenta la recuperación del tiempo ocioso en sistema multi-frecuencia. En la sección 4 se exhiben los resultados experimentales y, finalmente, en la sección 5 se exponen las conclusiones.

2 MODELO DEL SISTEMA

Para definir el modelo de sistema se tienen en cuenta las siguientes consideraciones:

- Se considera un microprocesador capaz de operar con m niveles de voltaje-frecuencia siendo $m > 1$ y $m=1$ el de mayor frecuencia y mayor voltaje.
- El sistema de tiempo real es del tipo duro donde no puede perderse ningún vencimiento.
- Un conjunto $\mathbf{\Gamma}(q)$ de q tareas, independientes, periódicas y apropiables. Cada tarea se especifica por el máximo tiempo de ejecución C_i, el período T_i y el vencimiento de la tarea D_i, donde $D_i \leq T_i$.
- El conjunto de q tareas de tiempo real $\mathbf{\Gamma}(q)$ debe ser planificable por lo menos, para la mayor frecuencia del microprocesador adoptado.

El planificador utiliza el tiempo ocioso del sistema a fin de reducir la frecuencia de operación del procesador. Aún cuando los métodos de cálculo de *Slack Stealing* (*SS*) aproximados [27, 28, 29] pueden resultar de utilidad, si logran una buena aproximación a la óptima con un bajo costo computacional, resulta deseable poseer un método de cálculo exacto a fin de tornar utilizable todo el tiempo ocioso del sistema, con el objetivo de reducir los niveles de voltaje-frecuencia de operación del microprocesador y por consiguiente la energía consumida.
En lo que sigue, se utiliza el método *SS* presentado en [18] por ser diseñado para sistemas embebidos y ser un método de cálculo exacto de baja carga computacional desarrollado para ser aplicado en línea.
Tomando este último trabajo como base, el *SD* en un instante t es $S(t) = min\ (S_i(t))$, siendo $S_i(t)$ el *SD* para la tarea τ_i.
Bajo estas condiciones el conjunto de tareas $\mathbf{\Gamma}(q)$ puede resultar planificable para algunas de las frecuencias de operación del microprocesador, pero no necesariamente para las menores, luego, a fin de minimizar la energía consumida el sistema, se ejecutará a la menor frecuencia en la cual el *STR* resulta planificable y se define como f_0.

2.1 Modelo de ejecución de tareas

Resulta evidente que, con fines prácticos, no se puede mantener actualizado en tiempo de ejecución, el valor del *SD* para cada tarea y en cada nivel voltaje-frecuencia del microprocesador. Esto se debe al alto costo computacional que demandaría el cálculo.
A fin de permitir su utilización en línea, se reduce el cálculo del *SD* al obtenido en la frecuencia f_0, donde es posible asegurar que el *STR* puede ser demorado en su ejecución $S(t)$ unidades de tiempo, a partir del instante t y que, por definición, $S_i(t) \geq S(t)$. Con estas condiciones, en lugar de demorar la ejecución de una tarea, podría reducirse la frecuencia de operación del sistema extendiendo así la ejecución de una instancia particular de dicha tarea sin alterar la planificabilidad.
La reducción en la frecuencia extiende el tiempo de ejecución de una tarea τ_i a la nueva frecuencia $f < f_0$:

$$C_i(f) = \frac{f_0}{f} C_i(f_0)$$

donde $C_i(f)$ representa el tiempo de ejecución de la tarea τ_i a la frecuencia f.
Para poder tratar este aumento en el tiempo de ejecución sin perder planificabilidad, se considera al tiempo de ejecución de cada tarea dividido en dos partes: la parte A_i representa el tiempo de ejecución de τ_i a la frecuencia f_0, y la parte B_i representa el aumento en el tiempo de ejecución producido por la reducción de frecuencia.(Ver Figura 1)

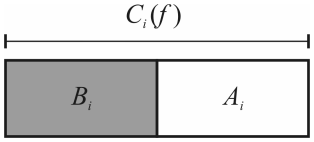

Figura 1. Modelo de Tarea.

luego

$$A_i = C_i(f_0) \qquad B_i = C_i(f_0).\left(\frac{f_0}{f} - 1\right) \tag{1}$$

Lema 1 :

Dado $\mathbf{\Gamma}(q)$ planificable a la frecuencia f_0, se podrá ejecutar una tarea τ_i a una frecuencia f si $B_i \leq S(t)$.

Prueba:

Por definición, el valor de $S(t)$ es el máximo intervalo en que es posible demorar la ejecución del sistema a partir del instante t con lo que la prueba resulta trivial.□

La ejecución de B_i implica la utilización del *SD* del sistema, por lo tanto esta porción del tiempo de ejecución de $C_i(f)$ no puede ser apropiativo, ya que no se puede garantizar que al retomar el control a la tarea τ_i, exista aún suficiente *SD* para proseguir con su ejecución en la nueva frecuencia. Dicho de otra manera, es posible que una tarea de mayor prioridad tome el control y gaste el *SD* para reducir el voltaje-frecuencia de su ejecución. Cuando devuelve el control a la tarea τ_i puede no existir más *SD* para que pueda continuar en su nuevo nivel voltaje-frecuencia. La ejecución de la parte A_i, por el contrario a B_i, no reduce el *SD*, debido a que es parte del tiempo de ejecución original de τ_i en la frecuencia f_0, lo cual significa que puede ser desalojada sin afectar la planificabilidad del sistema.

Como una tarea puede ser interrumpida durante el tiempo A_i por tareas de mayor prioridad, la misma puede fragmentarse en varias instancias hasta completar su ejecución y en principio, cada una de ellas podrá ejecutarse en diferentes frecuencias. Por lo tanto, definimos como Ar_i^n al tiempo restante que falta ejecutar de la tarea i en el nivel n y a A_i^n como la parte que sí se ejecuto en el nivel n y a B_i^n como el aumento en el tiempo de ejecución en el nivel n. En cada instancia debe considerarse como compuesta nuevamente por una parte B_i^n y otra A_i^n calculadas con la última frecuencia a la cual se ejecutó (Figura 2).
Se hace notar que si una tarea se divide en n partes, con la frecuencia original hay n+1 frecuencias.

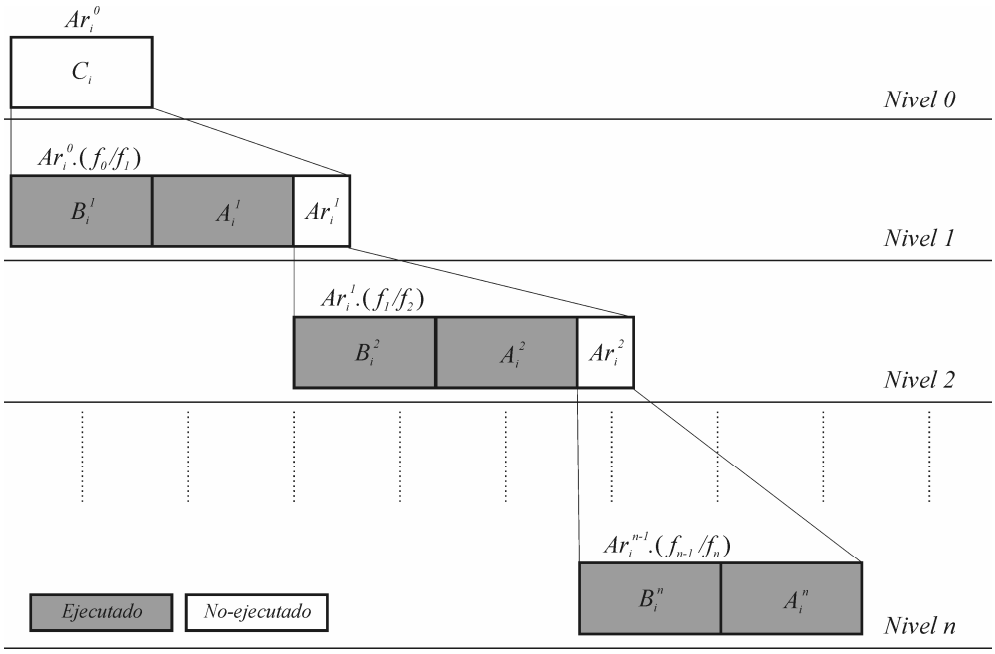

Figura 2. Múltiples fragmentos de la misma tarea en distintos niveles de tensión-frecuencia.

3 RECUPERACIÓN DEL TIEMPO OCIOSO CON TIEMPOS DE EJECUCIÓN VARIABLES EN SISTEMAS MULTI-FRECUENCIA.

Los sistemas de tiempo real duro son parametrizados, por definición, teniendo en cuenta las condiciones temporales de mayor exigencia para el sistema, a fin de garantizar su correcto funcionamiento en éste y en todo otro estado alcanzable. De este modo, los tiempos de ejecución son tomados, para el análisis de planificabilidad, como los de mayor duración. En general, esta situación se presenta ocasionalmente por lo que resulta de interés analizar cómo reducir los efectos producidos por el pesimismo en la utilización de los parámetros de peor estado, en cálculo del tiempo ocioso [17, 19, 20, 21, 22].
El tratamiento de este problema debe extenderse al caso particular de los sistemas que operen en múltiples frecuencias ya que en éstos se produce una fragmentación en los distintos niveles de los tiempos ganados al sistema por reducción en los tiempos de ejecución. Nótese que, según se indicara en el Lema 1, se podrá reducir la frecuencia de operación del sistema si existe suficiente tiempo ocioso disponible con lo que, como el cálculo es pesimista, se habrá sobrestimado el tiempo ocioso necesario para esta operación si la tarea se ejecuta en un tiempo menor que el previsto.
Considérese el siguiente ejemplo: Una tarea con $C_i = Ar_i^0 = 10$ unidades de tiempo a una frecuencia de 1000 es ejecutada inicialmente a una frecuencia de 800 ya que se dispone de suficiente *SD*.
En cada nivel, el cálculo del *SD* necesario para reducir en un nivel el valor de voltaje-frecuencia, se realiza con el peor tiempo de ejecución restante (Ar_i^n).

Nivel	f_n	f_{n-1}/f_n	$Ar_i^{n-1}(f_{n-1}/f_n)$	B_i^n	A_i^n	Ar_i^n
0	1000	1	10	0	0	10
1	800	1000/800	12.5	2.50	2	8
2	600	800/600	10,6667	2.6667	2	6
3	400	600/400	9	3	1	0

Tabla 1.

Como se puede observar en la Tabla 1, para el nivel 1 (800) el tiempo de ejecución se extiende de 10 a 12,5 unidades de tiempo. El *SD* que se necesita para poder pasar de una frecuencia de 1000 a una de 800 es de 2,5. Si luego de 2,5 unidades de tiempo de ejecución de B_i^1 y 2 de A_i^1 se produce la activación de una tarea de mayor prioridad, a τ_i le restará ejecutar 8 unidades de tiempo. Supongamos ahora que, cuando τ_i recupere el control, se dispone de tiempo ocioso suficiente como para permitir una nueva reducción de la frecuencia (600), el nuevo tiempo de ejecución se extiende de 8 a 10,6667 unidades de tiempo a esta frecuencia y el tiempo ocioso necesario es de 2,6667. En este nivel τ_i es desplazada por una tarea de mayor prioridad luego de 4,6667 unidades de tiempo restándole 6 unidades por ejecutar. Retomado el control la ejecución continúa a una nueva frecuencia en el nivel 3, extendiendo el tiempo de ejecución a 9 unidades. Si la ejecución de τ_i finalizara en un tiempo menor al de peor caso, por ejemplo luego de 3 unidades de B_i^2 y 1 de A_i^3, el *SD* utilizado ($\sum B_i^n$) es 8.1667. Las 5 unidades no ejecutadas representan 2 unidades en *Nivel 0*, (5*400/1000 = 2).

Repitamos ahora la planificación anterior considerando que el tiempo de ejecución de τ_i es de 8 unidades. El *SD* necesario para cada nivel es mostrado en la Tabla 2.

Nivel	f_n	f_{n-1}/f_n	$Ar_i^{n-1}(f_{n-1}/f_n)$	B_i^n	A_i^n	Ar_i^n
0	1000	1	8	0	0	8
1	800	1000/800	10	2	2.5	5.5
2	600	800/600	7.333	1.8334	2.8334	2.6667
3	400	600/400	4	1.3334	2.6667	0

Tabla 2.

Como se puede observar en la Tabla 1 y la Tabla 2, si bien el tiempo de ejecución en cada nivel para las dos tareas es el mismo, la suma del *SD* utilizado en cada nivel, es menor en la segunda (5.1667) en 3 unidades de tiempo.

Las 2 unidades de tiempo no utilizadas en la planificación de la Tabla 2 como excedente del tiempo de ejecución, más las 3 unidades de *SD* producto de la diferencia en el cálculo del tiempo ocioso (Tabla 2) por sobrestimación en el tiempo de ejecución de τ_i, deberán recuperarse a fin de permitir la ejecución de las tareas de menor prioridad a menores niveles de voltaje-frecuencia.

A continuación se adopta un microprocesador CMOS, cuyos parámetros de consumo se presenta en la Tabla 3 conjuntamente con los consumos registrados en cada nivel por la planificación del sistema del ejemplo.

Nivel	f_n	Potencia (Watts=W=J/seg)	Tiempo de Ejecución (seg)	Energía Consumida (Joules=J=W.seg)
0	1000	2	10	20
1	800	1.28	4,5	5.76
2	600	0.72	4,6667	3.36
3	400	0.32	4	1.28

Tabla 3.

Si la tarea se ejecuta en el nivel 0, el consumo registrado es de 20J. Si en cambio, su ejecución es dividida en tres niveles como esta descripta en la Tabla 1, se reduce el consumo a 10,4J. Las 3 unidades de tiempo ocioso excedentes a las previstas para el peor caso representan una innecesaria pérdida de energía que se encontrará comprendida entre 0,96J si el excedente es utilizado a una frecuencia f_3, y de hasta 6J si ocurre a la frecuencia f_0.

3.1 Cálculo del tiempo ocioso recuperado.

El *SD* producido por la reducción en los tiempos de ejecución es definido en [17, 21] para sistemas mono-frecuencia como *tiempo ganado*. La diferencia entre el tiempo de ejecución del peor caso y el tiempo de ejecución real se la denomina δ_i, siendo δ_i^n su valor calculado a la frecuencia f_n.

Luego, para las tareas de menor prioridad que τ_i:

$$i < j \le q$$
$$S_j(t) \leftarrow S_j(t) + \delta_i^0 \qquad (2)$$

Lema 2 :

Si τ_i divide su ejecución en n niveles de frecuencia y finaliza su ejecución δ_i^0 unidades antes del peor caso, el tiempo ocioso a recuperar para las tareas $\tau_j, \tau_{j+1}, \ldots, \tau_q$ con $j>i$, resulta:

$$i < j \le q$$

$$S_j(t) \leftarrow S_j(t) + \delta_i^0\left(\frac{f_0}{f_n} - 1\right) \tag{3}$$

Prueba:

Sea:

$\widehat{A_i^n}$ = Tiempo de ejecución real que aconteció de la parte A_i en el nivel *n*.

$\widehat{B_i^n}$ = Tiempo de ejecución real que aconteció de la parte B_i en el nivel *n*.

$\widehat{Ar_i^n}$ = Tiempo restante de ejecución real en el nivel *n*.

$Ar_i^0 = C_i(f_0)$ Peor tiempo de ejecución de la tarea

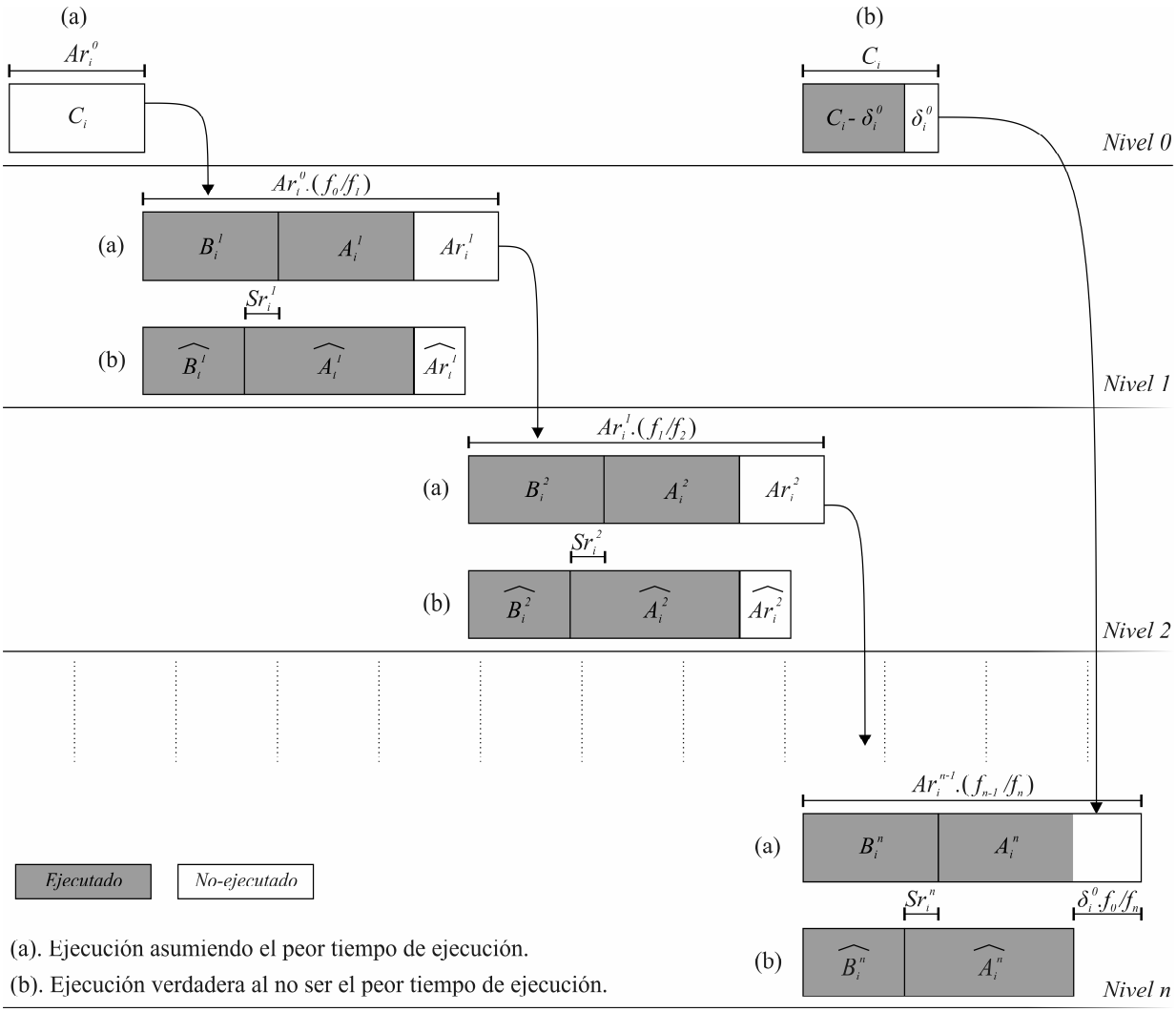

Figura 3. Ejecución con el peor tiempo y el ocurrido realmente, en múltiples fragmentos.

Para comenzar el tiempo restante en cada nivel es:

$$Ar_i^0 = C_i,\ \ Ar_i^1 = Ar_i^0\frac{f_0}{f_1} - A_i^1 - B_i^1,\ \ Ar_i^2 = Ar_i^1\frac{f_1}{f_2} - A_i^2 - B_i^2 \ldots\ldots\ Ar_i^n = Ar_i^{n-1}\frac{f_{n-1}}{f_n} - A_i^n - B_i^n$$

$$\widehat{Ar_i^0} = Ar_i^0 - \delta_i^0,\ \ \widehat{Ar_i^1} = Ar_i^1 - \delta_i^0\frac{f_0}{f_1},\ \ \widehat{Ar_i^2} = Ar_i^2 - \delta_i^0\frac{f_0}{f_2},\ldots\ldots\ \ \widehat{Ar_i^n} = Ar_i^n - \delta_i^0\frac{f_0}{f_n}$$

El tiempo ocioso gastado en cada nivel y el que tendría que haber sido es:

$$B_i^1 = Ar_i^0\left(\frac{f_0}{f_1}-1\right),\ B_i^2 = Ar_i^1\left(\frac{f_1}{f_2}-1\right),\ldots\ldots\ldots B_i^n = Ar_i^{n-1}\left(\frac{f_{n-1}}{f_n}-1\right)$$

$$\widehat{B_i^1} = \widehat{Ar_i^0}\left(\frac{f_0}{f_1}-1\right),\ \widehat{B_i^2} = \widehat{Ar_i^1}\left(\frac{f_1}{f_2}-1\right),\ldots\ldots\ldots \widehat{B_i^n} = \widehat{Ar_i^{n-1}}\left(\frac{f_{n-1}}{f_n}-1\right)$$

Se sabe que la cantidad que se ejecuta en cada nivel es la misma es decir:

$$B_i^1 + A_i^1 = \widehat{B_i^1} + \widehat{A_i^1},\ B_i^2 + A_i^2 = \widehat{B_i^2} + \widehat{A_i^2},\ldots\ldots\ldots\ B_i^n + A_i^n = \widehat{B_i^n} + \widehat{A_i^n}$$

Entonces

$$\widehat{A_i^1} = B_i^1 + A_i^1 - \widehat{B_i^1} = Ar_i^0\left(\frac{f_0}{f_1}-1\right) + A_i^1 - \widehat{Ar_i^0}\left(\frac{f_0}{f_1}-1\right)$$

$$\widehat{A_i^1} = Ar_i^0\left(\frac{f_0}{f_1}-1\right) + A_i^1 - \left(Ar_i^0 - \delta_i^0\right)\left(\frac{f_0}{f_1}-1\right) = A_i^1 + \delta_i^0\left(\frac{f_0}{f_1}-1\right)$$

$$\widehat{A_i^2} = B_i^2 + A_i^2 - \widehat{B_i^2} = Ar_i^1\left(\frac{f_1}{f_2}-1\right) + A_i^2 - \widehat{Ar_i^1}\left(\frac{f_1}{f_2}-1\right)$$

$$\widehat{A_i^2} = Ar_i^1\left(\frac{f_1}{f_2}-1\right) + A_i^2 - \left(Ar_i^1 - \delta_i^0\frac{f_0}{f_1}\right)\left(\frac{f_1}{f_2}-1\right)$$

$$\widehat{A_i^2} = Ar_i^1\frac{f_1}{f_2} - Ar_i^2 + A_i^2 - Ar_i^1\frac{f_1}{f_2} + Ar_i^1 + \delta_i^0\frac{f_0}{f_2} - \delta_i^0\frac{f_0}{f_1}$$

$$\widehat{A_i^2} = A_i^2 + \delta_i^0\frac{f_0}{f_2} - \delta_i^0\frac{f_0}{f_1} = A_i^2 + \delta_i^0\left(\frac{f_0}{f_2} - \frac{f_0}{f_1}\right)$$

…….

$$\widehat{A_i^n} = B_i^n + A_i^n - \widehat{B_i^n} = Ar_i^{n-1}\left(\frac{f_{n-1}}{f_n}-1\right) + A_i^n - \widehat{Ar_i^{n-1}}\left(\frac{f_{n-1}}{f_n}-1\right)$$

$$\widehat{A_i^n} = Ar_i^{n-1}\left(\frac{f_{n-1}}{f_n}-1\right) + A_i^n - \left(Ar_i^{n-1} - \delta_i^0\frac{f_0}{f_{n-1}}\right)\left(\frac{f_{n-1}}{f_n}-1\right)$$

$$\widehat{A_i^n} = Ar_i^{n-1}\frac{f_{n-1}}{f_n} - Ar_i^n + A_i^n - Ar_i^{n-1}\frac{f_{n-1}}{f_n} + Ar_i^n + \delta_i^0\frac{f_0}{f_n} - \delta_i^0\frac{f_0}{f_{n-1}}$$

$$\widehat{A_i^n} = A_i^n + \delta_i^0\frac{f_0}{f_n} - \delta_i^0\frac{f_0}{f_n} = A_i^n + \delta_i^0\left(\frac{f_0}{f_n} - \frac{f_0}{f_{n-1}}\right)$$

Despejando los $\widehat{B_i}$ en función de B_i y δ_i^0

$$\widehat{B_i^1} = B_i^1 - \delta_i^0\left(\frac{f_0}{f_1}-1\right),\ \widehat{B_i^2} = B_i^2 - \delta_i^0\left(\frac{f_0}{f_2} - \frac{f_0}{f_1}\right),\ \ldots\ldots\ \widehat{B_i^n} = B_i^n - \delta_i^0\left(\frac{f_0}{f_n} - \frac{f_0}{f_{n-1}}\right)$$

El *SD* que se debe recuperar es Sr_i, que resulta de la diferencia entre el tiempo ocioso utilizado y el tiempo ocioso que se tendría que haber consumido. Esta diferencia es parte del tiempo de ejecución de los A_i^n como se puede ver en la Figura 3.

$$Sr_i^n = B_i^n - \widehat{B_i^n}$$

$$Sr_i = \sum_{j=1}^{n} Sr_i^j = \sum_{j=1}^{n} B_i^j - \widehat{B_i^j}$$

$$Sr_i = \left(B_i^1 + B_i^2 + \ldots\ldots + B_i^n\right) - \left(\widehat{B_i^1} + \widehat{B_i^2} + \ldots\ldots + \widehat{B_i^n}\right) = \delta_i^0\left(\frac{f_0}{f_1}-1\right) + \delta_i^0\left(\frac{f_0}{f_2} - \frac{f_0}{f_1}\right) + \ldots\ldots + \delta_i^0\left(\frac{f_0}{f_n} - \frac{f_0}{f_{n-1}}\right)$$

$$Sr_i = \delta_i^0\frac{f_0}{f_1} - \delta_i^0 + \delta_i^0\frac{f_0}{f_2} - \delta_i^0\frac{f_0}{f_1} + \ldots\ldots + \delta_i^0\frac{f_0}{f_n} - \delta_i^0\frac{f_0}{f_{n-1}} = \delta_i^0\frac{f_0}{f_n} - \delta_i^0$$

$$Sr_i = \delta_i^0 \left(\frac{f_0}{f_n} - 1 \right) \square \tag{4}$$

Colorario 1:

El tiempo ocioso total recuperado en todos los niveles mayores a i, resulta:

$$i < j \leq q$$

$$S_j(t) \leftarrow S_j(t) + \delta_i^0 \frac{f_0}{f_n} = S_j(t) + \delta_i^n \tag{5}$$

El cálculo del tiempo ocioso recuperado en tiempo de ejecución resulta simple ya que el mismo se obtiene mediante la Ecuación 4 cuando τ_i finaliza su ejecución antes de lo previsto y dicho valor se adiciona al valor precalculado del tiempo ocioso para las tareas de prioridad menor a τ_i. Ello implica que la complejidad del procedimiento es O(q) =q.

4 RESULTADOS EXPERIMENTALES

Si a fin de recuperar el *SD* se realiza el cálculo del mismo para todas las tareas, cada vez que una determinada tarea termina su ejecución, el consumo de energía producido por este procedimiento podría resultar mayor, que el que se pretende ahorrar por recuperación del tiempo ocioso. A solo efecto de evaluar el costo computacional de la utilización de una técnica de tiempo ocioso como la presentada en [18] que recalcule el *SD* para todas las tareas, respecto de la propuesta en el presente trabajo, se realizan los siguientes experimentos.

Se utilizan grupos de 10, 20 y 50 tareas con 500 sistemas factibles por grupo. Los mismos son generados aleatoriamente, con periodos comprendidos entre 25 y 10000 unidades de tiempo. Como el sistema es empleado para aplicaciones con control de voltaje-frecuencia, el mismo operará a la menor frecuencia posible con lo que el factor de utilización rondará el 90%.

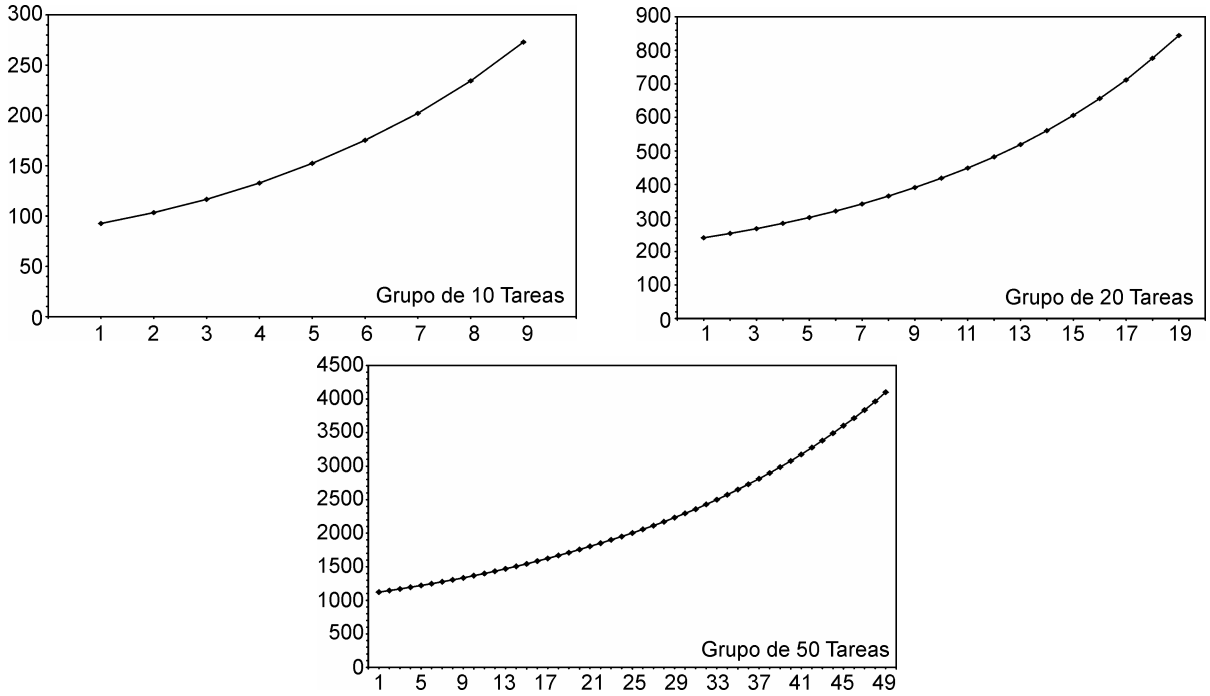

Figura 4. Costo computacional relativo Vs Número de tarea. Para sistemas con 10, 20 y 50 tareas

El método presentado en [18] es iterativo y su complejidad por iteración es del orden de la que posee el cálculo del procedimiento descrito en este trabajo. Debido a ello, en las gráficas se representa como elemento de comparación entre ambos, al costo computacional como el número de iteraciones requerido para calcular *SD* mediante el método iterativo para todas las tareas de menor prioridad a la tarea indicada en las ordenadas. Claramente se observa que el costo de recalcular el SD es de dos órdenes de magnitud superior en los casos mostrados.

5 CONCLUSIONES

La implementación de este método permite un cálculo exacto del tiempo ocioso en sistemas con tiempos de ejecución variables y con un bajo costo computacional. Estas características lo hacen especialmente apto para su aplicación a sistemas de propósito dedicado con DVS. El impacto de aplicar este trabajo a un sistema operativo de tiempo real, que sea capaz de regular el voltaje-frecuencia de operación del microprocesador, permitirá reducir el consumo de energía, como también el costo computacional involucrado en el cálculo del tiempo ocioso.

Para concluir, se hace notar que los saltos de frecuencias en los microprocesadores actuales son discretos, por lo cual pequeños errores en el cálculo del tiempo ocioso pueden tener una relevancia sustancial al determinar sobre qué nivel se podrá ejecutar una tarea o el sistema completo.

AGRADECIMIENTOS

A los revisores anónimos que con sus comentarios ayudaron a mejorar este trabajo.

Referencias

[1] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for Reduced CPU Energy," presented at 1st Symposium on Operating Systems Design and Implementation, Monterey, California, EUA, 1994.

[2] K. Govil, E. Chan, and H. Wasserman, "Comparing Algoritms for Dynamic Speed-Setting of a Low-Power CPU," presented at ACM International Conference on Mobile Computing and Networking, 1995.

[3] B. Novelli, J. C. B. Leite, J. M. Urriza, and J. D. Orozco, "Regulagem Dinâmica de Voltagem em Sistemas de Tempo Real," presented at XXXII Seminário Integrado de Software e Hardware (SBC 2005 SEMISH), Unisinos -Sao Leopoldo, Brazil, 2005.

[4] S. Saewong and R. Rajkumar, "Practical Voltage-Scaling for Fixed-Priority RT-Systems," presented at 9th IEEE Real-Time and Embedded Technology and Applications Symposium, Toronto, Canada, 2003.

[5] Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-time systems," presented at 36th Design Automation Conference, 1999.

[6] Y. Liu and A. K. Mok, "An integrated approach for applying dynamic voltage scaling to hard real-time systems," presented at Real-Time and Embedded Technology and Applications Symposium, The 9th IEEE , 27-30 May, 2003.

[7] J. M. Urriza, B. Novelli, J. C. B. Leite, and O. Javier Dario, "Economia de energia em dispositivos móveis," presented at VI Workshop de Comunicação sem Fio e Computação Móvel, Fortaleza, CE, Brasil, 2004.

[8] L. Bertini and J. C. B. Leite, "Um Breve Survey: Escalonamento em Sistemas de Tempo Real com Otimizacao do Consumo de Energia," presented at VI Workshop de Tempo Real, Gramado, RS, Brasil, 2004.

[9] P. Pillai and K. G. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems," presented at 18th Symposium on Operating Systems Principles, Banff, Alberta, Canada, 2001.

[10] C. M. Krishna and Y. a.-H. Lee, "Voltage-Clock Adaptive Scheduling Techniques for Low Power in Hard Real-Time Systems," presented at Real-Time Technology and Applications Symposium, 2000.

[11] Ala´Qadi, S. Goddard, and S. Farritor, "A dynamic voltage scaling algorithm for sporadic tasks," presented at Real-Time Systems Symposium (RTSS), 24th IEEE , 3-5 Dec, 2003.

[12] W. Kim, J. Kim, and S. L. Min, "A Dynamic Voltage Scaling Algorithm for Dynamic-Priority Hard Real-Time Systems Using Slack Time Analysis," presented at Conference on Design, Automation and Test in Europe, Washington, DC, EUA, 2002.

[13] M. A. Moncusí, A. Arenas, and J. Labarta, "Improving Energy Saving in Hard Real Time Systems via a Modified Dual Priority Scheduling," vol. 29, pp. 19-24, 2001.

[14] F. Yao, A. Demers, and S. Shenker, "A Scheduling Model for Reduced CPU Energy," presented at Foundations of Computer Science, 36th Annual Symposium 1998.

[15] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava, "Power Optimazation of Variable-Voltage Core-Based Systems," IEEE Transactions On Computer- AIDED Design of Integrated Circuits and Systems, vol. 18, pp. 1702-1714, 1999.

[16] T. Ishihara and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors," presented at International Symposium on Low-Power Electronics and Desing, Monterey, California, EUA, 1998.

[17] R. I. Davis, K. W. Tindell, and A. Burns, "Scheduling Slack Time in Fixed-Priority Preemptive Systems," Proceedings of the Real Time System Symposium, pp. 222-231, 1993.

[18] J. M. Urriza, J. D. Orozco, and R. Cayssials, "Fast Slack Stealing methods for Embedded Real Time Systems," presented at 26th IEEE International Real-Time Systems Symposium (RTSS 2005) - Work In Progress Session, Miami, EEUU, 2005.

[19] S. Ramos-Thuel and J. P. Lehoczky, "On-Line Scheduling of Hard Deadline Aperiodic Tasks in Fixed-Priority Systems," presented at Real-Time Systems Symposium, 1993.

[20] S. Ramos-Thuel and J. P. Lehoczky, "Algorithms for Scheduling Hard Aperiodic Tasks in Fixed-Prioriys Systems using Slack Stealing," presented at Real-Time Systems Symposium, 1994.

[21] J. P. Lehoczky and S. Ramos-Thuel, "An Optimal Algorithm for Scheduling Soft-Aperiodic Tasks in Fixed-Priority Preemptive Systems," presented at IEEE Real-Time Systems Symposium, Phoenix, Arizona, EUA, 1992.

[22] T.-S. Tia, J. W. Liu, and M. Shankar, "Aperiodic Request Scheduling in Fixed-Priority Preemptive Systems," Department of Computer Science, University of Illinois, Internal Report UIUCDCS-R-94-1859, 1994.

[23] J. P. Lehoczky, L. Sha, and J. K. Strosnider, "Enhanced Aperiodic Responsiveness in Hard Real-Time Enviroments," presented at IEEE Real-Time Systems Symposium, 1987.

[24] B. Sprunt, J. P. Lehoczky, and L. Sha, "Exploiting Unused Periodic Time For Aperiodic Service Using The Extended Priority Exchange Algorithm," presented at IEEE Real-Time Systems Symposium, Huntsville, Alabama, USA, 1988.

[25] L. Sha, B. Sprunt, and J. P. Lehoczky, "Aperiodic Task Scheduling for Hard Real-Time Systems," The Journal of Real-Time Systems, vol. 1, pp. 27-69, 1989.

[26] G. Fohler, T. Lennvall, and G. Buttazzo, "Inproved Handling of Soft Aperiodic Tasks in Offline Scheduled Real-Time Systems using Total Bandwidth Server," presented at 8th IEEE International Conference on Emerging Technologies & Factory Automation, Nice France, 2001.

[27] R. I. Davis, "Approximate Slack Stealing Algorithms for Fixed Priority Pre-Emptive Systems," Real-Time Systems Research Group, University of York, York, England, Internal Report 1994.

[28] R. I. Davis, "Dual Priority Scheduling: A Means of Providing Flexibility in Hard Real-Time Systems," Department of Computer Science, University of York, York, England, Internal Report 1995.

[29] J. D. Orozco, R. M. Santos, J. Santos, and R. Cayssials, "Taking advantage of priority inversions to improve the processing of non-hard real-time tasks in mixed systems," presented at WIP 21st IEEE Real-Time Systems Symposium, 2000.