Universitat
Autònoma
de Barcelona

4178-1: DEPENDENCIA DE LOS ALGORITMOS CON EL CONSUMO DE ENERGÍA EN SISTEMAS HPC

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Pedro Erencia Millán
i dirigit per
Diego Javier Mostaccio Mancini
Bellaterra, 22 de Junio de 2011

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, Diego Javier Mostaccio Mancini

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en

I per tal que consti firma la present.

Signat: .....

Bellaterra, ...de.....de 200.....

A mi familia.

Agradecimientos

Me gustaría agradecer la ayuda prestada por numerosos miembros y estudiantes del Departamento de Arquitectura y Sistemas Operativos. En primer lugar, a Diego Javier Mostaccio, tutor del proyecto, por sus consejos y ayuda en cuestiones de electricidad y electrónica. También quiero agradecer a Dolores Rexachs, Emilio Luque, Javier y Sandra, con quienes fui compartiendo mis avances y de quienes obtuve las ideas y visión crítica necesarias. A Juan Carlos Moure, por su ayuda y motivación. Finalmente, quiero agradecer también la ayuda de Alejandro Chacón, con quien discutí muchas de las cosas que no entendía y cuyas ideas siempre me resultaron beneficiosas, y a Javier Balladini, por sus observaciones sobre los resultados iniciales.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. El problema del consumo

La población mundial aumenta y cada vez más personas participan en sociedades altamente industrializadas; esto implica demandas de energía crecientes, tanto en el ámbito doméstico como industrial, mientras que los recursos naturales utilizados para generarla son finitos. Por todo esto, y más concretamente por la concienciación de un porcentaje cada vez mayor de la población sobre estos problemas, muchas empresas, especialmente en el sector de las Tecnologías de la Información y Comunicación(TIC), ámbito en el que se enmarca éste trabajo, utilizan el adjetivo *GREEN* para ofrecer una imagen responsable con el medio ambiente. Pero más allá de las cuestiones medioambientales, el consumo energético ha sido una barrera constante a superar en el desarrollo de procesadores cada vez más pequeños, versátiles y potentes, así como un gran problema en costes de operación.

1.2. Consumo en grandes infraestructuras TIC

Se utiliza el término “grandes infraestructuras” para hacer referencia a los complejos de alojamiento de servidores o centros de datos. Según datos de la EPA (Environmental Protection Agency) del 2007, estas infraestructuras consumieron

61 billones KW/h en 2006, lo que corresponde al 1.5 % del consumo eléctrico total de EEUU y equivale a 4.5 billones de dólares [APM].

Esta tendencia supone un grave problema de costes, necesitando además de la construcción de nuevas y poderosas plantas de energía. Por todo esto, grandes compañías como eBay, Yahoo o Google, han comenzado a construir sus nuevos centros de datos en lugares donde el coste energético sea lo más reducido posible; por ejemplo en el caso de Google, en el río Columbia en Washington, firmando acuerdos con los gobiernos locales para facilitar la construcción de las plantas generadoras necesarias[APM].

1.3. Consumo en HPC

1.3.1. Qué es HPC?

A grandes rasgos, la computación de altas prestaciones (HPC) consiste en la utilización de supercomputadores para la resolución de problemas que necesitan de una gran cantidad de cómputo. Estos problemas están relacionados por lo general con la simulación de procesos físicos, por ejemplo, la predicción del tiempo, la simulación de dinámica de fluidos, bioinformática (plegado de proteínas, alineamiento de secuencias de genes), etc... La posibilidad de realizar tales simulaciones en un tiempo razonable se ha convertido en una herramienta de gran utilidad científica.

Como ejemplo de uno de los muchos supercomputadores existentes puede citarse el Jaguar, situado en el Oak Ridge National Laboratory, Tennessee [ORNL]. Este supercomputador, en concreto el modelo XT-5, ocupa el segundo puesto en el top 500 [TOP500], lista que ordena los supercomputadores en función de la capacidad demostrada en los benchmarks Linpack. El XT-5 consta de 37.376 procesadores AMD Opteron de seis núcleos y 224.256 procesadores AMD Opteron, siendo capaz de alcanzar un pico de cómputo 2,332 teraflops. Las actividades a las que se dedica se muestran en la tabla 1.1.

Categoría científica	**Área de investigación**
Biología	Biología
Química	Química
Ciencias de la tierra	Seismología Clima Geociencias
Ingeniería	Combustión Dinámica de fluidos
Fusión	Fusion
Materiales	Materiales Nanociencia Materia condensada
Energía nuclear	Energía Nuclear
Física	Dinámica molecular Astrofísica Física nuclear

Cuadro 1.1: Areas de investigación del supercomputador Jaguar.

1.3.2. Consumo General

En lo que respecta al consumo, el XT5 tiene un consumo estimado de 5 MW/h, de los cuales un 30 % correspondería a la refrigeración. . Esto sitúa al supercomputador, a Noviembre de 2010, en la posición número 88 del GREEN500 con una eficiencia de 256.55 MFLOPS/W. El GREEN500 es una lista complementaria al TOP500 que ordena los supercomputadores en función de su eficiencia energética. Curiosamente, como se mostró, el XT-5, ocupaba el segundo puesto en el TOP500. El primer puesto del GREEN500 lo ocupa el Blue Gene/Q, con 1684.20 MFLOPS/W, situado en la posición 115 del TOP500. Una muestra del interés en la reducción de consumo es que el último supercomputador desplegado en el ORNL, un modelo Cray XT-6, ocupa la posición 32 en el TOP500 y la 43 en el GREEN500. En cuanto a los costes operativos, asumiendo un precio de 0.10 dólares por kW/h y un gasto de 5 MW mensuales, estos ascenderían a 0.5 M$ mensuales, unos 6 M$ anuales. [OPCOST]. Aunque el precio parece significativo, es difícil valorar su magnitud en el conjunto del gasto científico de un gobierno

como el de EEUU. Sin embargo, lo más alarmante no sea probablemente el gasto actual o los costes operativos, sino la tendencia que se adivina, tal como se muestra en la siguiente figura [HPCORNL]

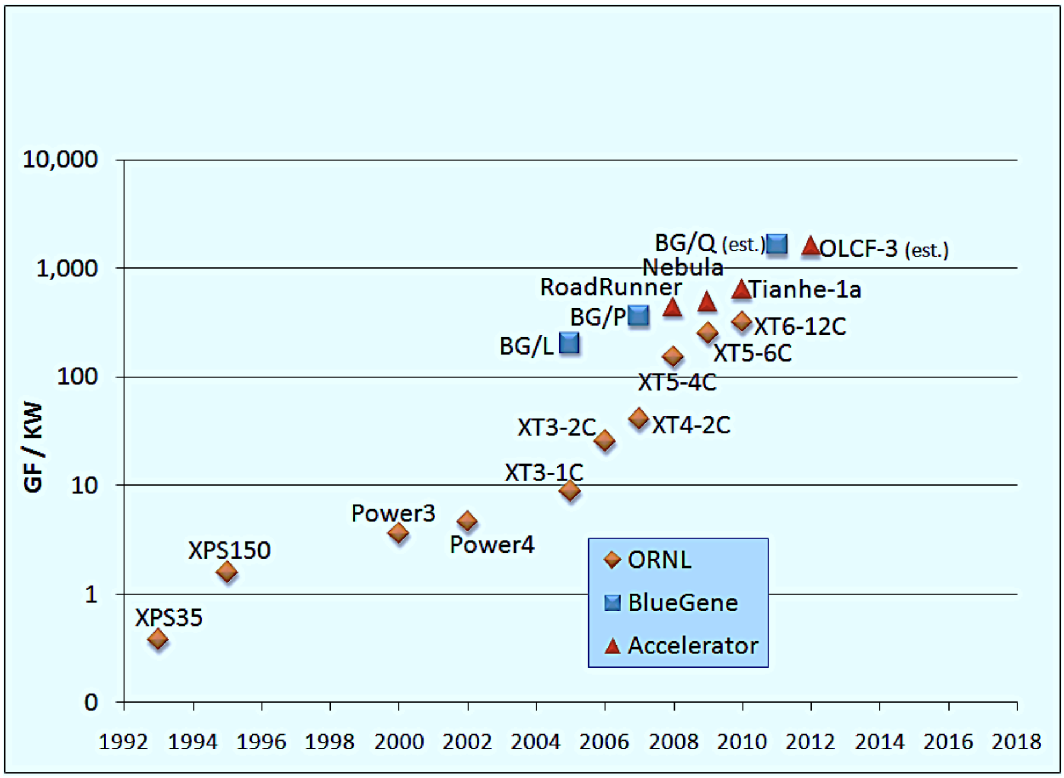


Figura 1.1: Evolución del consumo de supercomputadores.

Según el autor de la misma, con la tecnología actual y de seguir la tendencia, un supercomputador capaz de proporcionar un ExaFlop de cómputo ($10^{18} Flops$), necesitaría del orden de 50 a 100 MW en 2018 [HPCORNL].

1.3.3. Consumo útil y aplicaciones

Si se observa a una escala menor, dentro de una cabina del supercomputador, tenemos la distribución del consumo que se muestra en la figura 1.2 [HPCORNL],

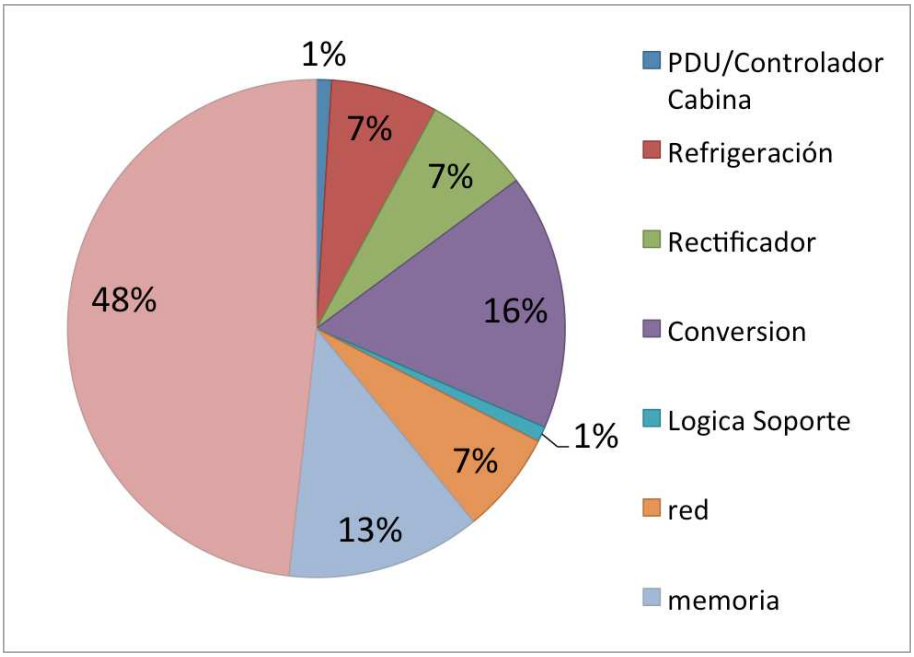


Figura 1.2: Distribución del consumo en el XT5.

Como puede apreciarse, únicamente el 70 % de la energía consumida se utiliza para los elementos de cómputo y comunicaciones, mientras que el 30 % restante se gasta en la infraestructura eléctrica (fuente de alimentación y refrigeración, principalmente).

Según [SUR], una aplicación típica de computo de altas prestaciones involucra gran cantidad de nodos coordinados, donde comunicación y sincronización se solapan, con tiempos relativamente largos. Estas características son coherentes con la figura 1.2, donde se aprecia como la cpu se lleva la mayor parte del consumo. Quizás aquello realmente interesante de este desglose es ver cómo el desarrollador de la aplicación, con un uso razonado y consciente de los recursos de memoria, cpu y comunicaciones, puede influir sobre ese 70 % de energía útil. En el capitulo 5 se abordará con mayor detalle el tema de las aplicaciones típicas de HPC con las que se experimentará en este trabajo.

1.4. Objetivo del proyecto

Este trabajo final de carrera pretende ser una introducción en el ámbito del estudio del consumo energético en procesadores. Mediante el análisis y experimentación con una de las técnicas más utilizadas para la gestión de dicho consumo, el escalado dinámico de voltaje y frecuencia, se pretende observar la eficacia de la misma en la ejecución de programas con diferentes cargas de trabajo típicas de HPC, esto es, intensivas en cómputo o memoria. Especialmente en el caso de las últimas, existen numerosos artículos que verifican la existencia de un compromiso entre consumo y rendimiento que produce grandes ahorros en energía con un incremento muy reducido del tiempo de ejecución. Se pretende probar de primera mano dicha hipótesis y, dado que la mayoría de procesadores actualmente son multinúcleo, extender la experimentación a éstas nuevas arquitecturas, para comprobar en que medida afectan en la aplicación de dicha técnica. La correcta utilización de dichas técnicas, especialmente en equipos dedicados a la computación de altas prestaciones (por su elevado consumo energético, tal y como se comentó previamente), puede derivar en un uso más eficiente de la energía, con los consiguientes beneficios tanto económicos como medio-ambientales.

1.5. Planificación

En la figura 1.3 se muestra la planificación inicial.

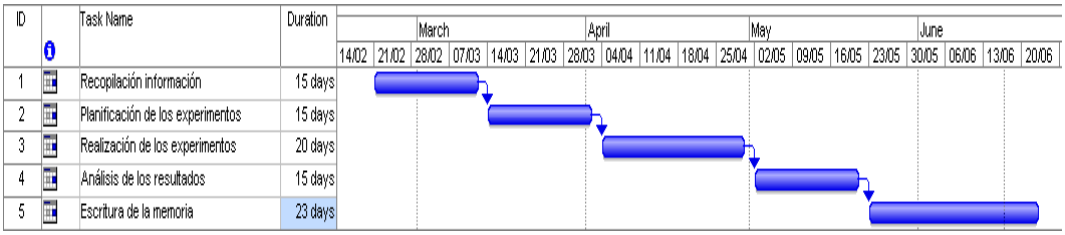


Figura 1.3: Planificación del proyecto.

El objetivo de este proyecto no es el desarrollo de una aplicación sino la realización de un trabajo de investigación introductorio. No hay por lo tanto análisis de

requerimientos ni diseño, aunque en cierto modo éstas etapas son sustituidas por la recopilación de la información y la planificación de los experimentos. Tampoco hay fase de test, sino de análisis de los resultados obtenidos.

1.6. Organización del trabajo

El resto de la memoria se organiza como sigue. En el capítulo 2 se introducirán las definiciones físicas pertinentes (trabajo, potencia y energía), primero de forma general y posteriormente en el contexto eléctrico. En el capítulo 3, se repasarán los aspectos básicos de la tecnología CMOS en relación al consumo. En el capítulo 4 se revisarán las técnicas aplicadas para la gestión del consumo por la industria, detallando la técnica principal con la que experimentaremos en este trabajo. En el capítulo 5, se explicarán los programas utilizados para la realización de los experimentos, así como la organización de los mismos. En el capítulo 6 se realizará una descripción detallada de los instrumentos y equipos utilizados. En los capítulos 7 y 8, se mostrarán y analizarán, respectivamente, los resultados de los experimentos, intentando extraer aquellos elementos comunes que definan y permitan agrupar los diferentes comportamientos exhibidos. Finalmente, en el capítulo 9, se realizarán las conclusiones.

Capítulo 2

Principios físicos

2.1. Conceptos generales.

2.1.1. Trabajo

Se realiza trabajo, en términos físicos, cuando se aplica una fuerza sobre un cuerpo, produciendo un movimiento. Por lo tanto, para su cálculo se consideran dos magnitudes, 1) la fuerza aplicada y 2) la distancia recorrida en dicho movimiento. El producto de ambas es el trabajo que se ha realizado.

$$W = F \times s \tag{2.1}$$

donde W es trabajo, F la fuerza aplicada y s la distancia recorrida.

La unidad de medida del trabajo es el Joule, que equivale a una fuerza de un Newton actuando a través de un metro. [FFUND]

2.1.2. Potencia

Como se ha visto el trabajo resulta de la aplicación de una fuerza y de un movimiento derivado de la misma. Sin embargo, el movimiento puede realizarse en más o menos tiempo. La potencia mide esta nueva variable, indicando la rapidez con la que se ejecuta un trabajo.

$$P = \frac{W}{t} \tag{2.2}$$

donde P es la potencia, W el trabajo y t el tiempo. La unidad de medida de la potencia es el Watt, que equivale a un Julio por segundo.[FFUND]

2.1.3. Energía

Finalmente, se define la energía como la capacidad para realizar un determinado trabajo, expresada también en Joules. La cantidad de energía es constante en un sistema y puede tomar diversas formas, en función del trabajo realizado por los agentes del mismo. Por ejemplo, el agua de una presa hidroeléctrica tiene una cierta capacidad de trabajo para mover las turbinas.

2.2. Potencia eléctrica

Hasta ahora se definió la potencia como la rapidez con la que se ejecuta un trabajo. En el contexto eléctrico, esto equivaldría a la rapidez con la que se desplaza una carga eléctrica Q desde un punto a otro de un campo eléctrico con una diferencia de potencial U.

Para un circuito como el de la figura 2.2 la potencia viene dada por la siguiente fórmula,

$$P = V \cdot I \tag{2.3}$$

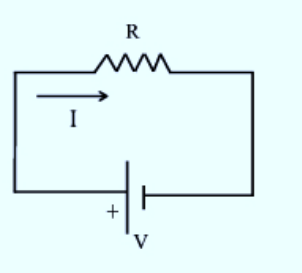


Figura 2.1: Circuito de tensión continua.

En el caso de corriente alterna hay que tener en cuenta el tipo de circuito con el que se trabaja. A continuación se examinarán los tres tipos básicos ideales.

2.2.1. Circuito con resistencia pura

El comportamiento es similar al de corriente continua, solo que se aplican los valores eficaces de corriente y tensión, que viene dados por la siguiente formula,

$$V_{ef} = \sqrt{\frac{1}{T}\int_{to}^{to+T} v^2(t)dt} \tag{2.4}$$

Esta potencia es aquella que se transforma de manera efectiva en calor o trabajo, por lo que se denomina **Potencia activa** y se expresa en **Watts(W)**.

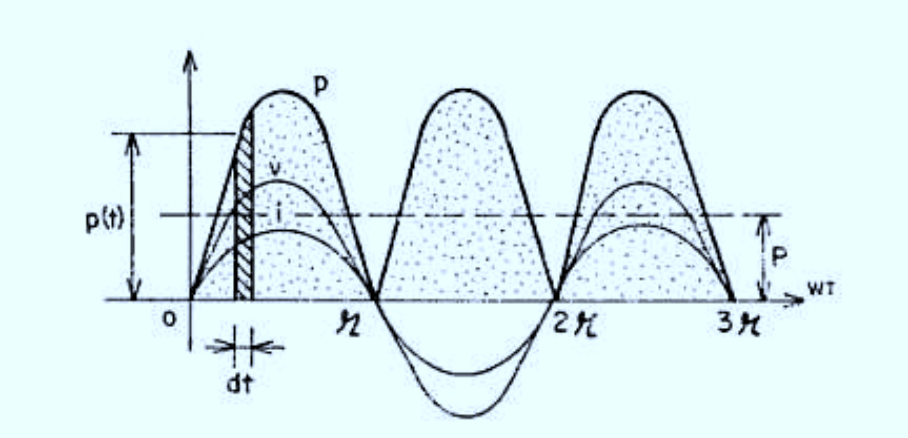


Figura 2.2: Potencia en un circuito con resistencia pura.

2.2.2. Circuito con bobina pura

Al contrario de lo que ocurre en una resistencia, en una bobina pura no se produce ningún consumo de energía calorífica. La corriente que recorre la bobina sirve únicamente para generar el campo magnético. Sin embargo, aunque la bobina no consuma energía real para su funcionamiento, las constantes cargas y descargas de la misma hacen que circule una determinada corriente por los conductores, y, por tanto, aparece una potencia que fluctúa entre los mismos. A esta potencia se la denomina **potencia reactiva** y se expresa generalmente en **voltiamperios reactivos(vars)**[CELC]

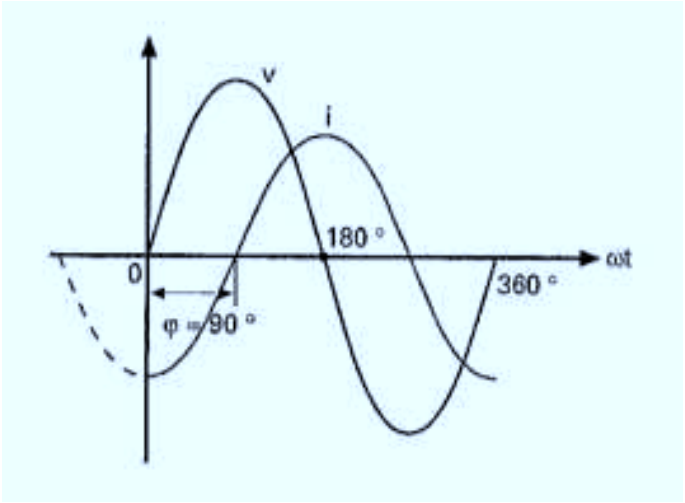


Figura 2.3: Relación entre tensión e intensidad en una bobina pura.

2.2.3. Circuito con condensador puro

En un condensador tampoco se produce ningún consumo de energía calorífica, simplemente toma la energía prestada en un cuarto de ciclo para devolverla en el siguiente cuarto de ciclo. Por ésta razón la potencia media consumida es también cero. Del mismo modo, aquí también aparece una potencia reactiva producida por la energía que se intercambia entre el condensador y el generador[CELC]

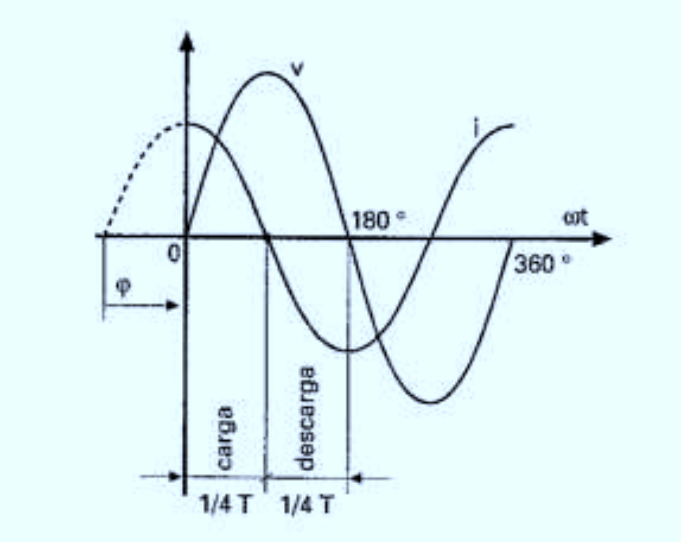


Figura 2.4: Relación entre tensión e intensidad en un condensador.

2.2.4. Relación entre las potencias activa, reactiva y aparente.

Hasta ahora se han examinado los tres tipos de circuito básicos. La pregunta que se formula es, en un circuito combinado ¿ como se mide la potencia activa final, aquella que realmente se consume y que genera trabajo ?

Como se vio existen dos tipos de potencia, la activa y la reactiva. Se puede por tanto considerar su suma vectorial, dando lugar a la que se denomina **potencia aparente**, como se puede ver en la figura 2.5

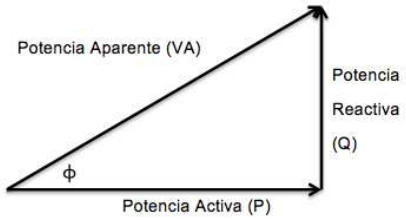


Figura 2.5: Relación entre potencias activa, reactiva y aparente.

donde el ángulo ϕ corresponde al desfase entre la tensión y la intensidad deri-

vado de los diferentes elementos del circuito. Por ejemplo, en el caso de la bobina la intensidad fluye con un retraso de 90º respecto a la tensión (ver figura 2.3) mientras que en el caso del condensador es la tensión que mantiene un retraso de 90º respecto a la intensidad (ver figura 2.4)

Se podrían resumir las potencias descritas del siguiente modo,

Potencia activa

Es la potencia útil, aquella que se transforma de manera efectiva en calor o trabajo. Se expresa en W, donde 1 W es igual a 1 julio por segundo. Su definición sería la siguiente,

$$P = V \cdot I \cdot \cos\phi \tag{2.5}$$

A la expresión $\cos\phi$ se la conoce como **factor de potencia (f.d.p)** e indica la relación entre las potencias aparente y activa. Éste es un factor importante típico de las fuente de alimentación, e indicaría cuanta energía ha de entregar la red eléctrica para que la fuente pueda satisfacer las demandas de energía de los componentes internos.

Potencia reactiva

Es la derivada de los elementos inductivos (bobinas) y capacitivos (condensadores). Se mide en voltamperios reactivos y su formula sería la siguiente,

$$Q = I \cdot V \cdot \sin\phi \tag{2.6}$$

Potencia aparente

Es la potencia total que transportan los conductores que alimentan al circuito.

2.3. Relación entre Energía y Potencia eléctrica

Si la potencia eléctrica nos dice cual es la energía entregada por unidad de tiempo, la energía consumida durante un intervalo de tiempo será igual al área bajo la curva que dibuja la potencia, esto es, su integral.

$$E_{t0-t1} = \int_{t0}^{t1} p(t)\,\mathrm{d}t. \tag{2.7}$$

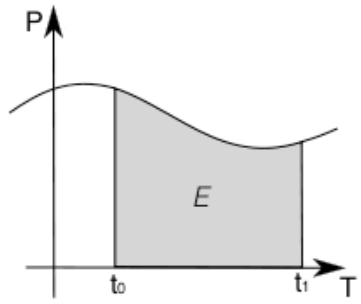


Figura 2.6: Energía en función de la potencia y el tiempo.

Capítulo 3

Consumo en CMOS

3.1. Qué es CMOS

Complementary metal-oxide-semiconductor (CMOS) es la tecnología dominante en la fabricación de circuitos integrados (microprocesadores, memorias y circuitos de aplicación específica (ASIC)). La base de los circuitos CMOS son los metal–oxide–semiconductor field-effect transistor (MOSFET), cuya unión forma las puertas lógicas [CMOS]. Estos transistores se dividen en dos tipos, p y n.

3.1.1. Transistor de tipo n

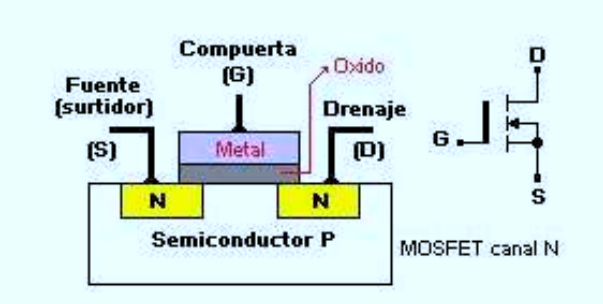


Figura 3.1: Estructura de un transistor de tipo N.

Como se muestra en la figura 3.1[MOSFET], este transistor se compone de

una fuente y un drenaje, cargados negativamente, y un sustrato, formado por un semiconductor cargado positivamente. Al mismo tiempo, una capa de Oxido actúa como aislante.

Cuando se le aplica una tensión positiva a la compuerta (G), los electrones son atraídos desde el sustrato hacia la misma, formando un canal por el que circulan la corriente desde drenaje hasta la fuente, como se muestra en la figura 3.2 [MOSFET].

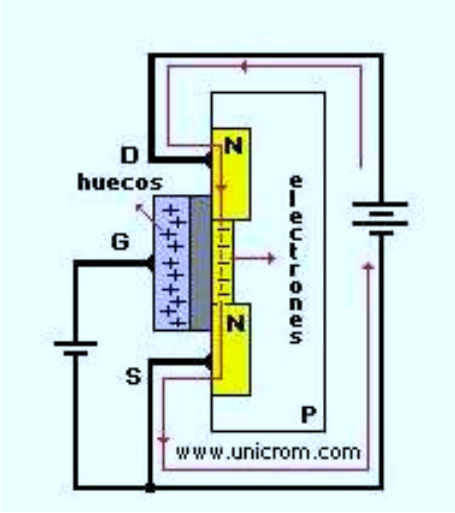


Figura 3.2: Funcionamiento de un transistor N.

3.1.2. Transistor de tipo P

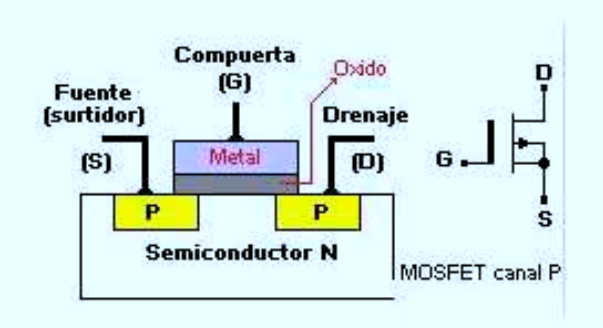


Figura 3.3: Estructura de un transistor de tipo P.

El funcionamiento es el mismo, pero con cargas invertidas. Es decir, drenaje y fuente están cargados ahora positivamente y el sustrato, negativamente. Por lo tanto, al aplicar una tensión negativa, los electrones del sustrato son repelidos, formando un canal cargado positivamente por el que la corriente fluye desde la fuente hasta el drenaje, como puede apreciarse en la figura 3.4[MOSFET].

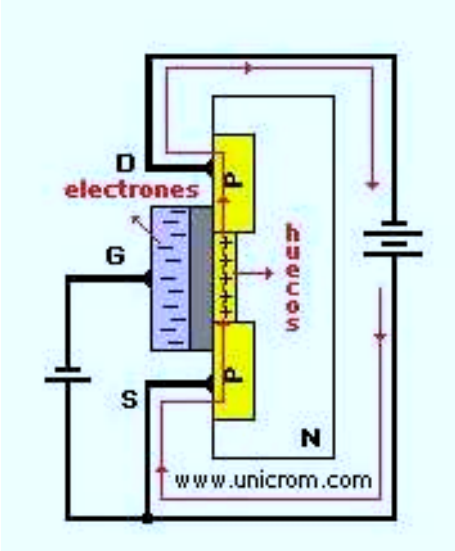


Figura 3.4: Funcionamiento de un transistor P.

3.2. Consumo en CMOS

Hay tres fuentes principales de consumo en la tecnología CMOS, representadas en la siguiente ecuación[CMOSPOW]

$$P_{avg} = P_{leakage} + P_{switching} + P_{short\_circuit} \tag{3.1}$$

donde $P_{leakage}$ es el consumo estático, $P_{switching}$ el consumo dinámico y $P_{short\_circuit}$ el consumo debido al corto-circuito que se da durante el cambio de estado. A continuación se detallarán cada uno de ellos.

3.2.1. Consumo Estático

Una ventaja de la tecnología CMOS, frente a alternativas como NMOS o transistor-transistor logic (TTL), y que ha contribuido en gran medida a su adopción, es un reducido consumo estático. Mientras que en un circuito NMOS, por ejemplo, se disipa energía incluso cuando el circuito se encuentra en estado estacionario, en un circuito CMOS la mayor disipación se da cuando el circuito alterna entre estados, de 0 a 1 y de 1 a 0, siendo el consumo estático prácticamente nulo. Esto es debido a como se conectan los transistores de tipo p y n. En la siguiente figura se muestra la puerta lógica NOT implementada con un transistor de tipo p y otro de tipo n.

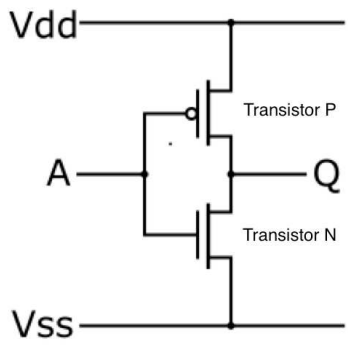
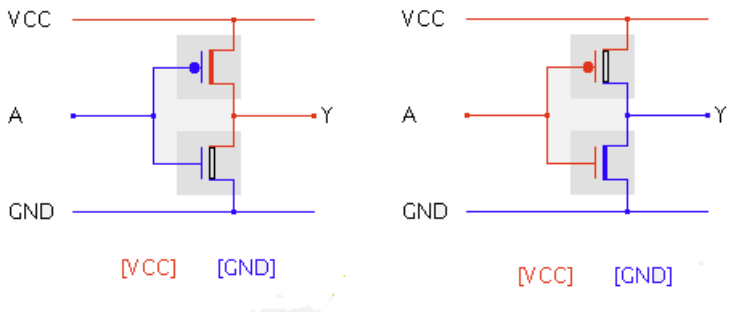


Figura 3.5: Puerta lógica NOT en CMOS.

Cuando la entrada A se conecta a la tensión positiva o alta, el MOSFET de tipo n conducirá corriente mientras que el de tipo p no. Cuando la entrada es baja, el comportamiento será el inverso. Este comportamiento se ilustra en la figura 3.6, con VCC como tensión alta y GND como tensión baja [CMOSDEMO]



(a) Inversor con entrada baja (b) Inversor con entrada alta

Figura 3.6: Inversor CMOS en funcionamiento

Mientras la entrada se mantenga, bien a alta o a baja, no habrá conexión entre VCC y GND, y por lo tanto no circulará corriente. Esto es, en estado estacionario el consumo será prácticamente nulo.

Sin embargo, hay que hacer notar que dicho consumo estático, aunque siga siendo una ventaja evidente de la tecnología CMOS y se haya calificado como prácticamente nulo, no es despreciable. De hecho, ha cobrado cada vez una mayor importancia, debido a la creciente densidad de transistores presente en los circuitos integrados. Actualmente, representa un 20 % del consumo en diseños actuales y se espera que crezca [CATPE].

El más relevante de este tipo de consumo es el de “sub-threshold”, consistente en la energía disipada por un transistor, entre la compuerta y la fuente, cuando éste se encuentra “off”. Aunque se trate a los transistores como dispositivos ideales, que se encuentran en estado apagado o encendido, su comportamiento real incluye corrientes parasitarias, tal y como puede apreciarse en la figura 3.7 [CATPE]. Por debajo del umbral (Vthreshold) donde se consideraría el transistor como “apagado”, hay un consumo de energía, mínimo quizás, pero la acumulación del cual en circuitos con millones de transistores puede derivar en una cantidad considerable.

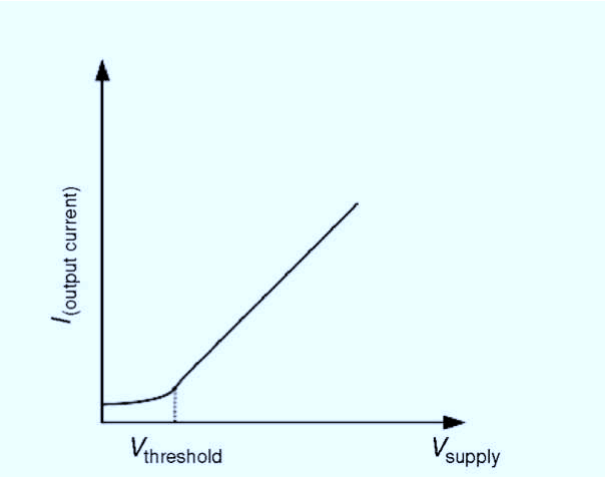


Figura 3.7: Ejemplo de una curva IV de un diodo semiconductor.

3.2.2. Consumo Dinámico

Es el consumo debido al cambio de estado de los transistores. Ha sido el dominante durante muchos años y se define mediante la siguiente expresión CMOS-POW,

$$P = \alpha_{0->1} \cdot C_L \cdot V_{dd}^2 \cdot f_{clk} \tag{3.2}$$

El cambio de estado y el correspondiente consumo acontece cuando cuando el transistor p realiza el cambio de baja a alta tensión (GND a VCC). Para que esto suceda es necesario que se almacena la carga capacitiva del circuito (C_L), tal y como sucedería en un condensador, para producir el canal que posibilite la circulación de la corriente. En un circuito inversor como el de la figura 3.5, la energía que se requiere de la fuente para realizar dicha carga es de $C_L \cdot V_{dd}^2$, la mitad de la cual es disipada por el transistor. Posteriormente, en el paso de 1 a 0, la energía almacenada en el condensador ($1/2C_L \cdot V_{dd}^2$) es disipada por el transistor n. Si dichas transiciones suceden a una determinada frecuencia, la energía consumida será la de la ecuación 3.2. Sin embargo, no todos las puertas de un circuito cambian de estado con la misma frecuencia. Ese es el factor α, que determina que porción de los ciclos para una determinada frecuencia, de media,

contienen transiciones que consumen energía (de 0 a 1) [CMOSPOW].

En un circuito integrado como el de un microprocesador, la carga capacitiva (C_L) no depende únicamente de los transistores, también está directamente relacionada con la longitud de las pistas que los conectan. Una estrategia empleada para la reducción de dicha carga es la división de un núcleo monolítico en núcleos más pequeños, lo que produciría también conexiones más pequeñas[CATPE]. Respecto al factor de actividad (α), su reducción puede conseguirse mediante el uso de una señal de control combinada en una puerta AND con la señal de reloj que llega a la unidad que se desea controlar, por ejemplo una ALU; de esta forma, si la ALU no va a ser utilizada durante un ciclo o más, la señal de control se puede desactivar, evitando la activación de la ALU innecesariamente[CATPE].

3.2.3. Consumo de Corto-Circuito

Durante el tiempo en el que se produce el cambio de estado, hay un breve periodo en el que tanto el transistor p como el n se encuentran abiertos, abriendo una ruta por donde fluye la corriente desde Vdd hasta tierra. Este hecho se da cuando se cumple la siguiente condición [CMOSPOW],

$$V_{Tn} < V_{in} < V_{dd} - |V_{Tp}| \tag{3.3}$$

donde V_{Tn} y V_{Tp} son los umbrales de los transistores n y p respectivamente.

Capítulo 4

Técnicas de control del consumo

Como se ha visto en la introducción, la preocupación por un uso eficiente de la energía ha ido ganando importancia con los años, por lo que la industria se ha visto obligada a desarrollar una serie de estándares que permitan adecuar el consumo energético al uso que se está realizando del dispositivo. El más relevante de éstos es ACPI (Advanced Configuration and Power Interface), cuyo objetivo es permitir el desarrollo de las tecnologías de control del consumo independientemente de sistemas operativos y hardware[ACPIW]

Esencialmente, ACPI define una serie de estados de rendimiento y consumo. Hay cuatro estados globales, etiquetados desde G0 hasta G3, siendo G0 el estado completamente operacional y G3 el de apagado mecánico. Estos estados son inmediatamente percibidos por el usuario ya que afectan al sistema como un todo. Dentro del estado G0 hay cuatro estados de cpu (C0-C3) y dentro del C0 hay hasta 16 estados de rendimiento (P0-P15)[URL]. Éstos últimos son los que permiten modificar el voltaje y la frecuencia para adecuar al consumo de la cpu en respuesta a cargas concretas de trabajo y los que conforman la base de la técnica con la que se experimentará en éste trabajo.

4.1. Escalado dinámico de frecuencia y voltaje

Como comprobamos en (3.2)

$$P = \alpha_{0->1} \cdot C_L \cdot V_{dd}^2 \cdot f_{clk}$$

existe una relación cuadrática entre el voltaje y la potencia. Si se puede reducir el voltaje en un porcentaje pequeño se podrá reducir la potencia por el cuadrado de ese factor. Sin embargo, reducir el voltaje suministrado puede reducir el rendimiento del sistema; en particular, tal reducción a menudo ralentiza los transistores de modo que es necesario reducir la frecuencia. Todo lo comentado permite una potencial reducción cúbica del consumo.[CATPE]

4.2. Implementación

Tanto Intel como AMD ofrecen el escalado dinámico de frecuencia y voltaje en sus procesadores. En el caso de Intel, el nombre comercial es SpeedStep. Su primera versión se remonta a algunos modelos de Pentium III[ST]. Actualmente tiene por nombre Enhanced Intel Speed Step (EIST). Por su parte, AMD ofrece dos versiones, Cool n'Quiet, para servidores y equipos de escritorio, y PowerNow!, para dispositivos móviles.

4.3. Funcionamiento

En la figura 4.1 se muestra un esquema con los componentes básicos que intervienen en la comunicación entre cpu y memoria principal, El North Bridge (NB), también denominado Memory Controller Hub (MCH) se encarga de buscar en la memoria principal los datos pedidos por la cpu y dejarlos en el bus de datos, donde ésta los recogerá para utilizarlos. Toda la comunicación entre el NB y la cpu se realiza por el Front Side Buffer (FSB), el cual toma su frecuencia del reloj del sistema, al igual que la cpu.

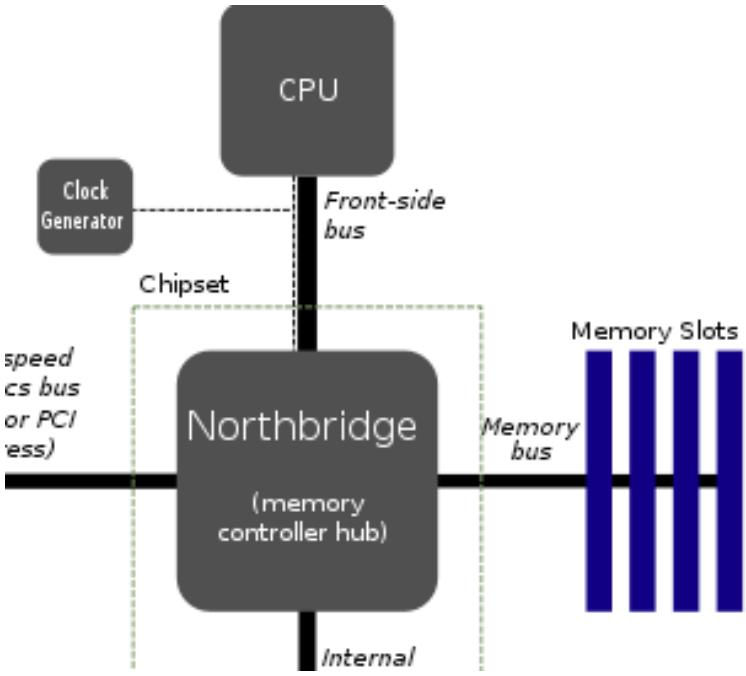


Figura 4.1: Diagrama de conexión entre memoria y cpu.

Ahora bien, como se ha comentado la cpu puede operar a frecuencias distintas. ¿ Cómo es posible si recibe una única señal de reloj ? en realidad, la cpu multiplica la frecuencia recibida mediante un circuito multiplicador, determinando el denominado *bus/core ratio*. Por ejemplo, en el equipo en el que se realizaron las pruebas [Q6600], la frecuencia del reloj es de 266 MHz y su bus/core ratio de 9, lo que significa una velocidad de cpu de 2.4 GHz. Ahora bien, si se rebaja el factor de multiplicación de 9 a 6, la cpu irá a una velocidad de 1.6GHz, la mínima a la que se podrá escalar en nuestra máquina. Haciendo esto se podrá también rebajar la tensión, como se comentó en el apartado anterior, y por tanto reducir el consumo. De esta forma, si se multiplica el reloj por 7, 8 y 9 tenemos las frecuencias disponibles en nuestro equipo (1.6GHz 1.8GHz 2.1GHz 2.4GHz). Por otra parte, la frecuencia de reloj domina la velocidad a la que pueden transmitirse datos por el FSB; en nuestro caso el FSB utiliza la tecnología Quad Pump [QPUMP] por lo que se dispone de una frecuencia efectiva en el bus de $266 \times 4 \approx 1066$ MHz.

Capítulo 5

Metodología

5.1. Programas utilizados

Como se comentó al principio de éste trabajo, el objetivo es investigar el impacto que una determinada técnica para reducir el consumo, el escalado dinámico del voltaje y la frecuencia, tiene sobre la energía consumida en cargas de trabajo típicas de entornos HPC; por lo tanto, se tenían que escoger diferentes programas que representasen éstas cargas de forma correcta. Para ello se decidió utilizar los denominados *Nas Parallel Benchmarks*, un conjunto de pequeños programas desarrollados por la NASA y diseñados para ayudar en la evaluación del rendimiento de supercomputadores paralelos, de uso frecuente en los artículos de investigación que se han consultado (p.ej [EET]) y cuyo código fuente puede ser descargado y utilizado libremente . Éstos programas derivan de la mecánica de fluidos y consisten en varios *kernels* y tres pseudo-aplicaciones[NASW].

A su vez, los kernels tienen definidos diferentes tamaños de problema, denominados *Clases*. En general estas clases han ido evolucionando con la librería, añadiendose a medida que surgían nuevos kernels o se modificaban los anteriores. En la versión 3.1, con la que se ha trabajado, se dispone de las clases S, W, A, B, C, D, E para los kernels SP EP CG BT FT y LU, que son los que han sido utilizados.

En la tabla 5.1 se muestran los tamaños de problema de las clases A y B para

Kernel	Tamaño Clase B	Tamaño Clase A
Embarrasingly parallel (EP)	2^{30}	2^{28}
Conjugate Gradient (CG)	75000	14000
3-D FFT PDE FT)	512x256x256	256^2x128
LU Solver (LU)	102^3	64^3
Pentadiagonal Solver (SP)	102^3	64^3
Block tridiagonal solver (BT)	102^3	64^3

Cuadro 5.1: Tamaños de problema de las clases A y B. [NAS]

los kernels mencionados. Por ejemplo, el cálculo del gradiente conjugado (CG) para clase B utiliza una matriz de entrada de $75000x75000$, mientras que la clase A lo hace sobre una de $14000x14000$. Además de los tamaños de problema, cada kernel tiene algunos parámetros específicos que varían entre clases. Por ejemplo, la clase B del kernel CG realiza 75 iteraciones, por 15 de la clase A. En nuestro caso se optó por la Clase B por dos razones: primero porque su tiempo de ejecución en las diferentes plataformas que se probaron era lo suficientemente grande como para eliminar, con mayor probabilidad, cualquier ruido a nivel de consumo que pudiera darse en el sistema durante la prueba. En segundo lugar, como también se comenta en [EET], éste tamaño no es lo suficientemente grande como para tener carga alguna de I/O en un sistema de 1GB como el de los equipos citados en el artículo, mucho menos en el nuestro, que dispone de 3 GB.

5.2. Organización de los experimentos

A la hora de realizar los experimentos era deseable obtener el conjunto de mediciones de energía que se correspondiese con el intervalo de ejecución del programa de la forma más ajustada posible. Realizar cada experimento de forma manual era algo tedioso y muy propenso a errores, ya que se tendría que estar alerta a la finalización del programa (y algunas ejecuciones podrían durar horas) para recoger las muestras en el momento adecuado. No era por lo tanto una opción viable. Por ello, y aprovechando que el medidor de potencia ofrecía una API para poder ser manejado programaticamente, se desarrollaron dos programas, uno

cliente y otro servidor, que coordinasen las ejecuciones y recogida de datos siguiendo un esquema como el de la figura 5.1. Este esquema se repite para cada programa, dentro de cada programa para cada frecuencia, y en el caso de pruebas con más de un núcleo, dentro de cada frecuencia para cada combinación de núcleos.

El lenguaje escogido para la realización de los programas fue Python. Aunque no se disponía de experiencia previa en el mismo, la facilidad en la creación de sockets para un modelo cliente servidor pesó suficientemente como para no considerar alternativas más complejas, como C o Java, que requerirían un mayor tiempo de desarrollo. Python cubrió sobradamente las necesidades y permitió un desarrollo y puesta en marcha muy rápido.

Al respecto de este esquema se ha de comentar una situación que entorpeció los cálculos en algunos momentos. La primera versión del servidor recogía la salida del programa y la enviaba al cliente, junto con la información de que el programa había acabado, para que éste la guardase como un registro y poder comprobar si había habido algún error posteriormente. Sin embargo, si se realizaba éste envío de la salida, que podía ser de algunos KB, el consumo en la máquina que se media se disparaba, y no recobraba la situación de reposo hasta varios minutos después, dando medidas erróneas para las pruebas que se ejecutaban a continuación. Para solventar éste problema simplemente se decidió no enviar la salida, únicamente verificando que ésta tuviese el literal "VERIFICATION SUCCESSFUL", que los benchmarks siempre mostraban si la ejecución había ido bien.

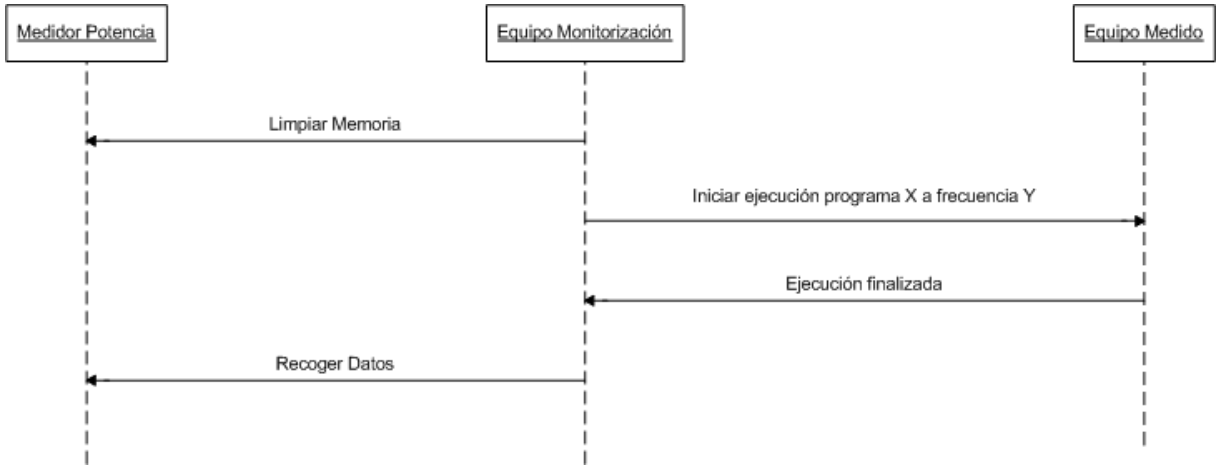


Figura 5.1: Esquema de comunicación entre los componentes del sistema.

Capítulo 6

Instrumentos y Sistemas analizados

6.1. Medidor de Consumo

El medidor que se ha utilizado es el modelo PRO de la marca Watts up?. Es capaz de proveer medidas de 19 factores (vatios actuales, vatios máximos, vatios hora, costes, etc..), de los que se utilizará principalmente los vatios actuales, que corresponden a la potencia activa (ver Capítulo 2). El tiempo de muestreo es de 1 segundo y tiene memoria interna para 2000 registros en modo automático. Si únicamente se registran vatios actuales la cantidad de registros se incrementa hasta 16384 o lo que es lo mismo, 4 horas y media, aproximadamente (los registros también se realizan en intervalos de 1 segundo).

Figura 6.1: Medidor Watts up?.

6.2. Equipos y procesadores

Inicialmente solamente se consideró realizar las pruebas en un equipo monoprocesador, un Intel Celeron, pues en principio era el único del que se disponía que ofreciese escalado de frecuencia. Tras los primeros intentos, y por las razones que se explican en la sección 7.1.2, se abandonó el Celeron en beneficio de un Core 2 Quad, cuyas características se muestran en la tabla 6.2[Q6600]

CPU	Intel® Core™2 Quad Processor Q6600
# of Cores	4
# of Threads	4
Instruction Set	64 bit
Max TDP	105 W
L2 Cache	4MB x Core
Frecuencias (Hz)	2403000 2136000 1870000 1603000
Main Memory	3GB, 2 DIMMS(1GB & 2GB) DDR2-667 PC2-5300

Cuadro 6.1: Características del Core 2 Quad.

Capítulo 7

Resultados

7.1. Intel Celeron

Como se comentó en el capítulo 6, en un principio el único equipo con el que se iba a probar era un Intel Celeron (Tabla 7.1[CELERON]).

CPU	Intel® Celeron® D Processor 346
# of Cores	1
# of Threads	1
Instruction Set	64 bit
Max TDP	84W [1]
L2 Cache	256KB
Main Memory	1GB
Frecuencias	3.06GHz

Cuadro 7.1: Características del Monoprocesador Celeron

[1]Tanto Intel como AMD lo definen como el máximo consumo energético para periodos de ejecución termalmente significativos en el peor caso de cargas de trabajo no sintéticas. Por lo tanto, TDP **no es** el consumo máximo del procesador.

Sin embargo, tras las primeras pruebas, se comprobó que no era el equipo adecuado. En este apartado se mostrarán las pruebas que llevaron a descartar dicho equipo.

7.1.1. Caracterización consumo en reposo

Para esto se midió el consumo del sistema en funcionamiento únicamente con el sistema operativo y a cada una de las frecuencias posibles, durante 30 minutos.

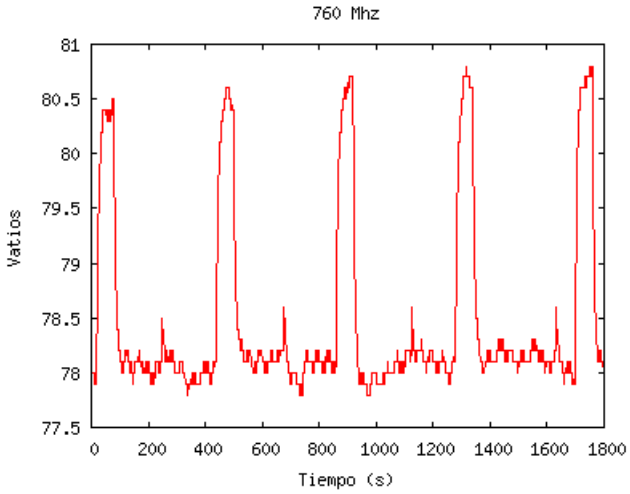
Las preguntas que se intentaban responder eran:

- ¿ Cual es el consumo en reposo del sistema ?
- ¿ Hay diferencias en el consumo en reposo a diferentes frecuencias ?

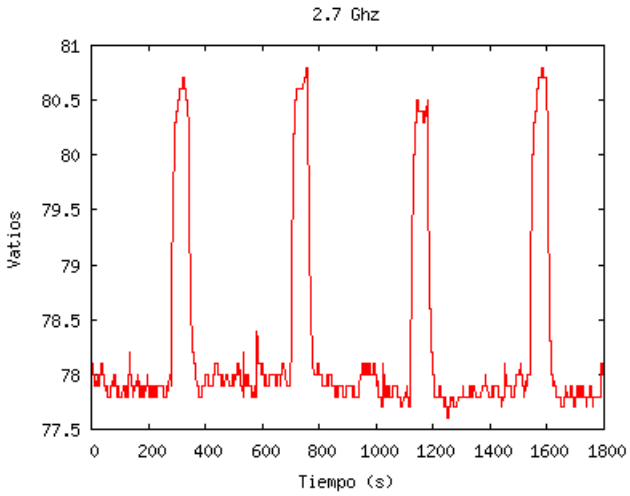
Como respuesta a ambas, se descubrió que el consumo era de unos 70 vatios, en todas las frecuencias. Esto no era lo esperado pues al bajar la frecuencia y el voltaje debería notarse una reducción en el consumo.

Además, se pudo comprobar como cada cierto tiempo, a intervalos regulares, había picos de 2 vatios, también en todas las frecuencias, para los que no se disponía de explicación. Tras barajarse varias hipótesis, principalmente el factor de potencia de la fuente y los procesos del S.O, ninguna de ellas resultó adecuada: en el caso del f.d.p se probó con diferentes fuentes con diferentes f.d.p sin hallarse un patrón al respecto; para los procesos del sistema operativo (S.O) se realizaron las pruebas directamente desde la BIOS, observando los mismos resultados.

En la figura 7.1 se muestran un par de ejemplos del consumo en reposo a dos frecuencias diferentes (para el resto de frecuencias el comportamiento es el mismo).



(a) 760Mhz



(b) 2.7GHz

Figura 7.1: Consumo en reposo del Intel Celeron para diferentes frecuencias.

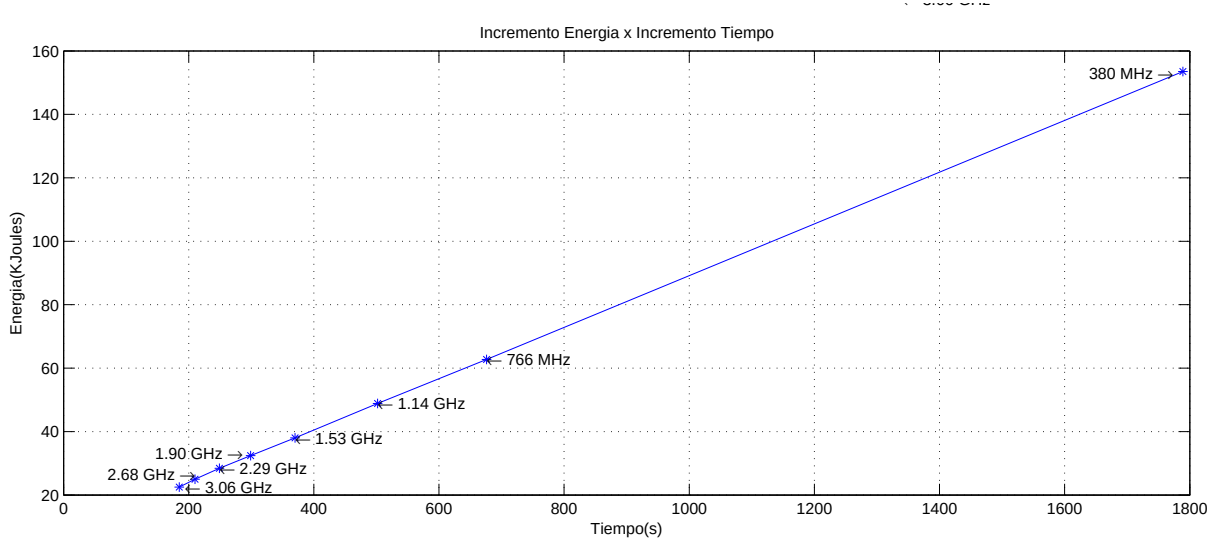
7.1.2. Experimentos y localización del error

En la figura 7.2 se pueden ver los resultados para dos de los kernels, EP y CG. Para ambos, *el aumento del consumo y el tiempo son proporcionales*.

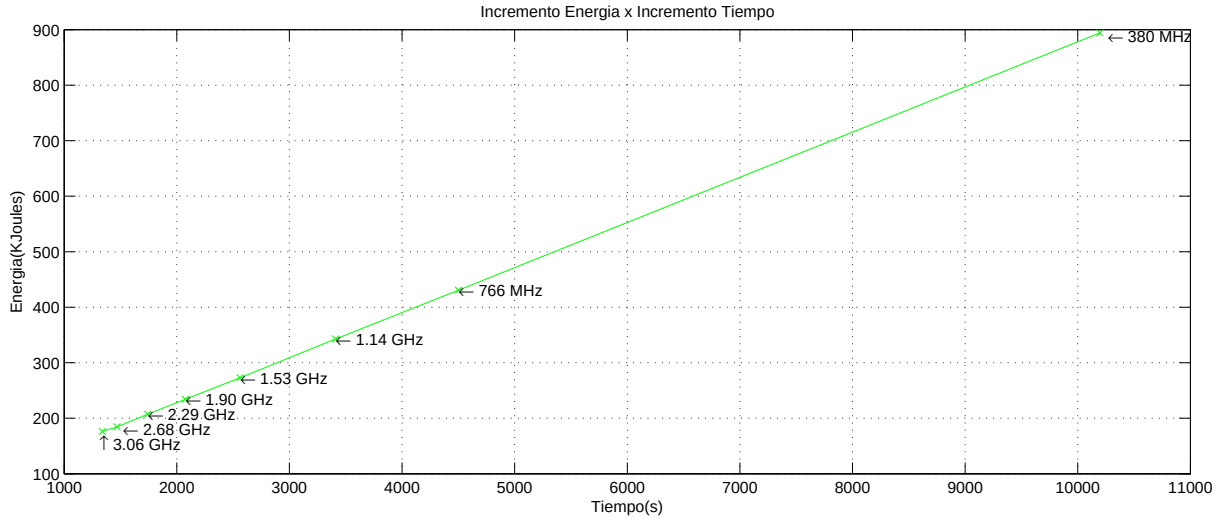
En cuanto al kernel EP, intensivo en cómputo, los resultados fueron los esperados. Sin embargo, para el CG, que es intensivo en memoria, contradecían toda la teoría que habíamos estado revisando al respecto. Según la misma, en ese tipo de cargas el escalado de frecuencia podría ayudar a rebajar el consumo con un incremento del tiempo moderado, mediante el rebajado de la frecuencia de la CPU para adecuarla a la latencia de memoria.[EET]. Por lo tanto, era muy probable que estuviésemos haciendo algo mal.

Tras consultar diferente documentación técnica se averiguó que el controlador para el escalado de frecuencia de Linux que se estaba utilizando (p4-clockmod) no realizaba un escalado real, sino que introducía periodos de inactividad. Tal y como se puede leer en el blog de uno de los desarrolladores del driver [P4CM], éste únicamente mantiene el procesador sin realizar trabajo durante un determinado porcentaje de tiempo, en ningún caso reduce la frecuencia ni mucho menos la tensión. Si la frecuencia es de 2 GHz y se rebaja a 1 GHz, la frecuencia sigue siendo de 2 GHz pero la cpu únicamente realiza trabajo el 50 % del tiempo. La utilidad de este controlador no es, por tanto, el escalado de frecuencia y voltaje reales, sino limitar la cantidad de calor disipada por la CPU para evitar su sobrecalientamiento [P4CM]

Se revisó entonces la documentación del microprocesador, comprobando que en realidad carecía de escalado de frecuencia. Por lo tanto, a partir de ese momento, se decidió trabajar únicamente sobre el Quad.



(a) EP Clase B



(b) CG Clase B

Figura 7.2: Consumo y tiempo para diferentes frecuencias (EP, un núcleo)

7.2. Quad

7.2.1. Caracterización consumo en reposo

En este caso se plantearon las mismas preguntas que para el Celeron, añadiendo la siguiente

- ¿ Hay diferencias en el consumo al cambiar el numero de cores ?

Para modificar el número de nucleos activos en cada momento se utilizó la instrucción

```
echo 0 > /sys/devices/system/cpu/cpuX>online
```

En cuanto al consumo en reposo a diferentes frecuencias se comprobó que, efectivamente, al reducir la frecuencia, manteniendo el mismo número de núcleos activo, había una reducción del consumo (ver figura 7.2.1). Se puede apreciar que entre la frecuencia más alta y la más baja hay una diferencia de unos 2 vatios. Otro dato que se pudo extraer y que resulta interesante, aunque en éste trabajo no se explote, es que cada core puede funcionar a una frecuencia diferente. Esto permite una enorme flexibilidad que se desconoce si realmente es aprovechada por los sistemas gestores de energía a nivel de sistema operativo, por ejemplo.

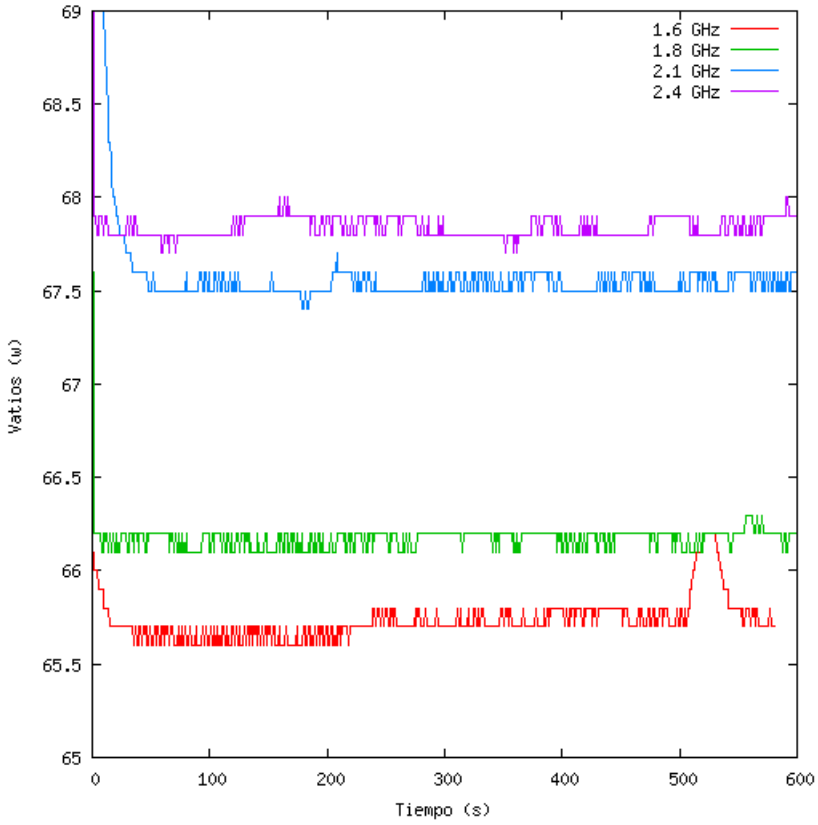
En lo que respecta al consumo en función del número de núcleos activos, no se aprecian cambios significativos, dentro de una misma frecuencia. Es decir, que la activación o desactivación de un núcleo, al menos tal y como fue realizada por nosotros, no tiene efectos sobre el consumo, ni siquiera cuando se deja un procesador completo desactivado.

En cuanto a los picos de 2 vatios del celeron, siguen apareciendo. Como lo hacen de forma constante se asumirá que afectarán igual a todas las pruebas, por lo que no serán considerados.

7.2.2. Bateria de Experimentos

Se utilizaron 6 benchmarks del NAS (ver capítulo 5). Para cada benchmark se realizaron 3 ejecuciones por frecuencia y número de núcleos utilizados (1,2 y 4),

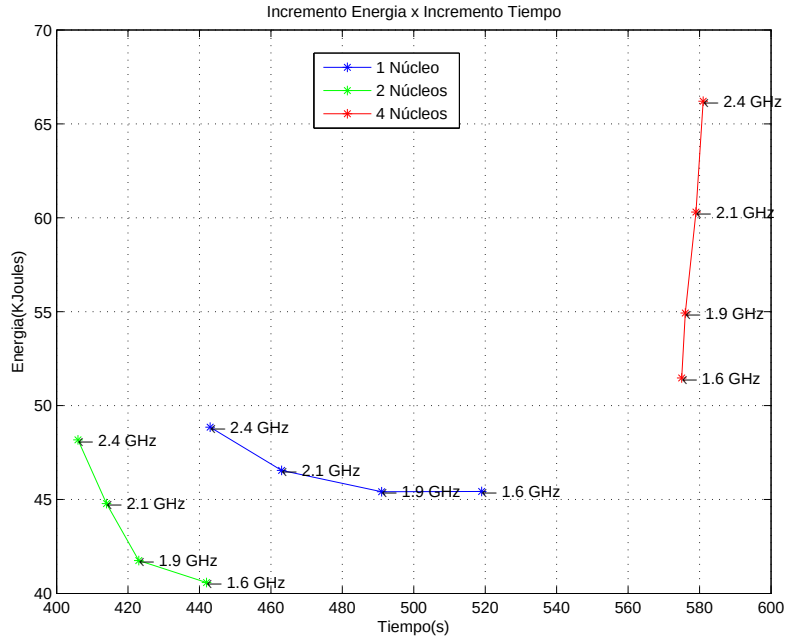
Figura 7.3: Consumo a diferentes frecuencias para el Quad.



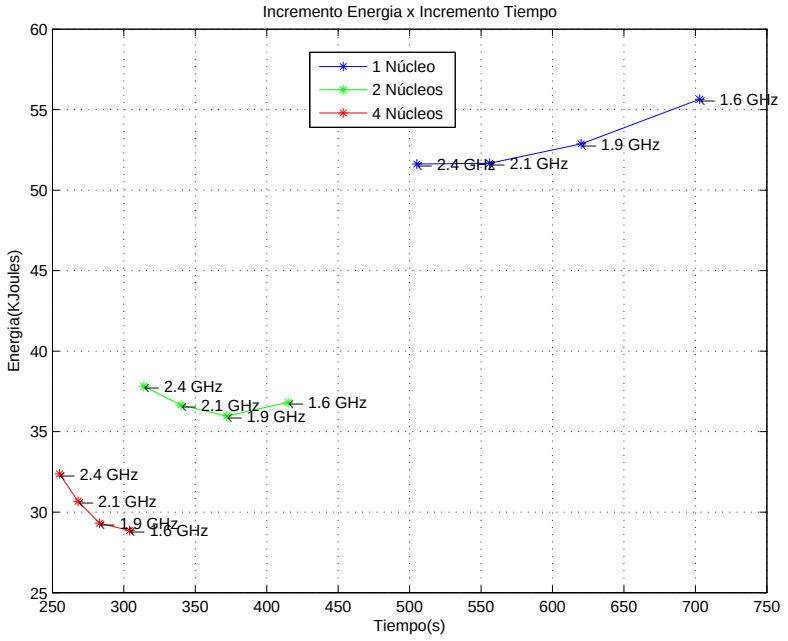
dando un total de 4 freqs x 3 comb. núcleos x 3 repeticiones = 36 ejecuciones por programa.

En todos los casos las tres repeticiones tenían un mismo patrón de comportamiento, con diferencias mínimas. Se muestra a continuación el resultado de una de las repeticiones para cada programa.

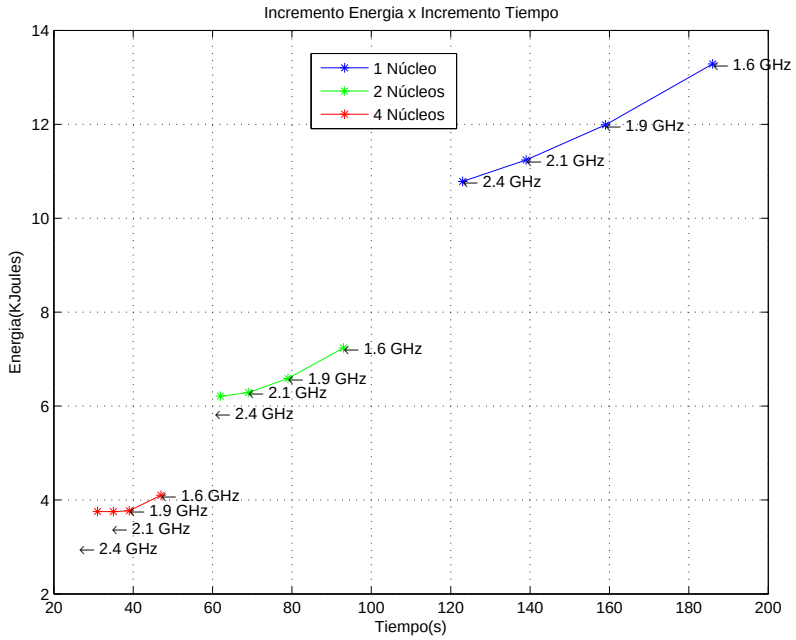
Figura 7.4: Resultados para los diferentes programas



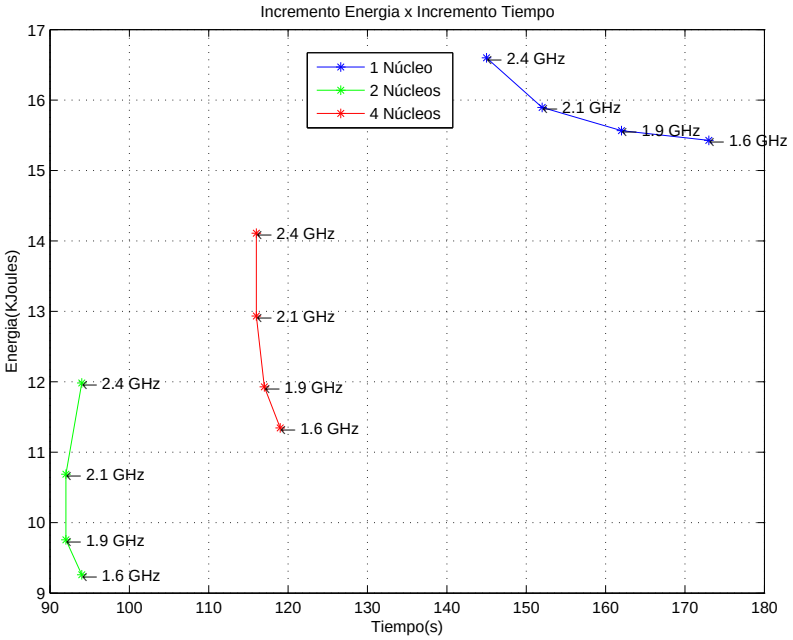
(a) SP



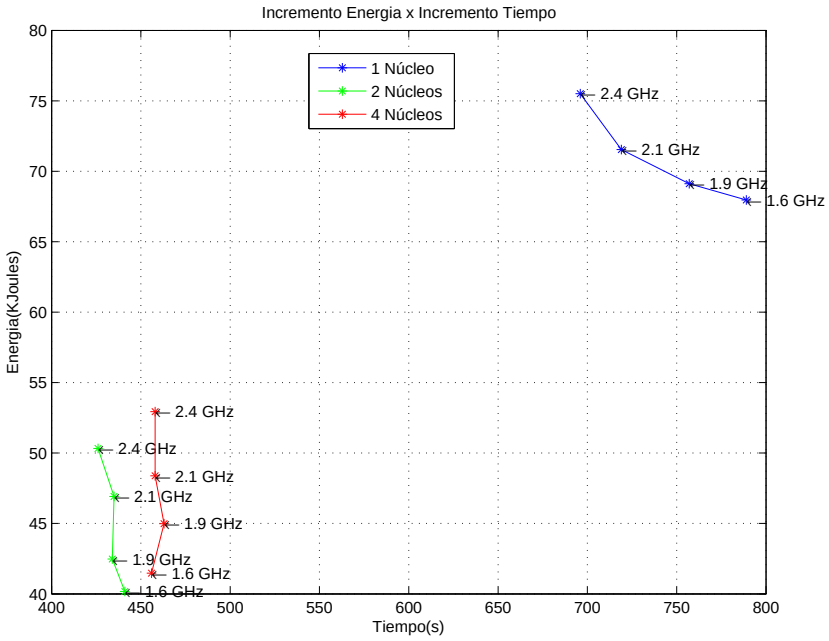
(b) BT


Incremento Energia x Incremento Tiempo
1 Núcleo
2 Núcleos
4 Núcleos
1.6 GHz
1.9 GHz
2.1 GHz
2.4 GHz
Energia(KJoules)
Tiempo(s)

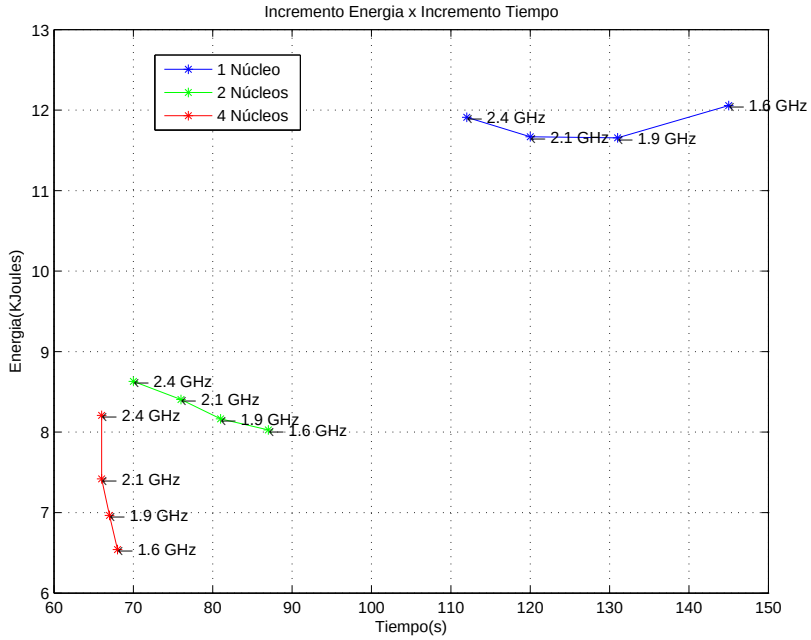
(c) EP


Incremento Energia x Incremento Tiempo
1 Núcleo
2 Núcleos
4 Núcleos
2.4 GHz
2.1 GHz
1.9 GHz
1.6 GHz
Energia(KJoules)
Tiempo(s)

(d) CG


Incremento Energia x Incremento Tiempo
1 Núcleo
2 Núcleos
4 Núcleos
2.4 GHz
2.1 GHz
1.9 GHz
1.6 GHz
2.4 GHz
2.4 GHz
2.1 GHz
2.1 GHz
1.9 GHz
1.9 GHz
1.6 GHz
1.6 GHz
Energia(KJoules)
Tiempo(s)
80
75
70
65
60
55
50
45
40
400
450
500
550
600
650
700
750
800

(e) LU


Incremento Energia x Incremento Tiempo
1 Núcleo
2 Núcleos
4 Núcleos
2.4 GHz
2.1 GHz
1.9 GHz
1.6 GHz
2.4 GHz
2.1 GHz
1.9 GHz
1.6 GHz
2.4 GHz
2.1 GHz
1.9 GHz
1.6 GHz
Energia(KJoules)
Tiempo(s)
13
12
11
10
9
8
7
6
60
70
80
90
100
110
120
130
140
150

(f) FT

Capítulo 8

Análisis de los Resultados

En primer lugar se analizarán los resultados para un único núcleo, exponiendo los diferentes comportamientos de los programas y agrupándolos en función de la frecuencia a la que se ejecutaron. Debido a que la comprensión en profundidad de cada uno de los kernels, con una caracterización adecuada de su rendimiento, requiere de un tiempo del que no se dispone, para cada grupo de programas se escogerá un representante, el más simple conceptualmente, para plantear las hipótesis que se cree explican dicho comportamiento.

Finalmente, se analizará el impacto que comporta el añadir más núcleos a la ejecución.

8.1. Relación potencia y tiempo

Tal y como se vio en el capítulo 2, la energía es la integral de la potencia en función del tiempo. En el caso de tensión continua, la energía será simplemente el producto de potencia por tiempo.

$$E = P \times t \tag{8.1}$$

Incrementar la frecuencia implica aumentar la potencia. Del mismo modo, si operamos a una mayor frecuencia es lógico pensar en una reducción del tiempo de ejecución, puesto que el procesador trabaja a mayor velocidad. Se observa por

tanto como la relación entre ambos factores determina el aumento o disminución de la energía consumida para una ejecución completa. A raíz de esta observación se plantea cual es el punto de equilibrio entre potencia y tiempo; de este modo, dados unos incrementos de potencia y decrementos de tiempo, derivados de un aumento de la frecuencia, se podrá determinar claramente si tal aumento ha supuesto un ahorro o gasto extra en energía.

Partiendo de la ecuación 8.1

$$E = P \times t$$

Dados los porcentajes de incremento de potencia α y disminución de tiempo β, la energía consumida se mantendrá si

$$P \times t = P(1+\alpha) \times t(1-\beta)$$
$$1 = (1+\alpha) \times (1-\beta)$$

Es decir, la conservación de energía se producirá cuando α y β mantengan la siguiente relación,

$$\alpha = \frac{1}{1-\beta} - 1 \tag{8.2}$$

Por lo tanto, se ahorrará energía cuando se cumpla la siguiente ecuación,

$$P \times t > P(1+\alpha) \times t(1-\beta)$$
$$\alpha > \tfrac{1}{1-\beta} - 1$$

y se estará gastando más de lo que ahorramos si se da el caso contrario

$$P \times t < P(1+\alpha) \times t(1-\beta)$$
$$\alpha < \tfrac{1}{1-\beta} - 1$$

En la figura 8.1 se muestra gráficamente la relación entre los dos incrementos.

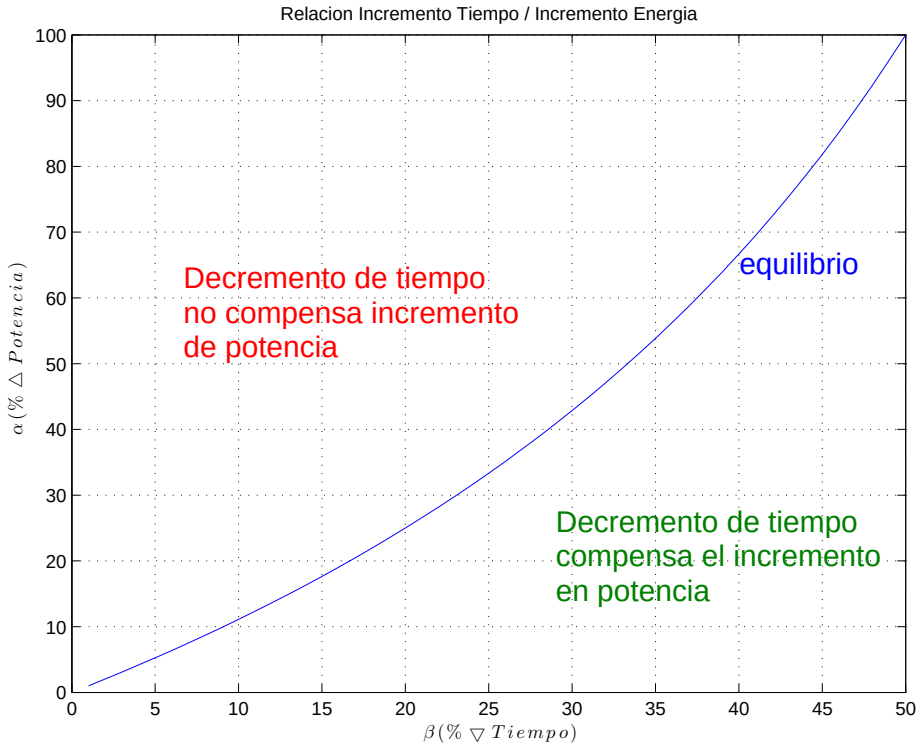


Figura 8.1: Relación entre los incrementos de potencia y tiempo.

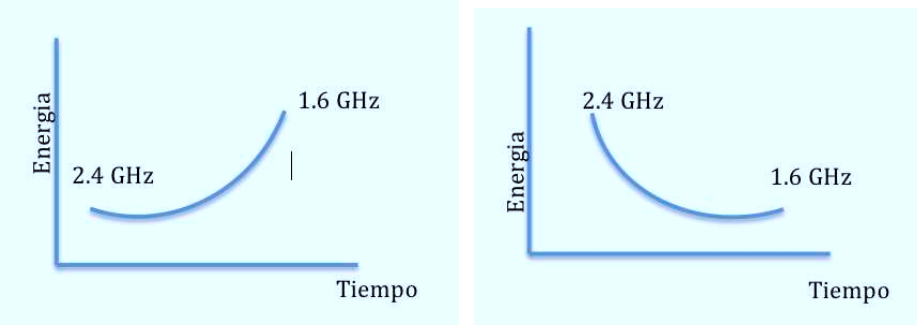
8.2. Un núcleo

A partir de los resultados mostrados se pueden extraer dos tipos de comportamiento, caracterizables mediante sus curvas. Tales curvas características se muestran en la figura 8.2.

8.2.1. Modelo a

Esta situación se produce en los kernels EP y BT. El kernel FT también muestra este comportamiento, aunque solo en una parte de la curva. De éstos, escogeremos para la explicación EP por ser el más sencillo conceptualmente.

El programa EP es un generador de números aleatorios; es por lo tanto un programa *intensivo en cómputo*. En la figura 8.3 se muestra como evolucionan potencia y tiempo en función de la frecuencia.



(a) (b)

Figura 8.2: Modelos de curva extraídos de los experimentos.

Se observa como el ahorro producido por la disminución del tiempo compensa el gasto derivado del aumento de la potencia, o lo que es lo mismo, el tiempo se reduce de forma más pronunciada de lo que incrementa la potencia.

Este comportamiento concuerda con el que se debería esperar de un programa que únicamente realiza operaciones de cómputo. Al no haber otros factores que limiten la disminución del tiempo, principalmente accesos a memoria, éste depende únicamente de la velocidad del procesador; incrementarla es reducir el tiempo de forma proporcional.

Concluimos, por lo tanto, que, al menos en un único núcleo, en un programa de este tipo lo ideal es ejecutarlo a la máxima frecuencia para hacerlo con el menor tiempo y el menor consumo posibles.

8.2.2. Modelo b

Aqui encontramos los kernels LU, CG, SP y parte del FT. Escogeremos en este caso el CG.

El kernel CG implementa el gradiente conjugado. Ésta operación comporta la multiplicación de un vector por una matriz dispersa, esto es, una matriz con la mayor parte de sus posiciones a 0. El acceso a los elementos no nulos se realiza mediante unos vectores de indices, con lo que patrón de accesos es prácticamente aleatorio con un escaso aprovechamiento de las cachés. Todo esto conformaría un programa *intensivo en memoria*, en contraste con el kernel EP.

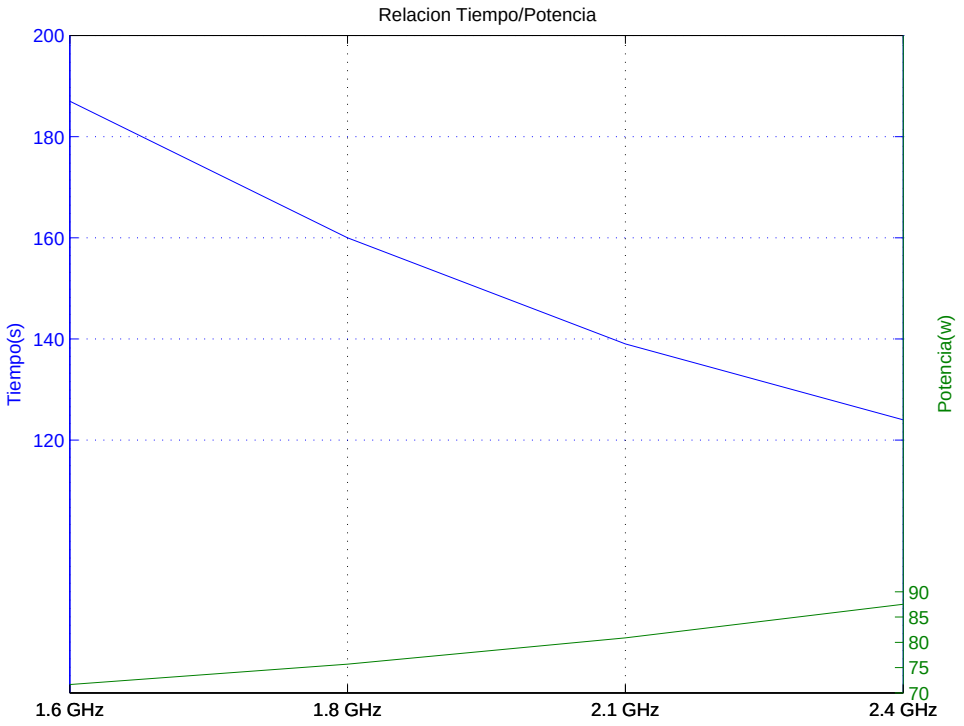


Figura 8.3: Incrementos de potencia y tiempo para el kernel EP.

Se muestra a continuación la figura que relaciona potencia y tiempo.

Como puede observarse, en este caso la pendiente de la curva de tiempo es menos pronunciada que la de potencia; es decir, el incremento en potencia no compensa la reducción de tiempo. En este caso, si tenemos un factor limitante, el acceso a memoria. Al incrementar la frecuencia solo podremos reducir el tiempo en una fracción del tiempo total, y cada vez en una fracción menor, siguiendo la ley de Amdahl, pues el tiempo de acceso a memoria permanece constante. Además, al incrementar la frecuencia, todo el tiempo que la cpu se encuentra ociosa esperando a memoria está operando a una frecuencia mayor, por lo tanto generando un mayor consumo sin realizar trabajo alguno.

Por todo lo dicho, en la ejecución de un programa de este tipo y con un único núcleo, la frecuencia debería escogerse en relación al aspecto que tenga mayor importancia. Por ejemplo, la mejor relación entre ahorro de energía e incremento de tiempo se encuentra en 2.1 GHz, aunque quizás interese ejecutar a menor fre-

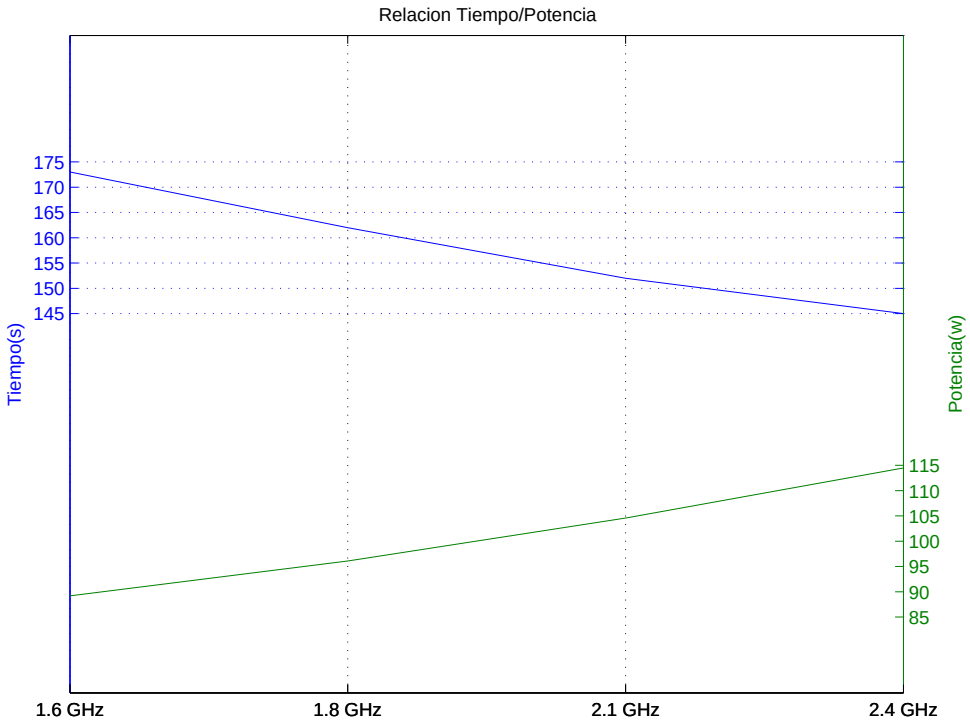


Figura 8.4: Incrementos de potencia y tiempo para el kernel CG.

cuencia durante más tiempo consumiendo menos, en cuyo caso habría de hacerse a 1.6 GHz.

8.3. Dos y cuatro núcleos

Se observa que en todos los casos, excepto en el kernel SP con 4 núcleos, las ejecuciones con dos y cuatro núcleos mejoran a las de un núcleo tanto en tiempo como en energía consumida. Asimismo, se aprecia una tendencia a la "verticalidad" a medida que se incrementa el número de núcleos, derivando en una curva como la de la siguiente figura,

8.3.1. Programas intensivos en cómputo

Estos corresponden a los que mostraban un comportamiento como el de modelo a de curva en el apartado anterior. Los programas clasificados de esta forma

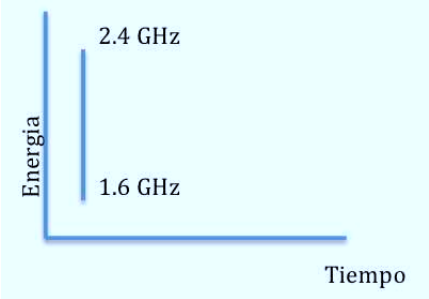


Figura 8.5: Modelo de curva predominante en 2 y 4 cores.

eran EP, BT.

En el caso de EP, donde no hay accesos a memoria, el escalado es perfecto: más núcleos y más frecuencia implican un menor consumo y menor tiempo. Parece claro que, si no hay factor limitante, incrementar el número de núcleos y la frecuencia siempre redunda en un menor consumo y tiempo. Sin embargo, en el caso de BT, dicha tendencia se invierte a partir de los dos núcleos, y se acentúa con cuatro, mostrando un comportamiento propio del modelo b, o intensivo en memoria. Se puede analizar este comportamiento del siguiente modo: el incremento de núcleos comporta una reducción del cómputo que realiza cada uno, lo que aumenta la proporción de tiempo que la cpu dedica a esperar a memoria en cada núcleo. Por lo tanto, la proporción inicial entre cómputo y memoria marcará cuando se llega al límite en el que el cómputo en cada núcleo no es el suficiente como para obtener ventajas al incrementar la frecuencia. En éste caso, parece que tal proporción es lo suficientemente favorable para el cómputo como para que de dos a cuatro núcleos no se consuma ni tarde más, justo lo contrario de lo que sucede en el FT. En éste último, la reducción del porcentaje de cómputo por núcleo con cuatro núcleos parece tal, que su reducción mediante el incremento de frecuencia no contribuye a reducir el tiempo total de ejecución, únicamente a incrementar el consumo.

8.3.2. Programas intensivos en memoria

En este caso, es predecible una tendencia más rápida hacia un modelo de curva como el mostrado en la figura 8.5. Si se examinan los programas que mostraban tal comportamiento (CG, LU y SP), es notable que en todos los casos el rendimiento en tiempo es mejor para dos que para cuatro núcleos, lo que explicaría el aumento de consumo también de dos a cuatro núcleos. Éste aumento de tiempo es especialmente pronunciado en el kernel SP, donde se llega incluso a superar el tiempo con un único núcleo.

En cualquier caso, tanto en dos como en cuatro núcleos, la curva descrita corresponde al modelo 8.5, donde incrementar la frecuencia únicamente supone un aumento del consumo, sin disminuir el tiempo de ejecución. Parece que el aumento de núcleos contribuye a hacer de la memoria el cuello de botella. *Ésta es la situación en la que más provechosa resulta la disminución de la frecuencia, pues pueden conseguirse reducciones notables del consumo sin perjudicar el rendimiento.* Para comprobar como afecta al uso de memoria el aumentar los núcleos de ejecución se calcularon los anchos de banda consumidos por cada aplicación en cada combinación de núcleos. Para realizarlo se utilizó el software de profiling VTUNE, aplicando la siguiente formula [VTUNE]

$$\frac{cls \cdot BUS\_TRANS\_BURST.ALL\_AGENT \cdot freq}{CPU\_CLK\_UNHALTED.CORE} \tag{8.3}$$

donde cls es el tamaño de la linea de caché L2 y freq es la frecuencia de reloj de los núcleos. El primer contador, BUS_TRANS_BURST, refleja el número de transacciones de lineas de caché. ALL_AGENT indica que se contabilicen las transacciones iniciadas por todos los núcleos. Como en la arquitectura en la que se trabaja todos los núcleos están conectados al mismo bus (FSB), este contador nos dará las transacciones de lineas de cache totales. El contador CPU_CLK_UNHALTED.CORE indica cuantos ciclos el núcleo no estuvo parado, es decir, ejecutando instrucciones HLT. Por lo tanto, la ecuación anterior indicaría la proporción entre la cantidad de memoria necesaria para realizar el cómputo. En la arquitectura bajo pruebas el tamaño de la linea de caché es 64 y

todas las mediciones se realizaron con todos los núcleos a la máxima frecuencia (2.4 GHz). Al mismo tiempo, se utilizo el benchmark STREAM [STREAM] para comprobar cual era el límite real de la arquitectura bajo pruebas. Los resultados se muestran en la siguiente figura,

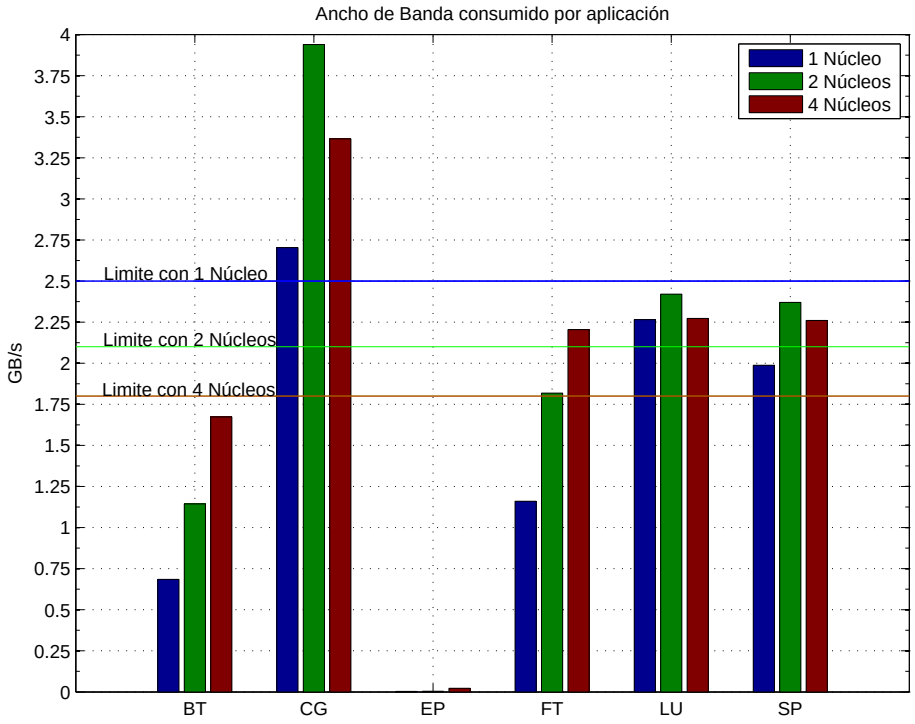


Figura 8.6: Anchos de banda por aplicación y núcleos

Se observa como para los programas que han sido catalogados como intensivos en computo (EP, BT y FT), únicamente el FT con 4 núcleos supera el ancho de banda. Este incremento coincide con el comportamiento limitado por memoria que muestra FT para los mismos núcleos. El kernel BT en ningún momento llega a superar los anchos de banda límite, aunque se acerca progresivamente. El comportamiento es coherente con el mostrado en la figura 7.4, donde se aprecia la misma tendencia. En cuanto a los programas catalogados como intensivos en memoria (CG, SP y LU), los datos de ancho de banda son algo menos regulares, aunque presentan ciertas semejanzas. En primer lugar, para los kernels SP y LU hay poca diferencia en los requerimientos de ancho de banda para 1, 2 y 4 núcleos.

Es decir, el programa ya se encuentra limitado por memoria con un único núcleo, con lo que añadir más no afecta demasiado. Sin embargo, esa cantidad en que se incrementa el ancho de banda requerido conlleva la superación del limite, tanto para 2 como para 4 núcleos, saturando el FSB. Esto podría explicar el comportamiento en dichos casos correspondiente al modelo de curva c, aunque ciertamente en el SP de dos núcleos la curva no sea tan clara, al menos no hasta su parte final.

El CG con un único núcleo es sin embargo un caso particular. Aunque con dos y cuatro núcleos se sobrepasan con creces los límites, lo que se refleja en la figura 7.4, con un único núcleo la superación del ancho de banda no se observa en modo alguno en la figura 7.4.

En todo caso, y exceptuando este último, parece claro que el incremento de núcleos comporta un mayor uso de memoria, como es lógico por otra parte, y que las situaciones en las que se satura el FSB son los casos extremos de la figura 7.4 en los que, independientemente de la frecuencia a la que se ejecute el programa, su tiempo apenas disminuye, ofreciendo mayores oportunidades para la reducción del consumo sin efectos secundarios en el rendimiento.

Capítulo 9

Conclusiones

Se ha constatado como el consumo está relacionado de forma directa con el programa que se ejecuta. Diferentes programas, con diferentes características muestran comportamientos muy diferentes en relación a su consumo energético. Es necesario, por lo tanto, considerar tales características, especialmente la proporción entre computo y memoria, para decidir la estrategia óptima. Se ha comprobado como, en el caso de programas altamente limitados por memoria, la ejecución a mayores frecuencias únicamente supone un aumento del consumo sin beneficio alguno en reducción de tiempo.

En cuanto a los problemas aparecidos, los ha habido de diversos tipos. El primero fue la confusión entre escalado y la introducción ciclos no operativos de la cpu, explicado en el capítulo 7. Además del tiempo dedicado a averiguar el error, el comprobar que no se disponía de una máquina con escalado de frecuencia y tensión reales forzó la necesidad de hacerse con un equipo que si la tuviese, puesto que en caso contrario no habría sido posible la realización del proyecto. Afortunadamente pudo conseguirse un equipo prestado. Otro problema, aunque éste de más fácil solución y, hasta cierto punto, previsible, fue la organización de los experimentos. La necesidad de sincronización entre el equipo que recoge los datos y el equipo bajo observación fue una tarea que llevo algo de tiempo, aunque se resolvió más rápido de lo esperado y resultó, a largo plazo, de gran utilidad, al proveer de un sistema automatizado para realizar las pruebas. Sin duda el tiempo

invertido valió la pena. En éste sentido fue de gran ayuda la interfaz de conexión con el medidor y su API de programación. El análisis de los resultados tampoco ha resultado tarea sencilla, especialmente los anchos de banda consumidos por las aplicaciones. En este sentido, queda alguna medida que requeriría de mayor estudio para su explicación, como las del kernel CG. Sin embargo, el tipo de análisis realizado, involucrando los contadores de rendimiento HW, y librerías de instrumentación y acceso a los contadores, como TAU, VTune y PAPI, ha resultado muy interesante. Saber utilizar dichas herramientas ofrece un gran potencial y es algo en lo que merece la pena profundizar.

Como futuro trabajo puede plantearse el estudio de la combinación del escalado de frecuencia en función de la fase en que se encuentre el programa. Por ejemplo, reducir la frecuencia si la cpu se encuentra inactiva esperando a memoria, y aumentarla durante el cómputo. Estrategias similares forman la base del proyecto Green Building Blocks [GBB]. Aunque no parece haber información actualizada, y es difícil saber en que estado se encuentra el proyecto, las ideas planteadas corresponden con lo comentado.

Sin embargo, y siendo realistas, es muy probable que los mayores avances en este sentido vengan del campo del hardware. Noticias como el desarrollo por parte de IBM de transistores de grafeno[TRGR], los cuales, según un estúdio de la Universidad de Illinois [GRCO], serían capaces de “auto-enfriarse”, abren la puerta a circuitos integrados con una mayor densidad de transistores y mayor eficiencia energética.

Bibliografía

[CATPE] Stefanos Kaxiras, Margaret Martonosi, *Computer Architecture Techniques for Power Efficiency*, Morgan & Claypool, 2008.

[APM] David J. Brown, Charles Reams, *Toward Energy-Efficient Computing*, acmqueue, Vol. 8 No. 2 – February 2010
<http://queue.acm.org>

[URL] Vincent W. Freeh, David K. Lowenthal et al., *Exploring the Energy-Time Tradeoff in MPI Programs on a Power-Scalable Cluster*, Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), 1530-2075/05.

[EET] Scott Mueller, *Upgrading and repairing laptops*, Que, 2004.

[ACPI] Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., Toshiba Corporation, Advanced Configuration and Power Interface Specification, Revision 4.0a April 5, 2010.
<http://http://www.acpi.info/>

[NAS] D.Bailey, E.Barszcz, J.Barton et al *The NAS Parallel benchmarks*, RNR Technical Report RNR, March 1994.

[ETC] Pablo Alcalde San Miguel *Electrotecnia*, Thomson Paraninfo, 4º Edición, 2003.

[CELC] José Gómez Campomanes *Circuitos Eléctricos, Volumen 1*, Universidad de Oviedo.

[FFUND] Víctor Manuel González Cabrera *Física fundamental*, Editorial Progreso, 2ª reimpresión, 2004.

[SUR] Yongpeng Liu1, Hong Zhu, *A survey of the research on power management techniques for high-performance systems*, Published online 13 January 2010 in Wiley Online Library (wileyonlinelibrary.com).

[ACPIW] "Advanced Configuration & Power Interface" Internet: http://www.acpi.info/, August 23, 2010.

[VTUNE] Dr. David Levinthal *Cycle Accounting Analysis on Intel® CoreTM2 Processors*, Intel Corp.

[CMOSPOW] Anantha P. Chandrakasan, Robert W. Brodersen *Minimizing Power Consumption in CMOS Circuits*, Department of EECS. University of California at Berkeley

[CMOS] R. Jacob Baker *CMOS: Circuit Design, Layout, and Simulation*, Wiley-IEEE, 2010

[ST] "SpeedStep" Internet: http://en.wikipedia.org/wiki/SpeedStep, Marzo, 15, 2011.

[Q6600] "Intel® Core™2 Quad Processor Q660" Internet: http://ark.intel.com/Product.aspx?id=29765", Mayo, 29, 2011.

[QPUMP] "Quad Data Rate" Internet: http://en.wikipedia.org/wiki/Quad_Data_Rate", Octubre, 10, 2010.

[NASW] "NAS PARALLEL BENCHMARKS" Internet: http://www.nas.nasa.gov/Resources/Software/npb.html", Junio, 29, 2010.

[CELERON] “Intel® Celeron® D Processor 34” Internet: http://ark.intel.com/Product.aspx?id=27125”, Mayo, 31, 2011.

[STREAM] “STREAM: Sustainable Memory Bandwidth in High Performance Computers” Internet: http://www.cs.virginia.edu/stream/, Junio, 11, 2011

[ORNL] “Oak Ridge National Laboratory” Internet: http://www.ornl.gov/, Junio, 13, 2011

[TOP500] “Top 500 SuperComputers site” Internet: http://http://www.top500.org/, Junio, 13, 2011

[P4CM] “Dave Jones & Linux Open Source Stuf” Internet: http://codemonkey.org.uk/2009/01/18/forthcoming-p4clockmod, Junio, 11, 2011

[MOSFET] “MOSFET” Internet: http://ccpot.galeon.com/enlaces1737099.html, Junio, 16, 2011

[CMOSDEMO] “CMOS gates demonstration” Internet: http://tams-www.informatik.uni-hamburg.de/applets/cmos/ Junio, 17, 2011

[GREEN] “The Green 500” Internet: http://www.green500.org/ Junio, 18, 2011

[HPCORNL] Buddy Bland, Presented November 16, 2010. “HPC @ ORNL. Where do we go from here?” Internet: http://computing.ornl.gov/SC10/documents/SC10_Booth_Talk_Bland.pdf, Junio, 18, 2011

[OPCOST] Jim Rogers, “Deploying Large Scale Cray XT Systems at ORNL” Internet: http://www.cug.org/5-publications/proceedings_attendee_lists/CUG09CD/S09_Proceedings/pages/authors/16-18Thursday/16B-Rogers/rogers_slides.pdf, May 7, 2010

[GBB] “Green Building Blocks - Software Stacks for Energy-Efficient Clusters and Data Centres” Internet: http://ercim-news.ercim.eu/en79/special/green-building-blocks Junio, 19, 2011

[GRCO] “An Incredible Discovery: Graphene Transistors Self-Cool” Internet: http://www.dailytech.com/article.aspx?newsid=21285 Junio, 18, 2011

[TRGR] “Nuevo transistor de grafeno (IBM)” Internet: http://http://www.neoteo.com/nuevo-transistor-de-grafeno-ibm, Junio, 18, 2011

Firmat: Pedro Erencia

Bellaterra, Juny de 2011

Resumen

El consumo energético es un aspecto cada vez más importante en el diseño de microprocesadores. Este trabajo experimenta con una técnica de control del consumo, el escalado dinámico de tensión y frecuencia (DVFS, siglas en inglés), para determinar cuan efectiva es la misma en la ejecución de programas con diferentes cargas de trabajo, intensivas en cómputo o memoria. Además, se ha extendido la experimentación a varios núcleos de ejecución, permitiendo comprobar en que medida las características de la ejecución en una arquitectura multicore afecta al desempeño de dicha técnica.

Resum

El consum energètic es un aspecte cada cop més important al disseny de microprocessadors. Aquest treball experimenta amb una tècnica de control d'aquest consum, l'escalat dinàmic de freqüència y memòria (DVFS sigles en angles), per tal de determinar com d'efectiva és aquesta en l'execució de programes amb diferents carregues de treball, intensives en còmput o memòria. A més, s'ha estes la experimentació a més d'un núcli d'execució, permetent comprovar en quina mesura les característiques de l'execució en una arquitectura multicore afecta el desenvolupament de l'esmentada tècnica.

Abstract

Power consumption is an increasingly important concern in processor design. This work experiences with one power control technique, dynamic voltage and frequency scaling (DVFS), in order to assess how effective it is in the execution of programs with different workloads, compute-bound or memory-bound. Moreover, the experiments have been extended to more than one core, allowing to investigate in which way the characteristics of such an execution affects the performance of that technique.