UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

GUSTAVO TREIN SALGADO

Estudo sobre o Impacto Energético de Máquinas Virtuais em um Sistema Computacional Físico

Trabalho de Graduação.

Prof. Dr. Alexandre Carissimi
Orientador

Porto Alegre, Julho de 2011.

AGRADECIMENTOS

Gostaria de agradecer a todos que de alguma forma contribuíram para este trabalho. Sobretudo, gostaria de agradecer ao meu orientador por toda a dedicação e atenção prestadas e pelos valiosos ensinamentos e orientações passados que foram fundamentais para a realização deste trabalho.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

EPA	US *Environmental Protection Agency*
TI	Tecnologia da Informação
MV	Máquina Virtual
J	Joule
W	Watt
kWh	quilowatt-hora
E/S (I/O)	Entrada/Saída (In/Out)
APM	*Advanced Power Management*
ACPI	*Advanced Configuration and Power Interface*
ARP	*Address Resolution Protocol*
MAC	*Media Access Control*
IP	*Internet Protocol*
WOL	*Wake on LAN*
ICMP	*Internet Control Message Protocol*
ARP	*Address Resolution Protocol*
ISA	*Instruction Set Architecture*
ABI	*Application Binary Interface*
API	*Application Programming Interface*
JVM	*Java Virtual Machine*
MMV	Monitor de Máquina Virtual
HAV	*Hardware Assisted Virtualization*
VMCB	*Virtual Machine Control Block*
GPL	*General Public License*
NPB	*NAS Parallel Benchmark*
NAS	*NASA Advanced Supercomputing*
CFD	*Computing Fluid Dynamics*
SPEC	*Standard Performance Evaluation Corporation*
UDP	*User Datagram Protocol*
TCP	*Transmission Control Protocol*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A utilização eficiente da energia é um assunto interdisciplinar. Na computação, o consumo, anteriormente problema exclusivo de sistemas embarcados e notebooks, está sendo objeto de estudo de todas as plataformas. Desta forma, a Computação Verde está ganhando espaço, diversas ferramentas e técnicas estão sendo propostas para garantir uma melhor utilização dos recursos energéticos em diferentes componentes do computador. Algumas destas técnicas baseiam-se na redução de desperdícios energéticos, outras permitem uma redução no desempenho em prol de uma menor dissipação de potência.

A virtualização é uma técnica que traz diversas vantagens para plataformas computacionais. No contexto da computação verde, a virtualização pode aumentar a eficiência energética de sistemas computacionais ao garantir uma utilização mais eficiente dos recursos de hardware.

Neste contexto, um estudo é proposto para investigar o consumo energético de máquinas virtuais e para avaliar a influência da redução da frequência de processamento no consumo energético de um *desktop*.

Palavras-Chave: computação verde, consumo de energia, virtualização, *frequency scaling*.

ABSTRACT

Power efficiency is an interdisciplinary subject. In computing systems, dissipation, previously an issue of embedded systems and laptops is becoming study subject of all platforms. Thus, Green Computing is gaining importance. Several tools and techniques are being proposed to ensure better power utilization in various computer components. Some of these techniques on reducing energy waste, others allow performance reduction in order to reduce the power dissipation.

Virtualization is a technique that brings many advantages to computer platforms. In Green Computing context, the virtualization can increase energy efficiency in computing systems by ensuring a more efficient usage of hardware resources.

In this context, a study is proposed to investigate the energy consumption of virtual machines and to evaluate how lowering the processing frequency can influence in the energy consumption.

Keywords: green computing, energy consumption, virtualization, frequency scaling.

1 INTRODUÇÃO

A geração de energia no mundo hoje está baseada, na sua maior parte, em combustíveis fósseis. Tais fontes são poluentes e não renováveis. Com a crescente preocupação ambiental observada nas ultimas décadas, a diminuição do consumo de energia e sua melhor utilização é um assunto em pauta em múltiplas áreas de estudo.

Na computação, o consumo de energia já é há algum tempo, motivo de preocupações em sistemas embarcados e computadores portáteis. Preocupados com a duração das baterias e a dissipação de calor de tais sistemas, estudiosos e desenvolvedores propuseram diversas técnicas para se reduzir o consumo energético em plataformas portáteis. Contudo, a constante demanda por melhor desempenho, aliada ao aumento do custo energético propôs o problema do consumo de energia aos desktops e servidores, anteriormente alheios a ele.

O conceito de *Green Computing* (Computação Verde) foi introduzido em 1992, quando o *US Environmental Protection Agency* (EPA) lançou o padrão *Energy Star* [RUTH, 2009]. Tratava-se de uma certificação fornecida a equipamentos eletrônicos energeticamente eficientes. A partir de então, o *Energy Star* passou a ser uma certificação importante e reconhecida no mercado. Diversos fabricantes concentraram esforços no desenvolvimento de produtos energeticamente mais eficientes para responder a essa demanda de mercado.

Um dispositivo energeticamente eficiente consome uma quantidade de energia proporcional à quantidade de trabalho que ele produz [GUNARATNE et al., 2005]. Desta forma, todo o aumento na quantidade de potência dissipada por esse equipamento é justificada por um aumento proporcional na quantidade de trabalho produzido por esse aparelho.

Em plataformas computacionais, a quantidade de trabalho executada pode ser – grosso modo – representada pelo número de instruções executadas pelo processador. Desta forma, melhorar a eficiência energética de um computador é equivalente a diminuir a relação energia por instrução executada. Computadores convencionais não são energeticamente eficientes. A relação energia consumida por trabalho produzido é bastante inferior em momentos de ociosidade do processador. Desta forma, a diminuição da quantidade de potência dissipada não se ajusta corretamente com a diminuição da quantidade de trabalho que o computador produz [BROWN et al., 2010]. Isso indica que, quando ociosos, os computadores desperdiçam energia. Esse problema é agravado, pois a maioria das plataformas computacionais passa uma grande parte do tempo operando com uma carga relativamente baixa de tarefas.

Levando-se em conta esse cenário, conclui-se que, para se utilizar a energia mais eficientemente em sistemas de Tecnologia de Informação (TI), deve-se evitar que os computadores fiquem ociosos. Para tal, duas medidas devem ser tomadas: i) quando não a trabalho a fazer, os componentes de hardware devem ser desligados (ou suspensos) e ii) quando ativado, o computador deve permanecer com alto nível de ocupação. Para se cumprir a primeira medida, soluções em hardware (em nível de processador, memória e

E/S) provêem aos dispositivos estados de inatividade, onde eles podem ter partes desativadas. A segunda medida é mais complicada de ser seguida. A quantidade de trabalho requerido a um computador em um determinado momento não é determinística. Os processos podem requerer uso do processador e acesso aos periféricos de forma imprevisível, segundo a vontade do usuário ou conforme o resultado de alguma computação.

Neste contexto, a virtualização surge como alternativa para se aumentar a eficiência energética de computadores. Máquinas virtuais (MV) fornecem uma plataforma abstrata de execução, independente da máquina física. Desta forma, diversas máquinas virtuais podem executar em uma mesma máquina física, garantindo uma maior utilização dos recursos dessa máquina e liberando outras máquinas físicas (possibilitando assim a sua desativação). Essa configuração é especialmente interessante em ambientes de *data centers*, onde diversas máquinas físicas provêm diferentes serviços. Ao se executar os serviços em máquinas virtuais, é possível desacoplá-lo da máquina física, permitindo-se que diversos serviços sejam executados em um mesmo servidor (quando a demanda no *datacenter* for pequena) e garantindo-se ainda um sistema robusto, capaz de suportar picos de ocupação dos servidores.

1.1 Objetivos do Trabalho

Este trabalho tem dois objetivos principais. Primeiramente, motivado pelos benefícios energéticos providos pela virtualização, pretende-se investigar o impacto energético resultante da execução de máquinas virtuais em uma máquina física. Para tal, propõem-se uma análise baseada nos conceitos fundamentais da virtualização e em dados reais sobre consumo energético de máquinas virtuais. O objetivo deste estudo, em suma, é analisar o custo energético de uma máquina virtual, mostrar que, como mencionado no início deste capítulo, a adição de novas máquinas virtuais aumenta a eficiência energética da máquina física e mensurar o sobrecusto oriundo do gerenciamento das máquinas virtuais.

O segundo objetivo deste trabalho é analisar o impacto da redução da frequência de processamento no desempenho e consumo energético de um computador. Para tal, determinadas aplicações serão executadas em duas frequências diferentes. O tempo e consumo energético resultante da execução de cada aplicação serão comparados para as duas frequências. Os benefícios e prejuízos de cada abordagem serão comentados.

1.2 Organização do Trabalho

O restante deste trabalho está dividido da seguinte forma: o capítulo 2 apresenta a questão do consumo energético em plataformas computacionais. Além disso, são apresentadas diversas técnicas para se reduzir o consumo de energia em nível de hardware e software. O capítulo 3 é sobre virtualização. A teoria que possibilita a compreensão dos conceitos básicos de virtualização será discutida. O capítulo 4 consiste na apresentação e análise dos resultados experimentais. Dois cenários de testes diferentes foram propostos para se mensurar o consumo de máquinas virtuais e o impacto da redução da frequência de processamento no desempenho e consumo energético de uma máquina física. Finalmente o capítulo 5 apresenta as conclusões do trabalho.

2 CONSUMO DE ENERGIA EM COMPUTAÇÃO

Nos últimos anos, o consumo energético se tornou um requisito imprescindível no projeto de sistemas computacionais. Estima-se que cerca de dois por cento de toda a emissão de dióxido de carbono no mundo deva-se à tecnologia da informação (TI) [NEWING, 2010]. Um *desktop* moderno, por exemplo, pode consumir uma tonelada de CO_2 durante sua vida útil [CARISSIMI et al., 2010]. Neste contexto, a computação verde - área que estuda alternativas para uma utilização mais eficiente dos recursos energéticos em sistemas de TI – está ganhando grande importância.

As técnicas adotadas pela computação verde podem ou não basear-se na diminuição do desempenho da máquina para aumentar a sua eficiência energética. Soluções que não degradam o desempenho baseiam-se na identificação e eliminação de fontes de desperdício de energia. Em um computador, por exemplo, um dispositivo que não está sendo utilizado pode ser desativado (ou colocado em um estado de baixa atividade) gerando menos consumo, mas mantendo o desempenho máximo para a aplicação em execução. Alternativamente, existem abordagens que se baseiam na redução do desempenho visando otimizar o consumo energético. Ao processar uma tarefa cujo tempo de execução é flexível, um processador pode ter a sua frequência reduzida, se isso representar um menor consumo energético global na execução da tarefa.

Nesta seção, serão apresentadas algumas das técnicas de computação verde existentes. Primeiramente será feita uma pequena introdução sobre o conceito físico de energia e como ela é utilizada em diferentes componentes computacionais. Em seguida, serão apresentadas técnicas em nível de hardware e software que diminuem o consumo energético em plataformas computacionais. Por fim, serão mencionadas algumas ferramentas para se medir consumo em computadores.

2.1 Conceitos Básicos

Energia é uma grandeza física que representa a capacidade de se realizar trabalho. Ela é medida em Joules (J) ou em quilowatt-hora (1 kWh = 3600 kJ). Potência é a taxa com que a energia é transferida, medida em Watts (W = Joules por segundo). A relação entre energia e potência é a seguinte:

$$P = \frac{dE}{dt} \qquad \text{(expressão 2.1)}$$

Sendo *P* a potência, E a energia e *t* o tempo.

Mais especificamente, de acordo com [append CARISSIMI et al., 2010], a energia utilizada para a realização de um programa executado em um processador baseado em *switches* CMOS pode ser expressa como:

$$E_{op} = C * {V_{dd}}^2 * f * \left(\frac{N}{f}\right) + I_{fuga} * V_{dd} * \left(\frac{N}{f}\right) \qquad \text{(expressão 2.2)}$$

Onde E_{op} é a energia, *C* a capacitância de chaveamento, V_{dd} é a voltagem na qual o circuito está operando, f é a frequência de operação, I_{fuga} a corrente de fuga e *N* o número de ciclos requeridos para executar o programa.

Analisando-se a expressão, uma boa ideia para se diminuir o consumo é reduzir a voltagem de alimentação (V_{dd}). Contudo, por razões físicas, a frequência de processamento deve ser diminuída para que tal mudança seja possível. A redução da frequência faz com que o processador execute instruções em um ritmo menor, prejudicando o seu desempenho. Surge então, um compromisso consumo energético versus desempenho.

Em termos de componentes de hardware, em geral, o maior dissipador de potência em uma plataforma computacional é o processador. Processadores modernos podem dissipar mais de 100 Watts de potência, consumindo energia e gerando calor [GARRET, 2008]. Os processadores de hoje são formados de vários núcleos que podem rodar a altas frequências mesmo se a demanda de processamento não for condizente com este desempenho.

Outros componentes de um computador também merecem ter seu consumo energético avaliado. Uma unidade de memória DDR3 pode dissipar cerca de 3,3 Watts [GARRET, 2008]. Placas gráficas, como por exemplo a GeForce GTX 590 pode consumir impressionantes 365 W [NVIDIA, 2011]. Dispositivos de entrada e saída também consomem uma significativa quantidade de energia. Uma placa de rede, por exemplo, pode consumir 14 Watts e uma unidade ótica de disco, 5,6 Watts em um estado normal de leitura [CHIN, 2010]. Um monitor de LCD de 15,6 polegadas consome 16W [SAMSUNG, 2011].

Unidades de disco rígido são também grandes consumidores de energia. Em computadores de uso pessoal, um disco rígido consome tipicamente entre 5 e 15 Watts, dependendo do modelo e da velocidade de rotação [TOMSHARDWARE,2011]. Em *data centers*, onde vários discos são utilizados para o armazenamento de dados, o consumo de potência pode chegar a valores alarmantes. Em 2006 a taxa média de consumo de servidores era de 14 kWatts por *rack* [TOMSHARDWARE, 2011]. Além disso, boa parte dessa potência é dissipada em forma de calor, fazendo-se necessário o uso de sistemas de refrigeração que também consomem energia.

2.2 Estratégias Básicas para a Redução do Consumo de Energia

Dissipação de potência é um problema que deve ser considerado em dispositivos de hardware. Visando aumentar o tempo de carga das baterias, reduzir o consumo energético, diminuir a temperatura de operação dos dispositivos, aumentar a vida útil dos sistemas, aumentar o tempo de operação das baterias, os projetos de hardware apresentam cada vez mais suporte ao gerenciamento de consumo energético.

De forma simplificada, as funcionalidades oferecidas pelos componentes de hardware se dividem em dois grupos: as que exploram períodos de ociosidade e as que se aproveitam da possibilidade de se utilizar os recursos computacionais abaixo do seu limite. Os *idle states* e *frequency scaling*, respectivamente, ilustram essas duas abordagens.

Reduzir o consumo quando ocioso – *idle states:* Pouco trabalho demanda pouca energia. Se um recurso não é necessário, ele não deve ser utilizado. Estas, provavelmente, são as idéias mais básicas e naturais que existem em relação a consumo e desperdício. Contudo, a maioria dos componentes de hardware não as leva em conta, apresentando baixa eficiência energética quando operados em baixo nível de utilização [BROWN et al., 2010].

Para evitar desperdício energético, o hardware, quando ocioso, deve consumir muito menos energia do que em estado pleno de execução. Para isso, o dispositivo deve ter a capacidade de desativar os seus componentes que não estão em uso. Em longos períodos de ociosidade, diferentes estados de inatividade – *idle states* - podem ser definidos. Desta forma, o dispositivo pode desativar gradualmente as suas funcionalidades. Esta estratégia considera que, se o dispositivo está a um longo tempo sem uso (o conceito de longo tempo varia - cerca 20 ms para o processador e 2 segundos para o disco [LESS WATT, 2011] ele provavelmente ficará ainda muito mais tempo sem ser utilizado. Deve-se, contudo, mensurar o custo – em termos energéticos e de tempo – da reativação do dispositivo. Quanto mais profundo o *idle state*, menos energia ele consome, contudo, maior é o custo da reativação do componente. O dispositivo deve permanecer sem uso tempo suficiente para que tais custos sejam compensados. Processadores, memórias e discos utilizam *idle state* para reduzir consumo energético.

Reduzir o consumo à custa do desempenho - *Frequency Scalling:* Diminuir a frequência de processamento reduz linearmente o seu consumo de energia (expressão (2)) e permite a redução da tensão de alimentação do processador. Desta forma, reduzir a frequência de processamento provoca um grande impacto na potência dissipada. Fazendo-se um cálculo simples (utilizando-se a expressão (2)), conclue-se que a redução da tensão de alimentação de 1,3V para 1V garante, grosso modo, 40% de redução na potência dissipada. Levando isso em conta, se uma aplicação não impuser uma forte restrição no seu tempo de execução, ela pode ser processada mais lentamente, gerando um grande ganho na relação instrução por energia consumida.

Processadores modernos apresentam ferramentas de hardware que permitem variar a frequência de processamento. Essa variação pode ser feita sob demanda ou conforme alguma política definida pelo sistema operacional ou, ainda, pelo usuário. A seção 2.3.1. detalha a implementação em hardware desse mecanismo e a seção 2.4.1. ilustra a sua utilização em um sistema operacional GNU/Linux.

2.3 Mecanismos de Hardware para Reduzir o Consumo de Energia

O hardware de sistema computacional pode ser dividido em processador, memória e dispositivos de entrada e saída (E/S). O processador é o componente principal de um computador. Ele executa as tarefas, gerencia o sistema e dá ao usuário acesso às outras partes do hardware. A memória é espaço de trabalho do processador. Ela provê um armazenamento volátil onde o processador mantém os dados importantes para a execução das tarefas. A memória também é utilizada para comunicar dispositivos de entra e saída e o processador. Os dispositivos de entrada e saída servem para comunicar o computador com o mundo externo. Os discos, por exemplo, armazenam permanentemente dados e programas. As interfaces de rede permitem ao computador se comunicar com outros computadores.

Nesta seção, serão apresentadas funcionalidades e ideias que visam melhorar a eficiência energética de dispositivos de hardware tais quais processadores, memórias e dispositivos de E/S.

2.3.1 Reduzindo o Consumo Energético do Processador

Considerando a grande quantidade de potência dissipada por processadores, várias soluções foram introduzidas em CPUs modernas para se melhorar a sua eficiência energética.

APM (*Advanced Power Management*): A APM, criada pela Intel em 1992, foi a primeira interface de gerenciamento de consumo entre o firmware e o sistema operacional. A APM utiliza uma abordagem em camadas para comunicar dispositivos com o sistema operacional. O hardware é controlado pela APM-aware BIOS que se comunica com um driver APM específico para o sistema operacional. A comunicação ocorre em ambos os sentidos: eventos de gerenciamento de energia são enviados a partir do BIOS para o driver APM, o driver APM envia informações e pedidos à BIOS através de chamadas de função.

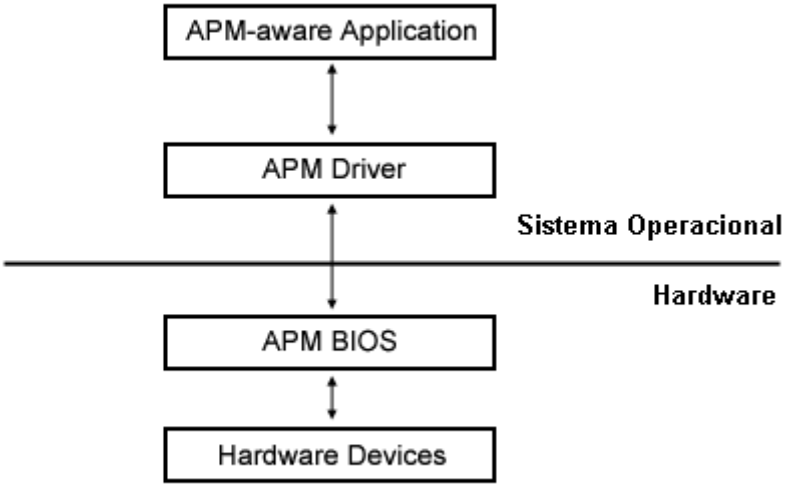

Figura 2.1: Camadas do APM

Através da APM, o sistema operacional pode obter informações pertinentes à gerência de consumo (status da bateria, por exemplo) e colocar o sistema em diferentes estados de menor atividade.

ACPI (*Advanced Configuration and Power Interface*): Lançada em dezembro de 1996, a ACPI é o sucessor do APM. Ela especifica interfaces e conceitos para gerenciamento de potência, integrando sistema operacional, drivers, hardware e aplicação. A ACPI dá ao sistema operacional toda a responsabilidade do gerenciamento de consumo, tendo em vista que é ele que está na melhor posição para avaliar o estado de funcionamento global do sistema. A ACPI utiliza duas modalidades básicas para gerenciamento de consumo que serão apresentadas a seguir.

A primeira forma é denominada *running x suspended*. Esta é a implementação dos *idle states* (seção 2.2). Nesta modalidade, o dispositivo (ou todo o sistema) pode ser desativado ou executar com menor desempenho. Para tal, a ACPI define *power states*.

Existem 7 *power states* para o sistema (C0 a C6). C0 indica o estado normal de execução. Os demais estados permitem diferentes níveis de redução de consumo (*suspended states*). Na CPU, a execução de instruções pode ser suspensa e o circuito de relógio desligado. Em estados que implementam um gerenciamento de consumo mais agressivo (C5, C6), algumas partes do hardware são desativadas (caches, buffers e controladores de memória). Quanto mais partes do dispositivo são desativadas, mais energia é economizada. Contudo, é necessário ter em mente que estados de desativação mais profundos requerem mais tempo, e mais energia, para voltarem ao estado normal de execução. Cabe ao sistema operacional avaliar o nível de atividade do processador e fazer a troca de estado. Para permitir um gerenciamento energético eficiente, o hardware deve informar ao sistema operacional sobre o consumo para cada um dos estados, o tempo de transição entre os eles e a quantidade de energia gasta nas trocas de estado.

A segunda forma utilizada pela ACPI para gerenciar o consumo energético é baseada na ideia de *frequency scalling* (seção 2.2). São definidos *P states* (P0 a P15). A cada estado está associado uma frequência de processamento. P0 é o estado de maior desempenho do dispositivo, enquanto os demais estados vão progressivamente reduzindo a frequência e a tensão de alimentação, garantindo menor dissipação de potência. O ajuste de desempenho é transparente ao usuário, pois ele não interrompe a execução do processador. Pode-se definir uma frequência de processamento constante ou variá-la conforme alguma política determinada pelo sistema operacional ou pelo usuário. Na seção 2.4.1, o gerenciamento da frequência de processamento será detalhado. Processadores modernos apresentam a possibilidade de *frequency scaling*. SpeedStep da Intel e Power Now! da AMD são exemplos de tecnologias que possibilitam o uso deste conceito.

2.3.2 Reduzindo o Consumo Energético da Memória

Enquanto muitos mecanismos para o gerenciamento energético em processadores foram desenvolvidos, pouco foi feito em relação às memórias RAM. O consumo de potência em dispositivos de memórias tem crescido muito nos últimos anos. Em alguns sistemas portáteis, as memórias podem ser responsáveis por mais de 50% do consumo total do sistema [SAXE, 2010].

Baseando-se na idéia de *idle states*, fabricantes de memória estão desenvolvendo dispositivos com múltiplos estados de consumo energético. Um exemplo disso é a tecnologia Direct Rambus DRAM (RDRAM) que define quatro estados de desempenho: *active* , *standby*, *nap* e *power-down*. Para funcionamento normal de leitura e escrita, o chip deve estar no estado *active.* Os demais estados estão em ordem decrescente de consumo energético, mas em ordem crescente latência de reativação. O estado *power-down* desliga a maioria dos circuitos possíveis da memória, realizando apenas *refresh* nas células [FAN et al., 2002].

Ao utilizar esta possibilidade, o controlador de memória deve considerar o fato de que o funcionamento de um computador baseia-se na constante interação entre processador e memória. Sendo assim, um baixo desempenho da memória pode representar um gargalo no processamento.

2.3.3 Reduzindo o Consumo Energético do Disco

Comparado com o processador, os discos consomem menos energia. Um disco rígido comum de 7.200 RPM consome aproximadamente 8 Watts, cerca de 10% do que um processador multicore consome [BROWN et al., 2010]. Apesar disso, a crescente demanda por espaço de armazenamento em servidores e o rápido aumento da potência dissipada em discos de última geração, tem tornado o consumo energético um problema a ser considerado nos projetos de disco rígido. Discos modernos podem ser utilizados em diferentes modos de operação. Em modos de menor consumo, pode-se limitar a utilização de *caches*, se reduzir o tempo de ociosidade necessário antes de se desativar o disco, etc. Algumas ferramentas existentes permitem a exploração dessas funcionalidades.

Hdparm é uma ferramenta que permite parametrizar o funcionamento do disco rígido. Através dele podemos gerenciar *caches*, *sleep modes*, consumo energético etc. Hdparm permite duas abordagens diferentes para o gerenciamento de consumo. Primeiro, pode-se definir 256 valores diferentes de controle de consumo. Valores baixos permitem um controle de consumo agressivo reduzindo bastante a velocidade de rotação do disco, valores altos permitem um melhor desempenho. A segunda abordagem utilizada pelo Hdparm, baseia-se no tempo que o disco deve passar inativo para que o motor seja desligado. O *timeout* pode variar entre 5 segundos e várias horas. Contudo, deve-se novamente mensurar o custo temporal e energético de recolocar o dispositivo no seu funcionamento normal.

Um bom gerenciamento energético do disco requer considerações sobre seu funcionamento. Na falta de dados na cache, uma leitura obriga acesso ao disco. Uma aplicação pode, no início de sua execução, ler todos os dados que ela vai potencialmente precisar, evitando futuros acessos e deixando o disco mais tempo em modo de menor consumo.

Laptop Mode é uma ferramenta GNU/Linux que foi criada com o intuito de manter o disco inativo por mais tempo. Considerando o custo de recolocar o disco em pleno funcionamento antes de um acesso, o sistema pode manter em cache os dados a serem escritos no disco até que o acesso ao disco seja inevitável (quando um processo faz uma leitura ou quando a *cache* fica completamente cheia). Deve-se, neste caso, considerar o risco de perda de dados em caso de falha no sistema ou falta de energia.

2.3.4 Reduzindo o Consumo Energético na Rede

O grande tráfego de dados da Internet consome uma enorme quantidade de energia em ambientes de rede. Os protocolos padrões nos quais as redes de computadores se baseiam, em geral, dão pouca importância à eficiência energética. Um exemplo disso é o protocolo ARP (*Address Resolution Protocol)*. Este protocolo é usado para determinar o endereço de nível de enlace (MAC) do receptor a partir do endereço da camada de rede (IP). Ao executar o protocolo, o emissor envia por *broadcast* um pacote ARP para todas as máquinas da rede perguntando qual o endereço da interface de rede que possui o endereço IP informado. Desta forma, um pacote ARP vai chegar em todas as máquinas da rede e gerar uma interrupção pela placa de rede que deverá ser tratada pelo sistema operacional. Apenas o receptor (ou o *default gateway*, no caso da máquina alvo estar em outra rede) responderá à requisição. As demais máquinas descartarão o pacote. Esse esquema evidencia a péssima eficiência energética do protocolo.

Apesar de pouco suporte por parte dos protocolos de rede, algumas boas ideias para melhorar a eficiência energética na transmissão de dados surgiram, a maioria contando com pequenos suportes de hardware.

***Wake On Lan* (WOL):** Em 1995 a AMD desenvolveu um método conhecido como WOL [GUNARATNE et al., 2005] para "acordar" computadores em uma rede, a partir de sua interface de rede. Em um computador em estado *sleep*, a interface de rede, ao receber um pacote WOL, vai gerar uma interrupção no computador para que este seja reativado. Esse esquema, contudo, exige o conhecimento do endereço MAC da interface de rede. Como já explicado, o emissor deve utilizar o protocolo ARP para obter tal informação. Desta forma, ao tentar acordar uma única máquina, o emissor vai acabar acordando todas as máquinas da rede, ao enviar um ARP em *broadcast*. Para que o esquema de WOL fosse realmente eficiente, a interface de rede deveria ser mais seletiva com os pacotes que exigem reativação de todo o sistema. Além disso, ela deveria ter a capacidade de responder a requisições simples (ICMP, por exemplo), sem a necessidade de reativar o computador. Neste caso, pequenas mudanças no hardware padrão de uma interface de rede (como a adição de um pequeno processador) seriam necessárias para que ela pudesse efetuar esses procedimentos básicos.

Gunaratne et al. [GUNARATNE et al., 2005] analisaram os pacotes recebidos durante 12 horas e 40 minutos em um computador em *idle* conectado a uma rede. Constatou-se, nessa análise, que cerca de 91% dos pacotes poderiam ser descartados (na sua maioria pacotes ARP) ou exigiam mínimo tratamento. Sendo assim, ao se utilizar uma interface de rede capaz de pré-tratar os pacotes recebidos, seria possível reduzir-se a frequência de reativação do sistema pela rede em cerca de dez vezes.

Concentrar Interrupções: Placas de rede modernas, ao invés de interromper o processador a cada pacote que chega, o fazem em intervalos de tempos definidos. Desta forma, possivelmente, o processador será interrompido com menos frequência e tratará vários pacotes a cada interrupção. Deve-se tomar cuidado, no entanto, com aplicações de tempo real devido à latência temporal introduzida por tal técnica.

Escalar a Velocidade da Conexão: A potência consumida para transportar dados cresce com a largura de banda usada. Uma conexão de 1 gigabit, por exemplo, consome cerca de 3 Watts a mais que uma conexão de 100 megabit (que tipicamente consome cerca de 5W a 9W) [GUNARATNE et al., 2005]. Uma abordagem interessante (baseada na redução do desempenho visando uma melhor eficiência energética) seria adaptar a taxa de transmissão de uma conexão em função da demanda. Gunaratne et al. [GUNARATNE et al., 2005] propõe um algoritmo que analisa o tamanho da fila de pacotes em um PC, ou *switch*, para determinar a demanda de pacotes e alterar a velocidade da conexão.

Protocolo PS-Poll (*Power Save Poll*): Em uma rede IEE 802.11, um adaptador de rede sem fio emite constantemente ondas eletromagnéticas para se comunicar com o *access point*. A idéia do PS-Poll é fazer com que a interface de rede emita de forma intercalada, mantendo-se inativa por intervalos de tempo. Para tal, o *access point* deve ser informado sobre o tempo em que a interface ficará inativa para armazenar todos os pacotes destinados a ela durante o período e os enviar posteriormente. Novamente neste caso, deve-se ter cautela com a latência temporal na distribuição dos pacotes.

2.3.5 Reduzindo o Consumo Energético em Outros Componentes de Hardware

Os monitores de LCD são grandes consumidores de energia em CPUs. Visando aumentar o tempo de carga da bateria em sistemas portáteis, sistemas operacionais diminuem o brilho da tela e até a desligam quando o usuário para de interagir com o sistema. Além disso, a Intel introduziu uma tecnologia que monitora a distribuição de cores na tela e permite a diminuição da luz de fundo quando a maior parte da tela está escura. Outra idéia é comprimir o conteúdo do *framebuffer* (*buffer* que armazena um quadro inteiro antes dele ser mostrado na tela). Desta forma, a quantidade de dados a serem lidos pelo hardware é diminuída. Esta otimização pode diminuir o consumo em até 5W[GARRET, 2008]. Existem ainda diversas pequenas idéias e dicas para se reduzir o consumo energético em diferentes plataformas como desabilitar *Wi-Fi*, *Bluetooth* que pode aumentar bastante o tempo de carga da bateria de sistemas embarcados. Desabilitar os *logs* (syslogd) do sistema pode evitar acessos ao disco e, dessa forma, poupar energia [LESS WATT, 2011].

2.4 Mecanismos de Software para Reduzir o Consumo de Energia

Aproveitando-se das funcionalidades fornecidas pelo hardware e do comportamento energético dos seus componentes, projetistas e desenvolvedores de software também são capazes de tornar seus produtos energeticamente mais eficientes. Existem técnicas de software de redução de consumo energético em nível de sistema operacional e de aplicação.

2.4.1 Reduzindo o Consumo Energético do Sistema Operacional

O sistema operacional, através de chamadas de sistema e instruções, se comunica com o processador e outros componentes físicos. Em última análise, é ele que tem o controle sobre o hardware e detém todas as informações úteis para gerenciá-lo.

Mantendo o processador ocioso por mais tempo - *Dynamic Tick Model:* Essa idéia baseia-se no conceito de aproveitar melhor os momentos de ociosidade do processador. Como mencionado anteriormente, transitar entre diferentes estados *idle states* consome tempo e energia. Observando-se especificamente o processador, um estado de inatividade profundo (onde mais partes do hardware são desativadas) otimizará o consumo energético apenas se a CPU ficar tempo suficiente neste estado. Sendo assim, um escalonador que gera interrupções contínuas a cada preempção (tipicamente 100 vezes por segundo) pode ser bastante inconveniente quando a CPU está em estados de baixo consumo. Ao invés de utilizar um temporizador estático, o núcleo do sistema operacional, através do seu escalonador, pode observar os processos em execução e disparar interrupções quando considerar mais conveniente. Idealmente, um temporizador dinâmico permite ao processador ficar quase inteiramente desativado até que um usuário interaja com o sistema (interrupção de hardware).

Gerenciando a Frequência do Processador – *CpuFreq:* A partir do kernel 2.6.12, o GNU/Linux passou a oferecer uma infra-estrutura chamada *cpufreq*. Essa ferramenta se baseia no modelo de redução do consumo energético em detrimento do desempenho. A solução baseia-se no conceito de governadores. Um governador é um algoritmo que calcula a frequência que ele considera adequada para a CPU em um determinado momento. Além dos governadores, é possível selecionar, manualmente, uma frequência específica de processamento. A tabela 2.1 mostra exemplos de governadores.

Tabela 2.1: Governadores para CPU *frequency scaling*. Fonte THINK WIKI, 2010

Governador	Descrição
Performance	Coloca o processador na sua frequência máxima de operação.
Powersave	Coloca o processador na sua frequência mínima de operação.
Ondemand	Verifica a taxa de utilização do processador regularmente. Compara esta taxa com um valor pré definido. Quando a taxa é inferior a este valor, configura o processador na mínima frequência de operação. Quando a carga é superior a este valor, coloca o processador na sua frequência máxima de operação.
Conservative	Define 2 valores: *up_threshold* e *down_threshold*. Verifica a taxa de utilização regularmente. Se a taxa é inferior ao *down_threshold*, coloca a CPU para executar na primeira frequência mais baixa que a atual. Analogamente, se a taxa é superior ao *up_threshold*, coloca a CPU na primeira frequência mais alta que a atual. É mais custoso computacionalmente que o Ondemand.
Userspace	Configura a CPU em uma frequência definida por um programa em espaço de usuário. Permite a uma aplicação controlar a frequência da CPU.

2.4.2 Reduzindo o Consumo Energético da Aplicação

As aplicações não têm acesso direto ao hardware e interagem com ele através do sistema operacional. Contudo, existem diversas técnicas e recomendações que um desenvolvedor pode seguir para tornar a sua aplicação mais verde, isto é, energeticamente eficiente.

Agrupar *Timers*: Aplicações de usuário podem interromper o processador por várias razões: verificar se o usuário moveu o *mouse*, verificar se alguma tecla foi pressionada, se o volume do áudio foi alterado, etc. *Polling* é uma forma simples de se verificar todos esses eventos. Nesta técnica, para cada verificação que deve ser feita, a aplicação gera interrupções contínuas em períodos de tempo definidos. Uma forma de se diminuir o número de interrupções geradas pela aplicação (consequentemente diminuir o consumo energético) é agrupar todas essas verificações em uma mesma interrupção. A figura 2.2a ilustra o exemplo de uma aplicação que interrompe o processador em três intervalos de tempo diferentes. Cada *timer* T1, T2 e T3 representa uma verificação (*polling*) diferente. Em um segundo de execução, a aplicação interrompe o processador nove vezes. Na figura 2.2b, a aplicação é implementada de tal forma a agrupar as interrupções. Isso pode ser feito definindo-se períodos de interrupção que são múltiplos inteiros dos períodos menores (exemplo T3 = 2*T2 = 4*T1) ou flexibilizando-se os períodos, permitindo-se assim, atrasar ou adiantar uma determinada interrupção. Em um segundo de execução, a aplicação da figura 2.2b interrompe o processador 4 vezes.

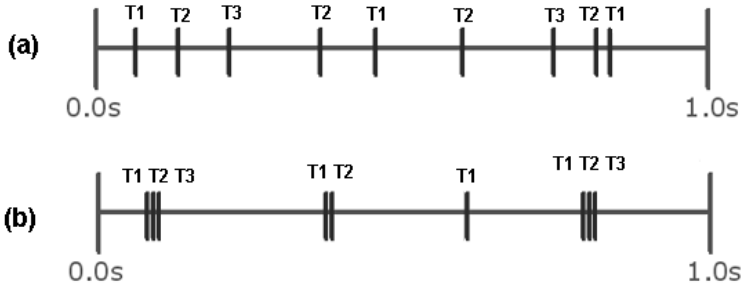

Figura 2.2: Agrupar *timers* para se reduzir o número de interrupções de CPU geradas pela aplicação. (LESS WATT, 2011)

Utilizar *Buffers* Maiores: Dados que são transferidos entre dispositivos de entrada e saída e a memória são armazenados em regiões de memória temporária - *buffers*. Ao definir o espaço de memória ocupado pela aplicação, os desenvolvedores devem criar *buffers* suficientemente grandes para minimizar o número de acessos ao dispositivo de entrada e saída. Uma aplicação que lê e executa faixas de um DVD, por exemplo, deve armazenar em *buffer* pelo menos vinte minutos de vídeo para ter uma boa eficiência energética [LESS WATT, 2011].

Desativar Dispositivos Após o Uso: Uma aplicação, ao terminar o uso de um dispositivo de hardware, deve informar ao sistema operacional que não necessita mais daquele dispositivo. Se isso não for feito, mesmo que a aplicação seja terminada, o sistema operacional vai manter aquele dispositivo em estado de ativação, desperdiçando energia.

Race to Idle: A idéia é finalizar as tarefas no menor tempo possível, para se retornar a um estado de ociosidade o mais rápido. Desta forma, o processador pode passar mais tempo nos *idle states* de mais baixo consumo energético. No exemplo a seguir [LESS WATT, 2011], foi executado um algoritmo de decodificação de duas formas: processador em máxima frequência de operação e na metade dela. O processador consome 34 Watts em frequência máxima, 24 Watts com metade da frequência máxima e 1 Watt em *idle*. Abaixo, o cálculo da energia total consumida pelo processador, durante um segundo.

- metade da frequência total de operação:

$$0{,}5\ segundos * 24\ watts + 0{,}5\ segundos * 1\ watt = 12{,}5\ Joules$$

- frequência total de operação:

$$0{,}25\ segundos * 34\ watts + 0{,}75\ segundos * 1\ watt = 9{,}25\ Joules$$

Com este simples exemplo é possível observar que a economia de energia deve-se à grande diferença de potência consumida pelo estado normal de operação do processador e o estado de *idle*.

2.5 Ferramentas para Medir e Monitorar Consumo de Energia

Dispositivos de hardware, em geral, oferecem poucas ferramentas para medir o seu consumo energético. Alguns softwares existentes fazem uma estimativa de consumo a partir de dados obtidos pelo sistema operacional. Quando se necessita de uma boa precisão e confiabilidade nas medidas de consumo energético deve-se utilizar componentes eletrônicos específicos. A seguir, serão apresentadas algumas das ferramentas existentes.

PowerTop: Powertop é uma ferramenta do GNU/Linux que provê informações sobre o consumo energético dos processos em execução. Basicamente, o software informa quais os processos que mais estão "acordando" a CPU e, consequentemente, consumindo mais energia. Além disso, ele mostra, dentro de um intervalo de tempo definido, a parcela de tempo que a CPU passou em cada um do C-States (seção 2.3.1). Para estimar a potência instantânea dissipada pela máquina, o PowerTop utiliza o nível de carga da bateria, informação fornecida pela ACPI.

Joulemeter: O Joulemeter [JOULEMETER, 2011] é uma ferramenta desenvolvida pela Microsoft para sistemas Windows. Através dele, é possível estimar a quantidade de energia utilizada por um computador, servidor, aplicação ou máquina virtual. Além disso, o Joulemeter também estima o consumo individual de componentes de hardware como CPU, memória, disco rígido e monitor. A ferramenta faz a estimativa através dos níveis de utilização dos componentes de hardware e baseada em dados realísticos de consumo.

Wattímetro: wattímetros são dispositivos eletrônicos capazes de medir a potência instantânea dissipada por um aparelho elétrico. É um instrumento simples e preciso usado para se medir o consumo energético de uma máquina como um todo. Ao ligar o wattímetro à fonte de alimentação de um *desktop*, por exemplo, é possível medir a potência total dissipada por todos os componentes de hardware da máquina. Além disso, de forma simples (através de um circuito integrador), o wattímetro pode informar a quantidade de energia consumida em um determinado intervalo de tempo.

Google Powermeter: Google PowerMeter [GOOGLE PM, 2011] é uma ferramenta de monitoração gratuita que permite observar o consumo energético de uma infraestrutura (uma casa ou um prédio, por exemplo) via Internet. A ferramenta não exige adição ou instalação de nenhum equipamento na residência ou instalação do usuário final. As informações são coletadas na infraestrutura da distribuidora de energia. O Google PowerMeter ainda está em fase inicial de uso, e só está disponível em alguns países (EUA, Canadá, Reino Unido, Alemanha, Austrália, Nova Zelândia). Apesar de, na atual implementação, não ser possível medir o consumo de um dispositivo isolado, o Google PowerMeter pode ser um precursor de uma solução de gerenciamento energético de dispositivos via Internet.

2.6 Considerações gerais

A exemplo de outras áreas de conhecimento, a computação está entrando na era do desenvolvimento sustentável. Projetistas e usuários propuseram e implementaram algumas formas de se utilizar mais racionalmente e eficientemente os recursos energéticos em sistemas de TI. As técnicas implementadas abrangem os níveis de hardware e software. Contudo, ainda há muito o que evoluir antes que se possa considerar os computadores equipamentos energeticamente eficientes. Poucas soluções de redução energética são oferecidas por dispositivos de E/S. Os protocolos de rede são, em sua grande maioria, alheios ao consumo energético e não provêem nenhum suporte neste sentido. Processadores em geral não oferecem nenhum sensor interno para se monitorar o seu consumo energético.

A virtualização é uma técnica promissora para se aumentar a eficiência energética em plataformas computacionais. Várias máquinas virtuais podem se concentrar em uma mesma máquina física. Além disso, elas podem ser migradas de um computador a outro. Desta forma, é possível se imaginar várias máquinas virtuais em uma única máquina física, aumentando a sua taxa de utilização e desocupando outras máquinas físicas. A virtualização é o assunto do próximo capítulo, que descreverá a técnica e suas vantagens.

3 VIRTUALIZAÇÃO

Virtualização não é um conceito novo na computação. O termo máquina virtual é oriundo dos anos 60 para se referir a uma abstração em software de um sistema operacional em hardware. Na época, a computação se baseava em *mainframes*. Esses gigantescos computadores atingiram uma velocidade de processamento relativamente grande que era ineficientemente aproveitada, pois o gerenciamento de processos era feito manualmente por um operador. Para melhor aproveitar os recursos de processamento dos *mainframes*, sugeriu-se que vários processos fossem executados ao mesmo tempo. Desta forma surgiu o conceito de tempo compartilhado (*time sharing*). Neste caso, a CPU executa cada processo durante uma fatia de tempo, se revezado entre eles. Se esta fatia de tempo for pequena o suficiente, um usuário tem a impressão que todos os processos estão executando paralelamente na máquina. Além disto, com esse esquema, múltiplos usuários podem executar seus processos ao mesmo tempo, compartilhando o *mainframe*.

Neste contexto, a virtualização surgiu como uma importante ferramenta. Ao prover uma camada de software que oferece um ambiente completo de execução (sistema operacional, bibliotecas e aplicações), equivalente ao de uma máquina física, as máquinas virtuais garantiam isolamento entre os diversos usuários da máquina física. Além disto, os *mainframes* tinham o hardware bastante diferente uns dos outros, cada um com seu próprio sistema operacional e aplicações. Neste sentido, as máquinas virtuais solucionavam o problema da portabilidade de aplicações e sistemas legados, pois permitiam abstrair os detalhes de implementação das camadas inferiores de software e hardware.

A IBM foi a empresa mais importante nesta área naquela época, desenvolvendo vários sistemas baseados em máquinas virtuais: CP-40, CP-67, VM/370 – primeiro computador comercial inteiramente projetado para a virtualização [BUENO, 2009] - entre outros. Basicamente, as máquinas virtuais propostas pela IBM eram cópias do hardware. Um componente chamado monitor de máquina virtual (MMV) executava diretamente sobre o hardware. Múltiplas máquinas virtuais podiam ser criadas pelo MMV, cada instância rodando o seu próprio sistema operacional. Muitas máquinas virtuais de hoje baseiam-se neste modelo.

Na década de 80, o emprego de máquinas virtuais perdeu importância. Com popularização dos computadores e uniformização de seus hardwares, os sistemas operacionais se restringiram a algumas poucas famílias (Unix, Machintosh e Microsoft), cada um com seu público alvo e suas aplicações.

Entretanto, graças ao aumento do poder computacional dos processadores modernos e a popularização dos sistemas distribuídos, a virtualização ressurgiu como uma ferramenta para explorar mais eficientemente os recursos computacionais. Os benefícios

e aplicações dessa tecnologia em computadores de mesa, servidores, *clouds*, aplicações são muitos: segurança, portabilidade, encapsulamento de um estado de computação, tolerância a falhas, criação de um ambiente de testes seguro, possibilidade de limitação de recursos para QoS, possibilidade de se executar múltiplos sistemas operacionais, consolidação de servidores redução do custo de gerenciamento e de energia, etc. Na seção 3.4 voltaremos a discutir as aplicações da virtualização de forma mais detalhada.

3.1 Conceitos básicos

Um sistema computacional, em sua essência, é constituído por diversas camadas de abstração hierarquicamente sobrepostas. Cada camada executa a sua função e comunica-se com as camadas vizinhas através de interfaces bem definidas. Desta forma, uma camada de abstração é um subsistema que fornece serviço aos demais componentes do sistema, sem precisar conhecer os seus detalhes de implementação.

O funcionamento de um computador pode ser visto, em um nível muito baixo de abstração, como a movimentação de elétrons e lacunas entre transistores e barramentos. Juntando-se alguns transistores é possível construir uma porta lógica. Para tal, não é necessário conhecer-se os níveis de dopagem dos transistores, basta compreender-se o seu comportamento quando se aplica determinadas tensões em seus componentes. Subindo-se mais no nível de abstração, conhecendo-se para um conjunto de entradas, a saída que cada porta lógica gera, é possível construir unidades lógicas mais complexas, como unidades aritméticas, multiplexadores, comparadores, memórias, etc.

Os processadores são formados por diversas unidades lógicas ligadas por barramentos. Para o processador, cada unidade é uma "caixa preta" da qual ele conhece, para cada combinação de entrada, o resultado da lógica aplicada. O processador, contudo, não conhece os detalhes de funcionamento das unidades que o compõem. Subindo-se ainda mais no nível de abstração, encontrando-se o núcleo do sistema operacional que é onde ocorre a comunicação com o processador, a memória e os dispositivos de entrada e saída. A interface utilizada entre sistema operacional e processador é o conjunto de instruções do processador (*Instruction Set Architecture* – ISA). Através da ISA, o sistema operacional e os processos de usuário executam os seus códigos. Um programador de uma linguagem de alto nível, em geral, não precisa estar ciente das instruções de máquina que seu código vai gerar, nem mesmo qual o modelo ou fabricante do processador. Para criar uma aplicação, ele apenas deve conhecer os detalhes da linguagem de programação. O núcleo do sistema operacional vai lhe fornecer diversas abstrações para manipular o hardware – sistema de arquivos, interfaces de rede, chamadas de função – e o compilador vai traduzir estes comandos em código binário para a arquitetura alvo.

Apesar de ser um conceito extremamente poderoso e fundamental para o desenvolvimento de sistemas complexos, como computadores, a abstração tem suas limitações. Subsistemas e componentes estão limitados às interfaces as quais eles foram projetados para se comunicar. O código binário de uma aplicação, por exemplo, está limitado à arquitetura - ISA - a qual ele foi compilado e a um sistema operacional. A virtualização aparece como solução para eliminar esta dependência ao mapear uma interface para outra com a introdução de uma camada intermediária de adaptação. Esse mapeamento baseia-se no conceito de isomorfismo. Para um conjunto de ações *s* que faz um sistema real passar de um E_i a um estado E_j, deve existir um conjunto de ações *s'* – possivelmente diferente de *s* - que faz o sistema virtualizado passar de um estado

E_i' (equivalente a E_i) a um estado E_j' (equivalente a E_j). Desta forma, o sistema real pode aparecer como um sistema virtualizado equivalente ou mesmo como múltiplos sistemas virtuais.

Ao contrário da abstração, a virtualização não visa esconder detalhes internos, mas sim mostrá-los de forma diferente da real. Um bom exemplo para se comparar abstração e virtualização são sistemas de arquivos e discos virtuais [MITH et al. 2001]. A figura 3.1a mostra a abstração do disco em um sistema de arquivos. Neste caso, o sistema operacional esconde do usuário o funcionamento interno do disco, provendo a ele uma interface simples de pastas e documentos. Já na figura 3.1b, a virtualização transforma um único disco em dois discos lógicos menores, cada um com suas interfaces virtuais equivalentes às interfaces reais de um disco real – segmentos, blocos e partições. O sistema de arquivos, neste caso, é uma camada de abstração intermediária para mapear os discos virtuais em um disco real. Diferentemente do particionamento, onde simplesmente dividiríamos o disco em partições estáticas, a virtualização consiste em interpretar arquivos como discos. Desta forma, os discos virtuais podem ser copiados, movidos e deletados de forma simples e flexível.

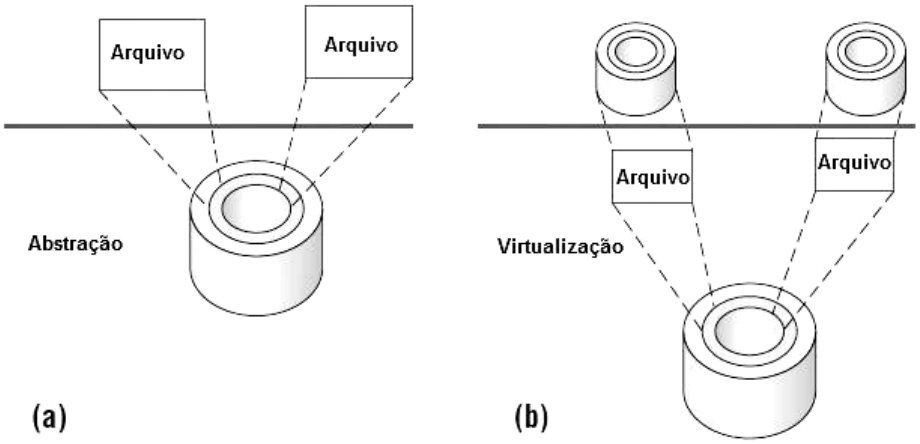

Figura 3.1: Abstração evVirtualização. a) Abstração provê uma interface com os dispositivos das camadas subjacentes. b) Virtualização provê uma interface diferente no mesmo nível de abstração. Adaptado de (MITH et al. 2001)

Como visto, um computador pode ser visto como uma composição de subsistemas que formam camadas de abstração que interagem entre si por interfaces bem definidas. Para se compreender o funcionamento e os conceitos relacionados à implementação de máquinas virtuais, é necessário o conhecimento de aspectos fundamentais de algumas destas camadas. Serão apresentados na próxima subseção conceitos básicos de arquitetura de computadores e sistemas operacionais.

3.1.1 Arquitetura de Computadores e o Sistema Operacional

O conjunto de instruções referentes a uma arquitetura específica de máquina –ISA - é a interface que delimita a fronteira entre o hardware e o software. O sistema operacional – ou mesmo uma aplicação – (peças de software) interage com o processador (peça de hardware) através de instruções de máquina que nada mais são que códigos binários. O processador garante também que o sistema operacional interaja com outros componentes de hardware (memória e dispositivos de entrada e saída) através destas instruções. A figura 3.2 ilustra um sistema computacional genérico no ponto de vista de seus componentes e suas interfaces.

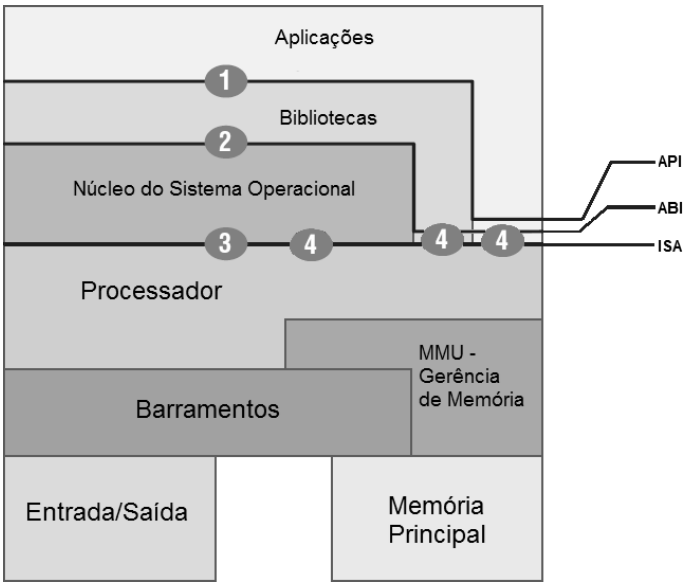

Figura 3.2 – Sistema computacional como um conjunto de camadas de abstração. Adaptado de (SMITH et al. 2001)

Tipicamente, o conjunto de instruções de um processador é dividido em dois subgrupos que definem dois modos de operação: modo não privilegiado e modo privilegiado. No modo não privilegiado, todos os processos podem executar o subconjunto de instruções não privilegiados – *user* ISA (interface 4 da figura 3.2). No modo privilegiado, além das instruções *user* ISA, é possível a execução de instruções capazes de configurar o comportamento do processador e acessar diretamente componentes de hardware (instruções de E/S, por exemplo). Esse subconjunto de instruções é denominado *system* ISA (interface 3 da figura 3.2) e só pode ser executado pelo núcleo do sistema operacional.

O sistema operacional é a camada de software inserida entre o hardware e as aplicações que executa tarefas para os usuários e cujo objetivo é tornar a utilização do computador, ao mesmo tempo, eficiente e conveniente [SILBERCHATZ, 2001]. Um conceito fundamental de sistema operacional são os processos. Um processo é a abstração de um programa em execução. Os processos de usuário executam apenas em modo não privilegiado. Contudo, em algumas situações, eles precisam executar instruções do conjunto *system* ISA, como por exemplo, para realizar E/S. A interface entre aplicações de usuário e hardware é provida por intermédio do sistema operacional pelas chamadas de sistema. A aplicação gera uma chamada de sistema para o sistema

operacional, e este acessa o hardware através da instrução adequada. Desta forma, uma aplicação de usuário interage com o hardware através do sistema operacional (chamadas de sistema) e através do *user* ISA do processador. A junção destas duas interfaces é denominada ABI (*Application Binary Interface*). A ABI define, portanto, a interface entre o processo e a plataforma onde ele está executando

Para simplificar o uso das chamadas de sistema, o sistema operacional provê códigos que encapsulam as chamadas ou desempenham funções compostas mais complexas. Esses códigos são chamados de bibliotecas de funções. O conjunto de funções de uma biblioteca é chamado de API (*Application Programming Interfaces*) (interface 1 da figura 3.2). Uma API dá ao programador acesso ao hardware através de chamadas de sistema encapsuladas para uma linguagem de alto nível. As chamadas de sistema são, em última instância, realizadas pelas bibliotecas (interface 2 da figura 3.2). O código objeto de uma biblioteca, juntamente com o código objeto da compilação de um programa fonte, forma o código executável.

A visão de um sistema computacional como uma sobreposição de camadas de abstração que se comunicam por interfaces é a base para a construção de máquinas virtuais. Pode-se construir uma máquina virtual, fornecendo-se a uma aplicação uma ABI virtual. Em contrapartida, uma camada de software, executando diretamente no hardware ou sobre um sistema operacional hospedeiro, pode oferecer às aplicações diversas ABIs diferentes [CARISSIMI, 2009]. Essas duas formas de se virtualizar sistemas são a base para a construção de máquinas virtuais de processo e máquinas virtuais de sistema, que serão apresentadas a seguir.

3.1.2 Máquinas Virtuais de Processo x Máquinas Virtuais de Sistema

Para se compreender o conceito de máquina virtual, é importante se considerar duas visões diferentes sobre o conceito de "máquina". Do ponto de vista de um processo executando um programa de usuário, uma máquina consiste em um espaço de endereçamento lógico onde as instruções e dados do programa residem. A interface entre o processo e a máquina é a ABI que ela fornece. Desta forma o processo pode ser executado através do conjunto de instruções não privilegiadas (*user ISA*) e do conjunto de chamadas de sistema oferecidas pelo sistema operacional.

Do ponto de vista do núcleo do sistema operacional, uma máquina é um ambiente completo de execução que pode executar múltiplos processos simultaneamente. Esses processos dividem os recursos de hardware do sistema. O sistema persiste, enquanto os processos são criados e terminados. Memória física e recursos de entrada e saída são alocados para cada processo do sistema. A interface entre a máquina e o sistema operacional é o conjunto completo de instruções da máquina – ISA.

Essas duas visões dão origem a dois modelos diferentes de máquinas virtuais: as máquinas virtuais de processo e as máquinas virtuais de sistema. As máquinas virtuais de processo executam um único processo. Elas são criadas e terminadas juntamente com o processo que executam. As máquinas virtuais de sistema, por sua vez, provêm um ambiente persistente e completo de execução para múltiplos processos.

O processo ou sistema operacional que executa sobre uma máquina virtual é denominado hóspede. A plataforma onde a máquina virtual executa é denominada hospedeiro ou sistema nativo. A camada de virtualização que implementa a máquina virtual é chamada de *runtime* para as máquinas virtuais de processo e de monitor de

máquina virtual – MMV – ou hipervisor (*hypervisor*) para as máquinas virtuais de sistema.

Máquinas Virtuais de Processo

Uma máquina virtual de processo oferece uma ABI virtual (*user ISA* e chamadas de sistema) a uma determinada aplicação [SMITH et al. 2005] (figura 3.3). Um dos desafios de tal técnica é a execução de programas compilados para um conjunto de instruções diferentes do ISA da arquitetura hospedeira. A maneira mais comum de resolver esse problema é através da interpretação. Um interpretador é um programa que decodifica um determinado código, convertendo-o em um código executável. Um interpretador pode emular um código binário compilado para uma determinada arquitetura para a arquitetura hospedeira - possivelmente diferente.

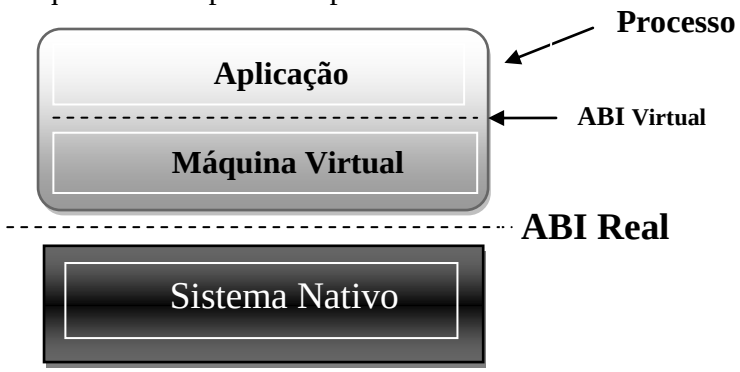

Figura 3.3. Máquina virtual de processo

Para máquinas virtuais de processo, a portabilidade é uma questão importante. Entretanto, a técnica mais simples de interpretação, emulação de uma arquitetura convencional em outra, restringe a compatibilidade apenas entre as duas plataformas envolvidas. Uma forma de se atingir compatibilidade completa entre diferentes plataformas é a interpretação para um código intermediário. O código intermediário resultante pode ser visto como um código binário para uma plataforma abstrata – uma máquina virtual. O conjunto de instruções dessa máquina virtual não corresponde ao ISA de uma máquina real. A máquina abstrata é executada em uma plataforma real como uma máquina virtual de processo. Uma aplicação cujo código binário execute sobre a máquina abstrata pode executar em todas as plataformas que suportam a máquina abstrata, garantindo que um mesmo código execute em diversas arquiteturas diferentes. Um exemplo clássico desta abordagem é a linguagem de programação Java. Um código fonte escrito em Java é convertido em um código binário específico – bytecodes – que é executado pela JVM (*Java Virtual Machine*). O código executável pode ser portado para qualquer arquitetura que seja capaz de executar a máquina virtual Java.

Máquinas Virtuais de Sistema

Uma máquina virtual de sistema oferece um ambiente completo no qual sistemas operacionais processos podem coexistir. Desta forma, uma única máquina física pode hospedar múltiplos usuários executando diferentes sistemas operacionais. Além disso, essa abordagem garante total isolamento entre processos de diferentes usuários ao executar-lhes em máquinas virtuais diferentes.

O principal desafio na implementação de máquinas virtuais de sistema é a divisão dos recursos de hardware entre os diferentes hóspedes. Cabe aos hipervisores esse controle. Existem duas formas básicas de se implementar um hipervisor de máquina virtual: hipervisores nativos (ou baremetal) e hipervisores hóspedes [SMITH et al. 2005].

Os hipervisores nativos executam diretamente sobre o hardware da máquina real. O hipervisor gerencia o compartilhamento dos recursos de hardware entre as diferentes máquinas virtuais fazendo com que elas tenham a ilusão de que esses recursos são privativos a elas. Esse tipo de abordagem foi inicialmente explorado pela IBM no início da década de 70. Hoje, o Xen, VMware ESX Server, são exemplos de tal técnica. A 3.4a ilustra um hipervisores nativos.

Os hipervisores hóspedes (figura 3.4b) executam sobre um sistema operacional nativo como se fosse um processo. Neste caso, o hipervisor oferece sistemas operacionais hóspedes – possivelmente diferentes do nativo – que executa sobre um hardware virtual criado a partir dos recursos oferecidos pelo sistema operacional nativo. O VMware Player, VirtualBox e MS VirtualPC implementam essa técnica.

Pode-se identificar ainda duas estratégias diferentes na implementação de hipervisores hóspedes. Uma possibilidade é oferecer uma ABI completamente diferente daquela da máquina física. A vantagem desse método é que ele permite que o sistema operacional e suas aplicações executem em plataformas diferentes das quais ela foi concebida. A desvantagem é que o hipervisor é obrigado a emular todas as instruções executadas, representando um custo no desempenho. A segunda possibilidade é oferecer um conjunto de instruções não privilegiadas idêntico ao da máquina hospedeira. Desta forma, as instruções não privilegiadas podem ser executadas diretamente, sem a necessidade de serem emuladas pelo hipervisor. A desvantagem de tal método é que as máquinas virtuais estão restritas a uma determinada arquitetura.

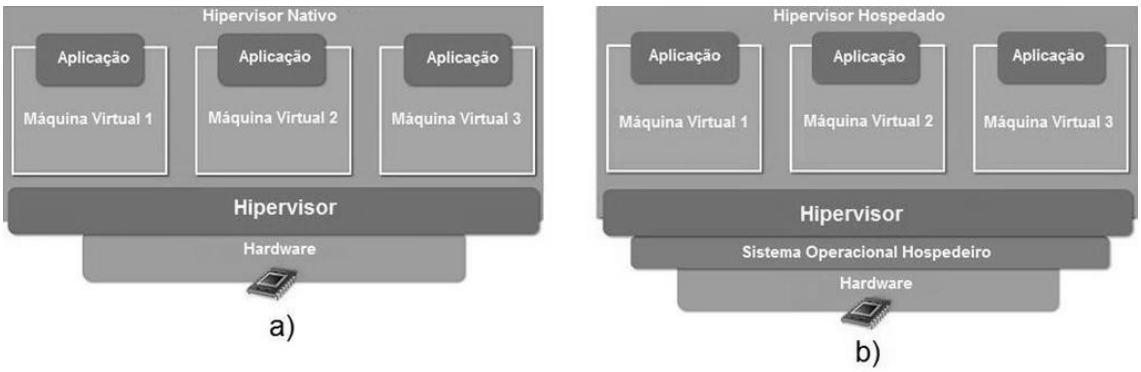

Figura 3.4. Diferentes modelos de hipervisores: a) Hipervisor Nativo, b) Hipervisor Hospedado. Adaptado de [VMWARE, 2011-a]

3.2 Virtualização Completa e Paravirtualização

Em 1974, Poek e Goldberg [Poek et al., 1974] estabeleceram três características essenciais para que um sistema computacional ofereça suporte adequado à virtualização, são elas:

1. *Equivalência*: A execução de uma aplicação em uma máquina virtual ou na máquina física deve apresentar resultados idênticos. Exceção é feita no tempo de processamento. Eventualmente, o tempo de execução de algumas instruções pode variar entre uma plataforma real e um sistema virtualizado.
2. *Desempenho*: Qualquer instrução que não afete o funcionamento do sistema deve ser executada diretamente pelo hardware e sem a intervenção do monitor de máquina virtual (MMV).
3. *Controle de Recursos*: O MMV deve ter o controle completo sobre os recursos virtualizados do sistema. Uma aplicação executando sobre uma máquina virtual não pode acessar diretamente o hardware.

A arquitetura x86 oferece quatro modos de operação representando diferentes níveis de privilégios, os anéis (*rings*) 0, 1, 2, e 3 [VMWARE, 2011-b]. Em um sistema tradicional, as aplicações de usuário normalmente executam no nível menos privilegiado (*ring* 3) e o núcleo do sistema operacional, que deve ter acesso direto ao hardware, no nível de maior privilégio - *ring 0*.

Para se garantir as três condições citadas no início da seção, propõem-se, em uma primeira análise, que o MMV execute as máquinas virtuais no *ring* 3. Desta forma, as instruções não privilegiadas - *user ISA* - geradas pela máquina virtual podem ser executadas diretamente, garantindo a propriedade 2. As instruções privilegiadas – *system ISA* -, quando executadas por uma máquina virtual, causariam uma exceção – *trap* - (estão sendo executadas em modo não privilegiado) sendo então tratadas pelo MMV da forma adequada, o que está de acordo com a propriedade 3. Com esse esquema, todo o conjunto de instruções de uma máquina física pode ser executado por uma máquina virtual (diretamente ou emuladas pelo MMV) o que garante a primeira propriedade.

No entanto, as arquiteturas x86 possuem instruções não privilegiadas capazes de alterar os recursos do sistema, as chamadas instruções sensíveis. Deste modo, a condição 3 – MMV deve ter o controle completo sobre os recursos virtualizados – não é conferida. A arquitetura IA-32, por exemplo, apresenta dezessete instruções não privilegiadas que são consideradas sensíveis [ROBIN at al., 2000]. Para se contornar esse problema, o MMV implementado para essas arquiteturas deve tratar, além das instruções privilegiadas, as instruções não privilegiadas sensíveis. Tal alternativa traz um custo em desempenho, visto que, as instruções *user ISA* devem ser analisadas e, as que forem consideradas sensíveis, emuladas pelo MMV.

Em suma, um monitor de máquina virtual deve ser capaz de gerar a execução direta de qualquer instrução que não afete os recursos do sistema (propriedade 2), tratar e emular todas as instruções sensíveis gerada por suas máquinas virtuais (propriedade 3) garantindo assim equivalência para todas as instruções da arquitetura real (propriedade 1). Para se construir um sistema virtualizado com tais propriedades, duas técnicas são normalmente usadas. Na virtualização completa, todas as instruções sensíveis são identificadas em tempo de execução e geram um desvio para o MMV tratá-las da forma

adequada. Na paravirtualização, o programa, ao ser executado pelo MMV, é modificado para que as instruções sensíveis sejam substituídas por chamadas ao MMV.

3.2.1 Virtualização Completa

Na virtualização completa ou virtualização total (figura 3.5.), a máquina virtual provida é capaz de simular de forma completa o hardware subjacente. Para tal, o MMV deve implementar todos os componentes e elementos relacionados ao hardware presentes na máquina física e que podem, potencialmente, serem usados pela máquina virtual. O sistema operacional e as aplicações executam como se estivessem rodando diretamente sobre o hardware.

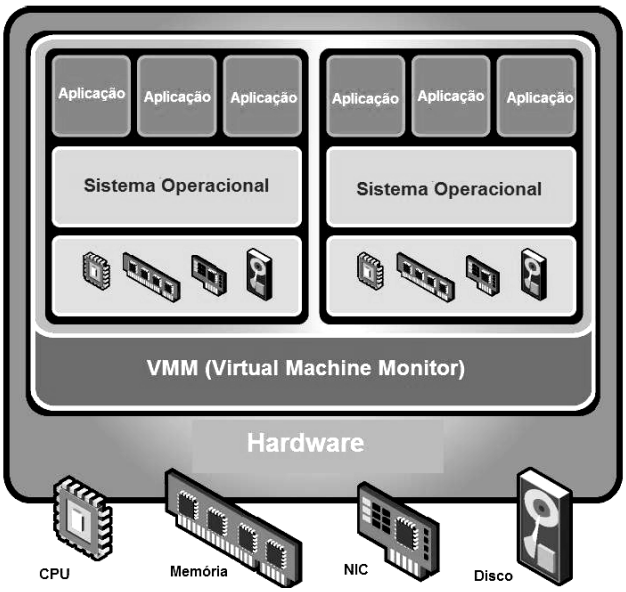

Figura 3.5 – Arquitetura baseada em virtualização completa. Adaptado de (VMWARE, 2011-b)

A grande vantagem desta técnica é que ela é transparente às aplicações. Desta forma, um sistema operacional pode executar em uma máquina virtual desse tipo sem ser alterado. Uma das desvantagens da virtualização completa é a dificuldade de simular a grande diversidade de dispositivos existentes. Um MMV desse tipo provê apenas um conjunto de dispositivos genéricos como teclado, mouse, controladores IDE, cdrom, portas seriais e paralelas, placa gráfica padrão e placas de rede. Além disso, existe um preço a se pagar no desempenho devido ao custo computacional de se detectar e emular as instruções sensíveis.

Existem duas formas de se implementar a virtualização completa. A mais antiga, e mais utilizada hoje em dia, é através da tradução binária [VMWARE, 2011-b] (figura 3.6b)**.** Nesta abordagem as instruções sensíveis são substituídas em tempo de execução por uma nova sequência de instruções de forma a emular o comportamento original do código. As demais instruções não privilegiadas são executadas diretamente, sem a intervenção do MMV. Todas as instruções devem ser testadas pelo MMV para se descobrir se são sensíveis. As instruções sensíveis devem ser interceptadas e emuladas no hospedeiro o que pode acrescer um alto custo computacional à solução.

A segunda forma de implementar um sistema baseado em virtualização completa, é a Virtualização Assistida por Hardware (*Hardware Assisted Virtualization* - HAV). Essa abordagem é possível graças ao suporte a virtualização fornecido pelos novos

processadores como extensões na arquitetura x86. A AMD, com sua tecnologia AMD-V e a Intel com a VT-x, desenvolveram duas soluções equivalentes para o problema das instruções sensíveis. Nestes processadores, o sistema de anéis, característico das arquiteturas x86, foi modificado para comportar a camada de virtualização (figura 3.6c).

O monitor de máquina virtual (MMV) executa em um novo modo de execução - *Root Mode* – posicionado entre o hardware e o anel 0. Além disso, algumas funcionalidades foram acrescentadas ao hardware para suportar a virtualização. Uma estrutura de dados chamada VMCB (*Virtual Machine Control Block*) registra dados de controle e de estado das máquinas virtuais presentes no sistema. Uma nova instrução chamada *vmrun* é usada para se passar do modo *root* para o modo não privilegiado – *non-root*. Ao executar *vmrun*, o hardware carrega o estado da máquina virtual a partir da VMCB. A execução em modo *non-root* prossegue até que o alguma operação especial (E/S por exemplo), seja feita. Neste momento, o MMV - usando os bits de controle do VMCB – detecta e intercepta a operação. Uma operação *exit* é gerada, passando o processador do modo *non-root* ao modo *root* e salvando o estado da máquina virtual na VMCB. O MMV, com o auxílio de informações obtidas na VMCB, trata a operação de *exit* de modo a emular o efeito da ação executada pela máquina virtual. Em seguida, o MMV executa *vmrun*, retornando ao modo non-root.

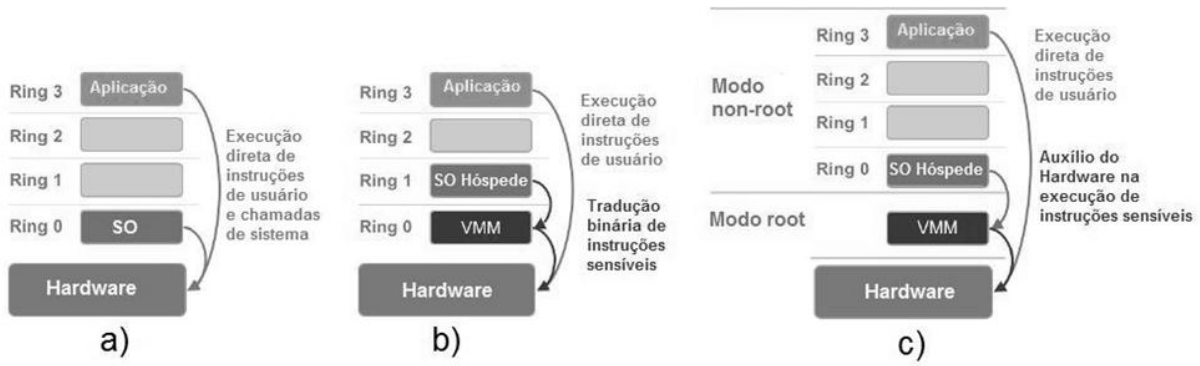

Figura 3.6 – Arquitetura x86 e Virtualização: a) sistema não virtualizado,
b) sistema virtualizado baseado em tradução binária,
c) sistema virtualizado assistido por hardware.
Adaptado de [VMWARE, 2011-b].

Ambas as soluções para a virtualização completa trazem sobrecarga ao sistema. A tradução binária, como já mencionado, pode ser onerosa devido ao custo para se analisar e emular cada instrução não privilegiada. Já na virtualização assistida por hardware, o custo no desempenho depende da freqüência que a máquina virtual irá realizar operações de *exit* e do tempo que o hardware levará para transitar entre os estados de privilégio. Uma forma de se atenuar essa sobrecarga no custo computacional seria utilizar um sistema operacional que estivesse ciente da camada de virtualização sob a qual ele está executando. Desta forma, ao se deparar com uma instrução sensível, o sistema operacional poderia chamar diretamente o MMV. Esta abordagem é a base para a paravirtualização que será apresentada na próxima seção.

3.2.2 Paravirtualização

A paravirtualização (figura 3.7) é uma abordagem alternativa que visa contornar os problemas de desempenho da virtualização total. Nesta técnica, o sistema operacional hóspede deve ser modificado de forma a se comunicar diretamente com o MMV. Na prática, o sistema operacional passa a substituir todas as instruções sensíveis por chamadas ao MMV (*hypercall*) que, por sua vez, as interpreta e as emula da forma adequada. As demais instruções são executadas diretamente no hardware. Desta forma, devido ao fato de o sistema operacional hóspede estar ciente que está executando em uma máquina virtual, o tratamento das instruções sensíveis representa um menor acréscimo computacional.

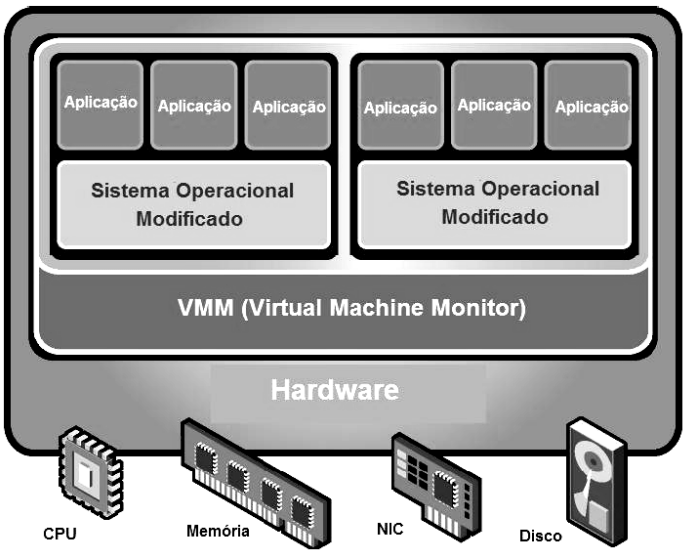

Figura 3.7 – Arquitetura baseada em paravirtualização.
Adaptado de [VMWARE, 2011-b]

Além disso, a paravirtualização provê aos sistemas hóspedes acesso aos recursos de hardware através de drivers do MMV. Desta forma, em contraste à virtualização completa, os sistemas hóspedes utilizam recursos reais da máquina. A grande desvantagem da paravirtualização é a necessidade de se alterar o sistema hóspede. Modificar o núcleo de um sistema operacional pode ser indesejado ou mesmo infactível, como no caso de sistemas de código proprietário como o Windows.

A tabela 3.1 apresenta de forma resumida as vantagens e desvantagens da virtualização completa por tradução binária, hardware assistido e da paravirtualização.

Tabela 3.1 - Comparação entre diferentes técnicas de virtualização em plataformas x86. Adaptada de VMWARE, 2011-b.

	Virtualização Completa por Tradução Binária	**Virtualização Completa Assistida por Hardware**	**Paravirtualização**
Técnica	Tradução binária de instruções sensíveis e execução direta das demais.	Execução de instruções sensíveis com auxílio do hardware e execução direta das demais.	Chamadas ao MMV para instruções sensíveis e execução direta das demais.
Compatibilidade	Excelente – Sistema operacional hóspede não modificado.	Boa – Depende de suporte de hardware, mas trabalha com sistema operacional hóspede não modificado.	Pobre – Demanda modificação do sistema hóspede.
Desempenho	Regular – sobrecusto da tradução binária.	Ainda sob testes, tende a melhorar com a evolução do suporte de hardware.	Bom – o melhor na maioria dos casos.
Implementações	VMware, Microsoft, Parallels	VMware, Microsoft, Parallels, Xen	VMware, Xen

3.3 Ferramentas Existentes

Nos últimos anos, a virtualização tem adquirido grande importância na área de TI. Aproveitando-se de toda a versatilidade dessa tecnologia, grandes corporações estão reformulando os seus ambientes TI para soluções baseadas em máquinas virtuais. A computação em nuvens, por exemplo, representa uma grande revolução na maneira com que compartilhamos dados e recursos computacionais. Esse conceito, fundamentado no uso de máquinas virtuais, é apenas um dos vários exemplos de tecnologias recentes que se aproveitam das vantagens da virtualização.

Atentos a esta revolução, diversas empresas e grupos de pesquisa desenvolveram produtos que implementam e facilitam o uso da virtualização. Nesta seção, quatro exemplos de produtos existentes ilustram o mercado de virtualização atual. São apresentados o Xen, explicando-se a sua arquitetura particular baseada nos domínios dom0 e domU; o VMware, abordando o ESX e o VMware Player; as soluções da Microsoft – Hyper-V e Virtual-PC e, por fim, as soluções Solaris (Oracle) com a Virtual Box.

Xen [XEN, 2011] é um monitor de máquina desenvolvido em software livre, nos termos da GNU General Public License (GPL). O hipervisor Xen é compatível com várias arquiteturas; x86, x86-64, Itanium, Power PC e ARM, por exemplo. Ele suporta uma variedade de sistemas operacionais hóspedes, tais quais GNU/Linux, FreeBSD, Solaris, Windows, etc.

A arquitetura Xen é um pouco particular e baseia-se em três componentes básicos: o hipervisor - um domínio privilegiado, o Dom0, e múltiplos domínios não privilegiados, os DomUs.

O hipervisor executa diretamente sobre o hardware (nativo) e tem por função controlar os recursos de comunicação, de memória e de processamento das máquinas virtuais hóspedes. Ele é a interface para todas as requisições de hardware, como CPU, E/S e disco.

O domínio 0 (Dom0) é disparado pelo hipervisor quando o sistema é iniciado. Sua função é gerenciar os domínios U – criar, iniciar, interromper, recomeçar, destruir, tratar requisições de E/S, etc. Para tal, mesmo sendo um domínio hospedado, ele tem privilégios únicos de acesso ao hipervisor. Na prática, o Dom0 é uma máquina virtual rodando um sistema operacional modificado (tipicamente GNU/Linux) que possui *drivers* de dispositivos de máquinas físicas e dois *drivers* especiais para tratar as requisições de acesso à rede e ao disco efetuados pelos outros domínios hóspedes, os domUs.

Os domínios Us, DomUs, são máquinas virtuais controladas pelo Dom0 e executam de forma independente umas das outros. Eles podem executar um sistema operacional modificado – paravirtualização – ou um sistema operacional não modificado com auxílio do hardware – virtualização completa assistida por hardware. A figura 3.8 ilustra a arquitetura Xen.

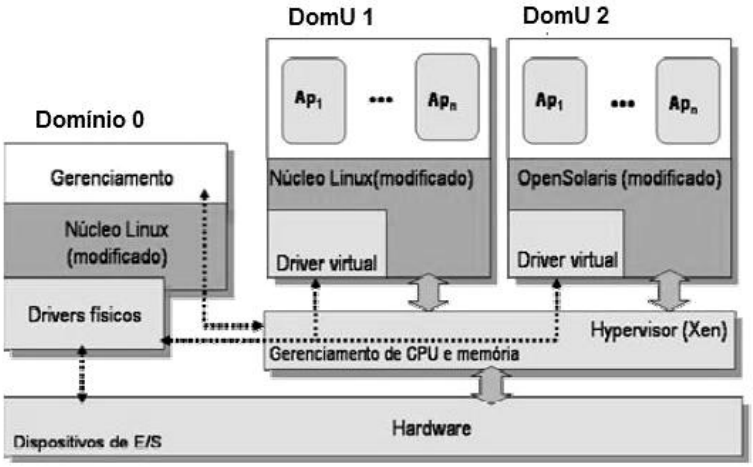

Figura 3.8 Arquitetura Xen [CARISSIMI, 2009]

VMware [VMWARE, 2011-c] oferece uma gama completa de produtos de suporte a virtualização, onde se destacam o VMware ESX e o VMware Player.

O VMware ESX é um hipervisor do tipo nativo usado para a criação de *datacenters* virtuais. Ele permite que um servidor físico seja particionado em vários servidores virtuais, permitindo assim a consolidação de servidores. O VMware Player é executado sobre um sistema operacional hospedeiro – hipervisor hospedado – e permite a criação de um ambiente de execução com várias máquinas virtuais. Ele é mais utilizado para se criar um ambiente virtual em *desktops*.

Microsoft: a exemplo da VMware, a Microsoft apresenta várias soluções para virtualização. O Microsoft Virtual PC [M. VIRTUAL PC, 2011] é um hipervisor hospedado, que implementa virtualização total. É destinado à virtualização em *desktops* que utilizam um sistema hospedeiro Windows. As máquinas virtuais podem executar Windows. Alguns outros sistemas operacionais, como GNU/Linux, são possíveis, mas não suportados oficialmente. Destina-se à execução de software legado, criação de ambientes de testes e treinamento. O Microsoft Hyper-V [M. HYPER V, 2011] é um produto destinado à virtualização de servidores. Através de um hipervisor nativo, o Hyper-V proporciona segurança, tolerância a falhas, confiabilidade e disponibilidade aos *datacenters*.

Oracle VM VirtualBox: a VirtualBox [VIRTUAL BOX, 2011] implementa virtualização completa através de um hipervisor hospedado para servidores, desktops e sistemas embarcados. Assim como o Xen, é desenvolvida em software livre nos termos da GPL. A VirtualBox executa em Windows, GNU/Linux, Macintosh e Solaris e suporta uma larga gama de sistemas operacionais hóspedes, incluindo Windows, DOS, Solaris, OpenBSD, etc. Além disso, ela suporta virtualização completa assistida por hardware através para as tecnologias Intel VT-x e AMD-V.

3.4 Cenários de Uso

Como visto anteriormente, as máquinas virtuais podem suportar processos individuais ou um sistema completo, dependendo do nível de abstração em que elas são empregadas. Pode-se listar alguns benefícios diretos do uso de máquinas virtuais:

- flexibilidade de hardware: ao prover uma camada de software entre o hardware e a aplicação, as particularidades de cada plataforma podem ser abstraídas ou emuladas para uma plataforma genérica;

- versatilidade: múltiplos sistemas operacionais podem ser executados em uma mesma máquina física provida de um hipervisor;

- portabilidade: a virtualização torna mais simples a migração de programas e sistemas, provendo mobilidade aos seus usuários;

- encapsulamento: o estado de um sistema virtualizado em execução pode ser completamente registrado. Desta forma, a execução de uma máquina virtual pode ser interrompida e reiniciada;

- tolerância a falhas e segurança: i) o estado do processamento de uma máquina virtual pode ser salvo e utilizado para *backup* do sistema. ii) ao se isolar os processos, a propagação de falhas é restringida à máquina virtual. iii) pode-se injetar falhas na camada de virtualização para se estudar o comportamento do sistema sob essa condição;

- QoS: a distribuição de recursos em um sistema virtualizado é facilmente gerenciada. Desta forma, é possível garantir uma determinada qualidade de serviço a uma aplicação, independente do nível de carga do sistema.

Devido a todos esses benefícios, a virtualização vem sendo largamente utilizada por diversas aplicações em sistemas computacionais. Nesta seção três cenários de uso são discutidos: consolidação de servidores, virtualização em desktops, desenvolvimento de testes.

3.4.1 Consolidação de Servidores

Data centers são grandes consumidores de energia. Estima-se que, nos Estados Unidos, estas infra-estruturas são responsáveis por 1,5 por cento do consumo energético do país [VMWARE, 2011-d]. Um dos fatores mais importantes para este cenário é o mau aproveitamento dos recursos de *hardware*. Tipicamente, por motivos que variam desde a heterogeneidade de clientes à segurança, os *data centers* são projetados para executar um único serviço em cada máquina física. Podemos imaginar um ambiente onde uma máquina será usada como servidor de e-mail, outra para servidor de arquivos, uma terceira máquina será usada como servidor *HTTP*, etc. O nível de carga de uma infra-estrutura de TI pode depender do dia do mês ou do horário. Um *data center* deve ser projetado de forma a garantir um bom nível de serviço o tempo todo. Em uma arquitetura do tipo um serviço por máquina física, os recursos de cada servidor devem ser mensurados de forma a suportar os picos de utilização. Desta forma, visto que os recursos serão plenamente utilizados apenas em momentos de pico, o *hardware* passará a maior parte do tempo subutilizado. Além disso, como foi comentado na introdução deste trabalho, em estado de baixa carga, a CPU apresenta baixa eficiência energética. Em outras palavras, a incapacidade da infra-estrutura de acomodar dinamicamente as flutuações no seu nível de utilização gera desperdício de hardware e energético.

Uma boa solução para este problema é a utilização de *data centers* baseados em máquinas virtuais. A idéia é fornecer um serviço por máquina virtual, ao invés de um serviço por máquina física. Um servidor pode executar diversas máquinas virtuais, portanto diversos serviços. Desta forma, esta nova arquitetura define *n* serviços para cada servidor. Com vários serviços executando em uma mesma máquina física, pode-se atingir um alto nível de utilização do seu hardware. Além disso, graças à flexibilidade de ambientes virtualizados, pode-se responder dinamicamente às variações de carga no sistema. Máquinas virtuais podem ser migradas de uma máquina física a outra sem grandes custos computacionais. Deste modo, o sistema pode ser gerenciado de forma a distribuir eficientemente os serviços entre os servidores. Pode-se concentrar todos os serviços em alguns servidores, permitindo que os outros sejam desligados.

A utilização de virtualização na consolidação de servidores garante uma utilização energética eficiente por dois fatores: i) menos hardware é necessário, pois ele é utilizado mais eficientemente e ii) conforme a demanda servidores podem ser desligados, poupando a energia da alimentação desses servidores e dos seus sistemas de refrigeração. Cada servidor virtualizado pode economizar 7.000 kWh de energia e deixar de emitir quatro toneladas de dióxido de carbono por ano [VMWARE, 2011-d].

3.4.2 Virtualização em Desktops

A flexibilidade e a portabilidade de máquinas virtuais tornam o seu uso interessante em *desktops*. Nesta aplicação, a camada de virtualização separa o ambiente pessoal de trabalho do hardware físico.

Assim como em um processo, a execução de uma máquina virtual pode ser interrompida e retomada mais tarde. Em um determinado momento, pode-se capturar o estado completo do sistema para posteriormente retomá-lo do ponto onde ele parou. Além disso, a independência de plataforma característica de uma máquina virtual provê a possibilidade de recomeçar a sua computação em uma máquina física diferente da qual ela estava executando. Os usuários de computadores demandam cada vez mais mobilidade dos sistemas de TI. Hoje em dia, queremos ter a possibilidade de trabalhar

em qualquer lugar, seja no escritório, em casa ou em lugares públicos. Neste contexto, a virtualização em desktops possibilita que usuários transportem seus ambientes de trabalho, sem a necessidade de carregar a mesma máquina física para todos os lugares.

3.4.3 Ambientes para Desenvolvimento de Testes

A virtualização é uma ferramenta muito interessante para a criação de ambientes de testes. Uma máquina virtual provê um ambiente isolado, que não interfere ou provoca danos no restante da infraestrutura. Além disso, uma aplicação que está sendo desenvolvida para diversos sistemas operacionais, pode ser testada em uma única máquina física. Para tal, a máquina hospedeira deve estar executando uma virtualização de sistema através de um hipervisor nativo ou hospedado. Este hipervisor será capaz de executar diversas máquinas virtuais com diferentes sistemas operacionais, onde a aplicação poderá ser testada.

Para testar uma aplicação em um único sistema operacional, é conveniente se utilizar uma máquina virtual de sistema com hipervisor hospedado. Desta forma, a aplicação será testada em um ambiente de trabalho bastante semelhante ao real, mas completamente isolado, provendo segurança às demais aplicações.

3.5 Considerações Gerais

Virtualização é um conceito antigo da computação que vem recuperando relevância nos últimos anos graças ao aumento do poder computacional, e a popularização de sistemas distribuídos. O principal objetivo de se desenvolver sistemas computacionais baseados em máquinas virtuais é prover a utilização mais eficiente e abrangente das plataformas. Ambientes virtualizados provêem independência de plataforma, isolamento entre usuários, facilidade de gerenciamento de recursos, etc. Além disso, ao aproveitar melhor os recursos computacionais, as máquinas virtuais buscam uma utilização mais eficiente de energia.

4 AVALIAÇÃO EXPERIMENTAL

As plataformas computacionais disponíveis hoje apresentam um mau desempenho energético, pois a quantidade de potência dissipada por um processador não é proporcional à sua carga de trabalho [ORGERIE, 2011]. Em baixos níveis de carga, os processadores apresentam uma eficiência energética pior do que quando completamente utilizados. Como será mostrado no decorrer deste capítulo, a potência consumida por um processador ocioso é equivalente a cerca de 50% da potência dissipada quando ele está completamente ocupado. Sendo assim, para se utilizar mais adequadamente a energia consumida por computadores, deve-se evitar que os processadores permaneçam muito tempo em estado ocioso ou de pouca carga. Para tal, quando não há trabalho a fazer, o dispositivo deve ser desativado (totalmente ou parcialmente) e, quando em funcionamento, é desejável que ele permaneça com alta taxa de utilização.

Neste contexto, a virtualização surge como ferramenta para se melhorar a eficiência energética de computadores. Como visto no capítulo 3, a utilização de virtualização em *datacenters* permite desacoplar os serviços das máquinas físicas. Cada serviço provido pelo ambiente será executado em uma plataforma abstrata, uma máquina virtual. Esta MV, por sua vez, poderá migrar entre diferentes máquinas físicas. Desta forma, várias máquinas virtuais – e consequentemente vários serviços – podem executar na mesma máquina física, garantindo um alto nível de ocupação do processador e desocupando outros servidores, que podem ser então desligados.

Para se compreender e investigar melhor a forma com que um sistema computacional consome energia, executou-se uma série de *benchmarks* em um ambiente de virtualização. Os dados sobre desempenho e consumo energético obtidos através desses testes serão apresentados e analisados neste capítulo.

O estudo apresentado aqui tem dois objetivos principais. Primeiramente, no **Objetivo I**, pretende-se mensurar o impacto energético resultante da execução de máquinas virtuais em uma máquina física. A intenção é observar o quanto *n* máquinas virtuais consomem em um sistema físico. Além disso, pretende-se comparar os desempenhos computacional e energético das máquinas virtuais com o de um sistema não virtualizado, para se avaliar o sobrecusto oriundo da virtualização.

Como visto no capítulo 2, reduzir a frequência de processamento diminui a potência dissipada pelo processador. Contudo, em menor frequência, menos instruções são executadas por segundo e uma computação leva mais tempo para ser efetuada. No **Objetivo II**, pretende-se medir o impacto da redução da frequência de processamento no consumo energético e no desempenho de aplicações. A análise pretende comparar duas técnicas diferentes: *race to idle* (seção 2.3.1.) onde executa-se as aplicações na maior velocidade possível para se voltar a um estado de *idle* e *frequency scaling* (seção 2.1.) onde a idéia é reduzir a frequência para se dissipar menos potência.

4.1 Metodologia

Para se responder aos questionamentos introduzidos no início do capítulo, um *desktop* foi utilizado para se criar um ambiente virtualizado. Uma série de testes foram feitos neste ambiente e, o consumo energético total da máquina durante a execução de cada teste foi medido.

Entre as ferramentas de medição de consumo energético em plataformas computacionais apresentadas na seção 2.4., foi optado pelo emprego de um wattímetro para se realizar a coleta dos dados apresentados neste capítulo. O processo de leitura de medidas é feito de forma manual - sujeito a imprecisões. O *desktop* é ligado ao wattímetro e são executados alguns *benchmarks*. A cada teste, o contador de energia consumida é zerado e o *benchmark* é iniciado. No momento que a aplicação é finalizada, o valor da energia consumida mostrado pelo wattímetro e o tempo de total que o *benchmark* levou para ser executado são anotados.

É importante destacar que o fato de o wattímetro ser um elemento externo e sem conectividade com a plataforma computacional, o que inviabilizou a automatização dos testes. Desta forma, dado o tempo de execução dos testes (alguns casos de teste levaram mais de 30 minutos) e a impossibilidade de se automatizá-los, os testes foram executados uma única vez, o que torna os resultados estatisticamente menos precisos. Exceção é feita para os *benchmarks* NAS LU classe A e Iozone, onde os dados apresentados são uma média de três medições subseqüentes.

Foram definidos dois cenários de testes que são descritos na seção 4.1.1. Os *benchmarks* utilizados nos cenários de teste foram selecionados por representarem aplicações bastante comuns em sistemas computacionais.

4.1.1 Cenários de Teste

Uma série de *benchmarks* foi executada em um sistema virtualizado. Os *benchmarks* utilizados se dividem em dois grupos: os que exigem uma computação intensiva – CPU *bound* e os que testam o desempenho e gerenciamento de dispositivos de entrada e saída. Para se responder aos objetivos I e II, dois cenários de testes foram desenvolvidos:

Cenário I – Consumo de uma máquina virtual: Para se medir o consumo energético de uma máquina virtual, os *benchmarks* foram executados nos seguintes sistemas:

I) Sistema nativo (não virtualizado).

II) Sistema virtualizado de duas formas:

II.1) Em um domínio privilegiado que possui *drivers* reais.

II.2) Em *n* máquinas virtuais (*n* variando entre 1 e 8).

Primeiramente observou-se as diferenças entre o consumo energético das máquinas virtuais e do domínio privilegiado para se mensurar o custo do gerenciamento das MVs. Em seguida, analisou-se como a potência de processamento varia, conforme se adiciona mais máquinas virtuais ao sistema. Por fim, para cada *n*, o custo energético médio de cada instrução de máquina virtual foi medido. O resultado foi usado para se descobrir se executar várias máquinas virtuais em uma mesma máquina física realmente melhora a sua eficiência energética.

Cenário II – Consumo X Desempenho: Para se mensurar o impacto da adoção de medidas de redução de consumo energético no desempenho, os *benchmarks* foram executados, em *n* máquinas virtuais (*n* entre 1 e 8), e para duas frequências de processamento : $F_{máx}$ = 3,06 GHz e : F_{min} = 1,197 GHz. Comparou-se, em cada caso, o tempo de processamento e o consumo energético. O objetivo desta análise foi avaliar o impacto no desempenho de execução das aplicações e os benefícios em consumo energético que a redução da frequência de processamento pode trazer.

4.1.2 Benchmarks Utilizados

Para se gerar carga computacional, foram executados os seguintes *benchmarks*:

NAS LU *Benchmark* [NAS, 2011-a] [NAS, 2011-b] [NAS, 2011-c]: *NAS Parallel Benchmark* (NPB) é um conjunto de *benchmarks* desenvolvidos pela NASA *Advanced Supercomputing* (NAS) para avaliar o desempenho de supercomputadores. Para os testes realizados neste trabalho, foi escolhido o *benchmark* NAS LU. Ele consiste em uma aplicação CFD (*Computational Fluid Dynamics*) que resolve problemas de dinâmica de fluidos através de um algoritmo baseado na decomposição matricial LU (*lower-upper*). A fatoração matricial LU é um método largamente utilizado na resolução de problemas científicos e aplicações reais. NAS LU possui diversas classes de execução que representam diferentes níveis de complexidade para o problema, a partir do tamanho da matriz a ser decomposta. Foram realizados testes utilizando NAS LU classes A (matriz 64x64x64) e B (matriz 102x102x102).

Spec CPU2006[SPEC, 2011]: CPU 2006 é um *benchmark* do tipo CPU intensivo desenvolvido pela SPEC (*Standard Performance Evaluation Corporation)*. O SPEC CPU2006 gera uma carga computacional que visa estressar processador e a memória através de aplicações reais de usuário. Os *benchmarks* avaliam operações aritmética com números inteiros (programas como compiladores, interpretadores, processadores de texto) e operações aritméticas em ponto flutuante (simulação física, processamento gráfico, soluções de problemas químicos complexos). Por representar uma aplicação bastante comum na computação, utilizou-se um *benchmark* que gera carga de compilação através do compilador gcc.

CPUburn [CPUBURN, 2011]: CPUburn é um *benchmark* do tipo CPU intensivo. Ele foi desenvolvido para testar o processador em situação de máximo estresse. O programa consiste em um código *assembly* realiza instruções de operações aritméticas simples e em ponto flutuante. O objetivo do *benchmark* é estressar o processador para maximizar a produção de calor e dissipação de energia.

Iperf [IPERF, 2011]: O Iperf é um *benchmark* de entrada e saída. Seu objetivo é medir a largura de banda da conexão e o desempenho dos protocolos TCP e UDP. O Iperf permite edição de diversos parâmetros e características de transmissão. Nos testes apresentados neste trabalho, o domínio privilegiado foi utilizado para servidor e as máquinas virtuais foram os clientes, ambos executando na mesma máquina física, criando-se assim, uma rede virtual. O *benchmark* consistiu em transmitir 25 Gbits entre cliente e servidor. O objetivo do teste era avaliar E/S. Desta forma, por gerar menos sobrecarga de processamento, escolheu-se UDP como protocolo de transporte.

Iozone [IOZONE, 2011]: Iozone é um *benchmark* projetado para se avaliar o desempenho de sistemas de arquivos. Ele permite que diversas operações de disco sejam realizadas em diferentes configurações de funcionamento. Nos testes realizados, a *cache* de disco foi desabilitada e foram efetuadas escritas e leituras de blocos de 64kB de forma iterativa.

4.1.3 Plataforma Experimental

A plataforma em hardware utilizada para executar os testes foi um *desktop* dotado de um processador Intel Core i3-540 – arquitetura 64 bits, dois núcleos de processamento, frequência máxima de operação: $F_{máx}$ = 3,06 GHz e frequência mínima de operação: F_{min} = 1,197GHz com 4GB de memória principal. O *desktop* possui fonte de alimentação Cooler Master 400 W com eficiência superior a 70%.

Em termos de plataforma de software, o monitor de máquina virtual Xen *Hypervisor* 4.0 foi utilizado para se criar o ambiente virtualizado. O Dom0 - domínio privilegiado - utiliza GNU/Linux com a distribuição Debian 6.01 squeeze, executando o kernel 2.6.32-5 modificado para fornecer suporte a virtualização. As máquinas virtuais – domU – são paravirtualizadas e utilizam GNU/Linux distribuição com a mesma distribuição do dom0, mas com o kernel 2.6.32-5 modificado, neste caso, para se comunicar diretamente com o hipervisor. As máquinas virtuais possuem processador virtual com um núcleo, 256Mb de memória, e disco de 4Gb.

Para se medir o consumo do *desktop*, um wattímetro Multifunctional Mini Ammeter d02A foi utilizado. O equipamento foi escolhido levando-se em conta custo e fácil disponibilidade no mercado. Apesar de existirem equipamentos de melhor precisão e, consequentemente, maior custo, o wattímetro empregado é mais preciso que as ferramentas de software existentes. O aparelho permite medir potência instantânea com uma precisão de 0,01 W e a energia consumida em um intervalo de tempo (em kWh) com uma precisão de 0,0001 kWh (360 J). Para se utilizá-lo, basta conectar-se o wattímetro à tomada e conectar o aparelho cujo consumo será analisado ao wattímetro.

4.2 Métricas de Análise

As métricas escolhidas para a análise dos resultados obtidos através da realização dos testes propostos neste trabalho foram desenvolvidas com base na equação (2.1), apresentada no capítulo 2 e repetida aqui:

$$P = \frac{dE}{dt}$$

Isolando-se E e considerando-se a potência como uma função de t:

$$E = \int_0^T P(t)\, dt \qquad \text{(expressão 4.1)}$$

Onde T é o tempo total da execução. Aplicando-se o conceito matemático de média para a potência, tem-se:

$$P_m = \frac{1}{T} * \int_0^T P(t) dt \qquad \text{(expressão 4.2)}$$

Onde P_m é a potência média durante o período T. Utilizando-se a expressão 4.1 para substituir-se $\int_0^T P(t)dt$ por E na expressão 4.2, obtem-se, finalmente:

$$P_m = \frac{E}{T} \qquad \text{(expressão 4.3)}$$

A expressão (4.3) permite o cálculo da potência média dissipada pela máquina física durante a execução de uma aplicação a partir da energia total consumida por toda a infraestrutura e do tempo de processamento da aplicação. Para se estimar a energia e potência gasta apenas pela aplicação, um método bastante simples foi utilizado: considerou-se que o processador, quando ocioso, dissipa uma potência praticamente constante[1]. Sendo assim, é possível enunciar:

$$P_m \approx P \qquad \text{(expressão 4.4)}$$

Desta forma, o valor da potência dissipada pelo processador quando ocioso foi medida e chamada de P_{idle} ($P_{idle} = 40W$). A partir de P_m e P_{idle}, calculou-se o que se chamou de potência média da aplicação:

$$P_{m\,aplicação} \approx P_m - P_{idle} \qquad \text{(expressão 4.5)}$$

Além disso, calculou-se também a energia consumida apenas pela aplicação:

$$E_{aplicação} \approx E_{total} - P_{idle} * t \qquad \text{(expressão 4.6)}$$

A expressão 4.6 pode então ser reescrita como:

$$P_{m\,aplicação} = \frac{E_{aplicação}}{T} \qquad \text{(expressão 4.7)}$$

A expressão 4.7 permite o cálculo da potência média consumida por uma aplicação a partir da energia e o tempo total gasto por ela. As três variáveis da expressão 4.7 serão usadas como métricas nas análises das próximas seções. A variável T será útil para se compreender o desempenho da máquina física em cada caso de teste. As variáveis $E_{aplicação}$ e $P_{m\,aplicação}$ permitirão a análise do consumo e eficiência energética de uma aplicação.

4.3 Cenário 1: Consumo de uma Máquina Virtual

As medidas de consumo energético dos diferentes *benchmarks* realizados serão utilizadas nesta seção para se responder ao objetivo I: mensurar o impacto energético resultante da execução de máquinas virtuais em uma máquina física. Primeiramente, será feita uma comparação entre o consumo energético de um *benchmark* executando no domínio privilegiado (Dom0) e em uma máquina virtual (DomU). O intuito é compreender e medir o quanto o gerenciamento das máquinas virtuais custa energeticamente ao sistema. Em seguida, várias máquinas virtuais serão instanciadas para se analisar o desempenho e consumo energético de um sistema virtualizado mais complexo.

[1]Essa consideração mostrou-se coerente a partir de testes realizados com o auxílio do wattímetro. Observou-se a potência instantânea dissipada pelo desktop durante um período de ociosidade e a potência dissipada não variou mais do que 7,5% (entre 38,5 W e 41,5 W).

4.3.1 Consumo de Energia Comparado: Nativo, Dom0 e DomU

O gráfico da figura 4.1 mostra a quantidade total de energia consumida pela máquina durante a execução dos *benchmarks*. Todos os testes foram executados com o processador em $F_{máx}$.

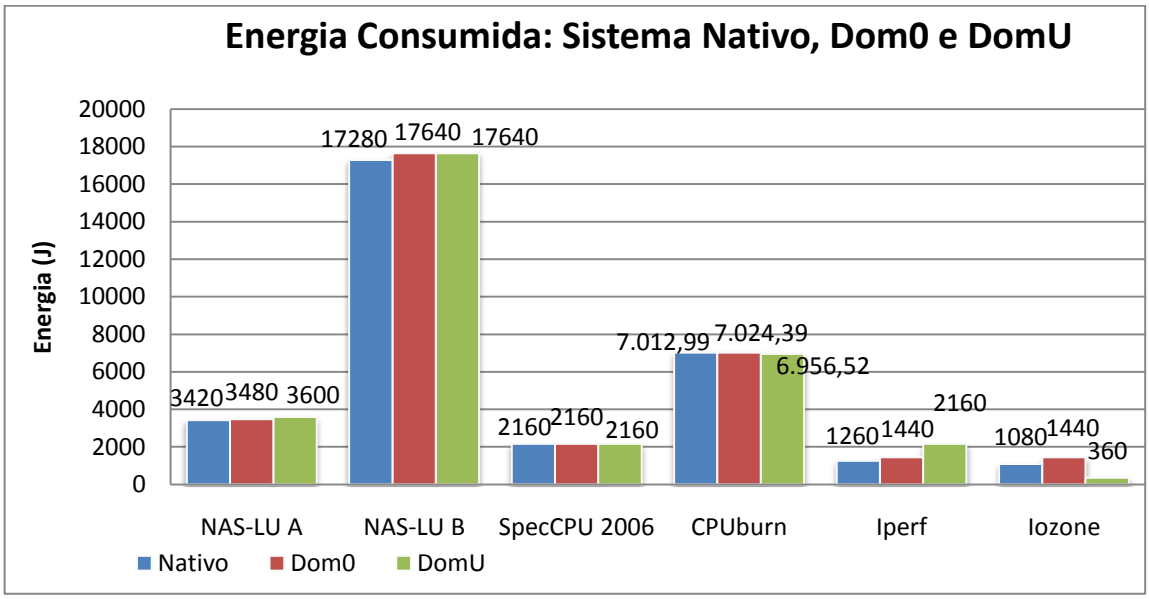

Figura 4.1: Comparação entre energia consumida pelo sistema nativo, Dom0 e DomU.

A tabela 4.1 mostra o tempo de execução dos *benchmarks* em para cada caso de teste.

Tabela 4.1: Tempo de execução dos *benchmarks* em sistema nativo, Dom0 e DomU.

Sistema	Tempo (s)
Nativo - 3,06GHz - NAS-LUA	56,8
Dom0 - 3,06GHz - NAS-LUA	57,57
1 DomU - 3,06GHz - NAS-LUA	59
Nativo - 3,06GHz - NAS-LUB	266,76
Dom0 - 3,06GHz - NAS-LUB	272,36
1 DomU - 3,06GHz - NAS-LUB	271,68
Nativo - 3,06GHz - Spec SPU 2006	44
Dom0 - 3,06GHz - Spec SPU 2006	45
1 DomU - 3,06GHz - Spec SPU 2006	45
Nativo - 3,06GHz - CPUburn	120
Dom0 - 3,06GHz - CPUburn	120
1 DomU - 3,06GHz - CPUburn	120
Nativo - 3,06GHz - Iperf	19,4
Dom0 - 3,06GHz - Iperf	22,5
1 DomU - 3,06GHz - Iperf	32,8
Nativo - 3,06GHz - IoZone	23,1
Dom0 - 3,06GHz - IoZone	27,67
1 DomU - 3,06GHz - IoZone	6,67

A grande variação no custo energético entre diferentes *benchmarks* é resultado da diferença de tempo de computação. Alguns *benchmarks* levam muito mais tempo para serem executados que outros. O NAS LU classe B, por exemplo, leva cerca de 6 vezes

mais tempo que o SpecCPU 2006 para ser computado. Além disso, como visto na seção 4.2., o instrumento de medida (o wattímetro) tem uma precisão de 0,0001 kWh (360 Joules). Portanto, é natural que alguns valores de consumo energéticos muito próximos acabem aparecendo como idênticos nos resultados das medições (exceção feita ao CPUburn, onde a energia total é calculada a partir da potência média dissipada).

Como foi explicado no capítulo 3, o monitor de máquina virtual implementado pela arquitetura Xen é um hipervisor nativo. As máquinas virtuais instanciadas no ambiente de testes são paravirtualizadas. Como ilustrado na figura 3.8, o Xen implementa uma arquitetura de virtualização um pouco particular. O acesso aos dispositivos de E/S são feitos pelo Dom0 (possui os *drivers* reais dos dispositivos) sem a intervenção do hipervisor. Sendo assim, ao executar uma instrução sensível, o sistema operacional modificado de uma máquina virtual Xen vai executar uma chamada ao Dom0 (no caso de instruções que acessem E/S) ou substituir a instruções sensíveis por chamadas ao hipervisor (nas demais instruções sensíveis). As instruções não sensíveis de uma máquina virtual, por sua vez, serão executadas diretamente no hardware.

Os *benchmarks* CPU *Bound* (NAS LU A, NAS LU B, SpecCPU 2006 e CPUburn) executam basicamente instruções aritméticas e lógicas (instruções não sensíveis). Sendo assim, espera-se que o custo energético da execução desses *benchmarks* em máquinas virtuais seja bastante parecido com o custo da execução em um sistema não virtualizado. No gráfico da figura 4.1, esse comportamento foi constatado. Para os quatro *benchmarks* CPU *bound* o sobrecusto energético oriundo da gerência das máquinas virtuais é bastante pequeno. A quantidade de energia consumida pela execução dos *benchmarks* em um sistema nativo, no Dom0 ou em uma máquina virtual varia menos de 5% para todos os casos de teste.

Os *benchmarks* I/O *bound* (Iperf e Iozone), por sua vez, executam uma grande quantidade de instruções de E/S. Como ilustrado na figura 3.8, na arquitetura Xen, uma instrução de E/S, executada por uma máquina virtual, gera uma chamada ao Dom0, que possui os *drivers* físicos dos dispositivos e é capaz de tratar a requisição. Sendo assim, é de se esperar um sobrecusto energético na computação desses *benchmarks* em máquinas virtuais. No caso do Iperf, foi observado um acréscimo de 50% na quantidade de energia gasta pela máquina virtual em relação ao mesmo *benchmark* executando no Dom0. O sistema nativo, por sua vez, consome cerca de 14,28% menos energia que o Dom0, o que evidencia um pequeno sobrecusto energético oriundo da gerência do domínio privilegiado.

O Iozone tem um comportamento diferente do previsto. Executar o *benchmark* em uma máquina virtual é, aparentemente, menos custoso energeticamente. Esse resultado é falso. Na realidade, aproveitando-se do tamanho reduzido do disco da máquina virtual (4 GB), o hipervisor coloca a maior parte desse disco em memória. Sendo assim, ao executar em uma única máquina virtual, o Iozone acaba transformando-se em um *benchmark* misto de disco e acesso em memória. Como consequência, o Iozone executa muito mais rapidamente em máquina virtual que no Dom0 (que efetivamente acessa o disco) consumindo, assim, menos energia. Quando compara-se a execução do Iozone no Dom0 e no sistema nativo, novamente é constatado um sobrecusto energético de cerca de 14% oriundo da gerência do domínio privilegiado.

4.3.2 Potência Dissipada por *n* Máquinas Virtuais

O gráfico da figura 4.2 mostra a potência dissipada pela máquina física durante a execução dos *benchmarks* em um sistema nativo, no Dom0 e em *n* máquinas virtuais.

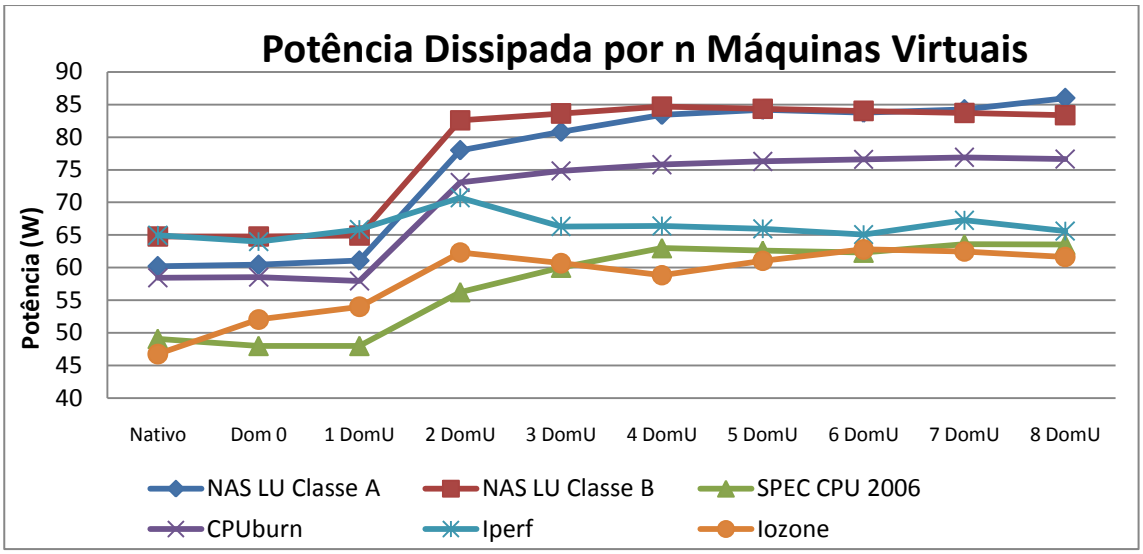

Figura 4.2: Potência dissipada por *n* máquinas virtuais

A tabela 4.2 ilustra o tempo de execução de cada *benchmark* em *n* máquinas virtuais. Por ser um laço infinito, não faz sentido comparar tempos de execução para o CPU burn. Apenas a potência média dissipada durante a execução do *benchmark* foi analisada.

Tabela 4.2: Tempo de execução dos *benchmarks* em *n* máquinas virtuais.

	Tempo (s)				
Sistema	**NAS LU Classe A**	**NAS LU Classe B**	**Spec CPU 2006**	**Iperf**	**Iozone**
Nativo - 3,06GHz	56,8	266,76	44	19,4	23,1
Dom0 - 3,06GHz	57,6	272,36	45	22,5	27,67
1 DomU - 3,06GHz	59	271,68	45	32,8	6,71
2 DomU - 3,06GHz	75	388	48	56	17
3 DomU - 3,06GHz	108	515	54	76	58
4 DomU - 3,06GHz	141	642	60	103	69
5 DomU - 3,06GHz	175	794	69	131	77
6 DomU - 3,06GHz	211	945	78	166	86
7 DomU - 3,06GHz	245	1097	90	182	81
8 DomU - 3,06GHz	276	1248	102	225	86

De acordo com as constatações da seção 4.3.1, a execução dos *benchmarks* CPU *bound* em uma máquina virtual consome praticamente a mesma potência que a execução dos *benchmarks* no sistema nativo ou no Dom0. No caso dos *benchmarks* I/O *bound*, observa-se o crescimento da potência na passagem de Dom0 para 1 DomU, conseqüente do custo de gerenciamento da máquina virtual. Além disso, a adição da segunda máquina virtual resulta em um aumento significativo na potência dissipada pela

máquina física. A adição das demais máquinas virtuais não representa grande aumento na taxa com que a máquina física consome energia.

Para se entender esse comportamento, deve-se considerar a arquitetura do processador e o cenário de testes. O ambiente virtualizado de testes consiste em um hipervisor, um domínio privilegiado (Dom0) e um número variado de máquinas virtuais (DomU). O processador da máquina física que hospeda esse ambiente possui dois núcleos de processamento físicos e quatro lógicos (tecnologia *Hyper Treading*). Quando apenas uma máquina virtual está executando o *benchmark*, em média, um dos núcleos do processador será plenamente ocupado pela máquina virtual e o outro passará a maior parte do tempo ocioso, responsável apenas por eventuais tarefas de sistema (gestão de hardware pelo hipervisor ou gestão da máquina virtual pelo Dom0). Ao se adicionar a segunda máquina virtual, os dois núcleos do processador serão plenamente utilizados, um por cada MV. Isso explica o grande aumento na potência dissipada pela máquina oriundo da adição da segunda máquina virtual. A partir de então, com o processador plenamente ocupado, cada máquina virtual adicionada resultará em um aumento no tempo de execução (as MV's acabarão concorrendo entre si pelo processador) mas a potência dissipada permanecerá praticamente constante. Em alguns casos, ainda, (como no Spec CPU2006) observa-se um crescimento na potência dissipada entre 2 DomU e 3 DomU. Isto é conseqüência da tecnologia *Hyper Treading* [C HARDWARE, 2011] que simula em um núcleo de processamento físico dois processadores lógicos, provendo ao sistema operacional uma maior possibilidade de paralelismo e aumentando o desempenho global do processador.

Além disso, o gráfico da figura 4.2 sugere que a potência dissipada por uma máquina não é proporcional à utilização da CPU. Quando se tem um grande número de máquinas (entre 6 e 8) virtuais executando aplicações CPU *bound* na mesma máquina física, a ocupação do processador é praticamente 100 por cento o tempo todo. Apesar de estar ocupado sempre, alguns *benchmarks* fazem o processador dissipar mais potência que outros. O NAS LU classe A, por exemplo executando em 8 MVs dissipa 35% mais potência que o Spec CPU2006 em 8 máquinas virtuais. Essa diferença deve-se ao fato que duas instruções diferentes podem consumir uma diferente quantidade de energia. Dessa forma, o gráfico da figura 4.2 indica que o NAS LU classe A executa instruções que consomem mais energia que as instruções executadas pelo SPEC CPU2006.

4.3.3 Custo Energético Médio de uma Instrução

O gráfico da figura 4.3 mostra o custo energético médio da execução de uma instrução em uma máquina virtual e no Dom0, durante a execução do *benchmark* NAS LU classe A. Os dados foram obtidos a partir do número médio de operações realizadas por segundo durante a execução do *benchmark* (dado informado pelo relatório disponibilizado pelo NAS LU a cada execução) multiplicado pelo tempo total da execução.

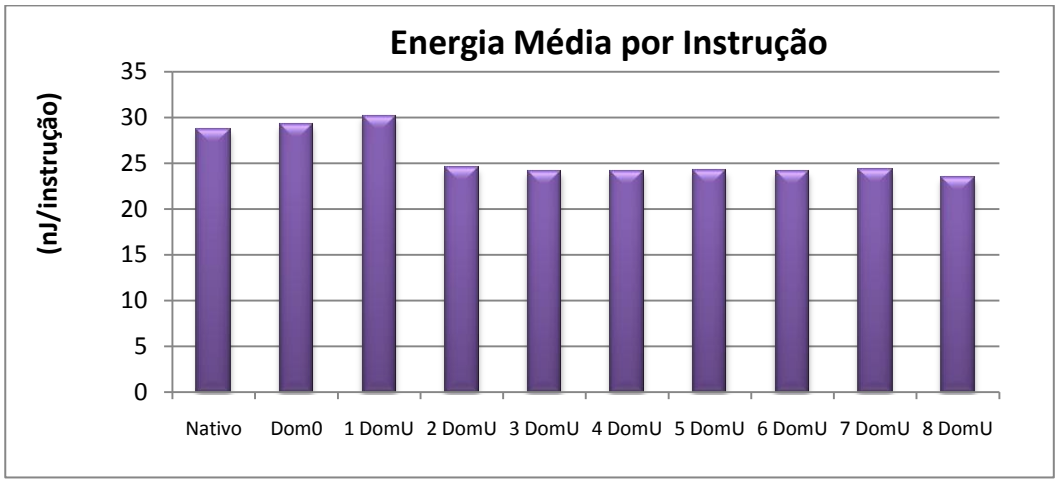

Figura 4.3: Energia média por instrução para o *benchmark* NAS LU classe A

Pelo fato do *benchmark* NAS LU classe A executar basicamente instruções não sensíveis, as instruções das máquinas virtuais serão executadas diretamente no processador. Isso se reflete em custo energético parecido para as instruções executadas pelo sistema nativo, no Dom0 e por uma MV. Como havia sido previsto na introdução deste trabalho, o acréscimo de máquinas virtuais em uma máquina física aumenta a sua eficiência energética. Conforme mais máquinas virtuais são colocadas para executar, a ocupação da máquina física aumenta e, em geral, a relação joules/instrução diminui, aumentando a eficiência energética do sistema.

A tabela 4.3 fornece os dados utilizados para a construção do gráfico da figura 4.3.

Tabela 4.3: Energia por Instrução para o *benchmark* NAS LU classe A

	Energia Consumida (J)	Número de Instruções	Energia por Instrução (nJ/Instrução)
Nativo	3420	1,192E+11	28,681
Dom0	3480	1,192E+11	29,185
1 DomU	3600	1,193E+11	30,161
2 DomU	5880	2,400E+11	24,491
3 DomU	8700	3,619E+11	24,039
4 DomU	11760	4,878E+11	24,105
5 DomU	14760	6,107E+11	24,167
6 DomU	17640	7,320E+11	24,096
7 DomU	20640	8,515E+11	24,238
8 DomU	23760	1,014E+12	23,410

4.4 Cenário II: Consumo versus Desempenho

Com o intuito de mensurar o impacto da redução da frequência de processamento no consumo energético e no desempenho de aplicações, os *benchmarks* CPU *bound* e I/O *bound* foram executados em *n* máquinas virtuais, para duas frequências de processamento: $F_{máx}$ = 3,06 GHz e F_{min} = 1,197 GHz .

O gráfico da figura 4.4 mostra o tempo que os *benchmarks* NAS LU classe A e Spec CPU 2006 levaram para serem executados em *n* máquinas virtuais para $F_{máx}$ e F_{min}.

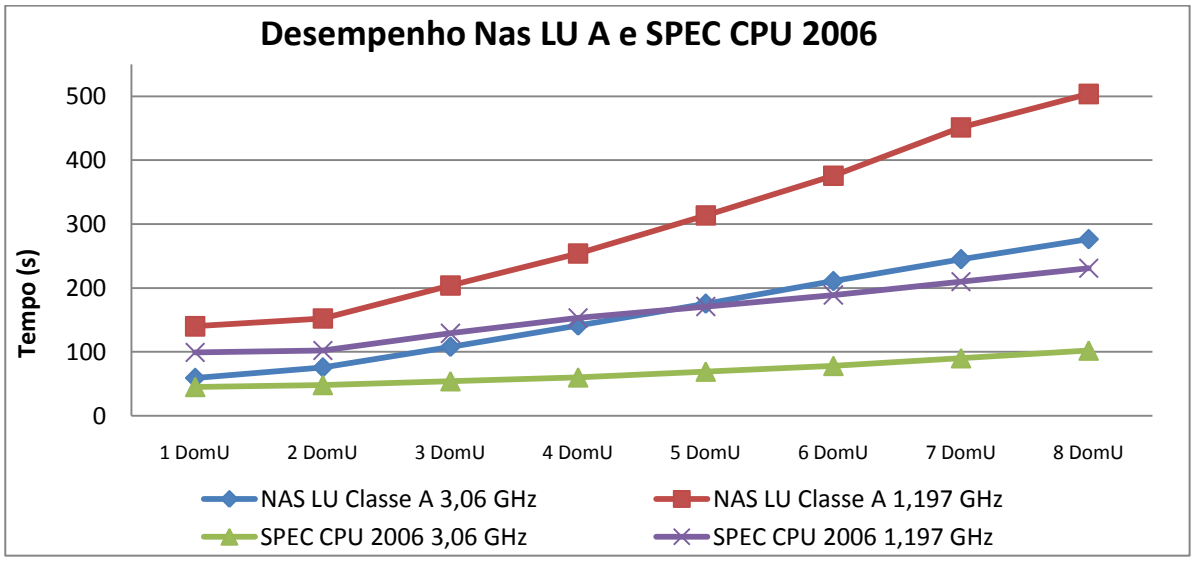

Figura 4.4: Desempenho NAS LU classe A e SPEC CPU2006

Como era esperado, a redução da frequência de processamento diminui o desempenho da execução. Os tempos de execução dos benchmarks NAS LU classe A e Spec CPU2006 em F_{min} são, em média, respectivamente, cerca de 82% e 133% maiores do que os tempos de execução em $F_{máx}$. Além disso, observa-se que a adição da segunda máquina virtual aumenta pouco o tempo de processamento. A partir da adição da terceira máquina virtual, o crescimento das curvas acentua-se. Esse comportamento é o reflexo da arquitetura do processador e está intimamente relacionado com o comportamento das curvas de dissipação de potência apresentadas na figura 4.2 Quando executa-se apenas uma MV, a tendência é que um dos núcleos de processamento fique ocioso a maior parte do tempo. Executando duas MVs, o processador fica com seu dois núcleos ocupados. A partir da adição da terceira MV, as máquinas virtuais começam a ser escalonadas, o que se reflete em um pior desempenho global.

O gráfico da figura 4.5 mostra o tempo que os *benchmarks* I/O *bound* levaram para serem executados em *n* máquinas virtuais para cada frequência de processamento.

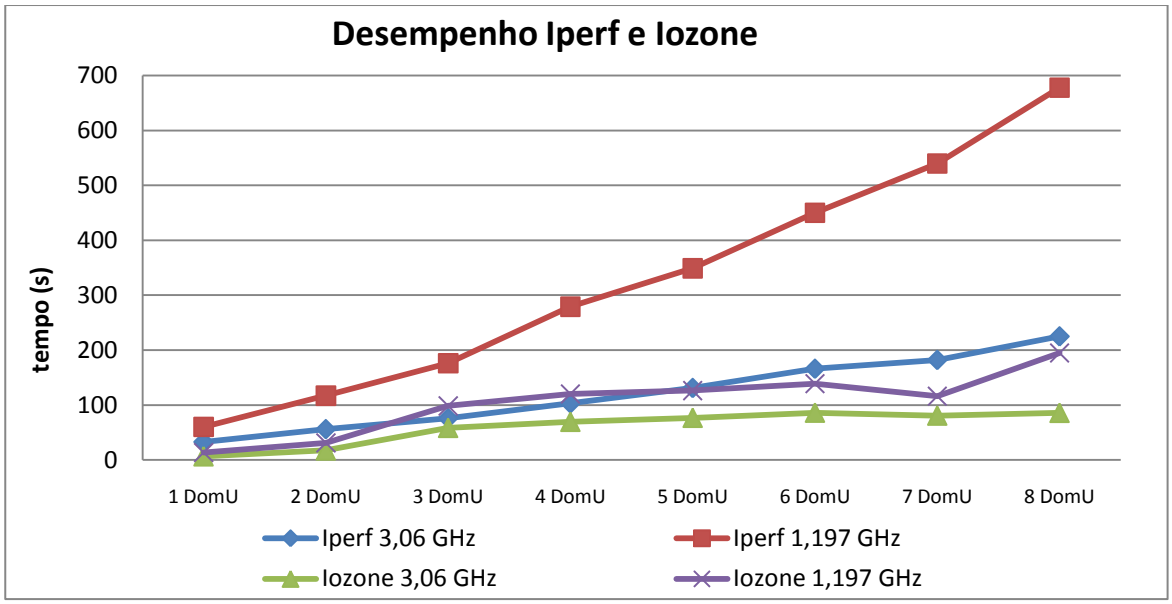

Figura 4.5: Desempenho Iperf e Iozone

Novamente o desempenho do processador em $F_{máx}$ é melhor do que em F_{min}. Apesar dos *benchmarks* principalmente lidarem com componentes de E/S, algum processamento é exigido para se gerenciar o sistema de arquivos (no caso do Iozone) ou para se construir os pacotes UDP (Iperf).

Percebe-se um crescimento um pouco desordenado e inconstante do tempo de execução dos *benchmarks* I/O *bound*. Por depender de elementos externos à CPU (dispositivos de E/S), o comportamento da execução é menos determinístico e menos linear. Na leitura e escrita de arquivos em disco (Iozone), por exemplo, dado a *cache* do sistema operacional e o tempo de acesso ao disco influenciam diretamente no tempo de execução. No caso do Iperf o servidor terá que lidar com vários clientes, o que pode representar um gargalo como sugere a curva Iperf 1,197 GHz. Além disso, vale ressaltar que, apenas o Dom0 possui os *drivers* reais dos dispositivos. Sendo assim, para acessar dispositivos de E/S, as máquinas virtuais terão de fazer uma requisição ao Dom0.

A figura 4.6 mostra a quantidade de energia consumida pela execução dos *benchmarks CPU* bound (NAS LU classe A e Spec CPU2006) em *n* máquinas virtuais para $F_{máx}$ e F_{min}.

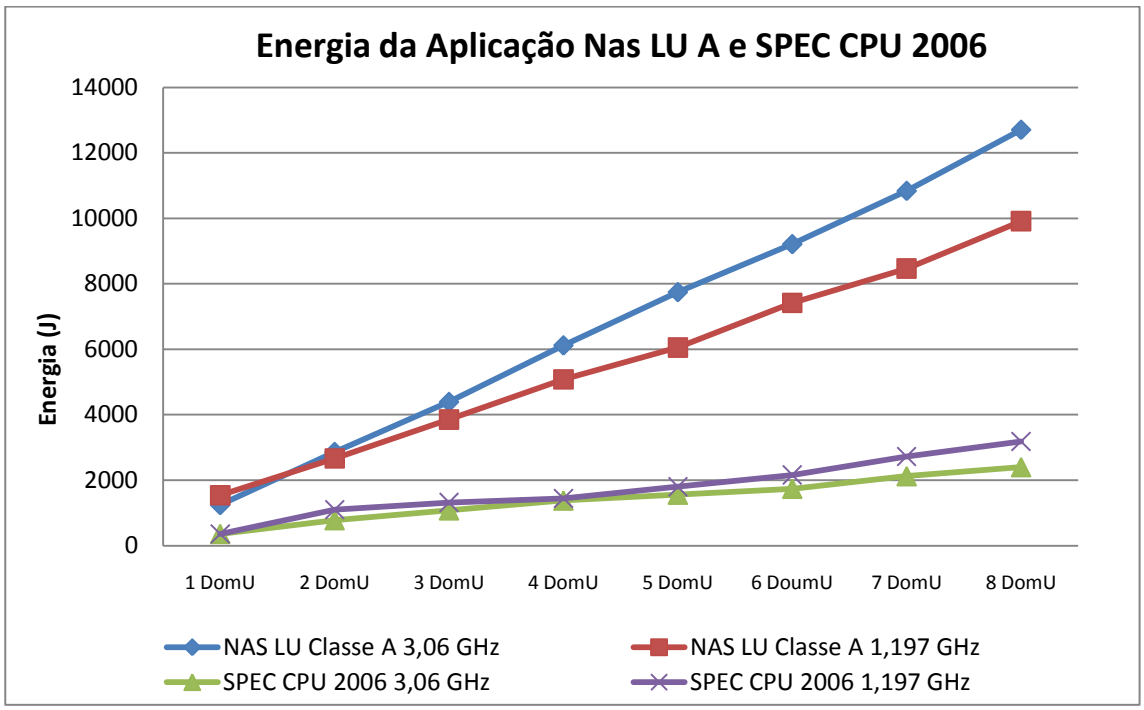

Figura 4.6: Energia consumida pelos *benchmarks* NAS LU classe A e SPEC CPU2006

A tabela 4.4 ilustra o gráfico da figura 4.6:

Tabela 4.4: Comparação entre desempenhos de $F_{máx}$ e F_{min} para os *benchmarks* CPU *bound*

	NAS LU Classe A			SPEC CPU 2006		
	Energia (J)		EFmin/EFmáx	Energia (J)		EFmin/EFmáx
	3,06 GHz	1,197 GHz		3,06 GHz	1,197 GHz	
1 DomU	1242,667	1540,000	1,24	360	360	1,00
2 DomU	2864,267	2666,667	0,93	780	1104	1,42
3 DomU	4393,333	3853,333	0,88	1080	1320	1,22
4 DomU	6120,000	5080,000	0,83	1380	1440	1,04
5 DomU	7746,667	6053,333	0,78	1560	1800	1,15
6 DomU	9213,333	7413,333	0,80	1740	2160	1,24
7 DomU	10840,00	8466,667	0,78	2124	2724	1,28
8 DomU	12706,67	9913,333	0,78	2400	3180	1,33
Média	6890,870	5623,330	0,82	1428	1761,00	1,23

Na tabela 5.2, a coluna Energia representa a $E_{aplicação}$. A coluna "$EF_{min}/EF_{máx}$" é a razão entre a quantidade de energia utilizada para se executar os *benchmarks* na mínima e máxima frequência de processamento.

Para o *benchmark* NAS LU classe A, constatou-se uma redução média de 18% no consumo energético quando o processador executa em menor frequência. Contudo, para apenas uma máquina virtual o consumo total de energia é superior para F_{min}. No caso dos SPEC CPU2006, no entanto, a aplicação consome em média, 23% mais energia quando executada em menor frequência. A explicação para esse diferente comportamento vem do fato de o SPEC CPU2006 ser um *benchmark* mais leve, dissipando menos potência e carregando menos a CPU (figura 4.2). Sendo assim, a defasagem entre a potência dissipada em $F_{máx}$ e F_{min} na execução do SPEC CPU2006 é menor do que a defasagem observada entre a execução do NAS LU classe A nas duas frequências. Para o NAS LU classe A, a execução em $F_{máx}$ dissipa, em média, cerca de 34% mais potência que a execução em F_{min}. A execução do SPEC CPU2006 em $F_{máx}$, por sua vez, dissipa, em média, cerca de 15% mais que em F_{min}. Sendo assim, no caso do Spec CPU2006, a pequena redução da potência gerada pela diminuição da frequência de processamento, não compensa o grande aumento no tempo de execução. Para o NAS LU classe A, a redução de potência compensa, em alguns casos, o aumento do tempo de execução, o que se reflete em um consumo total de energia ligeiramente menor para F_{min}. Deve-se no entanto avaliar, neste caso, se essa pequena economia de energia compensa o grande aumento no tempo de processamento da aplicação.

O gráfico da figura 4.7 mostra a quantidade de energia consumida pela execução dos *benchmarks* CPU *bound* em *n* máquinas virtuais para $F_{máx}$ e F_{min}.

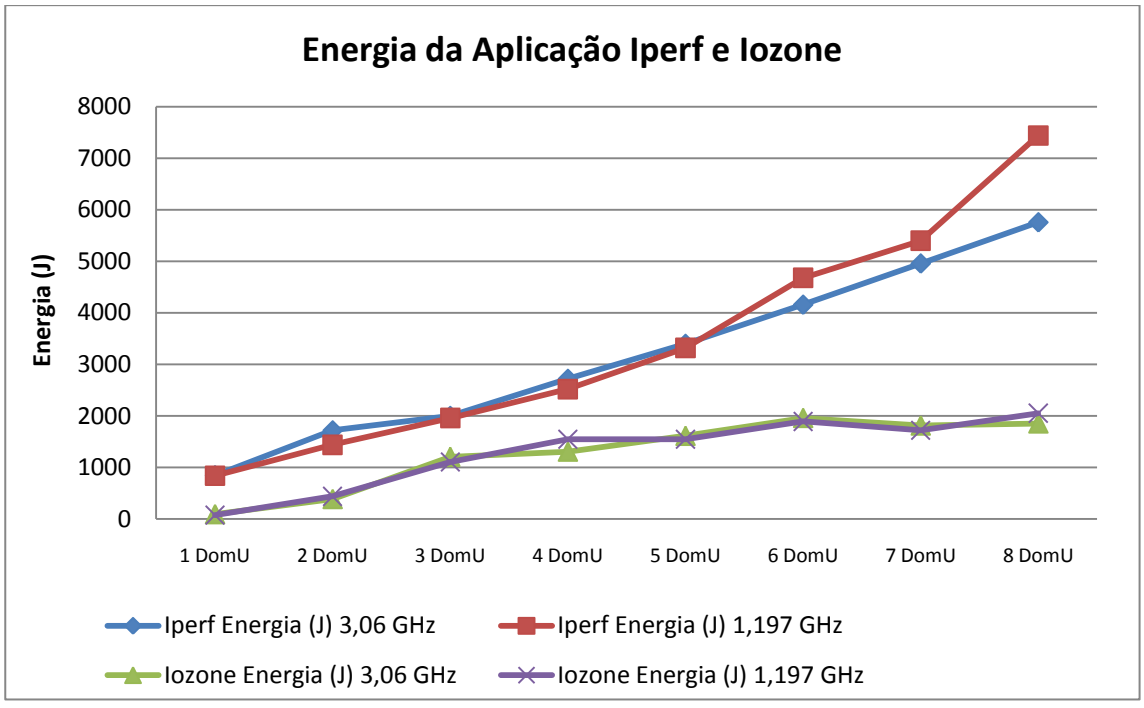

Figura 4.7: Energia consumida pelos *benchmarks* Iperf e Iozone

A tabela 4.5 sintetiza o gráfico da figura 4.7:

Tabela 4.5: Comparação entre desempenhos de $F_{máx}$ e F_{min} para os *benchmarks* I/O *bound.*

	Iperf			Iozone		
	Energia (J)		EFmin/EFmáx	Energia (J)		EFmin/EFmáx
	3,06 GHz	1,197 GHz		3,06 GHz	1,197 GHz	
1 DomU	848	840	0,99	93,33	77,33	0,83
2 DomU	1720	1440	0,84	386,67	440,00	1,14
3 DomU	2000	1960	0,98	1206,67	1106,67	0,92
4 DomU	2720	2520	0,93	1306,67	1546,67	1,18
5 DomU	3400	3320	0,98	1613,33	1546,67	0,96
6 DomU	4160	4680	1,13	1960,00	1893,33	0,97
7 DomU	4960	5400	1,09	1813,33	1720,00	0,95
8 DomU	5760	7440	1,29	1853,33	2053,33	1,11
Média	3196	3450	1,08	1279,17	1298,00	1,01

No caso dos *benchmarks* I/O *bound*, constata-se uma pequena economia de energia para o Iperf executando em até cinco máquinas virtuais.. Observa-se então que, com o aumento do número de máquinas virtuais, a relação $EF_{min}/EF_{máx}$ aumenta até o ponto em que a aplicação consome mais energia em F_{min} que em $F_{máx}$. Observando-se o gráfico da figura 4.6, conclui-se que esse comportamento é normal, visto que, o tempo de processamento dos *benchmarks* I/O *bound* cresce mais rapidamente com o processador em F_{min} que em $F_{máx}$. Sendo assim, é natural que a energia consumida (que é diretamente influenciada pelo tempo de processamento) apresente comportamento semelhante.

4.5 Considerações Gerais

O gerenciamento de máquinas virtuais apresenta um custo considerável apenas em aplicações que exigem grande quantidade de operações de E/S. Aplicações CPU *bound* apresentam desempenhos – computacional e energético – semelhantes quando executadas em máquinas virtuais ou sistemas nativos. O custo energético oriundo da adição de uma nova máquina virtual em uma máquina física depende da arquitetura do processador e do tipo de aplicação. Contudo, a adição de novas máquinas virtuais garante um melhor desempenho energético da máquina física.

A redução da frequência de processamento proporcionou, em alguns casos, uma pequena economia de energia. No entanto, o tempo de processamento acabou aumentando demasiadamente. Dificilmente será viável arcar com tal perda de desempenho para se economizar tão pouca energia.

5 CONCLUSÃO

A computação verde é uma nova área de pesquisa que vem ganhando importância na comunidade científica. A crescente popularidade do conceito de desenvolvimento sustentável e a implantação das taxas de carbono (imposto cobrado referente à emissão de dióxido de carbono por uma companhia) em vários países, fizeram com que a indústria também virasse sua atenção à computação verde. Alguns esforços têm sido feitos para se incluir requisitos de eficiência energética nos projetos. Muitos componentes de hardware e software implementam técnicas que visam otimizar o seu consumo energético. Contudo, as implementações ainda são escassas e restritas a alguns poucos componentes. Ainda há muito o que se evoluir para que as plataformas computacionais possam ser consideradas energeticamente eficientes.

No contexto de utilização eficiente de recursos em ambientes distribuídos, a virtualização acabou ressurgindo depois de anos de esquecimento. O uso máquinas virtuais tem diversas vantagens, entre elas, a capacidade de se melhor aproveitar os recursos energéticos de uma plataforma. Além disso, a virtualização é a base para a utilização de *Cloud Computing*, pois os serviços prestados pelos *Clouds* são providos por técnicas de virtualização. Dessa forma, as máquinas virtuais podem migrar para servidores em diferentes locais do planeta, visando sempre estar consumindo energia no local onde a demanda é menor naquele momento.

Os testes executados neste trabalho serviram para se compreender como as máquinas virtuais consomem energia e como as técnicas de redução de consumo energético influenciam no desempenho computacional e energético do computador. Os dois objetivos propostos neste trabalho (seção 1.1) foram atingidos graças aos cenários de testes desenvolvidos.

Através do primeiro cenário de testes (cenário I, seção 4.3) foi possível compreender-se melhor o custo energético de máquinas virtuais. Constatou-se que o sobrecusto energético da virtualização é muito pequeno para aplicações CPU *bound*. Esse comportamento é explicado pelo fato que, essas aplicações executam basicamente instruções não sensíveis. Essas instruções, ao serem geradas por uma máquina virtual, podem ser executadas diretamente no processador, sem acarretar, desta forma, nenhum sobrecusto computacional (seção 3.2). Por outro lado, aplicações I/O *bound* apresentam um sobrecusto energético considerável quando executadas em máquinas virtuais. Uma instrução de E/S gerada por uma máquina virtual, tem de ser tratada pelo monitor de máquina virtual (ou pelo Dom0, no caso Xen), gerando assim, um sobrecusto computacional e, consequentemente, energético.

Adicionalmente, os testes realizados no cenário I, sugerem que o aumento do número de máquinas virtuais em uma mesma máquina física, tende a aumentar a eficiência energética da máquina física. A quantidade de joules utilizados para a

execução de uma instrução de máquina virtual tende a diminuir com o aumento do número de máquinas virtuais no sistema. Por fim, o primeiro cenário de testes mostrou que a quantidade de potência dissipada não é proporcional à utilização da CPU. Duas aplicações que ocupam completamente o processador podem dissipar diferentes valores de potência dependendo do tipo de instrução que cada aplicação executa.

O segundo cenário de testes (cenário II, seção 4.4) mostrou o custo em desempenho e os benefícios em economia de energia oriundos da redução da frequência de processamento. A partir da análise dos resultados, concluiu-se que, em alguns casos, uma pequena economia de energia pode ser obtida. Contudo, a grande perda em desempenho dificilmente será compensada por essa pequena economia. Para os testes realizados, a tecnologia de redução de frequência, trouxe mais prejuízos que benefícios. Concluiu-se, portanto, que, para os testes realizados, a técnica alternativa à redução da frequência, o conceito de *race to idle* (onde a idéia é se executar as tarefas o mais rápido possível para se voltar ao estado de *idle*) é a abordagem mais interessante a se tomar.

Apesar dos testes com a técnica de redução de frequência não ter apresentado resultados satisfatórios, deve-se incentivar o desenvolvimento de medidas de redução de consumo em componentes computacionais. As técnicas existentes hoje evoluirão e novas surgirão para se aumentar as possibilidades de se reduzir consumo energético. No entanto, a cultura dos usuários também deve evoluir. Hoje, dificilmente escolhemos um produto por sua eficiência energética. Além disso, muitas vezes, as funções de redução de consumo são desabilitadas ou não utilizadas. Funcionários desperdiçam energia mantendo o computador da empresa ligado durante a noite. Até mesmo os *spams* são uma forma de desperdício. A McAfee estimou que, em 2008, cerca de 62 trilhões de *spams* foram enviados via *e-mail*. A energia consumida pelo envio dessas mensagens poderia abastecer cerca de 2,4 milhões de residências durante um ano [RUTH, 2009].

Para terminar, algumas sugestões para futuros trabalhos. Seria interessante fazer uma análise do consumo energético de máquinas virtuais executando uma aplicação de processamento paralelo, como por exemplo, o MPI, mesclando-se fases de computação e fases de E/S. Executar-se os mesmos testes do capítulo 4 em um sistema baseado em virtualização assistida por hardware. A idéia aqui seria comparar as técnicas de paravirtualização e virtualização completa assistida por hardware. Outro experimento interessante seria mensurar o custo energético de cada instrução de uma determinada arquitetura em um sistema nativo e em uma máquina virtual. Desta forma seria possível se fazer uma análise comparativa e se avaliar o custo energético da virtualização para cada tipo de aplicação.

REFERÊNCIAS

[BROWN et al., 2010] D. J. Brown e C. Reams, Toward Energy-Efficient Computing, ACM Queue, v.8, n.2, p. 1-13, fev. 2010

[BUENO, 2009] H. Bueno. Virtualização, um Pouco de História, 2009. Disponível em <http://hbueno.wordpress.com/2009/04/29/virtualizacao-um-pouco-de-historia/>. acesso em maio de 2011

[C HARDWARE, 2011] http://www.clubedohardware.com.br/artigos/163, acesso em: jun 2011

[CARISSIMI et al., 2010] A. Carissimi, et al. Energy-Aware Scheduling of Parallel Programs. Conferência Latino Americana de Computação de Alto Rendimento (CLCAR). Gramado,: [s.n], 2010. p. 95 – 101.

[CARISSIMI, 2009] CARISSIMI, A. S. Virtualização: conceitos e aplicações em processamento paralelo, 2009

[CHIN, 2010] M. Chin, Power Draw of Individual Componentes. Disponível em <http://www.silentpcreview.com/article265-page4.html>, acesso em: Nov. 2010

[CPUBURN, 2011] http://packages.debian.org/squeeze/cpuburn, acesso em: jun 2011

[FAN et al., 2002] X. Fan, C. S. Ellis, A. R. Lebeck, Memory Controller Policies for DRAM Power Management, 2002

[GARRET, 2008] M. Garret. Powering Down, ACM Queue, v.5, n.7, p. 17-21, jan. 2008

[GOOGLE PM, 2011] http://www.google.com/powermeter/about/index.html, acesso em: jun 2011

[GUNARATNE et al., 2005] C. Gunaratne, K. Christensen, B. Nordman. Managing Energy Consumption Costs in Desktop Pcs and Lan Switches With Proxying, Split TCP Connections, and Scaling of Link Speed. Int.J. Network. Manag., v. 15, n. 5, p. 297-310, 2005

[IOZONE, 2011] http://packages.ubuntu.com/dapper/iozone3, acesso em: jun 2011

[IPERF, 2011] http://packages.ubuntu.com/natty/iperf, acesso em: jun 2011

[JOULEMETER, 2011] http://research.microsoft.com/en-us/projects/joulemeter/default.aspx, acesso em: jun 2011

[LESS WATT, 2011] Less Watt, http://www.lesswatts.org/, acesso em: jun 2011

[M. HYPER V, 2011] Microsoft Server Virtualization, Disponível em <http://www.microsoft.com/virtualization/en/us/products-server.aspx>, acesso em: maio de 2011.

[M. VIRTUAL PC, 2011] Microsoft Desktop Virtualization. Disponível em http://www.microsoft.com/virtualization/en/us/products-desktop.aspx, acessado em: maio de 2011.

[NAS, 2011-a] http://www.ann.jussieu.fr/~lehyaric/NASfromMPItoBSP/lu.html, acesso em: jun 2011

[NAS, 2011-b] http://en.wikipedia.org/wiki/NAS_Parallel_Benchmarks, acesso em: jun 2011

[NAS, 2011-c] https://www.nas.nasa.gov/Resources/Software/swdescriptions.html, acesso em: jun 2011

[NEWING, 2010] R. Newing. Powerful Argument for cutting IT energy consumption. Finantial Times – Special Report: Green Inovation and Design, 16-09-2010, p. 2

[NVIDIA, 2011] http://www.nvidia.com.br/object/product-geforce-gtx-590-br.html, acesso em: jun 2011, acesso em: jul 2011

[ORGERIE, 2011] A. Orgerie, L. Lefèvre, J. Gelas, Demystifying Energy Consumption in Grids and Clouds. International Conference on Green Computing, p.335-342, ago. 2010

[Poek et al., 1974] G. J. POPEK, R. P.GOLDBERG, Formal requirements for virtualizable third generation architectures. [S.l.]: ACM v.17, n.7, p.412–421. 1974

[ROBIN at al., 2000] Robin, J.S.; Irvine, C.E. Analysis of the Intel Pentium.s Ability to Support a Secure Virtual Machine Monitor. Proc. 9a USENIX Security Simposium, [S.l.:s.n] 2000.

[RUTH, 2009] S. Ruth. Green IT - More Than a Three Percent Solution? IEEE Internet Computing, v.13 n.4 p. 74-78, jul. 2009.

[SAMSUNG, 2011] http://www.samsung.com/br/consumer/tv-audio-video/tv/lcd/, acesso em: nov. 2010

[SAXE, 2010] E. Saxe, Power-Efficient Software, Queue, V.8, N.1, p.44-48, jan. 2010

[SILBERCHATZ, 2001] A. Silberchatz,, P. Galvin, Sistemas Operacionais. (1a edição). Rio de Janeiro: Campus, 2001.

[SINGH, 2004] A. Singh, An introduction to Virtualization, 2004. Disponível em <(http://www.kernelthread.com/publications)>, acesso: maio 2011.

[SMITH et al. 2005] J.E. Smith, R. Nair .The architecture of virtual machines.. IEEE Computer, v.38, n.5, p. 32-38, 2005.

[SNOWDON, 2005] D. C. Snowdon, S. Ruocco, e G. Heiser. Power management and dynamic voltage scaling: Myths and facts. Workshop de Power Aware Real-time Computing, New Jersey, USA, [s. n]. Set. 2005.

[SPEC, 2011] http://www.spec.org/cpu2006/, acesso em: jun 2011

[THINK WIKI, 2010] http://www.thinkwiki.org/wiki/How_to_make_use_of_Dynamic_Frequency_Scalin, Acessado em Novembro de 2010

[TOMSHARDWARE,2011] http://www.tomshardware.com/reviews/energy-disk-drive,1944.html, Acessado em Novembro de 2010

[VAHDAT, 2000] A. Vahdat,A . Lebeck, C. S. Ellis. Every Joule is Precious: the case for revisiting operating system design for energy efficiency. EW 9: 9° workshop ACM SIGOPS European workshop, New York, NY, USA: ACM, 2000. p. 31-36, index.idx?pagetype=subtype_p2 , Acessado em Novembro de 2010

[VIRTUAL BOX, 2011] Oracle VM VirtualBox User Manual. Disponível em <http://www.virtualbox.org/manual/UserManual.html>, acesso em: maio de 2011.

[VMWARE, 2011-a] VMWARE White Paper, Virtualization Overview. Disponível em < http://www.vmware.com/pdf/virtualization.pdf >, acesso em: maio 2011

[VMWARE, 2011-b] VMWARE White Paper, Understanding Full Virtualization, Paravirtualization and Hardware Assist. Disponível em < http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf >, acesso em: jun. 2011

[VMWARE, 2011-c] Virtualization Basics. Disponível em <http://www.vmware.com/virtualization/what-is-virtualization.html>, acesso em: maio de 2011

[VMWARE, 2011-d] VMWARE White Paper, How VMware Virtualization Right-sizes IT Infrastructure to Reduce Power Consumation. Disponível em < http://www.vmware.com/files/pdf/WhitePaper_ReducePowerConsumption.pdf>, acesso em: jun. 2011

[XEN, 2011] What is Xen? Disponível em <http://www.xen.org/files/Marketing/WhatisXen.pdf>, acesso em: maio 2011