Um framework para monitoramento do consumo energético em arquiteturas multicore

Bruno Giacobo Pinto[1]
Lucas Mendonça de Souza Xavier[1]
Rodolfo Migon Favaretto[2]
Gerson Geraldo Homrich Cavalheiro[1]

Resumo: Os processadores modernos oferecem informações sobre o seu consumo energético. No entanto, não há padronização da interface de acesso aos serviços que as disponibilizam e nem uma etapa de tratamento ou uniformização dos dados coletados por essas ferramentas. Isso constitui um inconveniente para o programador final e restringe a portabilidade das soluções que utilizam essas informações. Este artigo apresenta um framework que provê uma interface comum de serviços para acesso às informações sobre o consumo energético dos processadores. A solução desenvolvida fornece uma representação uniforme para os dados coletados e é desenvolvida utilizando uma arquitetura modular e extensível. Nessa abordagem, o consumo energético de programas paralelos, executando em arquiteturas baseadas em processadores multicore, pôde ser monitorado de forma dinâmica, independente da família de processadores suportando a execução. Esse framework é validado com dois estudos de caso que ilustram sua aplicabilidade.

***Palavras-chave*:** Análise de performance. Medição de energia. Monitoramento.

Abstract: *Modern processors provide a specialized interface of services to access data describing the energy consumption at runtime. However, such interfaces are not standardized nor they perform any data treatment or uniformization. This is undesirable to the end programmer and reduces the portability of solutions that are built upon that kind of information. This paper presents a framework that describes a common interface of services that expose processors' energy consumption. The developed solution has a modular and extensible architecture design and provides a uniform data representation for the collected data. In this approach, energy consumption of parallel programs running on a multicore-based architecture was monitored dynamically, independently from the processor family supporting the execution. Our work was validated through two case of studies that claim its applicability.*

***Keywords*:** *Energy Measurement. Monitoring. Performance Analysis*

1 Introdução

Arquiteturas baseadas em processadores multicore têm se consolidado como uma alternativa para o suporte computacional de um vasto leque de aplicações em função de seu custo abordável e da relação satisfatória entre sua capacidade de processamento e seu consumo energético. Em nível de software, diversas estratégias de escalonamento sensíveis ao consumo de energia (power ou energy-aware, em inglês) foram elaboradas tendo como base funcionalidades arquiteturais introduzidas nos processadores multicore que permitem obter dados sobre o consumo energético em tempo de execução. O emprego de tais heurísticas em nível de kernel de sistemas operacionais é recorrente em vários ambientes, como registra [1] e [2]. Em nível aplicativo, ou seja, no escalonamento de tarefas

[1]Centro de Desenvolvimento Tecnológico – CDTec/Computação, UFPel, Campus Porto - Rua Gomes Carneiro 1 - Pelotas (RS) - Brasil
{bgpinto, lmdsxavier, gerson.cavalheiro@inf.ufpel.edu.br}
[2]Programa de Pós-Graduação em Ciência da Computação – PPGCC, Pontifícia Universidade Católica do Rio Grande do Sul – PUCRS, Av. Ipiranga, 6681 – Prédio 32 - Porto Alegre (RS) - Brasil
{rodolfo.favaretto@acad.pucrs.br}

http://dx.doi.org/10.5335/rbca.2015.4886

no contexto de um programa em execução, o uso de tais heurísticas é mais recente, sendo apresentados trabalhos nesta direção em [3] e [4].

Entretanto, para o programador que deseja inserir no seu código sensível ao consumo energético, os benefícios ainda são apresentados de forma rudimentar. Atualmente, a interface de acesso às informações se apresenta sob a forma de chamadas de sistema a dispositivos de hardware especializados em contabilizar informações sobre o consumo de energia. A interpretação dos dados fornecidos por tais serviços devem ser ainda tratados e interpretados no próprio algoritmo do programa em execução. O uso prático de tais contadores é ainda mais complexo considerando que não há nem uma padronização seguida pelos diferentes modelos de processadores, nem mesmo o estabelecimento de uma unidade de medida de consumo uniforme. A carência que se observa é a inexistência de abstrações de mais alto nível para explorar informações sobre o consumo energético de programas em execução e realizar escalonamento em nível aplicativo [5]. Esse aspecto é crítico, pois em uma determinada máquina com uso específico para processamento dedicado e de alto desempenho, se o consumo energético gerado por um determinado programa paralelo não for considerado pela estratégia de escalonamento aplicativo, a consequência pode refletir um alto custo de operação com gastos com energia [6].

Para reduzir o consumo energético e maximizar o uso de recursos de um determinado programa em nível aplicativo, é necessário que as ações sejam tomadas em tempo de execução, particularmente se se tratar de uma aplicação com características irregulares [7, 8]. Nesse caso, independente da abordagem escolhida, é importante que se obtenha informações de consumo energético dos diferentes processadores e/ou *cores* envolvidos na computação, uma vez que essas informações podem servir como base em diversas estratégias de execução que tenham por objetivo reduzir esse índice. Possíveis cenários seriam: identificar pontos do sistema onde existe desperdício de energia, obter um perfil do consumo energético de uma aplicação ou alimentar decisões de escalonamento que realizam o mapeamento das atividades geradas pelos programas (threads) sobre os recursos computacionais disponíveis.

Neste trabalho, é apresentado um framework que oferece funcionalidades de alto nível para o programador incluir, no seu código de aplicação, estratégias de execução que considerem o consumo energético gerado pela execução de seu programa. Este framework foi implementado em C++ e modelado de forma a uniformizar e apresentar uma interface única de serviços para a coleta de dados de energia proporcionados pelas ferramentas presentes nas famílias de processadores mais recentes da Intel e da AMD. Também foi considerada a questão das diferentes unidades de representação do consumo energético nos diferentes modelos de processador e proposta uma forma de conversão entre essas unidades.

Na atual implementação do framework, os dados de energia coletados são provenientes de duas ferramentas presentes nas microarquiteturas dos processadores da Intel e AMD. As primeiras famílias a disponibilizar esses recursos foram a Sandy Bridge (da Intel) e a Bulldozer (da AMD). A solução projetada foi voltada para sistemas operacionais baseados em GNU/Linux.

Para validação do trabalho e verificação de suas funcionalidades, foram realizados dois experimentos. No primeiro, um benchmark sintético, no qual a carga computacional de diferentes threads pôde ser efetivamente estabelecida, permitiu verificar a variação do consumo energético dos diferentes processadores de uma arquitetura multicore, considerando variações na estratégia de execução. O segundo experimento considerou uma aplicação com estrutura irregular e dinâmica de geração do paralelismo, sendo monitorado o consumo energético consequente.

Este artigo se organiza como segue. A Seção 2 sumariza diferentes estratégias que podem ser empregadas para obter dados de consumo energético de computadores em tempo de execução. A Seção 3 apresenta outras ferramentas disponíveis para coleta de informações relacionadas ao consumo energético. A Seção 4 apresenta as ferramentas e os recursos utilizados na construção do framework proposto. A Seção 5 ilustra desenvolvimento do framework em si e apresenta sua estrutura. Na Seção 6, dois estudos de caso são conduzidos para exemplificar o uso do framework. Finalmente, a Seção 7 apresenta algumas considerações finais, discute os resultados apresentados na seção anterior e conclui o trabalho.

2 Métodos de obtenção de dados

A medida do consumo energético de um computador durante a execução de um programa pode ser obtida de diferentes maneiras. Equipamentos especializados em medir grandezas elétricas constituem técnicas de medição não invasivas. Modelos de estimativa de consumo podem ser derivados de metodologias que aproximam o consumo de potência a partir da observação de eventos indiretamente relacionados à energia. Ferramentas integradas na arquitetura são soluções em hardware que contam com recursos de contabilização de dados de energia. Esta seção dedica-se a apresentar e discutir as metodologias existentes para medição de consumo energético e a apresentar as vantagens e desvantagens dessas abordagens.

2.1 Medições não invasivas

Uma forma não invasiva empregada para obter dados sobre o consumo energético de um determinado elemento, seja um sistema completo ou um dispositivo que o compõe, pode ser feita pela introdução de um equipamento de hardware especializado na medição de grandezas elétricas (corrente, potência e tensão) entre a fonte energética e o elemento a ser aferido. Embora requeira um equipamento específico e dissipe ele próprio energia, esta forma de medição é dita não invasiva por não interferir nem no consumo, nem no desempenho do elemento objeto de estudo (no caso, o programa em execução). Considerando computadores executando programas, essa abordagem permite medir o consumo global da máquina, ou seja, do somatório do consumo dos processadores, discos, memórias e demais dispositivos que compõem o sistema computacional em uso, quando o equipamento de medição encontra-se ligado ao cabo que alimenta a fonte do computador. Caso o equipamento seja conectado entre a fonte deste computador e um de seus módulos (processador, placa ou disco, como exemplos), é possível verificar isoladamente a potência dissipada por componentes individuais de hardware. Em [9] são apresentados estudos de caso relacionados ao consumo de componentes específicos.

Entre os instrumentos que podem ser empregados para esse propósito estão equipamentos de propósito geral como wattímetros, amperímetros e osciloscópios, e medidores voltados para monitoramento de equipamentos individuais, como o Watts Up[3]. Esses dispositivos são úteis, em particular, para auxiliar na construção e no ajuste de parâmetros de modelos para estimar o consumo de energia, uma vez que servem como uma métrica de comparação consistente. Em [10] foi apresentado um estudo de caso no qual medidores de corrente e tensão foram utilizados para aferir o padrão do consumo energético resultante da execução de um programa paralelo em uma arquitetura multiprocessada considerando diferentes políticas de escalonamento.

Contudo, quando consideramos o monitoramento do consumo energético gerado pela execução de programas, essa abordagem apresenta algumas desvantagens. Especificamente, a necessidade de introduzir este equipamento de medição é um problema, pois costuma ser realizada de forma intrusiva e, não raro, exige a interrupção do sistema. Adicionalmente, deve ser considerado que os dados de consumo obtidos por este meio somente podem ser utilizados por um determinado programa em execução quando puderem ser acessados por este programa. Assim, se faz necessário um mecanismo de sincronização entre o dispositivo de monitoramento e o software cujo consumo gerado está sendo aferido. Nesse caso, a sincronização se torna um fator limitante e crítico para a consistência da medição, pois caracteriza uma fonte de imprecisão. Se empregado desta forma, consequências no consumo e desempenho da aplicação objeto de estudo serão observadas. Finalmente, a aquisição de equipamentos de medição implica em custos adicionais em hardware.

2.2 Modelos de estimativa

Nas arquiteturas das últimas gerações de processadores, foram introduzidos contadores de performance capazes de contabilizar eventos e obter informações em tempo real. A leitura desses contadores e a combinação dos valores obtidos por diferentes leituras permitem estimar o consumo de energia e/ou a dissipação de potência sem a necessidade de medição direta. Essas estimativas empregam modelos, baseados em regressão linear ou outras formas de correlação estatística para integralizar esses dados em informações de consumo. Entre os contadores de performance mais utilizados para a construção de modelos de consumo de energia, podemos mencionar faltas nas

[3] <http://www.wattsupmeters.com>

caches, número de operações de ponto flutuante realizadas e desempenho do preditor de desvios. Em [11] e [12] são apresentadas técnicas que utilizam tais contadores.

Ainda, diversos modelos, como o utilizado pelo software PowerTOP[4], utilizam informações de estados ACPI (Advanced Configuration and Power Interface) para observar a frequência de operação do processador e seu estado de funcionamento (*idle* ou *sleep*, por exemplo). Outros trabalhos empregam, para estimar o consumo, estatísticas como a taxa de utilização dos componentes em conjunto com outras informações do sistema.

Embora o uso de modelos dispense hardware para medição direta, normalmente, esse tipo de abordagem necessita de uma etapa de calibragem para adaptar os parâmetros para a configuração da máquina monitorada. Além disso, é mostrado em [13] que a medição de consumo por modelos pode não ser adequada para todos os tipos de aplicações devido a imprecisões. É aceitável, portanto, que um determinado modelo seja mais indicado para uma classe de aplicações que outro.

2.3 Ferramentas integradas

À medida que cresceu a demanda pelo desenvolvimento de software energy-aware, os fabricantes começaram a introduzir facilidades para medir o consumo em suas arquiteturas. Essa solução é particularmente interessante para desenvolvedores, já que torna mais conveniente a realização de medições de granularidade fina e o acesso aos dados de consumo coletados. Essa conveniência possibilita o uso, em tempo de execução, das informações de consumo obtidas, o que constitui uma vantagem na construção de aplicações energeticamente conscientes.

Exemplos de facilidades pertencentes a essa classe são encontradas em processadores modernos dos maiores fabricantes. A interface Application Power Management (APM) está disponível nas famílias de processadores da AMD [14], enquanto nos processadores da Intel encontra-se a interface Running Average Power Limit (RAPL) [15]. Embora com finalidades semelhantes, essas duas interfaces distinguem-se no modo em que apresentam os dados de consumo, o que dificulta a portabilidade de código sensível ao consumo de energia entre processadores desses dois fabricantes. No que diz respeito aos servidores, a questão da portabilidade de código pode ser tratada uma vez que os novos processadores para servidor de ambos fabricantes adotam o padrão Intelligent Platform Management Interface (IPMI) [16], que define uma interface comum para gerenciamento de servidores a qual inclui o acesso a sensores que monitoram o consumo energético no nível de sistema.

Neste trabalho, este conjunto de ferramentas é considerado, levando em conta a pluralidade de interfaces disponíveis e a dificuldade dos programadores em adequar seus programas à variedade de recursos apresentados. A proposta é fornecer um framework que ofereça uma interface comum para acesso de tais informações, independente do hardware utilizado.

3 Trabalhos relacionados

O trabalho cuja funcionalidade de coleta de dados de energia que mais se aproxima ao *framework* proposto é o módulo de monitoramento de consumo da Performance API (PAPI) [17]. PAPI é uma ferramenta que implementa uma interface em C para acessar contadores de performance disponibilizados pelo hardware. A ferramenta tem uma arquitetura modular, permitindo o desenvolvimento de novos componentes para estender a funcionalidade original. PAPI conta com componentes específicos para acessar dados de energia provenientes das interfaces RAPL e NVML (NVIDIA Management Library)[18].

Outros trabalhos propõem interfaces para acessar medições ou estimativas para o consumo do sistema, do processador ou das aplicações. A ferramenta Intel Power Gadget[5], compatível com sistemas Windows e Mac OS X, disponibiliza uma API que expõe os dados produzidos pela interface RAPL. Joule Watcher [19] é uma ferramenta baseada em contadores de performance que oferece estimativas para o consumo realizadas com o apoio de uma etapa de calibragem, na qual os coeficientes do modelo são ajustados com o apoio de medidores externos.

O software PowerTOP utiliza informações dos estados ACPI do processador, em conjunto com informações de descarga da bateria (se disponíveis) para estimar o consumo com granularidade de aplicação. e-Surgeon [20]

[4] <http://01.org/powertop>
[5] <https://software.intel.com/en-us/articles/intel-power-gadget-20>

é uma ferramenta de análise que, por meio da instrumentação de código Java, disponibiliza o consumo a nível de threads e métodos. PowerScope [21] é um software para análise *offline* de código que utiliza multímetros para coletar informações.

Em contraste com os trabalhos apresentados, o framework desenvolvido se diferencia dos demais por incluir a ferramenta APM, por uniformizar a representação dos dados e por fornecer uma abstração de mais alto nível para o programador final.

4 Delimitação de escopo

O escopo deste trabalho está restrito, inicialmente, a duas ferramentas integradas (APM e RAPL). A opção por essas duas ferramentas é motivada por sua disponibilidade, já que ambas estão presentes de forma nativa em processadores comerciais, e pelo fato de que ambas expõem estatísticas ligadas ao processador, o que é interessante para dar suporte a estratégias de escalonamento e aplicações CPU-bound. Adicionalmente, as interfaces escolhidas foram validadas em diversos contextos por trabalhos distintos [22, 23], o que indica sua confiabilidade e precisão. Em [24], validamos a interface RAPL quanto à sua viabilidade na estimativa do consumo de aplicações multiprocessadas. Finalmente, é possível destacar que as duas soluções geram as informações de consumo usando, para isso, hardware dedicado, reduzindo um possível sobrecusto na leitura dos valores estimados.

4.1 Application power management

A interface Application Power Management (APM) [14] foi introduzida pela AMD e os processadores da microarquitetura Bulldozer foram os primeiros a incorporar a ferramenta. A APM monitora a atividade dos *cores* de forma dinâmica e gera uma estimativa da potência consumida pelo processador baseada em contadores de performance, armazenando estas informações em registradores específicos. A taxa com que os dados são atualizados pela interface varia e depende de um período térmico significativo.

Essa ferramenta não se restringe a estimar potência. Ela também gerencia o consumo de energia do processador. Para realizar tal tarefa, a interface determina um limite de potência para que o desempenho não seja penalizado e o consumo seja minimizado. É importante citar que o mecanismo de gerência proporcionado pela APM é transparente ao kernel e à BIOS do sistema, delegando a responsabilidade ao processador.

4.2 Running average power limit

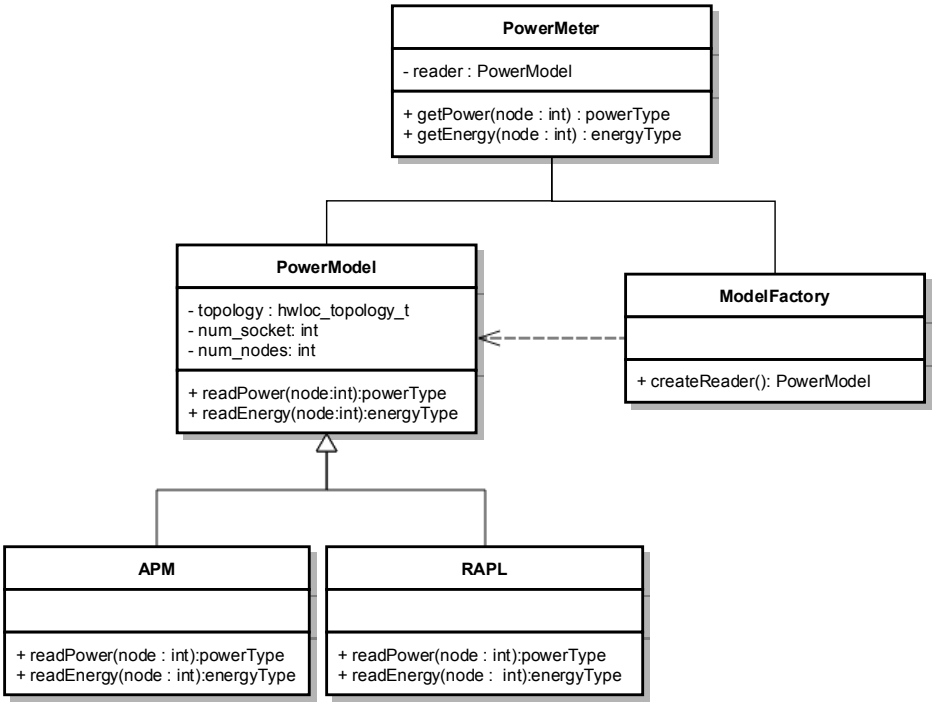
A interface Running Average Power Limit (RAPL) [15] foi incluída nos processadores da Intel a partir da microarquitetura denominada Sandy Bridge. A RAPL consiste em um conjunto de registradores que expõem estatísticas relacionadas ao consumo do processador e de seus componentes (como processadores gráficos integrados). Além disso, ela possui funcionalidades de gerenciamento de energia, permitindo a definição de limites para o consumo.

A interface da ferramenta com o sistema operacional é um conjunto de registradores não arquiteturais que podem ser acessados por meio de instruções privilegiadas de leitura e escrita. Entre esses registradores, estão contadores que acumulam continuamente a energia consumida pelo processador ou por seus subsistemas. Esses contadores são atualizados aproximadamente a cada milissegundo e representam estimativas calculadas a partir de um modelo linear baseado em contadores de performance. As estimativas são calculadas por uma unidade de hardware dedicada.

5 Modelagem do framework

Nesta seção, o processo de criação do framework proposto é descrito. São apresentadas as decisões de projeto tomadas para abstrair o programador da necessidade de conhecer a arquitetura e de tratar os dados resultantes da medição de consumo. No estágio atual, devido a limitações das ferramentas utilizadas, o framework retorna o consumo na granularidade de processador. A Figura 1 ilustra a relação entre as classes desenvolvidas.

Figura 1: Modelagem das classes do Framework



A classe abstrata *PowerModel* foi especificada com uma interface para padronizar a leitura do consumo e obter informações da arquitetura da máquina, possibilitando medir o consumo de um processador específico. Para coletar as informações da arquitetura, foi utilizada a API (Application Programming Interface) *Hwloc* [25]. *Hwloc* é implementada em C, está disponível para diversos sistemas operacionais e serve para obter a topologia da máquina de diversas formas.

A partir da classe *PowerModel* duas outras classes são especializadas: a classe *APM* e a classe *RAPL*. Fica a cargo dessas duas classes uniformizar a medição por meio da sobrecarga do método que lê o consumo energético. É por meio dessa estratégia que o framework desenvolvido permite acomodar novas ferramentas, desde que tais ferramentas estejam em conformidade com a interface proposta.

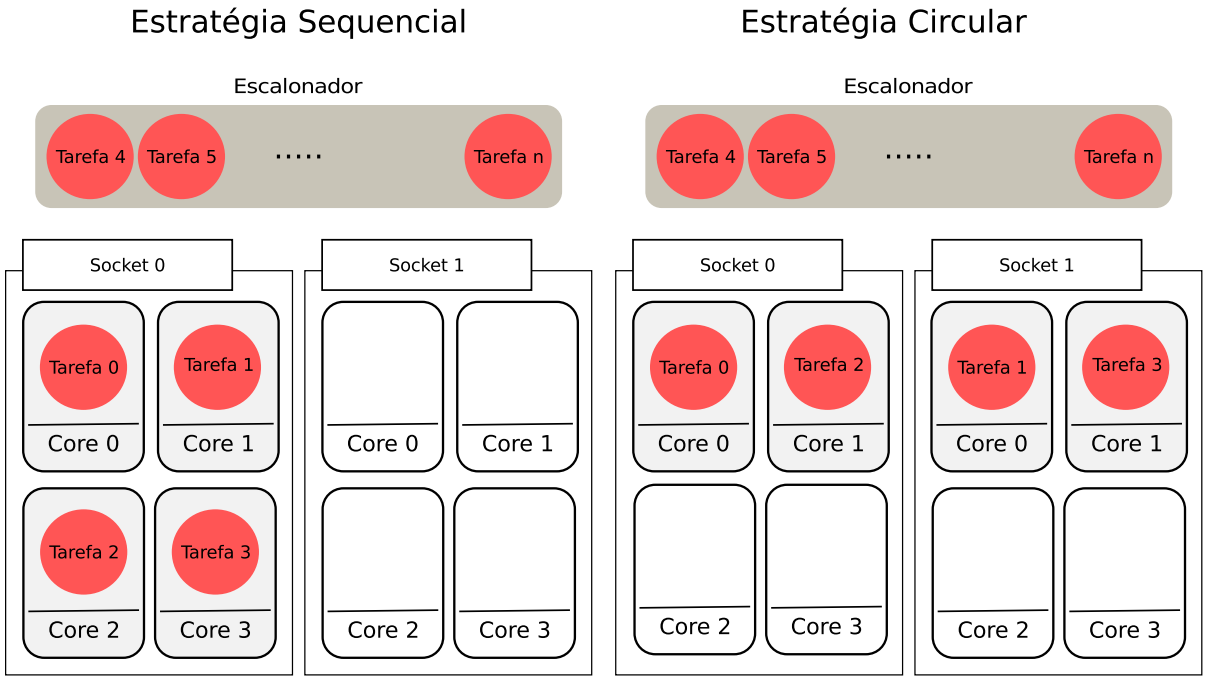
O método especializado da classe *APM* acessa um driver específico presente no kernel do Linux. A informação retornada pelo driver é a estimativa realizada pela APM para a potência dissipada pelo processador. Logo, para obter dados de energia consumida, é necessário integrar os valores de potência coletados. Em contraste, os drivers que alimentam a classe *RAPL* informam valores de energia; caso seja interessante obter valores de potência, deve-se monitorar a variação da energia consumida ao longo do tempo. Ambas as classes realizam amostragem periódica das estatísticas informadas para efetuar os cálculos que permitem uniformizar as grandezas coletadas.

A aplicação tem acesso a uma única classe, a classe *PowerMeter*, que permite consultar o consumo energético (em Joules) ou a potência dissipada (em Watts) de um processador especificado. No momento da construção de uma instância da classe, a arquitetura da máquina é verificada para que a ferramenta adequada seja utilizada. Isso é possibilitado por uma classe intermediária, a classe *ModelFactory*, que utiliza funções da arquitetura x86 para decidir qual classe especializada deve ser utilizada (*APM* ou *RAPL*). Dessa forma, a classe *PowerMeter* consegue proporcionar o mecanismo necessário para efetuar a leitura do consumo de forma transparente.

6 Avaliação e análise de resultados

Para analisar as funcionalidades do framework, foi avaliado o consumo energético de duas aplicações multithreaded: um benchmark sintético, com comportamento controlado, e o problema das n-rainhas, com compor-

Figura 2: Distrubuição de 4 tarefas seguindo as políticas desenvolvidas. Tarefas numeradas de 1 a n representam a carga computacional atribuída a cada *core*.



tamento irregular. Os benchmarks em questão são interessantes porque apresentam diferentes padrões de geração e distribuição de carga de trabalho, o que potencialmente leva a variações no consumo de energia observado nas unidades de processamento. O comportamento do consumo energético do processador foi monitorado ao longo de diversas execuções dos benchmarks, sob diferentes configurações de execução, objetivando demonstrar cenários de uso em que o framework é utilizado para levantar um perfil de consumo de aplicações multiprocessadas CPU-bound.

A arquitetura utilizada para rodar os experimentos é composta por quatro nós de processamento AMD Opteron 6276 com 2,3GHz e três níveis de cache (L3 de 6MB, L2 de 2MB e L1 de 64KB). Cada nó de processamento contém 16 *cores*, somando um total de 64. Essa máquina possui 128GB de memória principal, sendo 32GB por nó separados em blocos de 16GB. Utilizamos o sistema operacional GNU/Linux Ubuntu Server 12.04 LTS 64 bits com a versão 3.2.0-65 do kernel. Os resultados são apresentados em termos do tempo (em segundos) e energia (em quilojoules), representando uma média de 30 execuções cada caso, não sendo observado o desvio padrão maior que 2%.

A abordagem utilizada para expor diferentes padrões de distribuição de carga no processador foi implementar, em ambos os benchmarks, duas políticas simples para mapeamento de tarefas sobre os diferentes sockets disponíveis na plataforma de validação. O objetivo é observar variações no consumo e no tempo de execução em função da redução (ou aumento) de conflitos de escalonamento. A primeira política implementada, denominada *circular*, distribui a carga nos nós da máquina de forma homogênea, alocando tarefas nos nós de processamento disponíveis de forma alternada. Na segunda política, denominada *sequencial*, os threads da aplicação foram agrupados de forma a preencher todos os *cores* de um nó antes de distribuir tarefas para outros nós livres. A Figura 2 ilustra o funcionamento das políticas empregadas. Variando essas políticas, avalia-se o impacto do balanceamento da carga no consumo das duas aplicações em estudo, o que pode ser utilizado, por exemplo, para a construção de estratégias de escalonamento e gerência energeticamente eficientes e dinâmicas.

Figura 3: Resultados de tempo e energia para a aplicação sintética

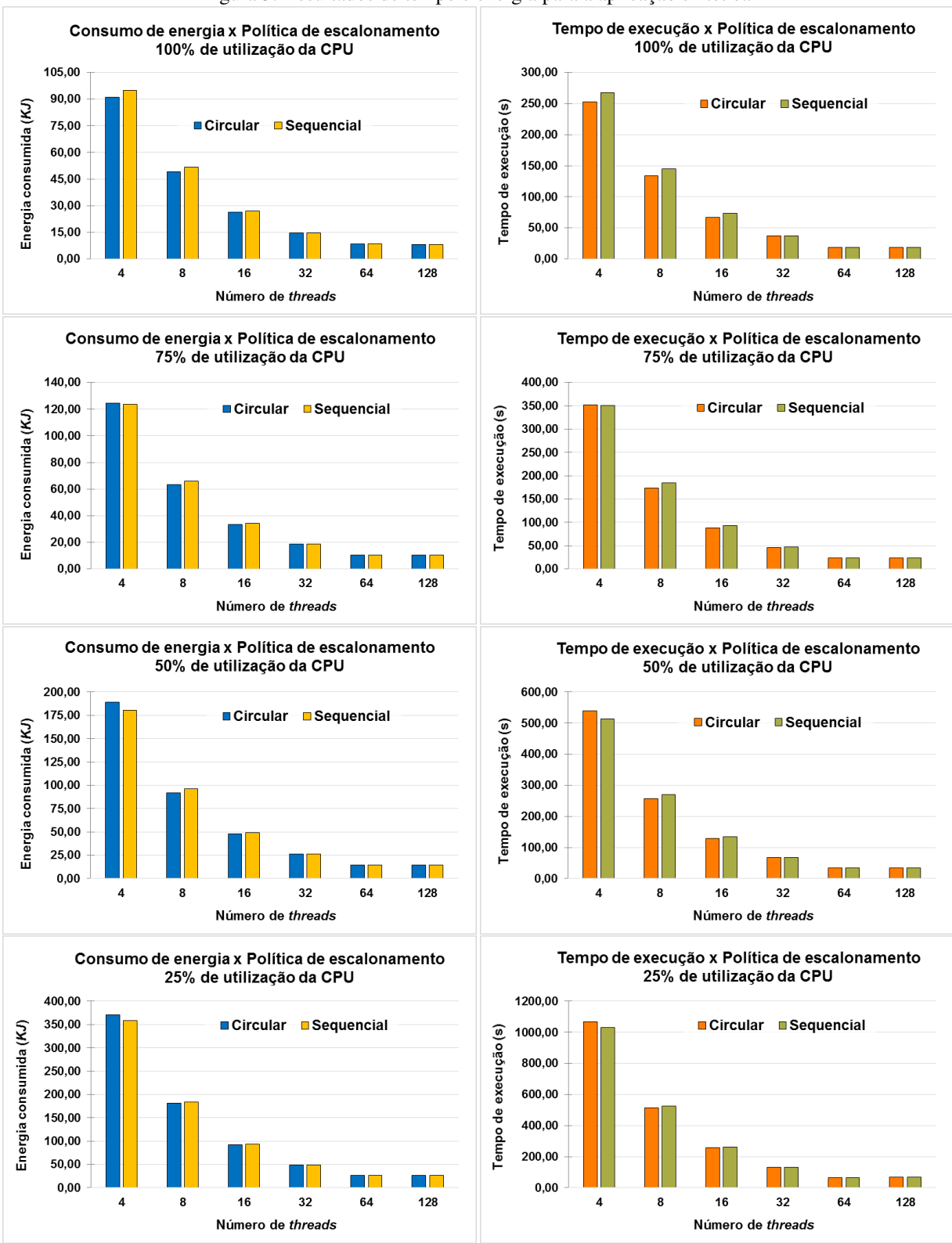
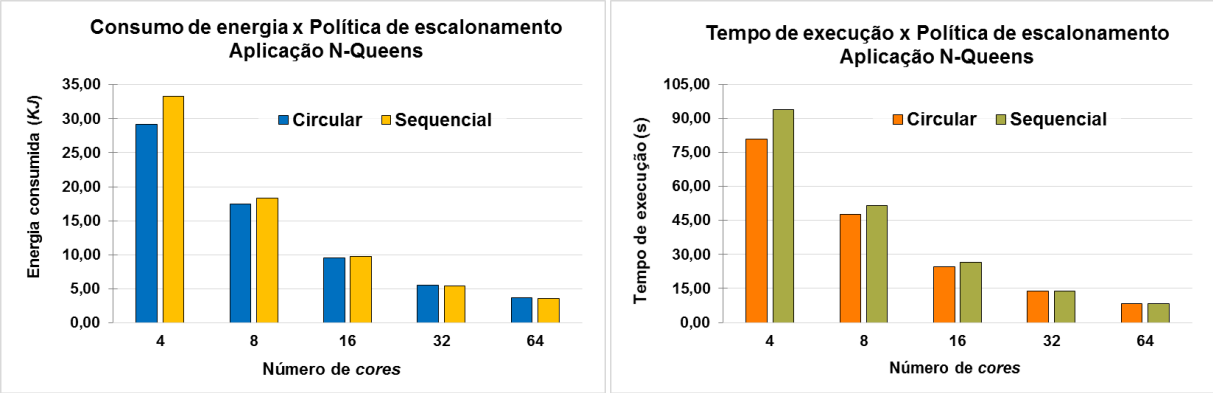


Figura 4: Resultados de tempo e energia para as n-rainhas



6.1 Aplicação sintética

O benchmark usado nesta etapa de análise emprega uma carga sintética que induz um nível de utilização controlado nos processadores da máquina. Níveis de uso de CPU reduzidos são obtidos bloqueando os threads da aplicação por períodos de tempo determinados dinamicamente, de acordo com as estatísticas de uso do sistema. A carga sintética, composta por instruções aritméticas simples, é dividida uniformemente no início da execução entre os threads criados.

Foram monitorados o consumo de energia e o tempo de execução do benchmark variando seus parâmetros: número de threads, que dividem entre si a carga total prevista, a política de escalonamento, circular ou sequencial, e uma taxa de uso, representando um limite artificial de utilização de cada core, fazendo com que a aplicação execute em mais ou menos tempo conforme a taxa de uso informada. As configurações analisadas foram: a variação do número de threads de 4 a 128 em potências de 2, a variação da porcentagem de uso de CPU de 25 a 100 e a política de distribuição (circular ou sequencial). Foram observados os resultados de medição de energia e tempo de 30 execuções, sendo que, os valores médios, estão apresentados na Figura 3.

Para a maioria dos casos, os valores de energia acompanharam os valores do tempo de execução. Reduzindo o número de threads e a taxa de uso da CPU, nota-se um aumento no consumo energético devido à subutilização dos recursos de processamento. Pode-se verificar, a uma significância de 99%, que na arquitetura testada o uso da distribuição circular resultou em uma execução mais rápida e em menor consumo de energia para os casos em que 8 ou 16 threads foram utilizadas. Esse comportamento reflete a proposta das distribuições de carga, já que, como a aplicação é composta por tarefas independentes e intensivas em processamento, a política circular reduz os conflitos de escalonamento e aproveita melhor o hardware subjacente. As diferenças chegaram a 8,20% para o tempo e 5,13% para a energia. Conforme o número de threads aproximou-se do número de *cores* da máquina, a diferença entre as políticas tornou-se menor, já que, para a distribuição sequencial, reduziu-se o efeito de agrupamento da carga em um número pequeno de nós. Para os casos com 64 e 128 threads, a diferença deixou de ser estatisticamente significativa.

Em alguns casos, foi possível observar pequenas diferenças em relação ao comportamento geral. Comparando, por exemplo, a utilização integral de 32 *cores* com a utilização de 64 *cores* trabalhando 50% do tempo, verifica-se que, no primeiro caso, o processador consumiu 1,2% a menos de energia; no segundo, a execução foi 7,15% mais rápida. Nesses casos, pela forma como a carga é dividida, é possível optar entre desempenho superior e menor consumo energético.

6.2 N-rainhas

Para ilustrar a utilização do framework em aplicações irregulares, realizou-se um estudo de caso sobre uma implementação concorrente do algoritmo das n-rainhas. Foi analisado o consumo de energia do processador enquanto o número de *cores* e a política de escalonamento da carga foram variados.

O problema consiste em posicionar n rainhas em um tabuleiro de tamanho $n \times n$ de forma que nenhuma das rainhas seja atacada por nenhuma das outras. Uma solução por força bruta para o problema envolve posicionar uma rainha por vez no tabuleiro, verificando, a cada peça, se a última rainha posicionada inviabiliza a solução do problema.

O algoritmo usado implementa uma busca em profundidade na árvore de pesquisa usando backtracking. Inicialmente, para cada nodo pesquisado, um novo thread é criado para pesquisar um dos ramos da árvore de busca de forma independente. A criação de threads é realizada até que um limite superior de 64 threads, equivalente ao número total de *cores* disponíveis, seja atingido. A partir desse ponto, a execução é serializada. Para avaliar esta aplicação, variou-se apenas o número de *cores* em que as threads poderiam ser mapeadas. O número de rainhas foi mantido em 15. Os resultados de tempo e energia são apresentados na Figura 4.

Foram verificadas, com relação às políticas de distribuição, diferenças maiores de tempo e energia quando comparadas às respostas dos experimentos com o benchmark sintético. Observou-se, também, que as diferenças em termos de tempo e consumo entre as políticas circular e sequencial reduzem conforme aumentamos o número de *cores* utilizados. Esses resultados decorrem do comportamento irregular da aplicação, que dificulta o controle sobre a distribuição da carga. A maior diferença, de 14,33% para a energia e 16,33% para o tempo, foi registrada para o caso em que quatro *cores* foram usados. A um nível de confiança de 99%, não foi observado diferença entre as políticas no que diz respeito a tempo e energia para casos com 32 ou mais *cores*.

7 Conclusão

Este trabalho apresentou a construção de um framework para monitoração do consumo energético do processador em tempo de execução e voltado para aplicações multiprocessadas. Para obtenção dos dados de consumo, foram utilizadas as interfaces APM e RAPL, que estão presentes nas arquiteturas multicore mais recentes. A ferramenta desenvolvida expõe informações de consumo de energia de forma uniformizada e conveniente para o programador final. Por seu design modular e orientado a objetos, o framework desenvolvido é flexível o suficiente para acomodar novas ferramentas e abstrair novas arquiteturas, exigindo, para isso, poucas alterações. Esse design, bem como sua implementação em C++, alinham-se com as tendências de projeto nas ferramentas modernas de programação multithread [26] proporcionando facilidade de integração do framework com aplicações concorrentes desenvolvidas com essas ferramentas.

Foi realizado, para avaliação, o estudo do consumo energético de duas aplicações distintas, buscando ilustrar o uso do framework em diferentes cenários. Com esses estudos de caso, objetivou-se demonstrar a aplicabilidade do framework na avaliação da eficiência energética de aplicações paralelas em sistemas multiprocessados. Os dados de consumo coletados poderiam ser consumidos por aplicações de profiling ou por escalonadores energy-aware, por exemplo.

Visando trabalhos futuros, pretende-se estender as funcionalidades do framework para dar suporte à ferramenta IPMI, já que essa interface possibilita a coleta de dados de forma análoga às interfaces aqui utilizadas. Avalia-se, ainda, a inclusão de uma funcionalidade para gerenciar a frequência do processador, objetivando dar suporte a ferramentas que implementem estratégias de escalonamento energy-aware. Por fim, considera-se estender o estudo do consumo energético de forma a contemplar aplicações intensivas em memória.

Agradecimentos

O presente trabalho foi realizado com apoio do Programa Nacional de Cooperação Acadêmica da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES/Brasil.

Referências

[1] CHU, S.-L.; CHEN, S.-R.; WENG, S.-F. CPPM: a comprehensive power-aware processor manager for a multicore system. *Applied Mathematics & Information Sciences*, Natural Sciences Publishing, v. 7, p. 793–800, mar. 2013.

[2] PALLIPADI, V.; STARIKOVSKIY, A. The Ondemand Governor: past, present and future. In: *Proceedings of Linux Symposium, vol. 2, pp. 223-238*. Ottawa: USENIX, 2006.

[3] LIBUTTI, S. et al. Exploiting performance counters for energy efficient co-scheduling of mixed workloads on multi-core platforms. In: *Proceedings of Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms*. New York: ACM, 2014. (PARMA-DITAM '14), p. 27:27–27:32.

[4] PETRUCCI, V. et al. Energy-efficient thread assignment optimization for heterogeneous multicore systems. *ACM Trans. Embed. Comput. Syst.*, ACM, New York, v. 14, n. 1, p. 15:1–15:26, jan. 2015.

[5] FEITELSON, D.; RUDOLPH, L.; SCHWIEGELSHOHN, U. Parallel job scheduling – a status report. In: *Job Scheduling Strategies for Parallel Processing*. Berlin: Springer, 2005, (Lecture Notes in Computer Science, v. 3277). p. 1–16.

[6] YANG, X. et al. Integrating dynamic pricing of electricity into energy aware scheduling for HPC systems. In: *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. New York: ACM, 2013. (SC '13), p. 60:1–60:11.

[7] MORARI, A.; VALERO, M. HPC system software for regular and irregular parallel applications. *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*, IEEE Computer Society, Los Alamitos, v. 0, p. 2242–2245, 2013.

[8] GAUTIER, T.; ROCH, J.; VILLARD, G. Regular versus irregular problems and algorithms. In: *In Proc. of IRREGULAR'95*. Berlim: Springer, 1995. p. 1–25.

[9] SCHUBERT, S. et al. Profiling software for energy consumption. In: *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*. Besançon: IEEE, 2012. p. 515–522.

[10] ARAUJO, A. S. *Anahy-3: Um novo Ambiente de Execução Otimizado para Arquiteturas Multicore*. Dissertação (Trabalho de Conclusão de Curso) — Universidade Federal de Pelotas, 2013.

[11] JOSEPH, R.; MARTONOSI, M. Run-time power estimation in high performance microprocessors. In: *Low Power Electronics and Design, International Symposium on, 2001.* Huntington Beach: ACM, 2001. p. 135–140.

[12] GOEL, B. *Per-core Power Estimation and Power Aware Scheduling Strategies for CMPs*. Dissertação (Mestrado) — Institutionen för data- och informationsteknik, Datorteknik (Chalmers), Chalmers tekniska högskola,, 2011. 70.

[13] MCCULLOUGH, J. C. et al. Evaluating the effectiveness of model-based power characterization. In: *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference*. Berkeley: USENIX Association, 2011. (USENIXATC'11).

[14] AMD. *AMD Family 15h Processor BIOS and Kernel Developer Guide*. [S.l.], 2013. Rev 3.14.

[15] Intel. *Intel 64 and IA-32 Architectures Software Developer's Manual*. [S.l.], 2013.

[16] Intel et al. *Intelligent Platform Management Interface Specification Second Generation v2.0*. Document revision 1.1. [S.l.], 2013.

[17] MUCCI, P. J. et al. PAPI: A portable interface to hardware performance counters. In: *In Proceedings of the Department of Defense HPCMP Users Group Conference*. Monterey: High Performance Computing Modernization Program Office, 1999. p. 7–10.

[18] WEAVER, V. et al. Measuring energy and power with PAPI. In: *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*. Pittsburgh: CPS, 2012. p. 262–268.

[19] BELLOSA, F. The benefits of event-driven energy accounting in power-sensitive systems. In: *In Proceedings of the 9th ACM SIGOPS European Workshop*. Kolding: ACM, 2000.

[20] NOUREDDINE, A. et al. Runtime monitoring of software energy hotspots. In: *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. New York: ACM, 2012. (ASE 2012), p. 160–169.

[21] FLINN, J.; SATYANARAYANAN, M. Powerscope: a tool for profiling the energy usage of mobile applications. In: *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*. New Orleans: IEEE, 1999. p. 2–10.

[22] HÄHNEL, M. et al. Measuring energy consumption for short code paths using RAPL. *SIGMETRICS Perform. Eval. Rev.*, ACM, New York, v. 40, n. 3, p. 13–17, jan. 2012.

[23] ROTEM, E. et al. Power-management architecture of the Intel microarchitecture code-named Sandy Bridge. *Micro, IEEE*, v. 32, n. 2, p. 20–27, March 2012.

[24] PINTO, B. G.; XAVIER, L. M. S.; CAVALHEIRO, G. G. H. Análise comparativa entre ferramentas para medição de energia em aplicações paralelas CPU-bound. In: *Anais da XIV Escola Regional de Alto Desempenho do RS*. Alegrete: SBC, 2014. p. 141–144.

[25] BROQUEDIS, F. et al. hwloc: A generic framework for managing hardware affinities in HPC applications. In: *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*. Pisa: IEEE, 2010. p. 180–186.

[26] CAVALHEIRO, G. G. H.; DU BOIS, A. R. Ferramentas modernas para programação multithread. In: SALGADO, A. C. et al. (Ed.). *Jornadas de Atualização em Informática*. Porto Alegre: Sociedade Brasileira de Computação, 2014. p. 41–83.