Multicores: Uma visão geral e novos desafios

Júlio Reis, RA: 044415
Instituto de Computação
Universidade Estadual de Campinas

juliocesardosreis@gmail.com

RESUMO

A fim de manter o desempenho exponencial dos processadores ao longo dos anos, as empresas encontraram-se incapazes de continuar a melhorar efetivamente o desempenho dos microprocessadores da maneira típica — por encolhimento dos transistores e mais encapsulamento destes em *chips* com um único núcleo. Esta abordagem mostrou-se inviável, pois gera muito calor e usa muita energia. Em resposta a isso, desenvolveram aumentando no desempenho através da construção de *chips* com múltiplos núcleos (multicore). Esta nova arquitetura de processadores é projetada para melhorar o desempenho computacional e minimizar o calor (aumentando a dissipação de energia com uma solução de gerenciamento térmico) através da integração de dois ou mais núcleos de processadores em um único *chip*. Estes microprocessadores com múltiplos núcleos são uma importante inovação na evolução dos processadores e esta tecnologia poderá revolucionar o modo de se desenvolver aplicações computacionais. Assim, o objetivo deste trabalho é desenvolver uma visão geral desta nova tecnologia, mostrando suas características, potencialidades, mas também os desafios introduzidos por ela que ainda não foram resolvidos de maneira adequada na literatura.

Palavras-Chaves

Multicores; microprocessadores; Paralelismo.

1. INTRODUÇÃO

Até alguns anos atrás, a comunidade de hardware (processador) traduziu a Lei sobre densidade de transistores de Moore diretamente em ganho de desempenho em um único processador como resultado do aumento de freqüência nestes. Ultimamente, essa tradução tem sido prejudicada pelos efeitos de freqüência em consumo de energia e geração de calor. A nova realidade é que o desempenho por processador é essencialmente estático, e o aumento no desempenho é dado por um aumento do número de núcleos de processador em um *chip* [1]. Para Gorder [2] o que impulsionou a indústria para a tecnologia de múltiplos núcleos não foi apenas a necessidade de uma maior velocidade de processamento, mas a necessidade de menos calor e mais eficiência em energia. Menos calor, porque os *chips* mais rápidos aquecem mais rapidamente do que os *coolers* podem arrefecer-los, e mais eficiência em energia, pois *chips* de apenas um núcleo gastam muita energia para concluir o trabalho.

De acordo com Agarwal *et al.* [3], a tecnologia de múltiplos núcleos refere-se a um único *chip* com vários "motores de processamento" visivelmente distintos, cada um com comando independente. Esta tecnologia melhora o desempenho através de diversas partes de uma aplicação em paralelo e *chips* com apenas um núcleo, por outro lado, realizam tarefas de maneira serial [4]. Processadores com múltiplos núcleos representam uma grande evolução na tecnologia da informação. Este importante desenvolvimento está chegando em um momento em que as empresas e os consumidores estão começando a exigir os benefícios oferecidos por estes processadores, devido ao crescimento exponencial de dados digitais e da globalização da Internet. Estes processadores acabarão por se tornar o modelo de computação pervasiva, pois eles oferecem o desempenho e vantagens em produtividade que vão além das capacidades dos processadores de apenas um núcleo [5].

Os processadores com múltiplos núcleos também irão desempenhar um papel central na condução de importantes avanços em segurança para os computadores pessoais (PCs) e tecnologias de virtualização, que estão sendo desenvolvidas para proporcionar uma maior proteção, utilização dos recursos, e valor para o mercado comercial da computação [5]. Consumidores gerais também terão acesso a um melhor desempenho, o que irá expandir significativamente a utilidade dos seus PCs domésticos e sistemas computacionais na mídia digital. Estes processadores terão o benefício de oferecer desempenho sem ter de aumentar os requisitos de potência, que se traduz em maior desempenho por *watt*. Colocar dois ou mais núcleos em um único processador abre um mundo de novas possibilidades importantes. A próxima geração de aplicações de *software* provavelmente será desenvolvida utilizando o recurso destes processadores por causa do desempenho e eficiência que podem proporcionar, em comparação com os processadores com um único núcleo. Seja para estas aplicações ajudarem as empresas profissionais de animação a produzirem filmes mais realistas de maneira mais rápida por menos dinheiro, ou para criar maneiras de tornar o PC mais natural e intuitivo aos usuários, a disponibilização generalizada de processadores com múltiplos núcleos mudará o universo da computação para sempre [5].

Neste cenário, o objetivo deste trabalho é desenvolver uma visão geral da tecnologia de múltiplos núcleos, elucidando desde aspectos de sua evolução, os tecnológicos, de performance e também os desafios desta nova tecnologia. Para isso o trabalho está estruturado da seguinte maneira: A seção 2 apresenta um breve relato sobre a evolução dos processadores e o surgimento da tecnologia de múltiplos núcleos; a seção 3 objetiva apresentar as principais considerações e algumas técnicas sobre esta tecnologia; a seção 4 visa observar questões sobre desempenho de arquiteturas com diversos núcleos; na seção 5 é apresentado os

desafios e questões em aberto sobre esta tecnologia e por fim na seção 6 o trabalho é concluído.

2. EVOLUÇÃO DOS PROCESSADORES E A TECNOLOGIA MULTICORE

A Intel fabricou o primeiro microprocessador, o 4-bit 4004, no início dos anos 1970. Pouco tempo depois eles desenvolveram o 8008 e 8080, ambos de 8 bits, e a Motorola seguiu o exemplo com os seus 6800 que foi equivalente ao da Intel 8080. As empresas, em seguida, fabricaram os microprocessadores de 16-bits. Motorola teve seu 68000 e a Intel o 8086 e 8088. Esta série seria a base para o Intel 80386 de 32-bit e mais tarde a sua popular linha Pentium que estava no primeiro PC para os consumidores. Cada geração de processadores crescia menor, mais rápido, dissipando mais calor e consumindo mais energia [6]. Desde a introdução do Intel 8086 e através do Pentium 4 um aumento no desempenho, a partir de uma geração para a seguinte, foi visto como um aumento na freqüência do processador. Por exemplo, o Pentium 4 variou de velocidade (freqüência) de 1,3 a 3,8 GHz em 8 anos. Enquanto o tamanho físico do *chip* diminuiu, o número de transistores por *chip* aumentou junto com a velocidade, impulsionando assim o aumento de dissipação de calor [7].
Historicamente, os fabricantes de processador têm respondido à demanda por maior poder de processamento, principalmente através da disponibilização de processadores mais rápidos. No entanto, o desafio do gerenciamento de energia e refrigeração para os processadores poderosos de hoje levou a uma reavaliação desta abordagem. Com o calor aumentando progressivamente mais rápido que a velocidade em que os sinais deslocam-se pelo processador, conhecido como velocidade de *clock*, um aumento no desempenho pode criar um aumento ainda maior no calor [8].

Esta discussão tem inicio em 1965, quando Gordon Moore observou que o número de transistores disponíveis nos semicondutores dobraria num período de 18 a 24 meses. Agora conhecida como a lei de Moore, esta observação tem orientado *designers* de computador nos últimos 40 anos. A métrica mais comumente utilizada na medição do desempenho da computação é o CPU (freqüência de *clock*) e ao longo dos últimos 40 anos esta tendeu a seguir a lei de Moore. Segundo Akhter e Roberts [9], embora a melhoria linear do fluxo de instruções e da velocidade de *clock* são metas que se valem lutar, arquitetos de computador podem tirar proveito desta melhoria em maneiras menos óbvias. Por exemplo, em um esforço para tornar mais eficiente o uso dos recursos do processador, os arquitetos têm utilizado técnicas de paralelismo em nível de instrução. Paralelismo em nível de instrução (ILP) dá a capacidade de reorganizar as instruções em uma ótima maneira para eliminar os *stalls* do *pipeline*. Para que esta técnica seja eficaz, múltiplas e independentes instruções devem ser executadas. No caso da execução em ordem, dependências entre instruções podem limitar o número de instruções disponíveis para a execução, reduzindo a quantidade de execução em paralelo que pode ocorrer. Uma abordagem alternativa que tenta manter as unidades de processamento de execução ocupada é a reordenação das instruções para que instruções independentes possam executar simultaneamente. Neste caso, as instruções são executadas fora da ordem do programa original.

Com a evolução do *software*, aplicações tornaram-se cada vez mais capazes de executar várias tarefas em simultâneo. A fim de apoiar o paralelismo, várias abordagens, tanto em *software* e *hardware* tem sido adotadas. Uma abordagem para enfrentar cada vez mais a natureza concorrente dos *softwares* modernos envolve usar uma preemptiva, ou fatia de tempo, e um sistema operacional multitarefa. Processamento com fatia de tempo permite desenvolvedores esconder latências associadas com I/O trocando a execução de múltiplos processos. Este modelo não permite a execução paralela. Apenas um fluxo de instrução pode ser executado em um processador em um único ponto no tempo [9].

Ainda de acordo com Akhter e Roberts [9], outra abordagem para resolver o paralelismo de processos é aumentar o número de processadores físicos no computador. Sistemas multiprocessadores permitem verdadeira execução em paralelo; múltiplos processos são executados simultaneamente em vários processadores. A perda de vantagem neste caso é o aumento do custo global do sistema. O próximo passo lógico para o processamento de processos simultâneos é o processador com múltiplos núcleos. Em vez de apenas reutilizar recursos dos processadores em um processador com um único núcleo, fabricantes de processadores aproveitaram melhorias na fabricação da tecnologia para implementar dois ou mais "núcleos de execução" independentes dentro de um único processador. Estes núcleos são essencialmente dois processadores em um único *chip*. Estes núcleos têm o seu próprio conjunto de execução e seus recursos arquitetônicos (veja a seção 3).

Um desempenho global de processamento excelente pode ser conseguido através da redução da velocidade de *clock*, enquanto se aumenta o número de núcleos de processamento e, conseqüentemente, a redução da velocidade pode conduzir a uma baixa produção de calor e maior eficiência do sistema. Por exemplo, ao passar de um único núcleo de alta velocidade, que gera um aumento correspondente no calor, para múltiplos núcleos mais lentos, que produzem uma redução correspondente no calor, pode-se melhorar o desempenho das aplicações, reduzindo simultaneamente o calor. Uma combinação que prevê dois ou mais núcleos com mais baixa velocidade poderia ser concebido para proporcionar excelente desempenho, reduzindo ao mínimo o consumo de energia e proporcionando menor produção de calor do que configurações que dependem de um único *clock* de alta velocidade [8]. Aumentar a velocidade de *clock* causa mudanças mais rápidas entre transistores e, assim, geram mais calor e consomem mais energia.

Segundo Geer [10], quando um *chip* com um único núcleo executa vários programas, ele atribui uma fatia de tempo para trabalhar em um programa e, em seguida, atribui diferentes fatias de tempo para outros. Isto pode causar conflitos, erros ou baixo desempenho quando o processador tem de efetuar múltiplas tarefas em simultâneo. Se você tiver várias tarefas que deverão executar todas ao mesmo tempo, poderemos notar que haverá uma melhora significativa em desempenho com múltiplos núcleos. Por exemplo, os *chips* poderiam usar um núcleo distinto para cada tarefa. Devido aos núcleos estarem nos mesmos *chips*, eles podem compartilhar componentes arquitetônicos, tais como elementos de memória e gerenciamento de memória. Assim, eles têm menos componentes e sistemas de custos mais baixos do que executar em múltiplos processadores distintos em *chips* distintos. Além disso, a sinalização entre os núcleos pode ser mais rápida e consumir menos eletricidade do que em uma abordagem em separado.

Acelerar a freqüência do processador executou o seu curso na primeira parte desta década; arquitetos de computador precisavam de uma nova abordagem para melhorar o desempenho. Adicionando um núcleo de processamento adicional no mesmo *chip* seria, em teoria, conduzir a duas vezes o desempenho e dissiparia menos calor, embora, na prática, a velocidade real de cada núcleo é mais lento do que o mais rápido processador de núcleo único. Em Setembro de 2005, constatou-se que o consumo de energia aumenta em 60% com cada aumento de 400MHz de velocidade, mas a abordagem de dois núcleos significa que você pode receber uma melhora significativa no desempenho, sem a necessidade de execução em baixas taxas de *clock* [11].

Se os núcleos utilizarem o mesmo número de transistores, mais núcleos poderão ser adicionados em um *chip* conforme o número de transistores disponíveis aumenta. De fato, usando um corolário da Lei de Moore, podemos dizer que o número de núcleos em um *chip* irá duplicar a cada 18 ou 24 meses. Dado que o *dual* e *quad* núcleos de liderança comercial já estão em uso generalizado hoje, e protótipos de investigação de 16 núcleos nas universidades provavelmente estarão disponíveis, é altamente provável que veremos processadores com 1000 núcleos na primeira parte da próxima década [3].

Mais importante do que isso é que processadores com múltiplos núcleos oferecem uma tecnologia imediata e eficaz para resolver os desafios de projeto de processadores de hoje, aliviando os subprodutos de calor e consumo de energia que existem quando continuamente se avança com processadores de maior freqüência e com apenas um núcleo. Ela ajudará a romper com as limitações de desempenho dos processadores de apenas um núcleo e fornecer capacidade de desempenho para enfrentar os *softwares* mais avançados de amanhã [5]. Isto ocorrerá, pois, estes processadores têm o potencial de executar aplicações de forma mais eficiente do que processadores de um núcleo único, fornecendo aos usuários a capacidade de continuar trabalhando mesmo durante a execução das tarefas que mais demandam processamento em segundo plano, como pesquisar uma base de dados, renderizar uma imagem 3D, mineração de dados, análise matemática e servidores *Web.*

A *Advanced Micro Devices* (AMD) [5] diz que a crescente complexidade dos *softwares*, bem como o desejo dos usuários para executar várias aplicações ao mesmo tempo, irá acelerar a adoção generalizada de sistemas baseados em processadores com múltiplos núcleos. Isso dará a aplicações comerciais a capacidade de lidar com grandes quantidades de dados e com mais usuários de forma mais rápida e eficiente. Os consumidores irão experiênciar mais funcionalidades e funcionalidades mais ricas, especialmente para aplicações como mídia digital e criação de conteúdos digitais.

Por fim, Merritt [12] argumenta que a tecnologia de múltiplos núcleos não é um novo conceito, uma vez que a idéia tem sido utilizada em sistemas embarcados para aplicações especiais, mas, recentemente, a tecnologia tornou-se integrar com as empresas Intel e AMD que tem disponibilizado comercialmente muitos *chips* com múltiplos núcleos. Em contraste com as máquinas de dois e quatro núcleos disponíveis comercialmente a partir de 2008, alguns especialistas acreditam que até 2017 processadores embutidos poderiam dispor de 4.096 núcleos, servidores poderão ter 512 núcleos e *chips* de *desktop* poderiam utilizar 128 núcleos. Esta taxa de crescimento é surpreendente considerando que os *chips* dos *desktops* atualmente estão no linear de quatro núcleos e um único núcleo tem sido utilizado ao longo dos últimos 30 anos [6]. Na próxima seção será descrito melhor sobre a arquitetura destes processadores.

3. ARQUITETURA MULTICORE: PRINCIPAIS CONSIDERAÇÕES TECNOLÓGICAS

Embora os fabricantes de processadores possam diferir um do outro em relação às arquiteturas dos microprocessadores, veja com detalhes em [6] diversos aspectos relacionados às diferenças entre as implementações específicas de arquiteturas de múltiplos núcleos de processadores da Intel, AMD e outros fabricantes. No entanto, as arquiteturas de múltiplos núcleos precisam aderir a determinados aspectos básicos. A configuração de um microprocessador pode ser visto na Figura 1.

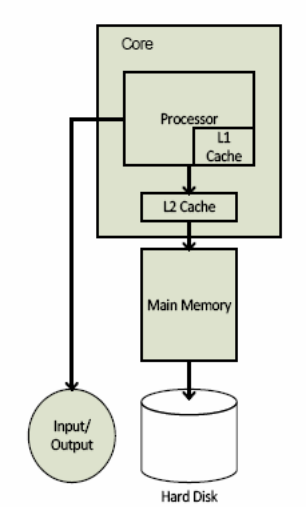

Figura 1. Configuração de um Processador Moderno Genérico [6]

As arquiteturas de computadores podem variar sobre diversos aspectos (Veja Figura 2). Em específico nas arquiteturas de múltiplos núcleos, além do modelo de memória compartilhada e modelo de memória distribuída que será discutido a seguir (Veja Figura 3), eles podem variar quanto ao modelo de cachê também. Assim, dependendo da arquitetura, estes processadores podem ou não compartilhar uma *cache*. As diferentes arquiteturas de processadores estão destacadas na Figura 2.

Em relação à Figura 1, mais próximo do processador está a *cache* de nível 1 (L1), que é uma memória muito rápida que é utilizada para guardar dados freqüentemente utilizados pelo processador. A *cache* de Nível 2 (L2) esta fora do *chip*, é mais lenta do que a *cache* L1, mas ainda muito mais rápida que a memória principal. A *cache* L2 é maior que a *cache* L1 e é utilizada para a mesma finalidade. A Memória principal é muito grande e mais lenta do que a *cache*, e é utilizado, por exemplo, para armazenar um arquivo atualmente sendo editado em um editor de texto. A maioria dos sistemas tem entre 1GB para 4GB de memória principal, em comparação com cerca de 32KB de L1 e 2MB de *cache* L2. Finalmente, quando os dados não estiverem localizados na *cachê* ou na memória principal, o sistema deve recuperá-lo a partir do disco rígido, o que leva exponencialmente mais tempo do que a partir da leitura da memória principal.

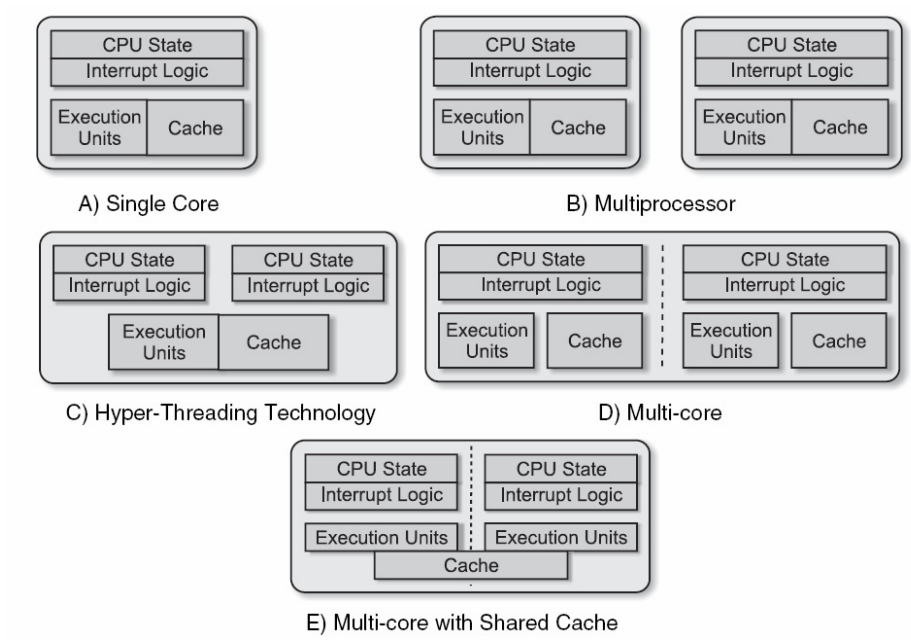

Figure 2: Comparação entre arquiteturas de um único núcleo, com múltiplos processadores e múltiplos núcleos [9]

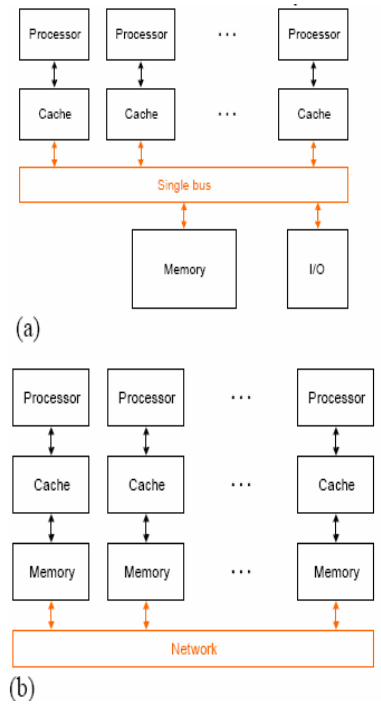

Figura 3: Modelo de memória compartilhada e modelo de memória distribuída [6]

Se estabelecermos dois núcleos lado a lado, pode-se ver que um método de comunicação entre os núcleos, bem como para a memória principal será necessário. Isso é normalmente realizado usando um único barramento de comunicação ou uma rede de interconexão. O método por barramento é usado com um modelo de memória compartilhada, enquanto que a abordagem de interconexão de rede é utilizada com um modelo de memória distribuída. Após cerca de 32, núcleos o barramento estará sobrecarregado com a quantidade de processamento, comunicação e concorrência, o que leva a diminuição do desempenho; portanto, uma comunicação pelo barramento tem uma capacidade limitada de expansão. Observe a Figura 3, que ilustra os dois tipos de comunicação possíveis [6].

4. O DESEMPENHO DOS MULTICORE

Segundo Geer [10], de acordo com o professor assistente Steven Keckler da Universidade do Texas, um *chip* com dois núcleos executando várias aplicações é cerca de 1,5 vezes mais rápido comparável com um *chip* com apenas um núcleo. Os *chips* de múltiplos núcleos não necessariamente rodam tão rápido como o melhor desempenho dos modelos com apenas um núcleo, mas melhoraram o desempenho global através do tratamento de mais trabalhos em paralelo, conforme ilustra a Figura 4. *Chips* de múltiplos núcleos são a maior mudança no modelo de programação desde que a Intel introduziu a arquitetura 386 de 32-bit. Além disso, estes processadores são uma maneira de estender a lei de Moore.

A melhor maneira de apreciar os benefícios obtidos por uma implementação com múltiplos núcleos sobre a abordagem de apenas um único núcleo é comparar a forma como os transistores seriam utilizados. Por exemplo, suponha que há um único núcleo de tecnologia de 90-nm que ocupa uma área A do *chip* e que as porções do processador e a memória *cache* ocupam cada uma a metade da superfície do *chip*. Vamos chamar isto de caso base. Com a tecnologia de 65-nm, os arquitetos têm o dobro do número de transistores para a mesma área A. Para tirar proveito do dobro de transistores, arquitetos podem manter a mesma arquitetura do CPU e triplicar o tamanho da *cache* (Caso 1). Alternativamente, os arquitetos podem dobrar o número de núcleos e manter o mesmo tamanho da *cache* para cada núcleo (Caso 2).

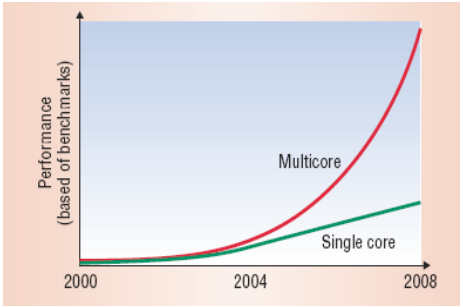

Figura 4. *Chips* com múltiplos núcleos têm um melhor desempenho [10]

Podemos aplicar um modelo simplificado para demonstrar as implicações de desempenho das duas alternativas. Suponhamos que a taxa de perda de trabalho para o cenário base é de 1% e a

latência da memória principal para uma *cache miss* é de 100 ciclos. Também assumimos que cada uma das instruções idealmente executa em um ciclo. Começando com o cenário base, as instruções por *clock* (IPC) será 0,5 (IPC pode ser calculado como 1 / (1 + 0,01 × 100) = 0.5). Supondo que a carga de trabalho tem grande paralelismo, o IPC dobra, aplicando o caso 2 (2 × 1 / (1 + 0,01 × 100) = 1), uma vez que a taxa de execução de instruções é o dobro do cenário base. Por outro lado, assumindo que a latência de memória não mudou, o IPC do Caso 1 será 0,62, ou 1 / (1 + 0,006 × 100), que mostra um desempenho mais baixo do que a alternativa de múltiplos núcleos (Caso 2). Este cálculo é obtido a partir de uma regra comumente utilizada para taxas de *cache miss* que sugere que a taxa de *miss* segue a regra de raiz quadrada. Assim, a taxa de *cache miss* no Caso 1 será de 1%/*sqrt*(3) = 0,6%. O Caso 2 também contribui para demonstrar que *caches* oferecerem retornos decrescentes.

Se o argumento de múltiplos núcleos é válido, devemos ser capazes de reduzir o tamanho da *cache* e colocar mais núcleos no mesmo *chip* para um melhor desempenho. Vamos tentar três núcleos, cada um com um terço do tamanho da *cache* do cenário de base. A taxa de *cache miss* será de 1%×*sqrt* (3) = 1,7%, e o IPC aumenta ainda mais, para 1,1, ou 3 × 1 / (1 + 0,017 × 100). No entanto, utilizando o mesmo tamanho de *chip*, a tendência é continuar? Com quatro núcleos, há pouco espaço para *cache*, mas presume-se que se pode espremer em uma *cache* do que um décimo sexto do tamanho do cenário de base para cada núcleo. A taxa de *miss* será de 1%×*sqrt* (16) = 4%, e o IPC cai para 0,8, ou 4 × 1 / (1 + 0,04 × 100). Observe que, neste exemplo, o Caso 3 tem o número ideal de cores e tamanho de cache. Isto demonstra que o balanço de recursos em um núcleo é de grande importância em arquiteturas de múltiplos núcleos [13].

Uma profunda e bem desenvolvida análise de medida de desempenho dos processadores de múltiplos núcleos foi desenvolvida por Muthana *et al.* [14], na qual diversas considerações são feitas em diversos aspectos. Foi observado que o desempenho de um processador com múltiplos núcleos é muito superior à de um processador com apenas um único núcleo, e uma taxa de banda de 1 *terabyte/s* é possível com uma considerável economia de energia. Capacitores embutidos com uma densidade de capacitância de *1uF/cm2* podem ser fabricados e a combinação desses capacitores em paralelo podem proporcionar dissociação para futuros processadores. Já sobre a dissipação de calor, comparado com vários *chips* com apenas um único núcleo, os *chips* com múltiplos núcleos são mais fáceis de resfriar, pois estes processadores são mais simples e utilizam menos transistores. Isto significa que eles usam menos energia e dissipam mais calor global [2].

Contudo, devido à limitada largura de banda e esquema de gerenciamento de memória que são mal adaptados a supercomputadores, o desempenho dessas máquinas estaria um nível abaixo com mais núcleos podendo até diminuir. Embora o número de núcleos por processador esteja aumentando, o número de ligações a partir do *chip* para o resto do computador não está. Então, devido a isso, é um problema manter todos os núcleos alimentados com dados a todo tempo. A chave para resolver esse gargalo pode ser uma melhor integração entre memória e processador [15]. Observe a Figura 6 que ilustra este cenário.

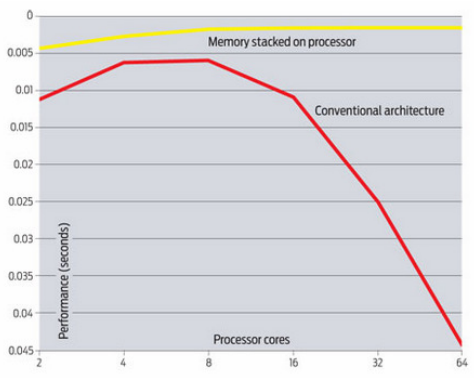

Figura 6. Mais núcleos por *chip* deixará alguns programas mais lentos ao menos que exista um grande aumento na taxa de memória [15]

Segundo Agarwal *et al.* [3], nos dias de *design* de apenas um único núcleo, arquitetos aumentariam o tamanho dos recursos do núcleo (por exemplo, tamanho da *cache*), conforme ocorreu com os transistores, que aumentaram em cada sucessiva geração da tecnologia. Arquiteturas com múltiplos núcleos proporcionaram uma maior escolha de inclusão de outros núcleos. Desta forma, tendo mais espaço, aumentando o número de núcleos e mantendo o tamanho dos recursos relativamente pequenos, poderar-se resultar em mais desempenho do que aumentar o tamanho dos recursos e manter o número de núcleos constante. Esta nova dimensão para melhorar o desempenho e eficiência em energia em múltiplos núcleos nos obriga a repensar na arquitetura do processador e isto é capturado por um princípio simples chamada de "*Kill Rule*". Este princípio afirma que o tamanho de um recurso deve ser aumentado somente se, para cada 1% de aumento na área central, exista pelo menos um aumento de 1% no desempenho do núcleo.

Já sobre como medir o desempenho destes processadores, em Gal-On e Levy [15] é feito uma discussão sobre como desenvolver a medição do desempenho dos processadores com múltiplos núcleos a partir de diversos tipos de *benchmarks* e de distintos critérios, tais como banda de memória e escalabilidade. Os autores explicitam que estes benchmarks poderiam ser baseados em arquiteturas de memória distribuída ou compartilhada. Ainda, eles poderiam consistir em núcleos homogêneos ou heterogêneos e poderiam empregar uma variedade de tecnologias interligadas. Podendo a tecnologias de múltiplos núcleos terem abordagens muito diferenciadas, então, *benchmarks* sobre elas precisam ser altamente diferenciados também.

Por fim, mesmo com desempenhos apurados e sendo esta uma arquitetura sofisticada, ainda há diversos desafios e questões em aberto nas arquiteturas de múltiplos núcleos, tópico que será mais bem discutido na próxima seção.

5. DESAFIOS E QUESTÕES EM ABERTO

Embora a tecnologia de múltiplos núcleos ofereça oportunidades para melhorias no desempenho de processamento e de eficiência em energia, ela também traz muitos novos desafios de programação e de *design*, devido a indústria de processadores avançar rumo a eficiência em energia, desempenho e programabilidade. Curiosamente, a eficiência e o desempenho não são apenas as maiores oportunidades oferecidas pela tecnologia de múltiplos núcleos, mas também, o mais importante desafio que se

tem refere-se ao número de núcleos que vão além dos projetos de único dígito de hoje. Outro desafio envolve o modelo de programação necessário para a utilização eficiente de múltiplos núcleos em um *chip* [13] (veja a subseção 5.5). Além disso, ter múltiplos núcleos em um único *chip* dá origem a alguns problemas como: gestão de energia e temperatura, que são duas preocupações que podem aumentar exponencialmente com a adição de múltiplos núcleos; coerência em memória cache, além do uso de processador com múltiplos núcleos no seu pleno potencial que é uma outra questão [6]. Observe a seguir alguns desafios e problemas relacionados a esta tecnologia.

5.1 Energia e Temperatura

O calor sofre a influência de um conjunto de fatores, dos quais dois são: densidade do processador e velocidade de *clock*. Outros controladores incluem tamanho da *cache* e tamanho do núcleo em si. Em arquiteturas tradicionais, o calor gerado a cada nova geração de processadores tem crescido a uma taxa maior do que a velocidade. Em contraste, usando uma cache compartilhada (em vez de separar *caches* dedicadas para cada núcleo do processador) e processadores de baixa velocidade, processadores de múltiplos núcleos podem ajudar os administradores a minimizar o calor, mantendo um elevado desempenho global [8].

Segundo Schauer [6], se dois núcleos forem colocados em um único *chip* sem qualquer modificação o *chip,* em teoria, consumiria duas vezes mais energia e geraria uma grande quantidade de calor. No caso extremo, se um processador sobre-aquecesse o computador poderia até queimar. Levando isso em conta, para cada projeto, os múltiplos núcleos são executados a uma menor freqüência visando reduzir o consumo de energia. Para combater o consumo de energia desnecessário, muitos projetos também incorporaram uma unidade de controle que tem o poder de parar núcleos não utilizados ou limitar a quantidade de energia. Ao alimentar núcleos não utilizados e usar "*clock gating"* a quantidade de perda no *chip* é reduzido. Para diminuir o calor gerado por múltiplos núcleos em um único *chip*, este é arquitetado para que o número de pontos quentes não cresça e não se torne muito grande, de modo que o calor seja espalhado por todo o *chip*.

5.2 Coerência de *Cache*

Coerência de *cache* é uma preocupação em um ambiente de múltiplos núcleos devido as *caches* L1 e L2 distribuídas. Uma vez que cada núcleo tem sua própria *cache*, a cópia dos dados naquela *cache* pode não ser sempre a versão mais atualizada. Por exemplo, imagine um processador com duplo núcleo no qual cada núcleo trouxe um bloco de memória em sua *cache* privada. Um núcleo escreve um valor para um local específico e, quando o segundo núcleo tenta ler o valor da sua *cache,* não vai ter a cópia atualizada, a menos que a sua entrada na *cache* seja invalidada e um *cache miss* ocorra. Este *cache miss* força a entrada da *cache* do segundo núcleo ser atualizada. Se esta política de coerência não estava correta, dados inválidos poderiam ser lidos e resultados inválidos seriam produzidos, podendo deixar errado o resultado final do programa errado. Em geral, existem dois regimes de coerência de *cache:* o protocolo *snooping* e um protocolo baseado em diretório. O protocolo *snooping* só funciona com um sistema baseado em barramento, usa um número de estados para determinar se precisa atualizar as entradas da *cache* e tem controle sobre a escrita ao bloco. O protocolo baseado em diretório pode ser usado em uma rede arbitrária e é, portanto, escalável para vários processadores, ou núcleos, em contraste com o *snooping* que não é escalável. Neste esquema é utilizado um diretório que detém informações sobre a localização de memória que estão sendo compartilhadas em múltiplas *caches* e que são utilizadas exclusivamente por um núcleo da *cache*. O diretório sabe quando um bloco precisa ser atualizado ou cancelado [6].

5.3 Sistema de Memória

Muitos aplicativos não são capazes de tirar pleno proveito da velocidade das CPU atuais, pois os seus desempenhos são ditados pela latência de memória (por exemplo, os programas que passam a maior parte do de seu tempo em listas ligadas). Se muitas cópias em paralelo são executadas destes programas,,a latência pode ser superada — os núcleos coletivamente pode proporcionar cargas pendentes suficientes para "esconder" a latência do sistema de memória subjacentes. De fato, sistemas de múltiplos núcleos foram concebidos com esse trabalho em mente [1].

De acordo com Agarwal e Levy [13], existem potenciais problemas de desempenho devido ao gargalo da banda de memória relacionada com as instruções e dados. Processadores com múltiplos núcleos podem resolver um aspecto deste problema por meio da distribuição de memória *cachê* dentro dos núcleos dos processadores. Do mesmo modo, a taxa da memória principal pode ser aumentada por meio da aplicação de vários portos de memória principal e um grande número de bancos de memória. Assim, nos múltiplos núcleos, o chamado gargalo da taxa de memória não é realmente um problema de memória. Pelo contrário, é uma questão de interconexão e o problema reside na forma como as unidades de memória fazem interface com os núcleos. A interface entre os bancos de memória e os núcleos é a interconexão, o que inclui tanto a rede *on-chip* quanto os pinos.

Soluções anteriores para a questão da interconexão escalável de redes aplicam-se à questão de memória, utilizando-se pacotes simples de transporte de memória ao contrário de dados processador-a-processador. Assim redes *on-chip* que são escaláveis aliviam o problema da taxa de memória. Os pinos que permitem o acesso a um processador com múltiplos núcleos acessarem a *offchip* DRAM também fazem parte da rede de interconexão. Em conclusão, interfaces de memória serial de alta velocidade darão um impulso significativo a curto prazo na taxa de pinos disponíveis para a memória e ao longo prazo, porém, taxas *off-chip* irão continuar a ser significativamente mais caras do que a largura de banda *on-chip*. Portanto nossos modelos de programação necessitam se adaptar para substituir comunicação baseada em memória *off-chip* com comunicação direta processador a processador *on-chip* [13].

5.4 Sistema de Barramento e Redes de Conexão

Que tipo de interconexão é o mais adequado para processadores com múltiplos núcleos? Uma abordagem baseada em barramento é melhor do que uma interconexão de rede? Ou existe um híbrido como a malha de rede que funciona melhor? A questão permanece. Embora o desempenho e a eficiência em energia sejam as principais vantagens dos processadores com múltiplos núcleos versus a abordagem de um único processador, eles também são desafiadores para melhorar à medida que o número de núcleos aumenta para além dos dígitos simples e se move para dezenas ou mesmo centenas de núcleos. O principal desafio de

lidar com o desempenho de um maior número de núcleos refere-se à rede que liga os vários núcleos em si e a memória principal. Sistemas atuais com múltiplos núcleos dependem de barramento ou de anéis para a sua interligação. Estas soluções não são escaláveis e, portanto, se tornarão um gargalo para o desempenho. Uma topologia comum interliga um barramento igualmente compartilhado entre todos os núcleos que estão ligados a ela. Arbitragem para a utilização do barramento é uma função centralizada e apenas um núcleo é permitido usá-lo em um determinado ciclo. Alternativamente, um anel é uma ligação unidimensional entre núcleos locais e a arbitragem é uma função de cada um dos interruptores. A malha, uma extensão bidimensional de comutação por pacotes, é ainda uma outra topologia de interconexão. Esta solução funciona bem com a tecnologia 2D VLSI (escala muito grande de integração) e também é a interconexão mais eficaz quando escalar a um grande número de núcleos. A malha pode ser escalar a 64 núcleos ou mais, pois a sua taxa de bisseção aumenta à medida que são adicionados mais núcleos, e sua latência aumenta apenas como a raiz quadrada do número de núcleos. (a taxa de bisseção é definida como a largura de banda entre as duas metades iguais dos múltiplos núcleos).

Contrastando isto com a topologia de barramento que apenas pode tratar cerca de oito núcleos, e o anel é viável até cerca de 16 núcleos. Assim para ambos (barramento e anel), a taxa de bisseção é fixa mesmo quando são adicionados mais núcleos, bem como a latência aumenta em proporção ao número de núcleos [13]. Portanto, memória extra será inútil se a quantidade de tempo necessário para os pedidos à memória não melhorar também. Reprojetar a interligação entre os núcleos de rede é um grande foco dos fabricantes de *chips*. Uma rede mais rápida significa uma menor latência na comunicação inter-núcleos e em transações de memória. [6]

5.5 Programação Paralela

De acordo com Geer [4], para *chips* com um único núcleo, programadores escrevem aplicações e estes *chips* priorizam tarefas na ordem em que elas devem ser realizadas para fazer a atividade designada mais eficiente. O sistema operacional, em seguida, atribui as tarefas a executar seriadamente no processador. Desenvolvedores que escrevem aplicações para *chips* com múltiplos núcleos devem dividir em tarefas que o sistema operacional pode atribuir; que decorrem paralelamente em vários núcleos. Cada núcleo tem a sua própria capacidade de multi-processamento e, portanto, pode dividir em partes as suas próprias tarefas que podem funcionar em paralelo. Uma questão-chave para os programadores é como uma atividade pode ser dividida em sub-tarefas.

Seguindo a lei de Moore, o número de núcleos em um processador poderia ser definido para duplicar a cada 18 meses. Isto só faz sentido se o *software* em execução nestes núcleos levar isso em conta. Em última análise, os programadores precisam aprender a escrever programas em paralelos que podem ser divididos e executados simultaneamente em múltiplos núcleos, em vez de tentar explorar o *hardware* de *chips* com um único núcleo para aumentar paralelismo de programas seqüenciais. O desenvolvimento de *software* para processadores com múltiplos núcleos traz algumas preocupações, tais como: Como é que um programador garante que uma tarefa de alta prioridade receba prioridade no processador, e não apenas em um núcleo? Em teoria, mesmo se um processo tivesse a mais alta prioridade dentro do núcleo em que está executando, poderia não ter uma alta prioridade no sistema como um todo. No entanto, como garantir que todo o sistema pára e não apenas o núcleo em que uma aplicação está a executar? Estas questões devem ser abordadas juntamente com o ensino de boas práticas de programação de paralelismo para desenvolvedores [6].

Embora o objetivo da programação para múltiplos núcleos seja clara, fazê-la não é necessariamente fácil. Uma das principais preocupações para os programadores é que algumas atividades, tais como os relacionados com gráficos, não se divide facilmente em sub-funções, como ocorre naturalmente em pontos de início e término. Nestes casos, o programador deve executar complexas transformações que produzem pontos. Assim uma aplicação pode ser dividida em tarefas menores, dividindo o *software* em partes distintas que são menores e mais granulares do que em sub-funções difícil de dividir. Outra opção seria mudar todas as estruturas de dados [4].

Para Schauer [6], o último e mais importante problema será usar multi-processos, ou outras técnicas de processamento paralelo, para obter o máximo desempenho de processadores com múltiplos núcleos. Reconstruir aplicações para serem multi-processos implica em um completo retrabalho pelos programadores na maioria dos casos. Os programadores têm que escrever aplicações com sub-rotinas capazes de serem executadas em diferentes núcleos, o que significa que as dependências de dados terão de ser resolvidas ou contabilizadas (por exemplo, latência na comunicação ou utilizando uma *cache* compartilhada). As aplicações devem ser balanceadas. Se um núcleo está sendo usado muito mais do que outro, o programador não tirará plena vantagem do sistema com múltiplos núcleos.

5.6 Núcleos Homogêneos e Heterogêneos

Arquitetos têm debatido se os núcleos em um processador com múltiplos núcleos devem ser homogêneos ou heterogêneos, e ainda não há uma resposta definitiva. Núcleos homogêneos são todos exatamente o mesmo: freqüências equivalentes, tamanhos de *cachê*, funções, entre outras características. No entanto, cada núcleo de um sistema heterogêneo pode ter uma diferente função, freqüência, modelo de memória, etc. Há um aparente *trade-off* entre a complexidade do processador e sua customização. Núcleos homogêneos são mais fáceis de produzir, uma vez que o mesmo conjunto de instrução é usado em todos os núcleos e cada núcleo contém o mesmo *hardware*. Mas eles são os mais eficientes para o uso nas tecnologias de múltiplos núcleos? Cada núcleo, em um ambiente heterogêneo, poderia ter uma função específica e executar o seu próprio conjunto de instrução especializado. Um modelo heterogêneo poderia ter um grande núcleo centralizado construído para tratamento genérico e rodar um sistema operacional, um núcleo para gráficos, um núcleo de comunicações, um núcleo para reforço matemático, um de áudio, um criptográfico, etc. [16].

Alderman [17] discute algumas vantagens em diversos aspectos de desempenho em uma arquitetura heterogênea. Multiprocessadores heterogêneos (ou assimétricos) apresentam oportunidades únicas para melhorar a taxa de processamento do sistema e de reduzir a energia de processamento. Heterogeneidade *on-chip* permite ao processador uma melhor execução dos recursos correspondentes a cada uma das necessidades da aplicação e endereça um espectro muito mais amplo de cargas - de baixo para alto paralelismo de processos com alta eficiência.

Pesquisas recentes em microprocessadores heterogêneos identificaram vantagens significativas sobre os processadores com uma abordagem de múltiplos núcleos homogêneo em termos de energia e taxa de processamento, além de enfrentar os efeitos da lei de Amdahl sobre o desempenho de aplicações paralelas. Em um *chip*, no entanto, nenhum núcleo único precisa ser ideal para o universo das aplicações. A utilização da heterogeneidade explora este princípio. Inversamente, uma concepção homogênea na verdade agrava ainda mais o problema, criando um único projeto universal e, em seguida, replicando esta solução em todo o *chip*. O modelo heterogêneo é mais complexo, mas pode ter benefícios em eficiência, energia e calor que superam a sua complexidade. Com os principais fabricantes de ambos os lados desta questão entre a abordagem homogênea e heterogênea, este debate de prós e contras se estenderá ao longo dos próximos anos [6].

6. CONCLUSÃO

A tecnologia de múltiplos núcleos encapsulados em um único processador já é realidade na computação nos dias atuais. Objetiva-se com estes melhorar o desempenho das aplicações a partir de melhores potências e eficiência térmica, ou seja, ter o potencial de prover melhor desempenho e escalabilidade sem um correspondente aumento em consumo de energia e calor, conforme seria o caso do aumento de freqüência de *clock* nos processadores com apenas um único núcleo [8].

Os processadores com múltiplos núcleos são bem sofisticados e estão definitivamente arquitetados a aderir a um moderado consumo de energia, além de um melhor controle da dissipação de calor e melhores protocolos de coerência de *cache*. Apesar das claras evidências de seu uso hoje, já no mercado, e das potencialidades desta tecnologia, diversos desafios ainda necessitam ser desbravados tais como: sistema de barramento e interconexão com a memória, programação paralela e arquiteturas homogêneas verso as heterogêneas; conforme apresentado neste trabalho. Além disso, diversas questões arquiteturais sobre estes processadores permanecem em aberto e ainda não foram resolvidas de maneira plausível pela literatura.

Observa-se que a tecnologia de múltiplos núcleos em um único processador poderá revolucionar o modo de se desenvolver aplicações, no entanto o desempenho dependerá muito do modo como as aplicações aproveitarão os recursos e vantagens (por exemplo, paralelismo) em sua total capacidade. Há relativamente poucas aplicações (e mais importante, poucos programadores com este conhecimento) para escrever níveis de paralelismo. Finalmente, segundo Schauer [6], processadores com múltiplos núcleos são uma importante inovação na evolução dos microprocessadores e com programadores qualificados e capazes de escrever aplicações paralisáveis, a eficiência poderá aumentar drasticamente. Nos próximos anos, poderá haver muitas melhorias para os sistemas computacionais, e essas melhorias vão proporcionar programas mais rápidos e uma melhor experiência computacional para os usuários.

7. REFERENCIAS

[1] Kaufmann, R. & Gayliard, B. 2009. **Multicore Processors**. Dr. Dobb's Journal. June 2009, http://www.ddj.com/architect/212900103

[2] Gorder, P. F. 2007. **Multicore Processors for Science and Engineering**. Computing in Science & Engineering. vol. 9, issue: 2. pp. 3-7.

[3] Agarwal, Anant; Levy, Markus. 2007. **The KILL Rule for Multicore**. Design Automation Conference. DAC '07. 44th ACM/IEEE, vol., no., pp.750-753, 4-8 June 2007.

[4] Geer, David. 2007 "**For Programmers, Multicore Chips Mean Multiple Challenges,**" Computer, vol. 40, no. 9, pp. 17-19, Aug. 2007, doi:10.1109/MC.2007.311

[5] AMD. 2005. **Multi-Core Processors: the next evolution in computing**. AMD, June 2009, http://multicore.amd.com/Resources/33211A_Multi-Core_WP_en.pdf

[6] Schauer, Bryan. 2008. **Multicore Processors: a necessity**. June 2009, http://www.csa1.co.uk/discoveryguides/multicore/review.pdf?SID=sfjgtitfdsj6h998ji91ns5nd4

[7] Knight, W. "**Two Heads Are Better Than One**", IEEE Review, September 2005

[8] Fruehe, John. 2005. **Multicore Processor Technology.** June 2009, http://www.dell.com/downloads/global/power/ps2q05-20050103-Fruehe.pdf

[9] Akhter, Shameem & Roberts, Jason. 2006. **Multi-Core Programming: Increasing Performance through Software Multi-threading**. Intel Press; 1st edition. 360 pages.

[10] Geer, David. 2005. **Chip Makers Turn to Multicore Processors.** Computer volume 38, issue 5, May 2005 pp. 11–13.

[11] Knight, W. 2005. "**Two Heads Are Better Than One**", IEEE Review

[12] Merritt, R. 2008. "**CPU Designers Debate Multi-core Future**", EETimes Online, June 2009, http://www.eetimes.com/showArticle.jhtml?articleID=206105179

[13] Agarwal, Anant & Levy, Markus. 2007. **Going multicore presents challenges and opportunities.** Embedded Systems Design. June 2009, http://www.design-reuse.com/articles/15618/going-multicore-presents-challenges-and-opportunities.html

[14] Muthana, P.; et al. 2005. **Packaging of Multi-Core Microprocessors: Tradeoffs and Potential Solutions**. Electronic Components and Technology Conference

[15] Gal-On, Shay & Levy, Markus. 2008. "**Measuring Multicore Performance,**" Computer, vol. 41, no. 11, pp. 99-102, Nov. 2008, doi:10.1109/MC.2008.464

[16] Alderman, R. 2007. "**Multicore Disparities**", VME Now, June 2009, http://vmenow.com/c/index.php?option=com_content&task=view&id=105&Itemid=46

[17] Kumar, R.; Tullsen, D. M.; Jouppi, N. P.; Ranganathan, P. 2005. **Heterogeneous chip multiprocessors**. Computer. volume 38, issue 11, Nov. 2005, pp. 32 – 38.