Paralelismo em nível de threads

Tatiana Al-Chueyr
(017396)
CTI Renato Archer
Rodovia Dom Pedro I, km 143
Campinas, SP, Brasil
+55 19 3746-6035
tmartins@cti.gov.br

RESUMO

Neste artigo são apresentadas arquiteturas que empregam paralelismo em nível de threads (TLP). Inicialmente, analisa-se o histórico e as motivações que desencadearam o desenvolvimento desta forma de paralelismo. Então, são apresentados dois paradigmas para implementação de TLP (*Simultaneous Multithreading* e *Chip Multi-processing*). Com base em tais paradigmas, são descritos dois processadores comerciais que oferecem paralelismo a nivel de threads (Hyperthreading e Niagara). Então, discute-se o desempenho da aplicação deste tipo de arquitetura tanto em servidores, quanto em aplicações multimídia e desktop. Por fim, são apresentados alguns desafios que enfrentados no desenvolvimento e na utilização de paralelismo em nível de threads.

Categorias e Descritores de Assunto

C.1.2 [**Arquiteturas de Processadores**]: Arquiteturas de Mútiplos Streams de Dados

Termos Gerais

Modelagem (Design).

Keywords

Paralelismo em nível de threads, multiprocessadores.

1. INTRODUÇÃO

Um dos constantes desafios no desenvolvimento de novas arquiteturas de processadores, ao longo da história, é prover aumento de desempenho associado a baixo custo de produção.

Visando satisfazer tais desafios, nas últimas décadas os processadores passaram por muitas mudanças importantes, dentre as quais destacam-se: (a) a divisão da execução de instruções em estágios; (b) a criação de pipelines, que possibilitam a execução de múltiplas instrucões em um mesmo ciclo do processador; (c) o desenvolvimento de processadores superescalares, que além de permitir a execução paralela e simultânea de instruções, também oferecem a sobreposição parcial oferecida por pipelines.

Processadores superescalares exploram o paralelismo em nível de instrução (ILP, sigla para *Instruction Level Parallelism*), podendo utilizar uma série de recursos para isso, tais como: execução fora-de-ordem, hierarquia de memória, previsão de desvios, dentre outros. Entretanto, estas técnicas fazem com que os processadores tornem-se mais complexos e tenham mais transistores dissipando grande quantidade de energia e calor. Exemplos comerciais de processadores superescalares são o Intel Pentium e o Sun UltraSparc.

Uma das estratégias para melhorar o desempenho de processadores superescalares, historicamente, foi aumentar a frequencia de clock, de modo que as instruções pudessem ser executadas cada vez mais rapidamente. Entretanto, para que tal frequencia pudesse ser aproveitada, surgiram dois desafios: o uso eficiente de um elevado número de estágios de pipeline e o aumento de consumo de energia, decorrente de maiores áreas de silício, maior número de transistores e maior dissipação de calor.

Ainda, apesar do ILP disponível em procesadores superescalares ser adequado para muitas aplicações, ele é ineficaz para outras, como programas em que é difícil predizer código. Outras limitações da ILP podem ser lidas na literatura [1].

Em contrapartida ao paralelismo a nível de instruções, observou-se que muitas aplicações apresentam melhor desempenho quando executadas com outra forma de paralelismo: em nível de threads ou TLP (*Thread Level Parallelism*).

O TLP envolve controlar múltiplas threads do processador, permitindo que partes específicas do programa sejam distribuídas entre diferentes processadores e possam executar simultaneamente. A Figura 1 ilustra a transformação de uma execução serial em paralela, na qual uma iteração que demoraria seis ciclos de clock é dividida em três threads e é reduzida a dois ciclos de clock.

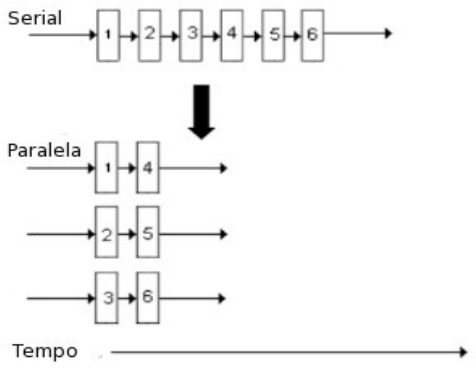

Figura 1. Transformação de código serial para paralelo

Há alguns contextos em que a TLP oferece melhor desempenho que a ILP, outros contextos em que a ILP provê melhor desempenho e, por fim, há situações em que ambas tem desempenho similar [2]. É fundamental que seja analisada a aplicação, para que então seja definida qual forma de paralelismo é a mais adequada.

Um método comum para aumentar o desempenho geral da TLP é o uso de múltiplas CPUs independentes ou multicores. De acordo com a taxonomia de Flynn [3] pode-se classificar multiprocessadores que dão suporte a TLP como MIMD (*Multiple instruction streams, multiple data streams*), onde cada processador controla seu próprio conjunto de instruções e opera seu próprio dado.

A seguir veremos dois paradigmas de processadores que permitem o paralelismo a nivel de threads. Na sequencia, serão apresentadas duas arquiteturas de processadores comerciais: Hyper-Threading (Intel) e Niagara (Sun Microsystem). Então será discutida a aplicação de TLP em contextos distintos, tais como em aplicações multimidia, servidores e aplicativos multimidia. Por fim, apresentamos desafios com os quais tanto arquitetos quanto programadores enfrentam ao lidar com TLP.

2. CLASSIFICAÇÃO

Processadores que implementam TLP são convencionalmente classificados segundo pelo menos um destes paradigmas: *Simultaneous Multithreading* (SMT) e *Chip Multiprocessing (CMP)*.

2.1 *Simultaneous Multithreading*

O SMT [4] é um paradigma que tem sido empregado industrialmente [5,6]. Ele permite que instruções de múltiplas threads, simultaneamente, sejam buscadas e executadas no mesmo pipeline, amortizando o custo de mais componentes distribuído entre mais intruções por ciclo.

O *Simultaneous Multithreading* propõe oferecer melhoria no *throughput* através da relação eficiência-área [7].

Uma arquitetura SMT é capaz de atingir alto IPC (instruções por ciclo) devido as seguintes razões:

1. A facilidade de paralelizar tarefas distintas, principalmente quando não há comunicação entre elas;
2. O escalonamento de instruções não-bloqueadas para o despacho pode esconder as altas latências de outras instruções bloqueadas, tais como as de acesso a memória;
3. Através da interpolação de instruções de diferentes tarefas, a execução da SMT otimiza os recursos que poderiam estar ociosos, tais como unidades funcionais, dada a maior variedade de tipos de instruções.

A arquitetura SMT é composta por unidades funcionais compartilhadas e por estruturas e recursos separados, logicamente ou fisicamente.

O Pentium 4 Extreme SMT permite a execução de duas threads simultâneas e proporciona um *speedup* de 1.01 no benchmark SPECint_rate e de 1.07 no SPECfp_rate. Ao executar o processador Pentium 4 para 26 benchmarks SPEC, o speedup varia entre 0.90 e 1.58, com média 1.20 [7].

Já no Power 5, um servidor executa 1.23 mais rápido o benchmark SPECint_rate com SMT, 1.16 mais rápido para SPECfp_rate.

Algumas dificuldades que podem emergir com esta abordagem [1]:

- escolher qual instrução será executada por vez;
- utilizar a estratégia de uma thread preferida faz com que o processador sacrifique parte do *throughput* quando ocorre *stall* de thread;
- criar arquivo de registradores para manter múltiplos contextos (gargalo); e, por fim,
- assegurar que os conflitos gerados pela a cache e a TLP não degradam o desempenho..

2.2 *Chip Multiprocessing*

O CMP é um multiprocessamento simétrico (SMP) implementado em um único chip [8]. Múltiplos núcleos de processador são interconectados e compartilham um dos níveis de cache (convencionalmente o segundo ou terceiro nível). Em geral cada core há b*ranch predictors* e caches de primeiro nível privadas.

O objetivo de uma arquitetura CMP é permitir maior utilização de paralelismo a nível de threads, sem contudo prover suporte a paralelismo a nível de instrução.

Para *workloads* multi-threaded, arquiteturas CMP amortizam o custo do chip entre os múltiplos processadores e permitem comparitlhamento de dados entre caches L2.

Alguns exemplos de arquiteturas comercciais que utilizam CMP: PA-RISC (PA-8800), IBM POWER 4 e SPARC (UltraSPARC IV).

Para que ocorra o uso efetivo de chips CMP é necessário que o sistema operacional empregado dê suporte a multiprocessamento.

2.3 *Comparação*

Em ambos os paradigmas, busca-se o aumento do throughput. A replicação de cores significa que a área e que o *overhead* de energia necessários para o CMP são muito superiores ao SMT. Para um determinado tamanho de chip, um SMT de apenas um core terá suporte a um tamanho maior de L2 do que em um chip multi-core.

Por outro lado, a falta de contenção e execução entre threads tipicamente existente no SMT permite um *throughput* bastante superior para o CMP. Uma das maiores preocupações em adicionar-se múltiplos cores ao chip é o aumento drástico de dissipação de energia, o que agrega a este tipo de processador custos adicionais para resfriamento.

Existem arquiteturas que abordam ambos paradigmas – tanto o CMP quanto o SMT. Um exemplo é a arquitetura Power 5 da Intel, onde há dois núcleos de processadores SMT

A partir da literatura [7], selecionamos dois gráficos que ilustram o desempenho e eficiência no consumo de energia do SMT e do CMP. Tais gráficos podem ser vistos nas Figuras 2 e 3. Para a elaboração destes gráficos foi considerado um processador Power 4.

Como pode ser observado, há situações em que um dos paradigmas apresenta melhor desempenho, e há situações em que o outro é vitorioso. O que reforça que a definição de qual tecnologia utilizar é intimamente relacionada a aplicação final e em que contextos o processador será empregado.

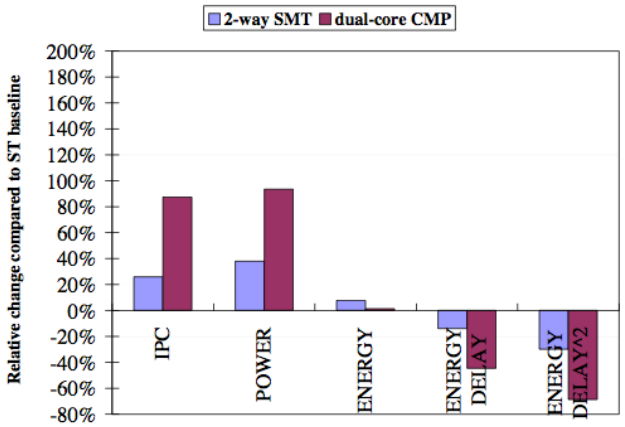

Figura 2. Comparação do desempenho e eficiência do uso de energia no SMT e no CMP para workloads com poucos miss na cache L2.

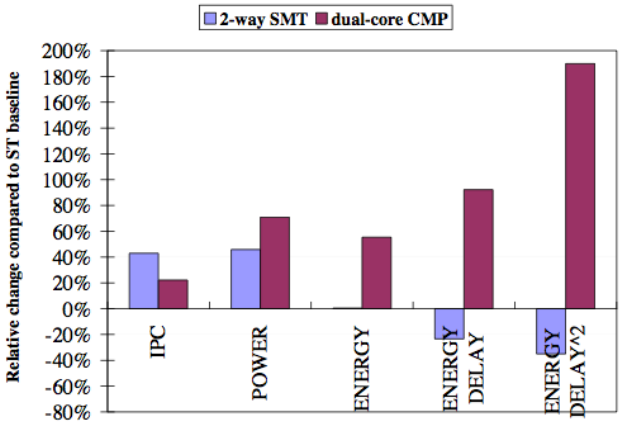

Figura 3. Comparação do desempenho e eficiência do uso de energia no SMT e no CMP para workloads com muitos miss na cache L2.

3. PROCESSADORES COMERCIAIS

3.1 HyperThreading

A tecnologia HyperThreading (ou HT), desenvolvida pela Intel, é a precursora dos processadores de núcleo duplo e múltiplo, tais como o Intel Core 2 Quad. Esta tecnologia se baseia na abordagem SMT.

A HT simula em um único processador físico dois processadores lógicos. Cada processador lógico recebe seu próprio controlador de interrupção programável (APIC) e conjunto de registradores. Os outros recursos do processador físico, tais como, cache de memória, unidade de execução, unidade lógica e aritmética, unidade de ponto flutuante e barramentos, são compartilhados entre os processadores lógicos. Em termos de software, o sistema operacional pode enviar tarefas para os processadores lógicos como se estivesse enviando para processadores físicos distintos, em um sistema de multiprocessamento real.

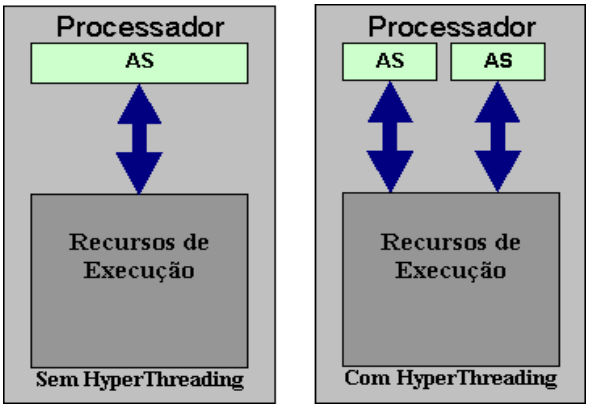

Figura 4. Comparação de um processador com e sem a tecnologia

A Figura 4 ilustra o funcionamento de um processador normal e um processador com a tecnologia HyperThreading. Os registradores e controlador de interrupção foram chamados de "AS". Na área denominada de "recursos de execução" estão todos os recursos que o processador necessita para executar as instruções. No processador com HP ocorre duplicação de registradores, controladores e compartilhado os recursos de execução entre os processadores lógicos, parecendo assim um sistema com dois processadores reais.

Para que se usufrua da tecnologia, tanto o sistema operacional utilizado quando os aplicativos de software têm que dar suporte a HyperThreading.

O primeiro processador da Intel a implementar a tecnologia HyperThreading foi o Intel Xeon. O Intel Xeon utiliza a arquitetura NetBurst e é voltado para o mercado de servidores. Apesar do foco inicial da tecnologia HyperThreading ser processadores para servidores de rede, forem feitos chipsets (Intel 845PE) para os processadores Pentium 4.

Sistemas operacionais como o Windowx XP e algumas distribuições de GNU Linux são SMP (Multiprocessamento Simétrico), ou seja, podem trabalhar com mais de um processador instalado no sistema, dividindo às tarefas entre os mesmos. A tecnologia HyperThreading estende essa idéia de forma que os sistema operacionais e software aplicativos dividam as tarefas entre os processadores lógicos.

As Figuras 5 e 6 ilustram as diferenças entre uma arquitetura multiprocessada (física) e a tecnologia HyperThreading.

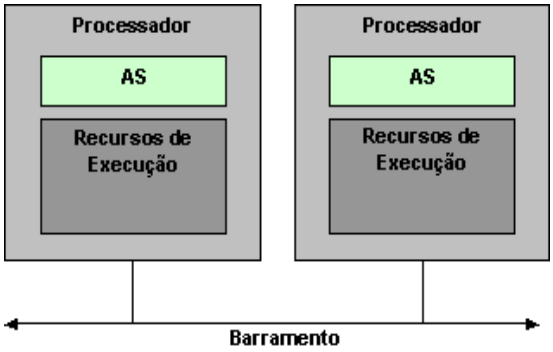

Figura 5. Sistema Multiprocessado sem tecnologia HyperThreading.

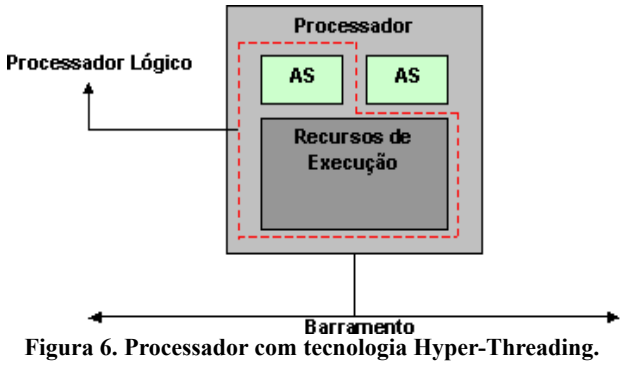

Figura 6. Processador com tecnologia Hyper-Threading.

Nos PCs desktop e workstations simples, a tecnologia HT aproveita da capacidade de multiprocessos, disponível no sistema operacional ou nos aplicativos, dividindo as cargas de trabalho em vários processos, que são designados e enviados independentemente. Num sistema de multiprocessador real, eles são executados em processos diferentes.

Nos servidores e workstations de alto desempenho, a tecnologia HyperThreading habilita a TLP, ao duplicar a arquitetura de cada processador e ao mesmo tempo compartilhar um conjunto de recursos de execuçãod o processador. Ao programar os processos, o sistema operacional trata os dois estados distintos de arquitetura como processadores lógicos ou virtuais separados, o que permite que softwares multiprocessados rodem sem modificações.

Embora a tecnologia HyperThreading não ofereça o nivel de escalonamento de desempenho alcançado ao adicionar um segundo segundo processador, testes de benchmark mostram que alguns aplicativos de servidor tem um desempenho 30% maior. [9].

Os usuários aproveitam do desempenho melhorado, executando múltiplos aplicativos simultaneamente, como, por exemplo, rodar uma análise de vírus ou codificação de vídeo no background e ao mesmo tempo continuar com um jogo. Para os gerentes da TI, a tecnologia HT significa um uso mais eficiente dos recursos do processador, maior saída e desempenho melhorado.

Conforme apresentado nos gráficos disponíveis nas Figuras 7 e 8, obtidas da literatura [9], a tecnologia hyper-threading pode ter ganhos consideráveis em aplicação típicas de servidores (banco de dados, servidores web). Por outro lado, em certas aplicações de servidores e aplicações típicas de desktop, a tecnologia pode apresentar perdas consideráveis de desempenho, conforme mostra a Figura 8.

3.2 Niagara

O Sun Niagara, ou UltraSPARC T1, é um processador que visa atender servidores de alto *workload*, como *webservers* e *datacenters*. Para isso, o Niagara possui 8 cores de processamento (são comercializadas versões de 4 e 6 cores, também), onde cada um executa 4 threads de hardware, totalizando 32 threads [10].

Este processador foi iniciado em um projeto de pesquisa chamado Hydra [11], desenvolvido pelo professor Kunle Olukotun, relacionado a CMP. Com a conclusão do projeto, a empresa Afara Websystems foi criada para comercializar o projeto Hydra. Entretanto, a empresa acabou sendo vendida para a Sun Microsystems, em 2002, onde o projeto foi a base do Niagara.

O Niagara tem como objetivo atingir altos valores de performance em paralelo a consumo adequado de energia. Assim, o processador consome, em média, 72W, chegando a um pico de 79W. O principal fator para isso é a sua arquitetura simples, composta por vários cores de baixa frequência, mas alto throughput. Entretanto, o Niagara possui algumas limitações – sua limitação mais séria é o fato de possuir apenas uma unidade de ponto flutuante (FPU) para 8 cores, e assim é incapaz de processar mais que 1-3% de operações de ponto flutuante.

A tecnologia CMT é utilizada no Sun Niagara. Esta tecnologia consiste na combinação de CMP e VMT (vertical multithreading - apenas uma das threads de cada core pode executar instruções em um ciclo, trocando para outra thread quando a thread ativa precisa buscar dados na memória). No Sun Niagara há 4 threads de hardware por core (com o intuito de maximizar a eficiência de cada core) e 8 cores (de forma a aumentar o poder de processamento).

Cada core possui um pipeline de 6 estágios, um instruction cache L1, um data cache L1 e uma unidade de gerenciamento de memória (MMU), que são compartilhados pelas 4 threads, e todos os cores são ligados ao cache L2, que é compartilhado por todos. Assim, o sistema operacional rodando na máquina enxerga 32 processadores virtuais (ou lógicos).

A motivação por trás desta técnica é o grande atraso causado por acessos a memória – normalmente na ordem de dezenas a centenas de ciclos. Utiliza-se então esta técnica para que instruções de várias threads de software sejam executadas simultaneamente, nas diversas threads de hardware (de forma que a pausa para I/O de uma não afete o throughput geral significativamente).

O Niagara é um processador voltado para aplicações de servidor – cujas cargas de trabalho são caracterizadas por altas taxas de TLP) e baixos níveis de ILP.

O desempenho de um sistema baseado em CMT é altamente relacionada ao sistema operacional utilizado – no caso do Niagara, o Solaris 10 (ou superior), sistema operacional da Sun, é bastante otimizado para isso – alocando corretamente as threads de software para os cores mais eficientes, reduzindo o atraso causado pelo uso de recursos compartilhados (o cache L2 e o bus).

Obtivemos a performance do Sun Niagara em testes realizados contra dois outros servidores: um Dell 2850 Dual 3.2 GHz Xeon com 12GB de RAM, rodando Debian, e um Dell 7250 Itanium (dual 1.5 GHz com 32GB de RAM). O Niagara foi testado no Sun FIRE T2000, uma das máquinas comercializadas pela Sun com ele. Os testes realizados utilizaram o Apache, ou seja, os servidores foram testados como webservers. Foram feitos por Colm MacCarthaigh, do webserver ftp.heanet.ie.

A performance do Niagara é consideravelmente superior a dos outros 2 em volume de transações por segundo (5.700 contra 2.700 do Itanium, o segundo maior – e o valor foi ampliado pra 22.000 em testes posteriores ao benchmark). Além disso, o servidor agüentou quase 150% dos downloads concorrentes do Itanium, e 300% do Xeon (83.000 para o Niagara, 57.000 Itanium, 27.000 Xeon). Além disso, a latência do Niagara, apesar de mais alta em valores mais baixos de downloads, aumenta de forma muito menos acentuada que os outros dois para maiores valores (chegando aos máximos que agüentam com valores perto de um minuto de latência).

Temos também que a energia consumida pelo Sun T2000 é muito mais baixa que a dos dois Dell: em média, aproximadamente 240 W, com picos de 300W, enquanto o Dell possui uma média de 384 W e picos de 480 W, e o Dell 7250 média de 432 W e picos de 538 W. [12]

Os problemas encontrados no Niagara foram, exatamente, em performance individual: quando foram testados I/O de apenas uma thread, ou apenas um download, o valor obtido foi muito abaixo dos outros.

Em 2007 foi lançado o Niagara-2, no qual houve melhoria de desempenho do processador, e diminuição ou manuteção no consumo de energia) [13].

O Niagara-2 tem 8 cores de 1.4 GHz, sendo capaz de executar 8 threads de hardware por core. Isso foi possível colocando uma segunda unidade de execução (EXU) em cada core. Assim, é possível que mais de uma thread esteja em execução simultaneamente em cada core, desde que todas as partes requeridas pela instrução estejam disponíveis em dobro no pipeline (a unidade de load/store, por exemplo, não está, logo duas instruções de load/store simultâneas não são possíveis).

Cada core matém o padrão da cache L1 do Niagara (8KB de data cache, 16KB de instruction cache), e a cache L2 aumentou para 4MB. Além disso, há uma floating point unit por core, corrigindo (ou amenizando) um dos maiores defeitos do Niagara. Também é possível desligar cores ou threads de um core, a fim de economizar energia.

A Figura 7 compara o desempenho do Niagara a outros processadores. Pode-se observar que o processador supera de modo bastante expressivo o desempenho das demais arquiteturas para os benchmarks: SPECJBB05, SPECWeb05 e TPC. No caso do SPECWeb05, uma das justificativas para uma diferença de desempenho tão significativa é que Solaris é bastnate otimizado para Java (linguagem de programação utilizada neste benchmark). Entretanto, devido as limitações de operações de ponto flutuante, apreseta desempenho sofrível no SPECFPRate.

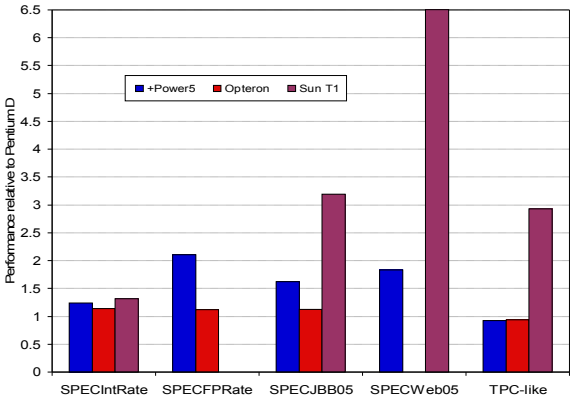

Figura 7. Comparação de desempenho utilizando benchmarks SPEC e TPC, comparando o Niagara ao Power 5 e ao Opteron.

Comparando o Niagara ao Xeon (MultiThreading), considerando o contexto de aplicação servidores web, para grande dos casos o Niagara é o mais indicado, conforme apresentado na Figura 8.

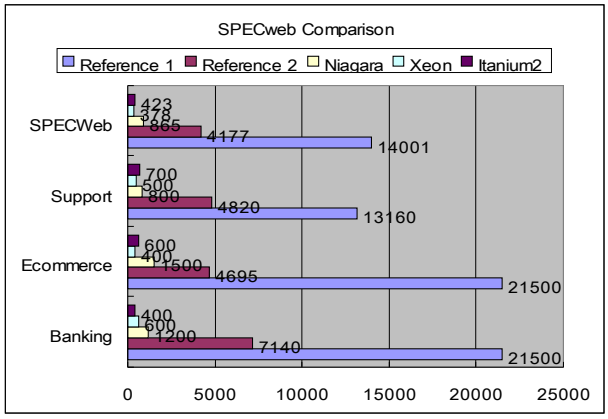

Figura 8. Comparação de desempenho utilizando benchmarks SPECWeb. A referência 1 consiste em um Niagara (1.2GHz CPU, Sun Java Web Server 6.1 + JSP) e a referência 2 é umXeon (2x3.8GHz CPU w/ HT, 2MB L2, Zeus + JSP)

.

4. APLICAÇÕES

Há diversos trabalhos que descrevem a eficiência ou a limitação na aplicação de tecnologias multi-threading a diversos contextos. Estudos demonstram restrições da aplicação de paralelismo em nível de thread em programas não numéricos [14] e outros discutem o real ganho em utilizar TLP em aplicativos Desktop [15], por exemplo.

Uma das principais motivações para a TLP foi melhorar o desempenho de servidores, como descrito anteriormente e exemplificado com o Niagara da Sun. Entretanto, há outras aplicações onde a tecnologia também provém ganhos. Elas são abordadas a seguir.

4.1 Aplicativos Desktop

Atualmente os multiprocessadores estão sendo amplamente comercializados em computadores de propósito geral. Entretanto, questiona-se até que ponto a TLP é aproveitada na execução de aplicações que exigem interatividade com o usuário, em oposição a servidores web.

Estudos [15] apresentam que usar dois processadores ao invés de um é uma forma de reduzir a duração de execução de aplicativos do dia-a-dia de usuários comuns de computadores. A TLP permite, em média, redução em 22% do tempo de execução (resultados variam entre 8% e 36%). Ainda, estas melhorias correspondem 16%-72% da máxima redução alcançavel por processadores dual-core (50%)[15]. Ou seja, utilizar dois processadores e TLP pode ser um meio eficiente e eficaz para melhorar o desempenho mesmo de aplicativos baseados na interatividade com o usuário.

Utilizar uma arquitetura Dual Core para executar um player de MP3 em background pode aumentar o tempo de resposta em 29%. Apesar do segundo processador eliminar a maior parte do overhead de tarefas em background, ele não absorve tais tarefas completamente.

Um aspecto importante apresentado nestes estudos é que não é vantajoso utiilizar mais de dois proessadores, para a maior parte das aplicações testadas.

Para os testes foi utilizado GNU Linux, o qual foi apresentado como uma plataforma SMP. Acredita-se que removendo o lock global do kernel seria possível melhorar ainda mais o paralelismo em nível de threads, mas que o maior pontencial é na reescrita de aplicativos visando o processamento multi-threading.

4.2 Aplicativos multimidia

Em contraposição a utilização de TLP em aplicações não adaptadas, como apresentado previamente, há testes de implementações específicas com suporte a TLP visando atender diversas aplicações, tais como codificação de video, processamento de imagens e de audio.

A literatura [16] apresenta que é possível reduzir drasticamente o tempo de execução de um codificador de MPEG2 utilizando TLP, onde são relatadas reduções de 96% para a utilização de 32 processadores (contextos) e uma faixa de 80 quadros ou imagens.

5. DESAFIOS

5.1 Paralelismo na Camada de Software

Para que se aproveite os benefícios propiciados pela TLP, é importante que existam programadores habilitados e que saibam usufruir da arquitetura.

A TLP pode ser aplicada em divrsos níveis de camada de software. A seguir são apresentadas algumas possíveis abordagens.

5.1.1 Sistema Operacional

Ao invés de todos os softwares terem que se preocupar com o paralelismo, a resposabilidade pode ser levada ao sistema operacional. Considerando um servidor que possui múltiplos cores, e que cada um pode rodar várias threads. Qual seria o melhor modo de aproveitar esta arquitetura para executar quatro threads simultâneas? Tudo depende do funcionamento delas. Se elas compartilharem memória, o ideal é que estejam próximas, caso contrário podem estar distantes, de modo a aproveitar o máximo de recursos independentes possíveis. Ainda considerando a memória, o programa de gerenciar o tráfego memória para CPU é bastente complexo neste tipo de proesador. Este tipo de problema, claramente, deve ser tratado pelo SO, e não pelas aplicações de nível mais alto.

Os sistemas operacionais recentes já oferecem suporte a multithreading (Unix, Solaris, Linux e Windows). Entretanto, ainda são necessários mais estudos e implementações para adequação dos OS existentes. Talvez uma refatoração bastante séria seja necessária.

Grande parte das aplicações é serial, não sendo vantajoso o recurso de TLP obeservando-as individualmente. Entretanto, os sistemas operacionais podem distrituir um conunto deste tipo de aplicação para ser executado paralelamente, lidando com as threads. Bons resultados já são obtidos com esta abordagem.

5.1.2 Compiladores

Hoje alguns compiladores, como o GCC, permitem a geração de código de máquina otimizado para suporte multi-threading, bastando para isso passar informações específicas na hora de compilar o programa.

Esta abordagem é intimamente dependente da linguagem, do compilador e do sistema operacional.

5.1.3 Softwares de infraestrutura

Aplicativos como banco de dados e web-servers tendem a ser projetados para dar suporte a threads, pois de um modo geral eles precisam oferecer alto desempenho.

Apache 2.0, por exemplo, foi estruturado de modo bastante inteligente, permitindo que sejam executados vários processos contendo algumas threads, ou poucos processos com várias threads. Entretanto, o Apache tende a ser reestruturado conforme programadores forem aprendendo a usufruir desta tecnologia, e novas demandas surgirem.

5.1.4 Camada da Aplicação

A princípio, por definição, todos os aplicativos de alto processamento de dados irão poder usufruir do paralelismo para obter melhores resultados. Softwares de processamento de imagens, videos, cálculos para física e matemática tem potencial para aproveitar esta tecnologia. Entretanto, um limitante é que programação baseada em threads é complicada.

Para se aproveitar ao máximo os chips TLP, é necessário uma grande evolução nas ferramentas de desenvolvimento, depuração e teste de software. Para bons programadores, é simples realizar "unit-tests" na maior parte das linguagens (JUnit aos programadores Java) para aplicações que rodam sobre uma única thread. Entretanto, quando o programa passa a ser paralelo, há um problema pois é difícil estabelecer um framework mental de como prever testes com múltiplas threads. Primeiro seria importante definir *design patterns*, para que então fosse possível desenvolver ferramentas de auxílio a testes e depuração.

Neste aspecto, a escolha da linguagem de programação pode afetar o resultado. É importante que a linguagem escolhida tenha um bom suporte para código paralelo multi-thread, facilitando que problemas de paralelismo, tal como a disputa de recursos, sejam encontrados.

5.2 Desafios e abordagens de hardware

Multiprocessadores é uma área ampla e diversa, sendo que grande parte dos desenvolvimentos são bastante recentes. Até recentemente havia mais casos de fracassos do que sucessos na área.

Um dos desafios em multiprocessadores é definir qual abordagem é mais adequada: processadores simétricos ou assimétricos. Ao longo deste artigo abordamos apenas arquiteturas simétricas, mas há implementações, como o Cell Hypervisor, na qual há 8 cores de uma categoria e um núcleo mais poderoso (Power) que repassa tarefa para os demais processadores.

Outras questões que merecem destaque [1] são o custo de comunicação entre processsadores e a troca de dados. Ambas são dependentes das arquiteturas adotadas – seja de comunicação entre processadores, seja da hierarquia de memória empregada. Em ambas situações surge latência e existem uma série de abordagens para tentar contornar tais problemas. Entretanto, estamos longe de termos achado uma solução definitiva.

6. CONCLUSÃO

Neste artigo apresentamos a importante abordagem de paralelismo a nível de thread, que tem sido amplamente empregada em processadores comerciais nos últimos anos. Foram analisadas as abordagens de implementação SMT e CMT.

Foi possível estudar como grandes fabricantes de processadores implementaram multithreading em suas arquiteturas, sendo que foram analisadas as tecnologias HyperThreading da Intel e a Niagara da Sun, exemplos comerciais das tecnologias SMT e CMT, respectivamente.

Observou-se que a eficiência da utilização destas tecnologias a servidores com alta carga de trabalho, em especial em circustâncias onde há níveis elevados de TLP e níveis relativamente baixos de ILP. Ainda assim constatou-se que o uso efetivo de TLP ainda está intimamente relacionado às implementações das aplicações.

Sem dúvidas o paralelismo em nível de threads e as arquiteturas multicore são uma grande oportunidade de pesquisa, mas também trazem consigo desafios, tanto na esfera de hardware quanto para o desenvolvimento de software.

7. REFERÊNCIAS

[1] Hennessy, J. L. and Patterson, D. A. 2007. Computer Architecture: A Quantitative Approach..Morgan Kaufmann Publishers, Inc. San Mateo, CA. Forth edition, 1995.

[2] Mitchell, N., Carter, L., Ferrante, J., Tullsen, D. 1999. ILP versus TLP on SMT. Supercomputing ACM/IEEE 1999 Conference. (Nov. 1999), 37-37.

[3] Flynn, M. 1972. Some Computer Organizations and Their Effectiveness, IEEE Trans. Comput., Vol. C-21, pp. 948.

[4] Tullsen, D. M.; Egger S., and Levy, H. M. 1995. Simultaneous multithreading: Maximizing on-chip parallelism. In *Proc. ISCA-22*, 1995.

[5] Kalla, R., Sinharoy B., and J. Tendler. 2003. Power5: Ibm's next generation power microprocessor. In *Proc. 15th Hot Chips Symp*, pages 292–303, August 2003.

[6] Marr D. T., Binns F., Hill D. L., G. Hinton, Koufaty D. A., Miller J. A. , and Upton M. 2002. Hyper-threading technology architecture and microarchitecture. *Intel Technology Journal*, 6(1):4–15, Feb. 2002.

[7] Skadron Y. L., Brooks K., Zhigang Hu D. 2005. Performance, energy, and thermal considerations for SMT and CMP architectures. *High-Performance Computer Architecture*, 2005. HPCA-11. 11th International Symposium , pages 71- 82, February 2005.

[8] Heo S., Barr K., and Asanovic K.. 2003. Reducing power density through activity migration. In *Proc. ISLPED'03*, Aug. 2003.

[9] Marr D., Binns F., Hill D., Hinton G., Koufaty D., Miller J. Upton M. 2002. Hyper-Threading Technology Architecture and Microarchitecture.ntel Technology Journal, vol.3, issue 1, ITJ 2002

[10] Nagarajayya, N. 2005. Improving Application Efficiency Through Chip Multi- Threading. DOI = http://developers.sun.com/solaris/articles/chip_multi_thread.html. Acessado 15/06/2009.

[11] Kanter, D. 2005. Niagara II – The Hydra Returns. DOI = http://www.realworldtech.com/page.cfm?ArticleID=RWT090406012516. Acessado 15/06/2009.

[12] De Gelas, J. 2005. SUN's UltraSparc T1 - the Next Generation Server CPUs. DOI = http://www.anandtech.com/printarticle.aspx?i=2657 Acessado 15/06/2009.

[13] Shankland, S.2006. Sun doubles thread performance with Niagara 2. DOI = http://news.zdnet.co.uk/hardware/0,1000000091,39281614,00.htm Acessado 15/06/2009.

[14] Nakajima A., Kobayashi R., Ando H. and Shimada T. 2006. Limits of Thread-Level Parallelism in Non-numerical Programs. IPSJ Digital Courier, Volume 2. Pages 280-288. May 2006.

[15] Flautner, K., Uhlig, R., Reinhardt, S., and Mudge, T. 2000. Thread-level parallelism and interactive performance of desktop applications. SIGPLAN Not. 35, 11 (Nov. 2000), 129-138. DOI= http://doi.acm.org/10.1145/356989.357001

[16] Jacobs, T.R. Chouliaras, V.A. Mulvaney, D.J. 2006. Thread-parallel MPEG-2, MPEG-4 and H.264 video encoders for SoC multi-processor architectures. Transactions on Consumer Electronics, IEEE. Volume 52, Issue 1. Pages 269-275. Feb. 2006