Prefetch-Aware Shared-Resource Management for Multi-Core Systems

Eiman Ebrahimi; Chang Joo Lee; Onur Mutlu; Yale N. Patt

ISCA'11, June 4–8, 2011, San Jose, California, USA.

INTRODUÇÃO

Circuitos integrados com múltiplos processadores internos (CMP) compartilham entre estes processadores grande parte dos subsistemas de memória como: último nível de *cache*, controladores de memória e memórias externas. Quando diferentes aplicativos estão sendo executados por diferentes *cores*, acessos de um *core* à memória podem interferir nos acessos de outros, causando atraso na execução dos aplicativos. Pesquisas recentes propõem técnicas para permitir alto desempenho e distribuição adequada (*fairness*) na utilização destes recursos, no entanto nenhuma delas leva em consideração acessos a memória antecipados (*prefetches*). *Prefetch* é uma técnica de acessar a memória antecipadamente de forma especulativa trazendo dados que o processador deverá utilizar no futuro, buscando minimizar o impacto do tempo de latência destes acessos no tempo de execução dos aplicativos.

OBJETIVO

O artigo analisa três destas técnicas de gerenciamento de recursos compartilhados, demonstrando que não trazem melhoria de desempenho e *fairness* aos CMPs quando aplicadas juntamente com a técnica de *prefetch*,

O artigo propõe então modificações aos algoritmos destas técnicas e demonstra que se adotadas é possível obter tanto a melhoria de desempenho e *fairness* no gerenciamento dos recursos bem como os benefícios da técnica de *prefetch*.

ANÁLISE TEÓRICA

Duas das técnicas analisadas são de agendamento e priorização de acesso à memória (*memory scheduling*).

A técnica PARBS (*paralelism-aware batch scheduling*) realiza esta priorização ao criar pacotes de acessos, incluindo um máximo de demandas por core/banco de memória no pacote, garantido que todos os *cores* sejam atendidos, e criando um novo pacote somente quando todos os acessos do pacote atual tiverem sido realizados.

A técnica NFQ (*network fair queuing*) realiza a priorização determinando o tempo virtual de término de cada demanda de acordo com a largura de banda alocada para cada tarefa e dá maior prioridade as demandas que tiverem o tempo virtual de término mais cedo.

Estas técnicas porém não fazem distinção entre acesso reais e *prefetches* na priorização dos acessos.

O artigo introduz então o conceito de *prefetches* úteis (*accurate prefetches*) que são acessos especulativos que posteriormente são efetivamente utilizados e de *prefetches* não úteis os quais acabam sendo descartados. Ao não distinguir *prefetches* dos acessos reais, os *prefetches* não úteis acabam sendo incluídos na priorização dos recursos compartilhados gerando perda de utilização efetiva e desequilíbrio na distribuição dos mesmos. Para solucionar este problema o artigo propõe que somente demandas reais e *prefetches* úteis sejam considerados na formação de pacotes da técnica PARBS e na priorização de tempo virtual de término mais cedo da técnica NFQ.

Ao se adicionar porém *prefetches* na priorização como sugerido, aplicações com freqüente acesso à memória podem atrasar demasiadamente aplicações de acessos pouco freqüentes. O artigo sugere então que solicitações de acessos de aplicações de baixa freqüência sejam atendidas prontamente (*demanding boosting*) nas técnicas PARBS e NFQ.

Desta forma elas não sofrerão atrasos na sua execução e como elas têm baixa freqüência de acesso à memória não provocarão atrasos freqüentes nas outras tarefas.

A terceira técnica analisada é uma técnica de controle de demanda na fonte FST (*fairness via source*-throttling) do sistema de memória como um todo. Esta técnica calcula um valor estimado de atraso (*slowdown*) de cada *core* devido à interferência de outros *cores* e baseado nestes valor determina qual o *core* que está experimentando o maior atraso e qual *core* esta provocando o maior número de interferências nos acessos à memória. Ao se atingir um limite máximo de atraso para um *core*, o controlador de demanda reduz a taxa de demanda do *core* que causa maior interferência e aumenta a taxa de demanda do *core* que atingiu o limite de atraso máximo.

Nesta técnica, *prefetches* devem ser considerados de maneira diferenciada. Devem ser levados em conta tanto demandas reais como *prefetches* do core gerador de interferência, porém no *core* que sofre a interferência não se deve considerar atrasos nas solicitações de *prefetches* mas somente nas demandas reais, pois atrasar acessos de *prefetch* não necessariamente reduz o tempo de execução das aplicações. Ainda nesta técnica, é sugerida uma coordenação entre o controlador de *prefetch* com o controlador de demanda do *core*. Quando o controlador de *prefetch* decidir que deve reduzir a taxa de solicitação de um core, deve-se somente fazê-lo se o controle de demanda indicar que este *core* é o que mais gera interferência no outros *cores*, caso contrário não é necessário reduzir a taxa de demanda de *prefetch* deste core pois não está causando atraso em outros *cores*.

EXPERIMENTO

Foi utilizado simulador de multi-processadores x86 com 4 *cores* e modelamento do sistema de memória com seguintes parâmetros : core 15 estágios execução fora de ordem, decodificação 4 instruções, emissão e execução 8 micro-instruções, 128 entradas de buffer de re-ordenamento; Fetch de até 2 desvios, 4K entradas de previsão de desvio BTB e 64K entradas de previsão de desvio híbrida; L1 I e L1 D cache 32KB, 4way, 2ciclos,64B linha; cache L2 compartilhada 2M, 16-way, 16 bancos, 20 ciclos, 1 porta 64B linha.

Para a avaliação de desempenho foram utilizados programas do SPEC CPU 2000/2006 sendo agrupados de 4 em 4 para formação de 15 *workloads* misturando programas de acesso freqüente à memória e que geram *prefetches* de maneira intensa com programas de acesso à memória pouco freqüentes.

METRICAS

Para medição de desempenho foram usadas média harmônica e média ponderada de s*peedup*. Em relação à distribuição justa (*fairness*) entre os processadores foram utilizados as medidas de *MaxSlowDown* and *Unfairness*.

CONCLUSÔES

O artigo demonstrou que utilizando a modificações sugeridas obteve-se melhoria nos resultados de desempenho de 11%, 10,9% e 11,3% de um sistema 4 cores utilizando as técnicas de NFQ, PARBS e FST respectivamente e significativa melhoria na distribuição de recursos, obtendo portanto alto desempenho no gerenciamento de recursos compartilhado juntamente com os benefícios advindos do uso da técnica de *prefetch*.