"Software-Controlled Priority Characterization of POWER5 Processor"
Boneti, C.; Cazorla, F.J.; Gioiosa, R.; Buyuktosunoglu, A.; Cher, C.-Y.; Valero, M.
Page(s): 415-426
Digital Object Identifier 10.1109/ISCA.2008.8
ISCA 2008

Carlos Eduardo Seo RA: 008278 MO401A

Resumo

O artigo em questão mostra uma caracterização dos efeitos do controle de prioridades de *threads* via *software* em um processador IBM POWER5. Os autores iniciam o artigo com uma descrição em alto nível do processador, aprofundando-se mais adiante no mecanismo de controle de *threads*: o POWER5 é um processador que, além do controle via *hardware*, apresenta um mecanismo combinado de *hardware*/*software* que controle a taxa de decodificação de instruções com 8 níveis de prioridades, que pode ser usado quando prioridades desiguais entre *threads* são desejadas. Além disso, os autores mostram que o uso desses mecanismos no mundo real é limitado devido à falta de resultados na literatura, sendo essa uma das motivações para o artigo. O artigo atinge seus objetivos provendo uma análise detalhada do efeito das prioridades de threads do POWER5 em um conjunto de *micro-benchmarks*, além de três estudos de caso nos quais é mostrado como o mecanismo de prioridades pode ser usado para melhorar o *throughput*, o tempo de execução e também como uma *thread* em *background* pode ser executada de forma transparente, sem afetar a *thread* principal

Para a análise, os autores tratam o processador como se fosse uma caixa-preta, observando como o desempenho da carga muda conforme a priorização de *threads* varia. O artigo utiliza a metodologia FAME (*FAirly MEasuring Multithreaded Architectures*) para determinar quantas vezes cada *benchmark* deve ser executado. Para diminuir o nível de ruído nos resultados, os experimentos com *single* e *multi-threading* foram executados no segundo *core* do POWER5, ao passo que todos os processos de usuário e IRQ foram isolados no primeiro *core*. Foram criados 15 *benchmarks* sintéticos, divididos em 4 categorias: inteiros, ponto-flutuante, acesso à memória e *branch*. Além dos *benchmarks* sintéticos, foram usados três estudos de caso: no primeiro, foram usados dois pares de *benchmarks* extraídos do SPEC CPU 2000 e 2006 para avaliar a otimização do *throughput* IPC; no segundo, foi usada uma decomposição LU de matrizes geradas por uma FFT, em um problema de análise espectral, para avaliar a otimização do tempo de execução; no terceiro, foi criado um cenário para execução transparente de *threads*, em que as *threads* em *background* podem usar recursos que a *thread* principal não precisa para execução em quase *full speed*. O *kernel* Linux também foi modificado de modo a agir de forma não intrusiva durante os experimentos.

Os resultados dos *benchmarks* mostram que cargas que apresentam muitas operações de alta latência são menos influenciadas pelas prioridades do que aquelas que apresentam operações de baixa latência. Além disso, foi mostrado que é possível usar o mecanismo de prioridades para aumentar o *throughput* geral em até duas vezes em alguns casos específicos, mas com drástica redução do desempenho de *low* IPC das *threads* – em casos menos extremos, é possível chegar em melhoria de até 40% do *throughput*. Por último, foi mostrado que, em vez de usar todo o espectro possível de prioridades, é mais vantajoso usar apenas as prioridades até +/- 2 – as outras prioridades devem ser usadas apenas quando o desempenho de uma das threads não é tão importante. Em relação aos estudos de caso, é mostrado que o mecanismo de prioridades pode ser usado para elevar o *throughput* geral em até 23,7%, reduzir o tempo de execução em até 9,3% e que é possível ter *threads* executando de forma transparente no *background*, desde que essas threads não apresentem alta latência ou alto desempenho.

Os autores concluem que o mecanismo de priorização de *threads* do POWER5 é uma ferramenta bastante poderosa para melhorar diferentes métricas e que esse trabalho abre caminho para uma maior utilização de técnicas combinadas entre *hardware* e *software* que permitam um melhor equilíbrio entre a utilização dos recursos entre *threads*.