ASF: AMD64 Extension for Lock-free Data Structures and Transactional Memory

Jaewoong Chung, Luke Yen, Stephan Diestelhorst, Martin Pohlack, Michael Hohmuth, David Christie e Dan Grossman

(43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Dec. 2010)

Resumo por Maycon Sambinelli – RA 134071

Processadores *multi-core* se popularizaram e hoje estão por toda parte. Porém nem todo o potencial dessa arquitetura é utilizado devido às dificuldades da programação paralela. Para tentar resolver esse problema surgiu a Memória Transacional (MT), um mecanismo onde o programador precisa apenas definir *blocos* que serão executados atomicamente, mas que necessita de algum tipo de aceleração por *hardware*.

Os autores do artigo projetaram uma extensão para a arquitetura AMD64 chamada *Advanced Synchronization Facility* (ASF), que provê regiões especulativas que executam acessos especulativos atomicamente. Essas regiões são utilizadas para construir primitivas atômicas multi-palavra para a programação *lock-free* e para prover a primeira geração de aceleração de *hardware* para memória transacional.

O ASF conta com a adição de 5 novas instruções: 1) *SPECULATE*, que inicia uma região especulativa; 2) *LOCK MOV*, que move dados entre os registradores e a memória; 3) *RELEASE*, que é uma dica para remover o isolamento de um *load* especulativo de um determinado endereço de memória; 4) *ABORT*, que reverte as alterações de uma região especulativa; e 5) *COMMIT*, que termina uma região especulativa. As regiões especulativas são criadas utilizando-se as filas *load/store* e a cache L1 como *buffers* para dados especulativos. Para sinalizar dados especulativos, cada entrada das filas recebe 1 bit, e cada linha da cache recebe 2 bits. O protocolo *Coherent HyperTransport* é utilizado para detectar conflitos de acesso à memória especulativa/não especulativa comparando mensagens que chegam do protocolo de coerência de cache contra esses bits. Se um conflito é detectado, a região especulativa é abortada, descartando-se os dados especulativos e limpados esses bits. Se a instrução *COMMIT* é atingida, os dados especulativos dos *buffers* são efetivados limpando-se o grupo de bits, e enviando os dados da fila de *store* para a cache L1.

Para avaliar o desempenho, os autores utilizaram um simulador interno da AMD e escreveram 4 versões do ASF: 1) ASF-LC, que usa tanto a cache L1 quanto a unidade LS como *buffer* para dados especulativos; 2) ASF-C8, que usa apenas a cache L1 *8-way* associativa como *buffer*; 3) ASF-C4, que usa uma cache L1 *4-way* associativa; e 4) ASF-AS, que é igual ao ASF-LC, porém não suporta o uso da instrução *LOCK MOV* e trata todos os acessos a regiões especulativas como acessos especulativos. Uma região especulativa pode esgotar os *buffers* de dados especulativos; nesse caso ocorre uma exceção de *overflow*. Nos testes, o manipulador de exceções usa um mecanismo de *software*, que trata o problema da seguinte maneira: 1) espera todos os outros núcleos saírem de suas regiões especulativas; 2) impede que outros núcleos entrem em regiões especulativas; 3) executa uma versão não especulativa da região especulativa que sofreu *overflow*; e 4) permite que os outros núcleos voltem a entrar em regiões especulativas.

O ASF foi testado utilizando programação *lock-free* e transacional. O teste utilizando programação *lock-free* teve por intenção avaliar como o ASF lida com um conjunto de dados randômicos. Uma tabela *hash* que trata os conflitos com uma lista ligada teve as chaves inseridas randomicamente por diversas *threads* até o primeiro *overflow* de capacidade. Os resultados mostraram que o ASF-LC pode conter 12,3x mais blocos do que ASF-C8 e 63,9x mais blocos do que o ASF-C4. Os testes transacionais foram utilizados para avaliar o *speedup*. Foram utilizados 3 grupos de aplicações do *benchmark* STRAMP, categorizados de acordo com suas escalabilidades. O primeiro grupo consistia das aplicações *bayes*, *labyrinth* e *yada*. Os resultados mostraram que o *ASF-AS* escala pobremente em comparação ao ASF-LC/C8/C4, devido ao fato dele não suportar a instrução *LOCK MOV*, que habilita o uso seletivo do acesso especulativo para reduzir o *footprint* da memória especulativa. O segundo grupo era formado pelas aplicações *intruder*, *kmenans* e *SSCA2*; obtiveram um resultado muito parecido, pois todas possuíam apenas transações de vida curta, e, desta forma, elas não tiraram vantagem do uso das instruções *LOCK MOVs*, nem de um conjunto associativo maior ou do uso das filas como um *buffer* especulativo. O terceiro grupo contém os problemas *Genome* e *Vacation*, que escalam melhor com ASF-LC e ASF-C8. Analisando esses resultados, os autores concluíram que o uso da unidade LS e L1 como *buffers* especulativos contribui muito para a robustez do sistema, e que o uso seletivo de acesso especulativo melhora sua escalabilidade.