Artigo: Cooperative Caching for Chip Multiprocessors

Citação Bibliográfica: ACM SIGARCH Computer Architecture News, Volume 34, Issue 2 (May 2006), Pages: 264 – 276, 2006

Autor / RA: Gabriela de Gáspari Martins / 047146

O artigo analisado apresenta o CMP *Cooperative Caching*, uma proposta para organizar a hierarquia *on-chip*, a fim de obter os benefícios dos *designs* de cache privado e compartilhado. O conceito básico dessa proposta é inspirado pelos algoritmos de software de cache cooperativo – caches individuais compartilham espaço, de forma cooperada, para alcançar seus requisitos de capacidade dinâmica, reduzindo o tráfego para os servidores remotos.

É apresentado o conceito de cache cooperativo para um CMP com 04 núcleos, onde cada cache L2 local dos núcleos está fisicamente próximo a eles, e somente o processador pode acessá-los diretamente. As falhas dos caches L2 locais podem ser tratadas por caches *on-chip* remotos através da transferência cache-a-cache. O cache cooperativo irá tentar fazer com que o efeito dos caches L2 privados pareça com um cache coletivo compartilhado através das ações cooperativas.

Nesse cenário, a otimização da latência total de uma requisição de memória é obtida combinando caches privados e compartilhados, através de 03 maneiras: (1) utilizando caches privados como a base da organização; (2) utilizando cooperação entre caches privados; (3) controlando o tamanho da cooperação. Além disso, são apresentadas as seguintes políticas para cooperação: (1) *Cache-to-cache Transfers of Clean Data:* facilita a transferência cache-a-cache de blocos *on-chip* limpos para eliminar acessos *on-chip* desnecessários aos dados que já estão em algum lugar do chip; (2) *Replication-aware Data Replacement:* troca blocos de dados replicados para dar lugar a cópias *on-chip* únicas (denominadas *singlets*), fazendo melhor uso dos recursos de cache *on-chip*; (3) *Global Replacement of Inactive Data:* permite que blocos *singlets* que foram evitados por um cache L2 local sejam armazenados em outro cache L2, possivelmente tomando o *slot* do cache de um bloco de dados inativo, mantendo os dados mais utilizados no chip.

Na estrutura *on-chip* apresentada para um CMP de 04 núcleos inclui-se o CCE (Central Coherence Engine), que abrange o diretório de memória e a ferramenta de coerência, e sua proximidade com todos os caches pode reduzir a latência da rede. O diretório de memória de um CCE é organizado como uma duplicação de todos os conjuntos de caches privados. As *tags* são indexadas da mesma maneira como em um núcleo individual. O diretório de memória é *multi-banked* utilizando *tag bits* de baixa ordem para providenciar alto *throughput* e um diretório de *lookup* é direcionado ao banco correspondente para procurar todos os caches relacionados em todos os conjuntos de *tag*. Tais resultados são usados para formar um vetor de presença/estado como em uma implementação de diretório convencional.

Outro fato é que o cache cooperativo requer que caches L1 sejam não-inclusivos com os caches L2 locais. Por isso, o CCE tem que duplicar as *tags* para todos os níveis de cache, que são atualizadas para refletir quais blocos estão armazenados em quais núcleos dos caches privados, e quais são seus estados.

Para suportar a transferência *on-chip* cache-a-cache de dados limpos, o CCE seleciona um dono limpo para uma requisição. Procurando pelo diretório de memória, qualquer cache pode ser escolhido. Por outro lado, o estado do diretório CCE deve ser atualizado quando uma cópia limpa é resgatada. Essa requisição é atingida estendendo o protocolo de coerência da *baseline* do cache com uma transação “PUTS”. Ao receber esse tipo de requisição, o CCE atualiza o estado do seu diretório.

Na implementação corrente, caches privados e CCE cooperam para implementar o *spilling push-based*, que consiste de 02 transferidores de dados. O primeiro é um escritor normal iniciado pelo cache privado, enviando o bloco para o CCE, que temporariamente armazena o dado e avisa quem o enviou. O segundo transfere o bloco do CCE para um cache escolhido randomicamente. O cache que recebe o bloco irá avisar ao diretório para liberar o local de armazenamento.

Os testes realizados mostram que o cache cooperativo pode reduzir o *runtime* de cargas de trabalho simuladas de 4-38%, e sua performance é 2.2% menor que os melhores caches privados e compartilhados em casos extremos. O cache cooperativo também melhora a performance do esquema de replicação de vítimas em 9% considerando uma mistura de *multi-thread*, *thread-simples* e cargas de trabalho programadas.