Relatório - Projeto 3
MORC: A Manycore-Oriented Compressed Cache

Paulo Henrique Junqueira Amorim - RA: 095431

Resumo. *Compressão de cache tem sido bastante utilizada para melhorar a performance de aplicações sigle-stream. Esse trabalho tenta reproduzir o artigo intitulado MORC: A Manycore-Oriented Compressed Cache. Serão apresentados conceitos breves sobre o artigo, bem como resultados e dificuldades encontradas.*

Introdução

A computação orientada a *throughput* está se tornando cada vez mais importante, na qual o foco é o processamento paralelo de grandes massas de dados. Exemplos emergentes de throughtput ocorrem em grandes data centers com aplicações map-reduce, computação cientifica como simulações, processamento gráficos entre outros. Geralmente essas aplicações são tolerante a latência uma característica muitas vezes utilizada para uma melhor eficiência energética. Um grande desafio com as futuras arquiteturas *manycore* é a largura de banda das memórias fora do chip de processamento (*off-chip*) para alimentar o número crescente de *threads* e núcleos em um único chip. Um segundo desafio é que os processadores orientados a *throughput* precisam enfrentar o grande custo de energia associado ao acesso à memória fora do chip de processamento. [Nguyen and Wentzlaff 2015].

O trabalho intitulado *MORC: A Manycore-Oriented Compressed Cache*, explora a compressão de cache como um meio para aumentar a quantidade de dados que podem ser inseridos em uma cacha e consequentemente aumentar o *throughput*. Para aumentar a taxa de compressão foi utilizada técnicas de compressão intra-line e inter-line. Para isso é utilizada uma arquitetura chamada MORC (*Manycore ORiented, compressed Last-Level Cache (LLC)*) que é uma arquitetura baseada em log. Ela permite comprimir linhas de cache similares juntas no caso da técnica inter-line.

Em uma cache baseada em log os dados são localizados baseados na associação de seus padrões, independente de seus endereços. A arquitetura MORC não usa logs somente para guardar dados comprimidos, mas também comprimi os tags junto usando compressão "base-delta". Para aumentar a compressão inter-line, MORC utiliza um novo algoritmo de compressão, o Large Block Encoding (LBE), que dinamicamente procura granularidade de palavras (32, 64, 128 ou 256 bits) para altas taxas de compressão. Um outro motivo para a utilização de logs é para manter o dicionário de palavras para compressão.

Objetivo

O objetivo do trabalho foi reproduzir o gráfico presente na figura 2.a do artigo, conforme é exibido na figura 1. Esse gráfico apresenta a taxa de ganha de compressão em relação ao tamanho original.

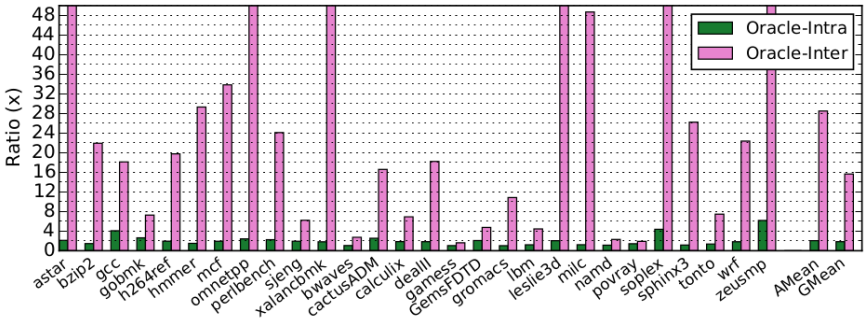

Figura 1. Gráfico a ser reproduzido.

Resultados experimentais

Para a reprodução do artigo foi utilizado o exemplo **allcache.cpp** presente no Intel pintool, foi necessário ler o conteúdo dos endereços no momento de cache miss nos níveis L1, L2 e L3, assim como as leituras para se determinar o tamanho ocupado de cada linha de cache e também enviar para a função que realiza a compactação. O código que realiza a compactação foi cedido pelo autor do artigo.

Foi utilizada a classe **CompressionLBE2** com o método **incrementalCompress** na opção de compressão *inter-line*, na opção *intra-line* foi utilizado o método **reset** em cada leitura, esse método realiza o reset do dicionário, dessa forma é levada em consideração apenas uma linha de cache. Segundo o autor para simular a cache ele utilizou o PrimeSim (http://primesim.princeton.edu/) mas no caso, esse simulador utiliza uma versão antiga do pintool, dessa forma optou-se por não utilizar.

Os programas de benchmark utilizados foram do SPEC2006, os mesmos utilizados no artigo. O autor utilizou simpoints e na reprodução também foi tentada a utilização mas por algum motivo o código falhava apresentando *segmentation fault* e por esse motivo a execução foi utilizada em modo *test* do SPEC2006, pois o modo *ref* também ocorreu o mesmo problema.

Na figura 2 é possível observar os resultados obtidos, entretanto somente alguns resultados do método *inter-line* (barra rosa do gráfico) acompanharam a proporção do gráfico original, como os programas astar, bzip2, cactusADM, namd e zeusmp. Já o intra-line (barra verde do gráfico) nenhum programa apresentou o resultado como o esperado do gráfico original. É possível observar na tabela 1 os dados utilizados para gerar o gráfico.

Para calcular a taxa de ganho do gráfico, foi utilizada a seguinte fórmula:

$$ratio = 100 - \frac{size_{compressed}}{size_{original}} * 100 \quad (1)$$

Conclusão

Foi seguida as orientações do autor do trabalho original, entretanto foi necessário utilizar um simulador diferente do utilizado e o modo SPEC2006 utilizado nas execuções foram diferentes, podendo isso ter afetado o resultado do trabalho. Entretanto foi possível entender o funcionamento de um sistema de compressão de cache e a importância de detalhar as ferramentas utilizadas em um artigo.

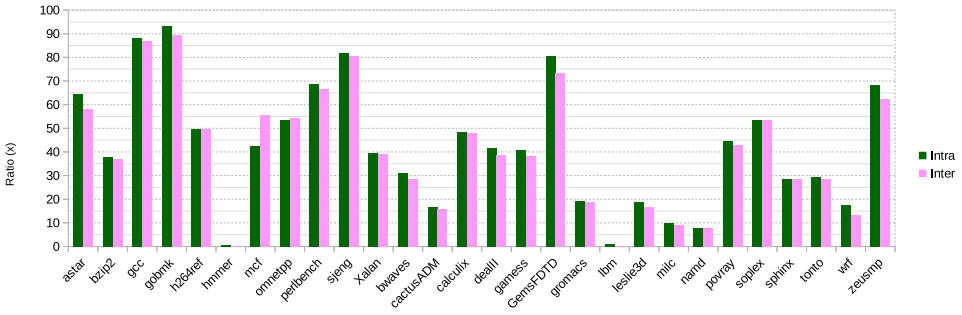

Figura 2. Gráfico produzido.

Tabela 1. Tamanho total de dados que passaram pela cache não comprimido e comprimido.

	Tamanho em bits		
Programa	**Original**	**Intra-line**	**Inter-line**
astar	39665108040	14047359141	16591646373
bzip2	7099706680	4417946061	4463728809
gcc	18963802568	2204535281	2445505472
gobmk	6707539296	442120308	716901009
h264ref	54337908120	27263381301	27250453189
hmmer	19616942296	19480562299	19553119137
mcf	71876774664	41323097966	31919114758
omnetpp	1440845056	671763886	660001409
perlbench	25159400	7881582	8362054
sjeng	16743600328	3060257285	3264864602
Xalan	1079459968	653154013	659003299
bwaves	125621459592	86417344266	89818944003
cactusADM	17490003816	14549950684	14718906045
calculix	131827824	68256483	68739212
dealII	94743881680	55090632535	58113485908
gamess	69228208	41076024	42589715
GemsFDTD	32751761440	6393223211	8711693268
gromacs	3641523800	2933891695	2957411365
lbm	84866751824	83949959661	84683286085
leslie3d	404011554856	327294797258	336451804102
milc	197714105272	177846153029	180009116505
namd	67286368920	62153875749	62102853581
povray	6151763088	3395923250	3505311275
soplex	223792264	103966943	104426190
sphinx	12052207448	8626300187	8598057808
tonto	2691832432	1896429137	1918274180
wrf	24902169272	20536288321	21638229102
zeusmp	165551091168	52230672826	62390652893

Referências

Nguyen, T. M. and Wentzlaff, D. (2015). Morc: A manycore-oriented compressed cache. In *Proceedings of the 48th International Symposium on Microarchitecture*, MICRO-48, pages 76–88, New York, NY, USA. ACM.