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

Paulo Henrique Junqueira Amorim - RA: 095431

Resumo. *Compressão de cache tem sido utilizada para melhorar a performance de aplicações que requerem altas taxas de throughput. Esse trabalho tenta reproduzir o artigo intitulado MORC: A Manycore-Oriented Compressed Cache que apresenta um método de compressão de cache. Serão apresentados breves conceitos sobre o artigo estudado, 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. É afirmado no artigo que 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 cache 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. Para aumentar a compressão inter-line, MORC utiliza o algoritmo de compressão 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.

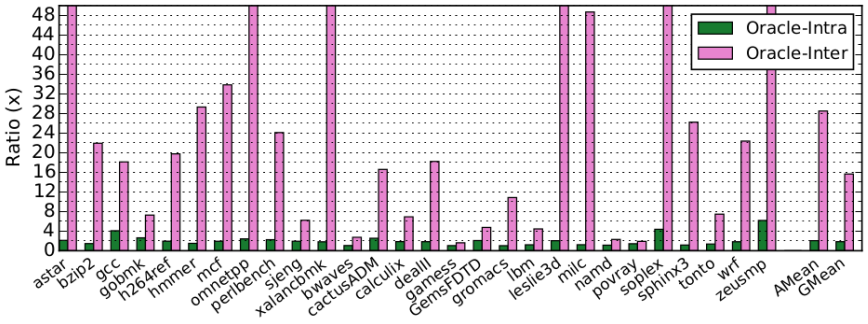

Figura 1. Gráfico a ser reproduzido.

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, na maioria dos programas executados o método *inter-line* teve um ganho em relação ao método *intra-line*, entretanto o ganho não foi expressivo como é apresentado no artigo original. É possível observar na tabela 1 os dados utilizados para gerar o gráfico, esses valores são a média de todas as execuções de um mesmo programa com múltiplas entradas.

Para calcular a taxa de ganho presente no 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.

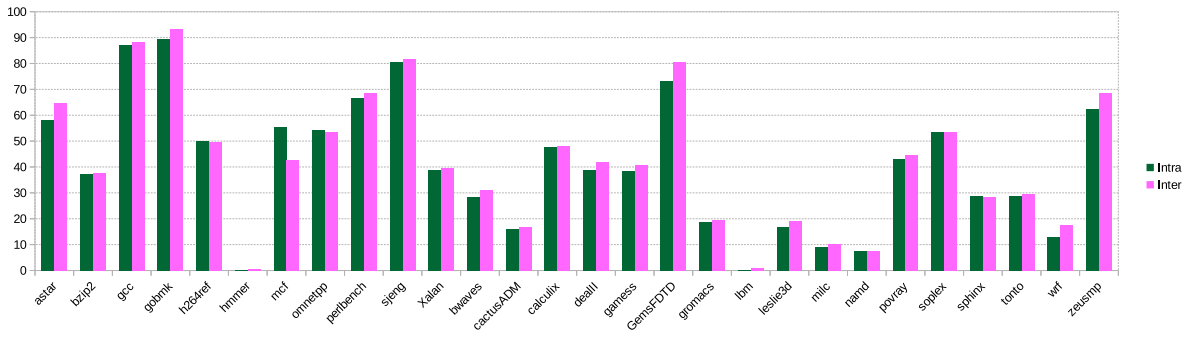

Figura 2. Gráfico produzido.

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

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.