Relatório - Projeto 4
Uso do algoritmo LZ4 para compressão de cache

Paulo Henrique Junqueira Amorim - RA: 095431

Resumo. *Esse trabalho emprega o uso do método de compressão LZ4 em caches de microprocessadores, os resultados são comparados com os obtidos no trabalho anterior que tenta reproduzir o artigo intitulado MORC: A Manycore-Oriented Compressed Cache. Serão apresentados conceitos breves sobre o artigo, bem como resultados e conclusão.*

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.

Nesse trabalho é utilizado o algoritmo de compressão LZ4 [blog on compression algorithms 2016], a vantagem que esse algoritmo atinge altas taxas de compressão em uma velocidade maior que o gzip.

No algoritmo LZ4 o bloco comprimido é composto por seqüências, onde cada seqüência começa com um token. O token é um valor de um byte, separado em dois campos de 4 bits (que vão de 0 a 15). O primeiro campo usa os 4 bits mais significativos do token e indica o comprimento dos literais. Se for 0, então não há literal. Se for 15, então é necessário adicionar mais alguns bytes para indicar o comprimento total. Cada byte adicional então representa um valor de 0 a 255, que é adicionado ao valor anterior para produzir um comprimento total. Quando o valor do byte é 255, outro byte é emitido [blog on compression algorithms 2016].

Objetivo

O objetivo do trabalho foi tentar melhorar a taxa de compressão obtida no trabalho 3, para isso foi empregado o algoritmo LZ4.

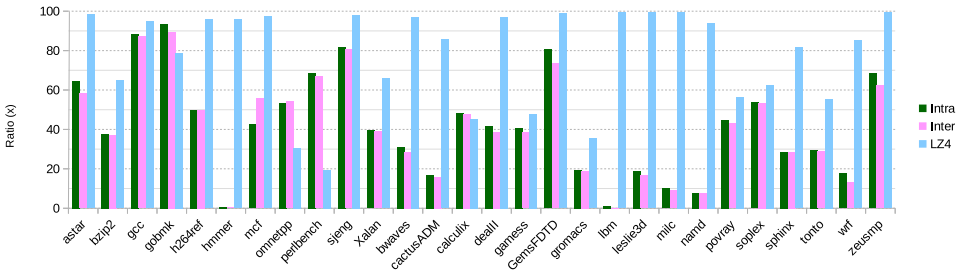

Figura 1. Gráfico com dados do trabalho 3 e trabalho 4 (barra em azul)

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.

A implementação utilizada do LZ4 está presente no link (https://github.com/lz4/lz4), foi necessário compilar o código e "linkar"estaticamente no makefile do allcache.cpp. Cada item lido da cache foi enviada para a função **LZ4_compressBound**

O benchmark utilizado foi o mesmo do trabalho 3, ou seja, programas do SPEC2006 sendo executados em modo test. É possível observar no gráfico presente na imagem 1 o comparativo entre os métodos inter-line e intra-line do trabalho 3 com o método LZ4 do trabalho 4. Na tabela 1 é possível verificar os dados utilizados para calcular 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

O método de compressão LZ4 utilizado para comprimir dados da cache apresentou ótimos resultados quando comparado com os método intra-line e inter-line, para trabalhos futuros é necessário explorar o método com o modo "ref" do SPEC2006. Os resultados mostram uma boa alternativa para compressão de cache visto que o algoritmo LZ4 possuí velocidade de descompactação e compactação superior aos algoritmos baseados em gzip.

Referências

blog on compression algorithms, D. (2016). Development blog on compression algorithms (tinyurl.com/qc9yve4). acessado em 04/12/2016.

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.

Tabela 1. Tamanho total de dados que passaram pela cache não comprimido e comprimido, incluindo o método LZ4.

	Tamanho em bits			
Programa	**Original**	**Intra-line**	**Inter-line**	**LZ4**
astar	39665108040	14047359141	16591646373	672259626
bzip2	7099706680	4417946061	4463728809	2502121044
gcc	18963802568	2204535281	2445505472	971787444
gobmk	6707539296	442120308	716901009	1422999212
h264ref	54337908120	27263381301	27250453189	2248579500
hmmer	19616942296	19480562299	19553119137	831419835
mcf	71876774664	41323097966	31919114758	1862350306
omnetpp	1440845056	671763886	660001409	999840709
perlbench	25159400	7881582	8362054	20305260
sjeng	16743600328	3060257285	3264864602	322173358
Xalan	1079459968	653154013	659003299	368690034
bwaves	125621459592	86417344266	89818944003	3589963860
cactusADM	17490003816	14549950684	14718906045	2499485842
calculix	131827824	68256483	68739212	72150867
dealII	94743881680	55090632535	58113485908	2758960402
gamess	69228208	41076024	42589715	36120063
GemsFDTD	32751761440	6393223211	8711693268	395450122
gromacs	3641523800	2933891695	2957411365	2353809810
lbm	84866751824	83949959661	84683286085	518039451
leslie3d	404011554856	327294797258	336451804102	2996653540
milc	197714105272	177846153029	180009116505	699599364
namd	67286368920	62153875749	62102853581	4081630174
povray	6151763088	3395923250	3505311275	2682138347
soplex	223792264	103966943	104426190	83563044
sphinx	12052207448	8626300187	8598057808	2225917242
tonto	2691832432	1896429137	1918274180	1199467929
wrf	24902169272	20536288321	21638229102	3710377355
zeusmp	165551091168	52230672826	62390652893	1003841909