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.

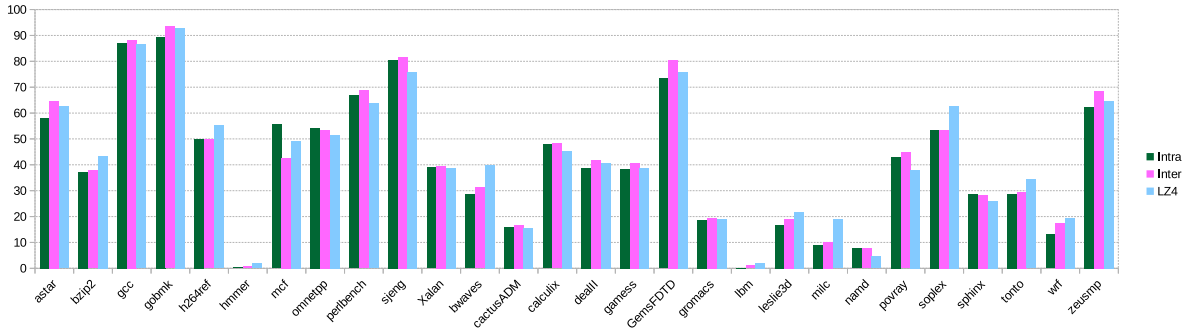

Figura 1. Gráfico com dados do trabalho 3 e trabalho 4 (barra em azul)

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 de compressão presente no 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.

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	16591646373	14047359141	14857609022
bzip2	7099706680	4463728809	4417946061	4016734101
gcc	18963802568	2445505472	2204535281	2555684589
gobmk	6707539296	716901009	442120308	484585257
h264ref	54337908120	27250453189	27263381301	24334674523
hmmer	19616942296	19553119137	19480562299	19244586239
mcf	71876774664	31919114758	41323097966	36453544121
omnetpp	1440845056	660001409	671763886	699145421
perlbench	25159400	8362054	7881582	9145113
sjeng	16743600328	3264864602	3060257285	4054054354
Xalan	1079459968	659003299	653154013	661245478
bwaves	125621459592	89818944003	86417344266	75524545678
cactusADM	17490003816	14718906045	14549950684	14756111214
calculix	131827824	68739212	68256483	72150867
dealII	94743881680	58113485908	55090632535	56124992547
gamess	69228208	42589715	41076024	42443212
GemsFDTD	32751761440	8711693268	6393223211	7881147379
gromacs	3641523800	2957411365	2933891695	2944877563
lbm	84866751824	84683286085	83949959661	83220257456
leslie3d	404011554856	336451804102	327294797258	316510542344
milc	197714105272	180009116505	177846153029	160547411254
namd	67286368920	62102853581	62153875749	64234544725
povray	6151763088	3505311275	3395923250	3815645256
soplex	223792264	104426190	103966943	83563044
sphinx	12052207448	8598057808	8626300187	8922542135
tonto	2691832432	1918274180	1896429137	1766420154
wrf	24902169272	21638229102	20536288321	20054654789
zeusmp	165551091168	62390652893	52230672826	58456544577

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.