FADA-CHL: FERRAMENTA PARA ANÁLISE DE DESEMPENHO E COMPARAÇÃO DOS ALGORITMOS DE COMPACTAÇÃO SEM PERDA HUFFMAN E LZW

Bruno Casemiro Kurzawe[1], Christine Vieira Scarpato[2]

[1]Curso de Ciência da Computação - Universidade do Extremo sul Catarinense (UNESC) - Criciúma - SC - Brasil

[2]Departamento de Ciência da Computação - Universidade do Extremo sul Catarinense (UNESC) - Criciúma - SC - Brasil

bruno.kurzawe@hotmail.com,cvs@unesc.net

Resumo. *A área de compactação de dados é uma das áreas mais antigas da computação, em 1950 já se estudavam métodos para isso. Nas décadas de 80 e 90 essa área teve um grande crescimento com o desenvolvimento de algoritmos que utilizamos até hoje como Huffman, LEMPEL-ZIV & WELCH LZW, Burrows-Wheeler Transform BWT e Prediction by Partial Matching PPM. Após ler sobre o assunto, e descobrir que compressores atuais como WinZip e WinRar utilizam esses algoritmos nos interessamos a conhecer melhor seu funcionamento, e através de técnicas de análise de desempenho, como a taxa de compactação, velocidade de compactação e uso da memória, descobrir o melhor algoritmo para cada tipo de arquivo. Neste trabalho implementamos uma ferramenta chamada FADA-CHL que implementa o algoritmo de HUFFMAN e o algoritmo de LZW, essa ferramenta analisa os resultados para os arquivos do benchmark da Canterbury Corpus desenvolvido pela universidade de CanterBury da Nova Zelandia. Durante o desenvolvimento da ferramenta obtivemos problemas com dois arquivos para analise por causa do tamanho desses arquivos.Através da ferramenta podemos concluir que quando aplicado a tabela CanterBury Corpus o algoritmo de HUFFMAN é mais eficiente devido a constituição dos arquivos.*

Palavras-chave*: Compactação de dados, algoritmos de compactação, análise de desempenho, taxa de compactação e tempo de execução.*

1. Introdução

Com a expansão da transmissão de dados em todo o mundo, o crescimento das redes de computadores e a expansão da internet, todos os dias surgem novas técnicas de processamento de dados a fim de encontrar melhores maneiras de transmiti-los ou armazená-los. HELD (1992) e FRANÇA NETO (1998) afirmam em suas obras que ao mesmo tempo em que a capacidade de transmissão e armazenamento cresce, as informações a serem transmitidas e armazenadas crescem mais rápido ainda, desta forma a compactação de dados se faz útil e muito necessária.

Ao citarmos compactação e transmissão de dados logo consideramos que sua aplicação é muito importante em arquivos de imagem e vídeo, pois nesse tipo de arquivo encontramos uma quantidade grande de informação, tornando-os arquivos de tamanho elevado e de alto custo para sua transmissão e armazenamento. Para isso buscamos na compactação de dados uma maneira de solucionar esse problema. Dentro da compactação de dados abordamos dois tipos, compactação com perda e compactação sem perda de informações. Neste projeto o foco será a aplicação dos algoritmos de compactação sem perda, será feita uma análise comparativa entre os dois principais algoritmos utilizados nessa área, que são o algoritmo de HUFFMAN e o algoritmo LEMPEL-ZIV & WELCH LZW que é chamado de LZW.

2. Compactação de Dados

De acordo com FRANÇA NETO (1998) a compactação de dados tem como intuito tentar reduzir ao máximo o espaço ocupado em um dispositivo de armazenamento de dados, por exemplo, discos rígidos ou pendrives. Esse tipo de operação é feita através de algoritmos de compactação de dados no qual o objetivo é reduzir a quantidade de bytes utilizados para representar determinados tipos de arquivos. Essa compactação de dados tem como objetivo reduzir a redundância de informação utilizada para representar um arquivo. Na figura 1 temos um esquema que FRANÇA NETO (1998) utilizou para representar a compactação e descompactação de determinado arquivo "D" onde este arquivo é comprimido se tornando o arquivo "C" e depois descomprimido voltando a ser o arquivo "D".

A partir das definições HELD (1992) podemos dizer que atualmente a compactação de dados está presente nas aplicações de comunicação, como redes de computadores, onde ela é utilizada em protocolos de comunicação para reduzir as informações na hora de enviá-las. Vários padrões de compactação estão presentes no nosso dia-dia, tais como o ZIP, RAR entre outros. A compactação de dados possui inúmeras aplicações podendo ser utilizada tanto no nível de hardware quanto no nível de software.

Dentre as técnicas de compactação de dados podemos classificá-las em compactação com perda e compactação sem perda.

HELD (1992) afirma que algoritmos de compactação com perda são conhecidos como quantizadores, neste tipo de compactação o dado original é processado através de um método de quantização, permitindo altas taxas de redução do tamanho original. Porém alta taxa de redução de tamanho tem um custo, que seria a perda de informação em relação ao dado original, o arquivo perde a fidelidade em relação ao arquivo original.

Seguindo as afirmações de HELD (1992) podemos dizer que algoritmos de compactação sem perda são conhecidos como codificadores universais, esse tipo de codificador mantém a fidelidade do arquivo em relação ao arquivo de origem. Porém essa fidelidade tem um custo, em que as taxas de redução são diminuídas, ou seja, quando utilizamos compressores que utilizam métodos sem perda obtemos taxas de compactação menores, os arquivos compactados possuem tamanho modestamente menor. Isso se dá devido ao limite entrópico, o limite entrópico é a quantidade de caracteres diferentes presentes dentro da cadeia de símbolos, a entropia pode ser caracterizada como a quantidade real de dados de uma informação existente.

3. Algoritmo de Huffman

O algoritmo de Huffman é segundo FRANÇA NETO (1998) um dos algoritmos de compactação sem perda mais conhecido, inicialmente este método foi aplicado à compactação de arquivos de texto onde buscava obter uma taxa de compactação ótima. David Huffman desenvolveu esta técnica que visava utilizar árvores binárias a fim de gerarem códigos binários. Dentro desta técnica se suponha que o arquivo de texto a ser compactado fosse composto por uma seqüência de caracteres ou símbolos, conhecendo a freqüência com que cada símbolo aparece seria possível atribuir um valor para cada símbolo através do número de ocorrências, desta forma compactando o arquivo inteiro. FRANÇA NETO (1998, 67p) afirma em sua obra que, "O algoritmo de Huffman é um aperfeiçoamento do algoritmo de Shannon-Fano, desenvolvido em 1952 por David Huffman. Desde então, surgiram diversas variações do algoritmo", apenas o algoritmo de Huffman foi desenvolvido por David Huffman, o Shannon-Fano foi desenvolvido por por Claude Shannon e Robert Fano.

Seguindo as afirmações do autor o algoritmo de Huffman revolucionou a compactação quando foi criado, se tornou um dos algoritmos mais conhecidos sendo implementado por universidades pelo mundo todo. Seguindo a mesma lógica do algoritmo de compactação de Shannon-Fano o algoritmo de Huffman tem como objetivo principal a codificação dos símbolos que aparecem com uma freqüência maior com um número menor de bits e aqueles que aparecem com menos freqüência com um número maior de bits. Com isso podemos afirmar que as taxas de compactação através desse algoritmo podem ser maiores ou menores dependendo da probabilidade da distribuição dos símbolos dentro de um arquivo. A compactação através deste algoritmo é dada através de dois passos, no primeiro determinamos a forma que a probabilidade é distribuída entre os símbolos da fonte, utilizamos essa distribuição para a geração da tabela de códigos. No segundo passo é feita uma varredura na tabela gerada anteriormente, desta forma determinamos a codificação e as fontes reduzidas, apões determinarmos as codificações a serem utilizadas o arquivo por fim é compactado.

Segundo FRANÇA NETO (1998) a diferença entre o algoritmo de Shannon-Fano e o de Huffman está basicamente na maneira em que a árvore binária é construída, a árvore de decisão de Huffman não é gerada baseando-se em divisões de dois grupos de símbolos que é utilizada pelo algoritmo de Shannon-Fano, no qual a soma das probabilidades tem que ser igual ou semelhante, mais sim na soma das menores freqüências de símbolos encontradas nos arquivos de entrada.

4. Algoritmo de LZW

De acordo com FRANÇA NETO (1998) com o surgimento da compactação baseada em dicionário de dados, também conhecida como cadeia de compactação, a compactação de dados teve uma revolução por todo o mundo, a maioria dos aplicativos de compactação utilizados hoje utilizam algum algoritmo da família LZW criados por Lempel-Ziv & Tery Welch.

Segundo o autor o algoritmo de Lempel-Ziv & Welch é uma variação do algoritmo Lempel-ziv32, este algoritmo foi desenvolvido pelo inglês Tery Welch em 1984, a contribuição de Tery Welch no desenvolvimento do aprimoramento do algoritmo de Lempel-Ziv32 foi tão significativa e importante que existe uma tendência

de se desconsiderar o algoritmo original desenvolvido por Tery Welch, por causa da melhora significativa que ele gerou ao algoritmo.

De acordo com WELCH (1984) o aprimoramento do método trabalha da seguinte forma: O algoritmo de Lempel-Ziv32 conhecido com LZ77 utilizava uma janela deslizante para codificar os blocos de caracteres iguais a fim de gerar um código de saída e esse código é o próprio dicionário de strings. O algoritmo trabalha direcionado aos blocos de caracteres mais recentes localizados dentro da janela. Por esse problema Tery Welch no ano seguinte abandonaram o conceito de janela deslizante e criaram o LZ7833 e começaram a utilizar o dicionário como uma lista ilimitada de blocos, onde os novos blocos, ou novas frases são inseridas em um dicionário e nas próximas vezes que esse bloco aparecer na fonte é gerado um código que as representa no dicionário.

Mesmo Terry Welch sendo o criador do método de compactação LZW ele não é o proprietário da patente, de acordo com a internet espanhola, na época que ele desenvolveu o algoritmo ele trabalhava como pesquisador na Sperry, uma empresa americana que adquiriu os direitos do algoritmo de LZW em 1981, a empresa patenteou o algoritmo, a empresa se fundiu com a Burroughs posteriormente. Atualmente a dona da patente é a UNISYS e ela delimita que todos precisam comprar uma licença para utilizar esse método em aplicações comerciais.

5. Desenvolvimento da Ferramenta

A ferramenta desenvolvida foi dividida em duas partes, onde na primeira parte tem-se a Interface e na segunda o núcleo funcional, o núcleo funcional será a parte interna do sistema que possuirá as funções de compactação e análise de desempenho, esse núcleo não possui telas, a Interface possui quatro telas. O usuário interage com o sistema através da Interface, nela seleciona os arquivos a serem comprimidos pelo compressor, seleciona as opções de análise de desempenho, seleciona quais os algoritmos serão utilizados, seleciona se irá compará-los, todas as operações a serem solicitadas na interface serão realizadas pelo núcleo principal.

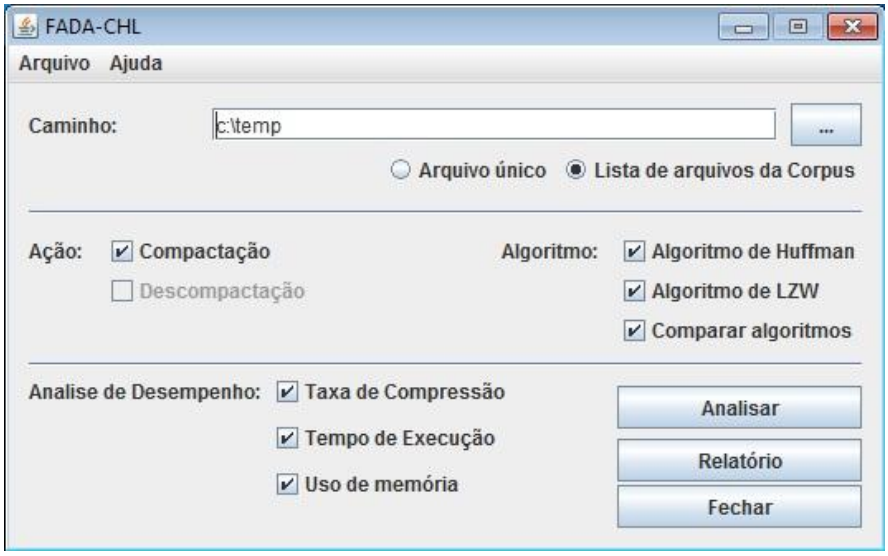

Figura 1. Tela inicial do sistema

Na figura 1 é mostrada a janela principal do sistema, onde pode-se encontrar as opções mencionadas anteriormente, parâmetros para análise de desempenho, abertura de arquivos e emissão do relatório de comparação.

Neste software foi feita a comparação dos métodos de Huffman e LZW, esses resultados também foram comparados com os de uma ferramenta de compactação, o WINRAR na versão 3.93. Esta comparação teve como base analisar certos parâmetros de desempenho como a taxa de compactação, tempo de execução e taxa de uso da memória.

Foi utilizado como benchmark o CanterBury Corpus para a comparação, pois é a principal coleção de arquivos utilizados para a comparação de algoritmos de compactação, essa coleção esta disponível na pagina na internet www.corpus.canterbury.ac.nz, os arquivos presentes nessa coleção possuem uma característica de possuírem resultados típicos atualmente nos algoritmos de compactação.

O Corpus Canterbury é uma referência para permitir que pesquisadores para avaliar métodos de compactação sem perdas. Corpus Canterbury, um substituto para o Corpus Calgary. O conjunto de arquivos Canterbury Corpus vem sendo desenvolvido especificamente para testar novos algoritmos de compactação. Os arquivos foram selecionados com base em sua capacidade de fornecer resultados de desempenho representativos.

Tabela 1. Relação de arquivos do Canterbury Corpus

Número	Nome arquivo	Tamanho(em bytes)
1	Alice29.txt	152.089
2	Asyoulik.txt	125.179
3	Cp.html	24.603
4	Fields.c	11.150
5	Grammar.lsp	3.721
6	Kennedy.xls	1.029.744
7	Lcet10.txt	426.754
8	Plrabn.txt	481.861
9	Ptt5	513.216
10	Sum	38.240
11	Xagrs.1	4.227

Os testes foram realizados em um computador no padrão IBM-PC com processador Intel Atom 1.6Ghz e 2Gb de memória e sistema operacional de Windows Seven.

6. Conclusões

Quando falamos em escolher um método ou outro método para compactação de dados concordamos que alguns parâmetros devem ser observados, a escolha em si não é muito simples, a escolha depende muito do tipo de arquivo que vamos compactar, ou o quando

queremos reduzir desse arquivo durante a compactação, ou até mesmo quantos tempos têm para que essa operação seja executada. Os recursos disponíveis para essa operação quanto a espaço em disco, memória suficiente para executar a operação ou outros recursos computacionais. Percebemos através de testes que um determinado algoritmo pode ter uma alta taxa de compactação quando se trata de arquivos de texto, paginas de internet ou executáveis porem podem não ser tão bons com planilhas ou documentos do Microsoft Office.

Determinados métodos podem ser muito rápidos, mais quando se trata de taxa de compactação não alcançam valores expressivos, alguns algoritmos podem compactar arquivos rapidamente e com alta taxa de compactação mais utilizarem muitos recursos dos computadores. Como já foi citado no trabalho alguns algoritmos de compactação possuem métodos patenteados o que pode influenciar na escolha pois algoritmos da família LZW por exemplo patente registrada. Um dado importante, porém, é que, para aplicações estáticas, os códigos de HUFFMAN são melhores, ao passo que, para aplicações adaptativas e online, a LZW funciona melhor.

Referente ao dados coletados pela FADA-CHL em relação ao algoritmos HUFFMAN e LZW percebemos que o algoritmo de HUFFMAN foi superior a taxa de compactação aplicado a tabela CanterBury Corpus reduzindo cerca de 43,71% dos arquivos da tabela em média, contra uma redução média de 38,58% do algoritmo de LZW. Quando ao tempo de tempo de execução o algoritmo de HUFFMAN também se mostrou superior com um tempo médio de 6,10 segundos contra 443,73 do algoritmo de LZW. Podemos observar que quando tratamos de arquivos de texto os algoritmos LZW e HUFFMAN possuem uma taxa de compactação muito parecida, mais o tempo de execução do algoritmo de LZW é muito maior. Quanto a taxa de uso de memória através da ferramenta FADA-CHL podemos observar que o uso de memória dos dois algoritmos é parecido sendo que para alguns arquivos o Huffman usa mais memória, e para outros arquivos o LZW usa mais memória, essa elevações na taxa de uso estão relacionadas a construção dos algoritmos em relação ao uso de dicionário de dados ou arvores binárias.

Ao cruzarmos os dados com os obtidos através do software WINRAR percebemos que ele busca obter menores tempos de execução possuindo uma taxa de compactação maior, ele consegue isso pois possui na sua estrutura mais de um algoritmo de compactação, possuindo um método de escolha do melhor em relação ao tipo de entrada selecionado. Os arquivos 6 e 10 da tabela CanterBury Corpus foram removidos do teste pois possuímos dificuldades na implementação do software de comparação para as extensões xls e sum.

Como uma conclusão podemos dizer que quando aplicados a tabela CanterBury o algoritmo de HUFFMAN é mais eficiente em todos os parâmetros abordados e a ferramenta desenvolvida atende os requisitos e funcionalidades para a qual foi desenvolvida. Como trabalhos futuros sugerimos a implementação de análise estatística sobre os resultados obtidos e fazer com que a própria ferramenta possa gerar esses resultados, desenvolver a parte de descompactação e gerar uma comparação entre os resultados obtidos na compactação e descompactação dos arquivos, cruzando as informações e obtendo os resultados de cada um, desta forma obtendo o melhor algoritmo dentro das duas finalidades.

6. Referências

BUSCHART, Cameron. “File Compression.[on line] 1997”. Disponível em: http://chesworth.com/pv/technical/file_compression.htm#duh . Acesso em: 23 mar. 2012.

FRANÇA NETO, Leopoldo Rodrigues. “Um Ambiente para Processamento de Grandes Acervos de Imagens.” Dissertação de Mestrado. Departamento de Informática, Universidade Federal de Pernambuco. Disponível em: http://www.netpe.com.br/~leopoldo/tese/ . Acesso em: 23 mar. 2012.

HELD, Gilbert; MARSHALL, Thomas R. “Compactação de dados: Técnicas e aplicações, considerações de Hardware e Software.” Trad. Andreia Dell’more Santos. Edição 10 Tatuapé-SP. Livros Érica Editora Ltda. 1998. 390p.

MULLER, Daniel Nehme. “Compactação de dados.” Disponível na Internet: http://www.ulbra.tche.br/~danielnm/ed/E/polE.html. Acesso em: 23 mar. 2012.

NELSON, Mark. “LZW Data Compression.” Disponível na Internet: http://www.dogma.net/markn/articles/lzw/lzw.htm Acesso em: 23 mar. 2012.