Estudo experimental do compressor BZIP2 em arquiteturas paralelas e distribuídas

Júnior F. Barros, Raffael B. Schemmer, Júlio C. S. dos Anjos, Cláudio F. R. Geyer

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.801-970 – Porto Alegre – RS – Brazil

{jfbarros,raffael.schemmer,jcsanjos,geyer}@inf.ufrgs.br

Resumo. *Este trabalho apresenta um estudo do compressor paralelo e distribuído BZIP2 baseado no algoritmo de Burrows-Wheeler utilizando uma arquitetura de memória compartilhada e duas distribuídas. A avaliação verificou escalabilidade e aceleração em todas as arquiteturas utilizadas. Em contexto distribuído obteve-se resultados mais homogêneos em cluster se comparado a cloud, uma vez que, a última faz uso de recursos compartilhados.*

1. Introdução e objetivos do trabalho

O uso de técnicas de compressão de dados permite reduzir o tamanho dos arquivos. Os ganhos relacionados a compressão estão ligados não somente ao aumento da capacidade de armazenamento mas também, na redução do consumo de largura de banda em redes de computadores. O compressor BZIP2 [Gilchrist 2012] é baseado no algoritmo de compressão Burrows-Wheeler [Burrows 1994]. Este algoritmo realiza a compressão de arquivos a partir da busca por padrões redundantes nos dados, substituindo redundâncias por caracteres especiais. Trabalhos relacionados apontam que a implementação BZIP2 possui maior eficiência em compressão se comparado a implementações ZIP e GZ baseadas no algoritmo LZW [Mahoney 2014].

A existência de versões paralelas e distribuídas do BZIP2, que no momento não são suportadas pelas mais recentes distribuições Linux baseadas em Debian e CentOS motivam os autores deste trabalho a avaliar a escalabilidade e aceleração desta versão do compressor. Diferentes comparações entre algorítimos de compressão foram realizados pelo trabalho onde que o algorítimo BZIP2 foi aquele que apresentou a melhor relação de eficiência entre o custo computacional necessário e a capacidade de compressão. A disponibilidade computacional de máquinas distribuídas que executam aplicações cliente/servidor e de máquinas cliente do tipo multicore, permitem executar o compressor BZIP2 de forma paralela podendo manipular dados comprimidos capazes de reduzir o consumo de rede e disco. Este trabalho tem como objetivo e contribuição gerar um estudo capaz de demonstrar a viabilidade do uso da aplicação paralela BZIP2 em máquinas paralelas e distribuídas, no intuito de apontar que ao usar-se a versão paralela, consegue-se reduzir o tempo de execução da compressão e descompressão.

2. Avaliação experimental

Esta seção avalia a escalabilidade da implementação BZIP2 para três tipos de máquinas, uma de memória compartilhada multicore e duas de memória distribuída, cluster e *cloud*. É importante frisar que a escolha destes três tipos de máquina é motivada pela necessidade de executar o BZIP2 em diferentes ambientes. Relacionando, em cenário assimilado

a um usuário comum/cliente temos a máquina multicore, a serviço temos um ambiente distribuído controlado (cluster) e não controlado (*cloud*). Obtem-se da avaliação experimental o fluxo de utilização de um serviço SaaS do inglês *Software as a Service*, no qual, o estudo experimental visa trazer métricas de uso da compressão e descompressão.

Especificando, a máquina de memória compartilhada possui processador Intel Core i5 3470S 2.9GHz *IvyBridge* Quad-Core, 4GBytes de memória RAM e 1TB de disco rígido. A implementação BZIP2 é codificada em linguagem C++, utilizando POSIX PThreads [Gilchrist 2012]. A máquina possui Linux CentOS 6, kernel 2.6.32 de 32bits e G++ (GCC) 4.46. Na compilação do código foram utilizados parâmetros de compilação para a arquitetura IvyBridge (Core-avx-i) e nível de otimização O2.

Este trabalho utilizou um cluster do tipo Beowulf e configuração IaaS em *cloud*. O cluster, chamado GradeP, de propriedade do GPPD (Grupo de Processamento Paralelo e Distribuído) da UFRGS composto por 10 computadores com processadores Intel Pentium IV *Prescott* 2.8GHz Single-Core, 2GBytes de memória RAM, 1TB de disco rígido e rede Gigabit. A implementação BZIP2 é codificada em linguagem C++, usando tecnologia MPI [Gilchrist 2007]. Cada computador possui software Linux Rocks 6.1 *Emerald Boa*, kernel 2.6.32 de 32bits e MPICH2. Na compilação foi feito o uso da otimização O2.

A configuração do tipo *cloud* utilizou infraestrutura Microsoft Azure em nível de IaaS, do inglês *Infraestructure as a Service* [Chappellm 2010]. Este trabalho buscou configurar a *cloud* semelhante ao cluster, formada por 10 computadores. Cada computador possui configuração A1, processador Intel Xeon E5 2660 *Sandy Bridge* 2.2GHz Octa-Core (um núcleo é vinculado a cada computador), 1.75GBytes de memória RAM e 30GB de disco rígido. Cada computador possui software Linux Ubuntu 14.04 LTS, kernel 3.2.0 64bits e OpenMPI. A mesma implementação C++ MPI usada no cluster.

Foi utilizada a *suite* de benchmark *Silesia Compression Corpus* para realização dos testes. O benchmark é composto por 12 arquivos totalizando 202MBytes. Cada arquivo representa o conteúdo padrão de um formato de arquivo utilizado em um cenário real. O conteúdo do arquivo têm relação direta com a taxa de compressão e descompressão. Este trabalho executou 72 testes para todas as configurações, destes, normalizados em 3 execuções para cada cenário objetivando reduzir o desvio padrão causado pela oscilação dos resultados durante execuções. Levou-se em conta que o uso de recursos distribuídos em *cloud* opera sob o modo de cobrança pelo uso da infraestrutura.

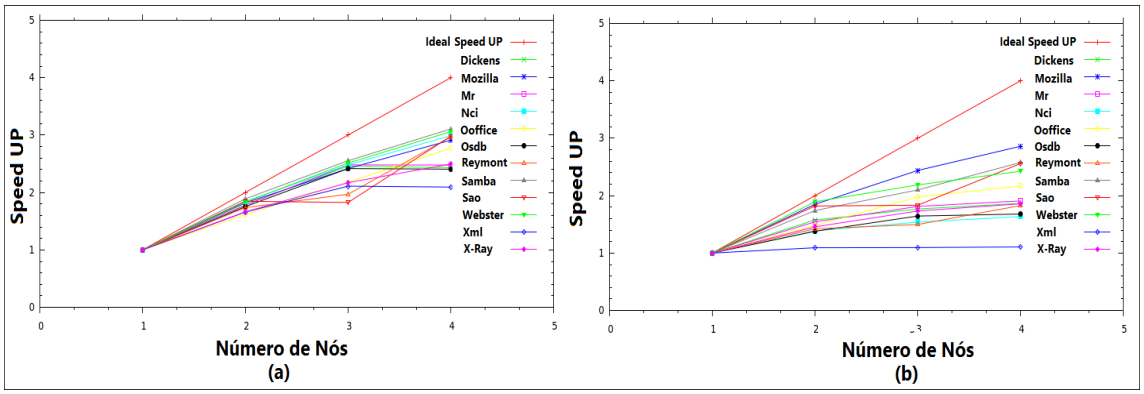

Figura 1. Fator de aceleração multicore: (a) Compressão; (b) Descompressão.

A Figura 1 apresenta o fator de aceleração, do inglês *Speedup*, referente a execução do algoritmo utilizando várias *threads*. Os resultados demonstram que a compressão apresenta melhor escalabilidade e menor espalhamento comparando os formatos de arquivos utilizados. Estas características apontam que a compressão é aquela que melhor esta preparada para suportar arquiteturas formadas por processadores extremamente paralelos.

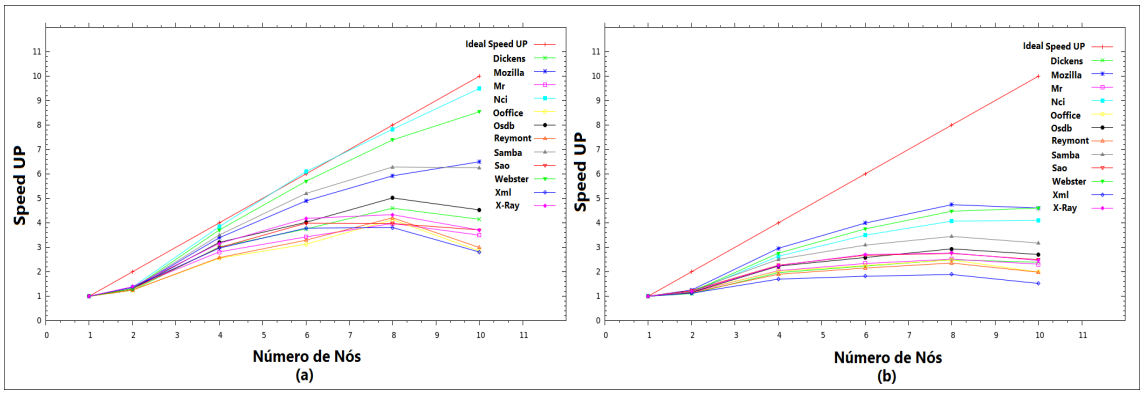

Figura 2. Fator de aceleração em cluster: (a) Compressão; (b) Descompressão.

A Figura 2 apresenta a aceleração no cenário de cluster. Os resultados apontam que a compressão novamente foi aquela que melhor atingiu aceleração existindo formatos de arquivos capazes de atingir resultados quase lineares. A descompressão seguiu as mesmas tendências encontradas no cenário multicore. Nota-se que para este tipo de máquina, o uso de tecnologias distribuídas exige que o tamanho da entrada consiga gerar volume de processamento suficiente para sobrepor o custo da coordenação, transmissão e sincronização dos dados entre os processos.

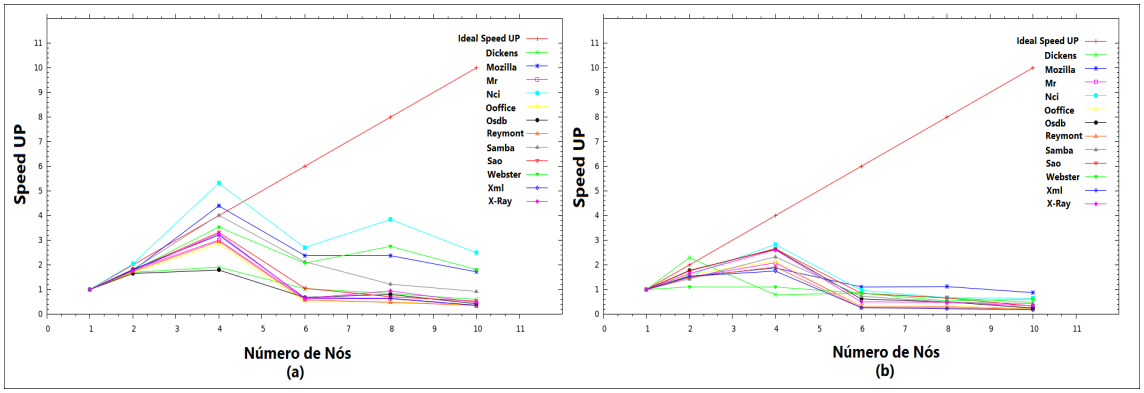

Figura 3. Fator de aceleração em *cloud*: (a) Compressão; (b) Descompressão.

A figura 3 apresenta o fator de aceleração em *cloud*, tanto para a compressão como descompressão. Os resultados da *cloud* Microsoft Azure levantam uma série de questionamentos interessantes. Os autores acreditam que a variabilidade nos resultados tenha influência direta, não somente pelos nós compartilharem uma rede Ethernet em um ambiente multiusuário mas também, por estarem operando sob uma configuração com uso de virtualização na qual pode apresentar influência direta de aplicações de terceiros executando na infraestrutura compartilhada. Por estas razões, a escalabilidade no uso de

recursos tanto para a compressão como descompressão ficou limitada a um conjunto de 2 a 4 nós, apresentando até superlinearidade em dois casos de compressão.

3. Conclusões e Trabalhos Futuros

Este trabalho apresentou um estudo experimental do compressor paralelo BZIP2. Os resultados e análises realizadas apontam que além do algoritmo ser eficiente em compressão, sua versão paralela apresenta ganhos na redução do tempo de compressão e descompressão. Também, a implementação apresenta escalabilidade quando variado o número de processadores em máquinas multicore e variado o número de nós de computação em arquiteturas distribuídas.

O crescimento das aplicações web do tipo SaaS, onde a maior parte da aplicação roda sob configuração de *cloud* ou cluster pode diretamente ser favorecida com o uso da compressão paralela. A redução dos arquivos além de aumentar a capacidade de armazenamento do lado do servidor, reduz o tamanho dos arquivos e consequente o consumo de rede, melhorando a performance da aplicação distribuída. Como verificado nos testes, a configuração de máquina distribuída cluster obteve resultados superiores a *cloud* e multicore em termos de *SpeedUp*, ou seja, melhor capacidade de redução do tempo ne compressão e descompressão com o uso de n nós controlados. Levando em consideração aplicações SaaS, teria-se melhor utilização dos recursos se a compressão fosse feita pelo lado do cluster, permitindo a transmissão dos dados de forma mais eficiente.

Trabalhos futuros poderiam avaliar o impacto causado pelos atrasos da rede nos resultados da *cloud* e também, tentar fazer uso do conceito de afinidade entre os nós na *cloud* Microsoft Azure visando reduzir a variabilidade detectada nos resultados. Ainda, sugere-se que grandes arquivos de entrada sejam usados na avaliação.

4. Agradecimentos

Os autores deste trabalho agradeçem o apoio da Microsoft e do GPPD da UFRGS pela cedência da *cloud* Azure e do cluster GradeP para a execução dos testes deste trabalho.

Referências

Burrows, M. (1994). "A block-sorting lossless data compression algorithm". `http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-124.pdf`.

Chappellm, D. (2010). "Introducing windows azure". `http://go.microsoft.com/?linkid=9682907`.

Gilchrist, J. (2007). "Parallel MPI BZIP2 (MPIBZIP2)". `http://compression.ca/mpibzip2/`.

Gilchrist, J. (2012). "Parallel BZIP2 (PBZIP2)". `http://compression.ca/pbzip2/`.

Mahoney, M. (2014). "Silesia open source compression benchmark". `http://mattmahoney.net/dc/silesia.html`.