An Approximation to the Greedy Algorithm for Differential Compression of Very Large Files [1]

Ramesh C. Agarwal, Suchitra Amalapurapu, Shaili Jain

We present a new differential compression algorithm that combines the hash value techniques of Ajtai et al. and suffix array techniques of Manber and Myers. Differential compression refers to encoding a file (a version file) as a set of changes with respect to another file (a reference file). Previous differential compression algorithms can be shown empirically to run in linear-time but they have certain drawbacks, namely they do not find the longest matches for every offset of the version file. In our algorithm, we first compute hash values for every block of the reference file. Next, we compute a suffix array on these block hash values. This helps in finding the longest matches. Generally, computing a suffix array of the reference file requires a large amount of storage and computing time. However, computing the suffix array of the block hash values reduces the storage and computing requirements roughly by a factor equal to the block size used. Our algorithm finds the longest matches for every offset of the version file, that are above a certain length threshold and with respect to a certain granularity (or *block size*). It has two variations depending on how we choose the block size. If we keep the block size fixed, we show that the compression performance of our algorithm is similar to that of the greedy algorithm of Reichenberger, without the expensive space and time requirements. If we vary the block size linearly with the reference file size, we show that our algorithm can run in linear-time and constant-space to compress very large files.

The performance of our algorithm rests upon the utilization of three new data structures, the block hash table, the quick index array, and the pointer array, which dramatically improve the run-time of our algorithm. We compare the performance of our algorithm with the algorithm of Ajtai et al, *vcdiff*, the greedy algorithm of Reichenberger, and *xdelta* on synthetic data as well as several real data sets. Additionally, we use *bzip2* on the output of our algorithm and find that, in general, *bzip2* compresses the output of our algorithm better than it compresses the output of other algorithms.

Our algorithm provides better compression than the implementation by Ajtai et al, *vcdiff*, and *xdelta* in most cases. Overall, our algorithm's run-time is somewhat slower than *xdelta*, but is significantly faster compared to *vcdiff* and the implementation of Ajtai et al. We have also shown that our algorithm is competitive with the greedy algorithm implemented by Ajtai et al. both empirically and theoretically, without the expensive space and time requirements. In most cases, our code provides slightly better compression compared to the greedy algorithm. We expect our method of differential compression not only to be used in compression applications but also as a general string matching technique to be used in web crawling as well as computational biology.

[1]Contact Authors: R. Agarwal, ragarwal@us.ibm.com, IBM Almaden Research Center. S. Jain, shailij@umich.edu, University of Michigan.