C-022

番組ファイルの差分検出のための高速ダイジェスト演算手法の検討
Study of Quick Calculation Method for Message Digest of Video Program Files to Detect Differences between two Files

和泉　吉則† 金子　豊† 小川　一人† 竹内　真也† 黄　珉錫†
Yoshinori Izumi Yutaka Kaneko Kazuto Ogawa Shinya Takeuchi Minsok Hwang

1. はじめに

ファイル化が進む放送局では、大容量の番組ファイルを扱うためファイル転送時間が課題である。筆者らは送出直前に修正された番組の差分を高速に検出し、差分のみを転送することで高速にファイルを更新する手法を開発した[1]。開発した手法では事前にファイル全体に、修正部分よりも小さなデータサイズ毎にメッセージダイジェストを演算しておく必要がある。このメッセージダイジェストには、コリジョンが少なく、均一な分布を持つ SHA-1 などのハッシュ関数が適している。しかし、大容量のデータに対するソフトウエアでのハッシュ演算は時間がかかる。そこで新たにダイジェスト演算の高速化手法を考案し、演算速度およびコリジョンについて実験と検討を行ったので報告する。

2. 番組ファイルの高速差替えシステムの課題

放送局では送出直前まで番組の手直しや修正が行われることが多い。番組のファイル化が進むと修正ファイルを再登録するためのファイル転送時間が課題となる。筆者らは修正部分が番組の一部であることを利用して、ファイルの差分を検出して、差分のみを転送してファイルを高速に差替えるシステムを開発した[1]。このために、メッセージダイジェストにハッシュ関数を利用[1]してハッシュ値の差分を高速に検出するアルゴリズム[2]と、差分のみを高速に書き換えるための挿入削除機能をもつファイルシステムを開発した[3]。開発したシステムではハッシュ演算はファイル作成時に行うことで作成時間を吸収しようと考えていた。しかし、50GB にもおよぶ番組ファイルに SHA-1 などのハッシュ演算を行うと十数分も演算時間がかかり、大幅な改善が望まれていた。

3. ハッシュ関数

3.1 ハッシュ関数 MD5、SHA-1 等の構成

暗号の分野で各種ハッシュ関数が研究され、規格化されている。ハッシュ関数とは、任意長の入力値を固定長に圧縮する関数で、出力であるハッシュ値から入力値を求めることが困難な一方向性関数である。最近の研究により脆弱性（コリジョンの発見）が指摘されているが、SHA-1 が現在のデファクト標準であり、以前のデファクト標準である MD5 も利用されている。これらのハッシュ関数では、(1) 演算に用いるデータ単位（ブロックサイズ）の倍数のデータ長にするため、入力データに 0 パディングする。(2)パディングした入力データに用意された初期値を用いてビット演算とビットシフトを組み合わせた処理を SHA-1 の場合で 80 回巡回して行うことで一様な分布のメッセージダイジェストを得る。(3)さらにデータ入力が続く場合は、得られたダイジェストを初期値として巡回演算し、最後に出力のデータ長に整形する。

3.2 ハッシュ関数 MD5、SHA-1 等の演算時間

図 1 にソフトウエア処理による各種ハッシュ関数の処理時間の測定結果を示す。(CPU:Xeon3.2GHz,2GB メモリ)

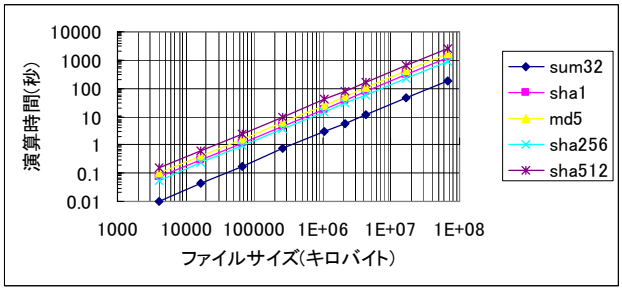

図1　各種ハッシュ演算時間とファイルサイズ

これよりハッシュ演算時間はファイルサイズ、つまり処理するデータ量に比例して大きくなることが分かる。

また、ファイルの差分を検出する目的では、差分検出を行うデータサイズ毎に(1)～(3)の処理を行うことになる。図 2 にこのデータサイズに対するハッシュ演算時間の実測を SHA-1 について示す。データサイズが小さいほど(1)～(3)の処理の呼び出し回数が多くなるため演算時間が大きくなっている。しかし、1 k バイト以上においては、データサイズに関わらず演算時間がほぼ一定であることが分かる。

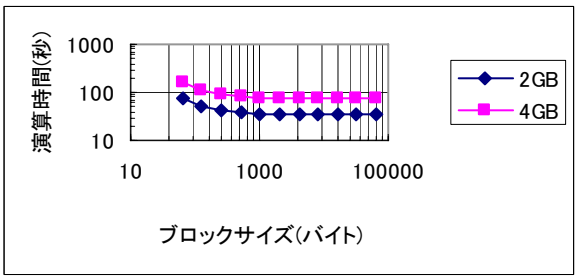

図 2　ハッシュ演算時間とブロックサイズ

4. ハッシュ演算の高速化

4.1 番組ファイルの差分検出におけるハッシュ演算時間

図 1 より、1 時間番組に相当するファイルサイズ 50GB においては、SHA-1、MD5 などのハッシュ関数では 1000 秒（約 17 分）かかることが分かった。差分検出のためのハッシュ演算としては、ファイルを全て転送するのにかかる時間より短時間でなければ意味がない。50GB のファイルを全

† 日本放送協会　放送技術研究所, NHK

て転送してファイルを上書きする場合、ギガビットイーサでの転送時間は約 400 秒(約 6 分半)である。さらに差分検出や差分データ転送のための時間短縮が必要である。

一方、ハッシュ演算の高速化については、CPU の最適化、ハード化、アッセンブラ化、並列化アルゴリズムなどの報告があるが、2～3 倍程度で不十分である。

4.2 ダイジェスト演算の高速化手法の提案

ハッシュ演算の時間は処理するデータ量に比例するので、ハッシュ演算の前にデータ量を削減すれば演算時間の短縮が可能となる。そこで、メッセージダイジェストとしては簡易だが高速演算であるチェックサムとハッシュ関数の組み合わせによる高速化手法を提案する。

図 1 にチェックサムの 32 ビットの加算処理（sum32)の演算時間とファイルサイズの関係を示しており、32 ビットのサム（sum32）の演算時間は、SHA-1 の約 1/10 である。

ここで、sum32 のデータ 1 バイトに対する演算時間を tu、SHA-1 の同データに対する演算時間を ta、ファイルサイズを F,sum32 によるデータ圧縮比を x　とすると

演算時間 T = F×tu+x×F×ta　　(1)

ただし　ta = 10×tu　　(2)

(1)式に(2)式を代入して　T = (x + 1/10)×F×ta　　(3)

ここで x→0 の時(3)式の括弧の中は 1/10 に近づき、データを大きく圧縮することにより sum32 と同程度への高速化が可能である。

4.3　提案手法の構成および実験結果

図 3 に提案方式の構成例を示す。

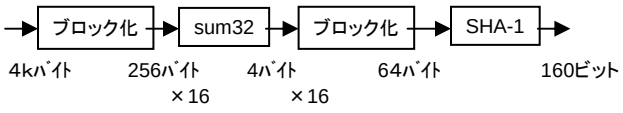

図 3　提案方式のブロック図

第 1 のブロック化は、データの圧縮率を決める。文献[1]への適用例では、差分検出のデータサイズは 4k バイトであり、sum32 を 256 バイトのブロック毎に演算して 16 個の 32 ビット(4 バイト)の sum32 が得られる。これを 4 バイト×16 個=64 バイトでブロック化して SHA-1 演算することで、4k バイトのデータサイズに対して 160 ビットのメッセージダイジェストが得られる。

次に提案方式での演算時間の測定結果を図 4 に示す。提案方式では、ほぼ SHA-1 の演算時間の 7～8 倍の高速化が達成されている。

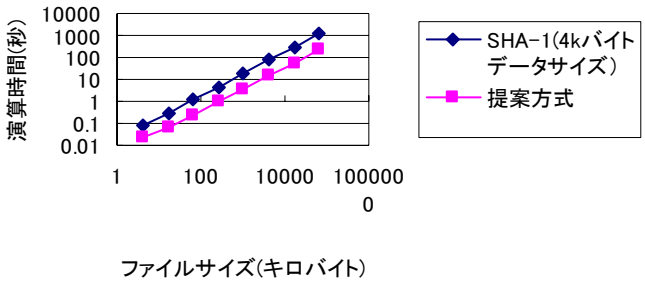

図 4　提案方式での演算時間とファイルサイズ

5. 提案方式におけるコリジョン

5.1　コリジョン

提案方式では、低コリジョンが保証されたハッシュ関数の前に、高速だがコリジョンで劣る sum32 を用いるため、sum32 のコリジョンが支配的となる。ファイルの差分の検出においてコリジョンはファイルの差分の見逃しにつながるため重要な性能である。

5.1.1 32 ビット sum のコリジョン

sum32 は 1 バイトずつを順次加算して 32 ビット（4 バイト）でオーバーフロする処理である。コリジョンは 2 つの 4 バイトの sum32 が同じで演算前の値が異なる場合である。

元の値の分布が一様と仮定するとコリジョン率は 1/(256 ^ n)となる。

一般に n バイトの sum ではコリジョン率 p=1/(256 ^ n)となり、n が大きいほどコリジョンは少なくなる。

ここでファイルサイズ M バイトのデータ列からブロックサイズ m バイトの sum を得る場合を考える。M/m 個のデータが得られ、ここから 2 個選んだ場合のコリジョンの期待値を求めると、

(M/m×(M/m-1))×p≒(M ^ 2/m ^ 2)/(256 ^ n)　となる。

n=4 バイト、M=50GB の場合、m=762939 バイト以下でコリジョンの期待値が 1 を越えてしまい、かなり大きなブロックでないとコリジョンを避けられない。

5.1.2 提案方式のコリジョン

提案方式では sum32 の演算結果を SHA-1 のために再びブロック化する。この段階でのコリジョンを求める。

n バイトの sum のコリジョン率が p の時、n バイトの sum が k 個連続した値のコリジョン率は p ^ k となる。

M/m 個から k 個連続した値を取り出した場合のコリジョンの期待値は(M/k/m)×(M/k/m-1)×p ^ k　となり、　k＞2.5 で SHA-1 のコリジョン率と同等となる。例に示した k=16 では sum32 のコリジョン率は非常に小さくなり、提案方式のコリジョン発生に支配的ではなくなる。

6. まとめ

番組ファイルの差分比較に用いるメッセージダイジェスト演算の高速化手法について提案を行い、7～8 倍の高速性、低コリジョン率を実現した。今後は放送局システムへ実装する予定である。

参考文献

[1]南浩樹, 金子豊, 竹内真也,藤沢寛,和泉吉則, “番組ファイルの高速差替え手法の検討”, 映メ放送技術研究会, Vol.32, No.42 (2008).

[2]南浩樹, 金子豊, 竹内真也,藤沢寛,和泉吉則, “番組ファイルの差分検出法の検討”, FIT2008, D-033, No.2 (2008).

[3].南浩樹, 金子豊, 竹内真也,藤沢寛,和泉吉則, “番組転送用ファイルシステム”, 放送技術, Vol.61, No.10 (2008).