4K-9

# 圧縮技術を用いた効率的なデータ通信方法の検討

表 雅之 † 大津 金光 † 大川 猛 † 横田 隆史 † 馬場 敬信 †
† 宇都宮大学大学院工学研究科情報システム科学専攻

## 1 はじめに

近年，大規模データを利用したビジネスが盛んに行われるようなり，ネットワークに掛かる負荷も増大している．そのため，ノード間における通信性能はシステム全体に影響する重要な要素となっている．大規模データを扱う際，データ量を削減するため，圧縮処理を施すのが一般的だが，圧縮アルゴリズムには圧縮率や速度に特化したものが存在するため，環境や扱うデータによって使い分けが必要である．また，圧縮データを転送する場合，圧縮，伸張および，データ転送に掛かる時間をすべて含めた場合でも，通常のデータ転送にかかる時間より短縮される場合がある．短縮される時間の程度は環境や圧縮アルゴリズムにより異なるため，出来る限り転送時間を短縮したい場合は，事前の調査が必要となり手間となる．そこで，環境により自動的に最適な圧縮アルゴリズムを選択し，データ転送時間を短縮するシステムを提案する．本システムは自動的に計算資源，実効帯域幅，対象データから圧縮方式を選び出し，出来る限りデータ転送時間を短縮する事を目的としたシステムである．

本稿では，本システムを作成するために必要となる，データ通信環境の違いによる圧縮アルゴリズムのデータ転送性能についての評価を行い，有効性について検討する．

## 2 データ転送方式

本データ転送方式では，圧縮と伸張処理のバックグラウンドで，データ転送を行い，データ転送時間の短縮を図る．圧縮処理を用いたデータ転送は，図1のように，送信側でデータを圧縮し，圧縮が完了出来次第，バックグラウンドでデータ転送を行う．受信側はデータを受信出来次第，データを伸張し，バックグラウンドでデータを受信し続ける．

本方式によりデータ転送を行う場合，重要となるのが圧縮率，圧縮速度およびネットワークの帯域幅である．圧縮率が高ければ，データ量が削減され，転送時間が短くなるが，並行してデータ転送も行っているため，ネットワークの帯域幅によっては，圧縮処理がボトルネックとなり，かえって転送時間が増加する．図1のデータ転送に掛かる時間に対して，圧縮に掛かる時間が大きい場合，圧縮処理がボトルネックとなり性能低下する．逆に，圧縮時間の方が，転送に掛かる時間より短い場合，高圧縮なアルゴリズムにより，データサイズを縮小する猶予があると考えられる．そのため，圧縮処理を利用して転送時間を最短にするには，環境毎に適切な圧縮アルゴリズムを選択する必要がある．

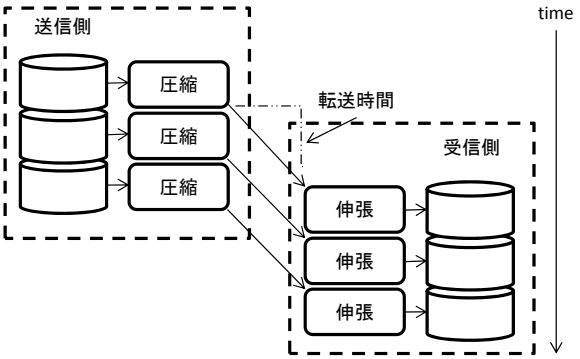


図 1: データ転送方式

## 3 データ圧縮

本研究ではデータ圧縮に，高速かつ実用的な圧縮アルゴリズムとして広く使用されている Lempel-Ziv 符号 [1] を利用した圧縮ライブラリを使用する．Lempel-Ziv 符号を利用したアルゴリズムの中でも，圧縮率や圧縮速度の面に特化させた特徴の異なるアルゴリズムが存在する．そこで，本研究では特徴の異なる圧縮アルゴリズム lz4，zlib，xz の 3 つを使用する．lz4 は Lempel-Ziv 符号を使用した，低圧縮率だが非常に高速な圧縮アルゴリズムである．また，zlib では，deflate と呼ばれる Lempel-Ziv 符号で圧縮したデータをさらにハフマン符号 [2] と呼ばれるエントロピー符号で 2 段階圧縮を行うアルゴリズムを用いており，中程度の圧縮率，速度を持つ．xz は lzma2 と呼ばれる巨大な辞書を生成する Lempel-Ziv 符号の改良アルゴリズムで圧縮したデータを，レンジコーダと呼ばれるエントロピー符号で 2 段階圧縮を行うアルゴリズムを用いている．高い圧縮率を見込めるアルゴリズムだが，低速である．加えてこれらのライブラリは，圧縮のレベルを調整できる．lz4 は圧縮のレベルを 2 段階で設定でき，zlib，xz は 9 段階で設定を行える仕様となっており値が大きいほど，低速だが高圧縮率が見込めるアルゴリズムとなる．

## 4 評価

本節では，本システムを作成するために必要となる，ネットワークの最大帯域幅の違いによる，圧縮アルゴリズムのデータ転送性能について明かにするため，実機でデータ通信し評価を行う．

Consideration of Efficient Data Communication Method using Compression Technology
†Masayuki Omote, Kanemitsu Ootsu, Takeshi Ohkawa, Takashi Yokota and Takanobu Baba
Department of Information Systems Science, Graduate School of Engineering, Utsunomiya University (†)

### 4.1 評価環境

評価環境は，送受信側共に，OS を CentOS6 とし，CPU に Intel Corei7 920(2.67GHz) を使用する．ネットワークの実効帯域幅は 920Mbps（netperf による計測値）である．データ転送速度によってのデータ転送時間を測定するため cbq および tc コマンドを使用し帯域制限を行い，制限した帯域毎にデータ転送時間の測定を行う．圧縮および転送を行う対象データは Squeeze Chart[3] のベンチマークデータ (3dgame, app, dna, gutenberg, installer, monile, pgm, sav, src, xml 合計約 4Gbyte) を使用する．また，本研究では，通信プロトコルに TCP/IP を使用する．圧縮済みのデータが 64kbyte 溜まった時点で，バックグラウンドでデータ転送する．その際の圧縮処理に使用する CPU コア数は 1 つである．また，各圧縮方式の圧縮レベルは，lz4 は 0, zlib は 5, xz は 5 を使用する．

### 4.2 各圧縮アルゴリズムの性能

各圧縮アルゴリズムがどの程度の帯域幅において有効な圧縮方式となるか見積りを立てるため，表 1 に使用する圧縮ライブラリの性能評価を行った結果を示す．

Squeeze Chart ベンチマークすべてのデータを圧縮した後のデータサイズおよび，圧縮に要した時間を測定した．その結果を元に圧縮率および，圧縮速度を算出した．圧縮率は，圧縮後のサイズを元のデータサイズで割った値として算出する．圧縮速度は圧縮後のサイズ (Mbit) を圧縮に掛かった時間 (sec) で割った値である．表 1 より，lz4 の圧縮速度は 1Gbps を越えているため，GbE 規格を利用した環境であれば，転送時間の短縮が見込めると考えられる．zlib，xz については，実効帯域幅が広い場合は転送時間が見込めないが，圧縮速度と近い実効帯域幅であれば圧縮率が高いため，lz4 より転送時間が短縮されると考えられる．

表 1: 圧縮アルゴリズムの性能

| ライブラリ | 圧縮率 (%) | 圧縮速度 (Mbps) |
|---|---|---|
| lz4 | 73.55 | 1460.54 |
| zlib | 60.96 | 101.39 |
| xz | 52.88 | 6.94 |

### 4.3 データ転送性能

図 2 にネットワークの最大帯域幅毎に，通常の無圧縮データ転送に対する，本圧縮データ転送手法の速度向上率を示す．図 2 は Squeeze Chart Courpus ベンチマークすべてのデータを転送する時間を計測し，算出した．縦軸に通常の無圧縮データ転送に対する，本圧縮データ転送手法による転送時間で割った速度向上率を示す．また横軸は使用できるネットワークの最大帯域幅を示している．図 2 より lz4 はすべての使用帯域において，1.4 倍程度の速度向上率となった．zlib に関しては 10〜100Mbps の範囲では lz4 より高い速度向上率を達成するが，100Mbps 以上の範囲では，lz4 の方が速度向上率が上回った．また，200Mbps 以上の範囲では，無圧縮のデータ転送より低速となった．また，200Mbps を越えたあたりからは，通常のデータ転送速度より低速となった．xz は，今回測定した，どの帯域でも lz4 または，zlib の方が，高速という結果となった．また，これらの結果と表 1 の圧縮速度を合わせると，圧縮速度が使用可能最大帯域を上回る場合に，速度向上の限界となることが分かる．

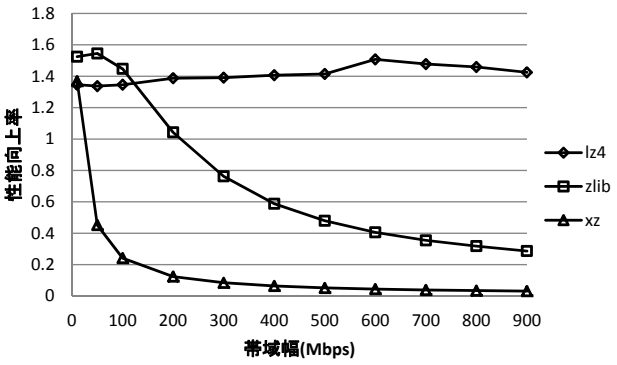


図 2: 速度向上率

評価結果より，本方式によるデータ転送は，転送時間を短縮する有効な手段といえる．図 2 および，表 1 より，圧縮速度により，ある程度の有効となる使用帯域が導き出せるため，想定する使用環境でベンチマークを取り，圧縮速度を求める事で，効率的な圧縮アルゴリズムの選択が実現出来ると考えられる．

## 5 おわりに

本稿では，本システムを作成するために必要となる，データ通信環境の違いによる圧縮技術を用いたデータ転送についての評価を行った．評価データより，圧縮データ転送はデータ転送時間を短縮する有効な手段であるとことを示した．今後は，異なるデータの種類を扱う場合についての適切な転送方式を検討していく予定である．

**謝辞**

本研究は，一部日本学術振興会科学研究費補助金 (基盤研究 (C)24500055，同 (C)24500054) の援助による．

## 参考文献

[1] Ziv. J, Lempel. A, “A Universal Algorithm for Sequential Data Compression,” IEEE Transactions on Information Theory, Vol. 23, pp.337-342, 1977.

[2] D.A. Huffman, “A method for the construction of minimum-redundancy codes,” Proceedings of the IRE, Vol. 40, pp.1098-1101, 1952.

[3] Squeeze Chart
http://www.squeezechart.com