5B-06

メーターデータ向け時系列データの格納方式

小川 康志† 松浦 陽平†

三菱電機株式会社 情報技術総合研究所†

1. はじめに

近年、スマートメーターの普及や電力自由化に伴い、メーターデータを活用するサービスや情報システムが増えつつある。30 分毎の検針値であるメーターデータは件数が多く、そのデータ処理は計算機リソースや処理時間を多く必要とするため、効率化が求められている。I/O やメモリ量削減による効率化を狙い、本稿では、高い圧縮率を少ない演算量で実現するメーターデータの格納方式を示す。

2. ターゲットと課題

センサーの計測値など時系列データは、各データが独立ではなく時間順序で隣り合うデータ間で相互に影響を有しているものが多い。このため、時系列データの前後値の差分は、絶対値が小さくなる傾向にある。時系列 DB では差分の特徴を用いてデータを効率的に圧縮する差分符号化方式[1]が広く利用されている。この方式では、時刻情報とデータそれぞれの差分を算出し、そのビット列を並べた塊(chunk)として扱う。差分の絶対値を小さくすることで必要となるビット数を減らし、圧縮率を向上できる。特にこの方式では、差分 0 値が最も多い分布となるデータが対象とされている。

本稿でターゲットとするメーターデータは、常に値が増加する累積値であり、差分は非 0 が大部分である。また、電力値(低圧)は少数点以下 2 桁を含む数値である。浮動小数点型データは差分では圧縮しにくく、XOR 演算を用いて差分のビット数を小さくする方式[1]もあるが、常に増分がある場合では高い圧縮率は期待できない。

また、メーターデータは欠測が発生すると欠測補完として後から欠測データが入力される場合があり、時系列データの入力順が保証されない特徴がある。

3. 提案方式

メーターデータは時間間隔が固定であるため、時刻情報をそのまま保持するのは非効率である。起点となる時刻情報のみを保持し、それ以外のデータは配置位置で時刻を判断できる構造とすることで時刻情報分のデータを削減することができる。例えば、30 分電力量であれば 1 日 48 コマとなるため、図 1 のように日付情報と配置位置からデータの時刻を特定可能である。

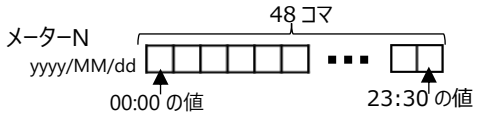

図 1：chunk 内のデータ位置と時刻の対応

また、浮動小数点型データは差分として扱いにくいが、ターゲットのメーターデータを少数点 2 桁固定とすると、2 桁シフトさせることで整数型と同様に扱うことができる。本稿では桁シフト後の整数型として差分を抽出する。

前述のとおり、メーターデータは累積値であるため、差分により絶対値の小さい値で表現できる。また、メーターデータの差分である電力消費量も時系列データであるため、二重差分により値をさらに小さくできる可能性がある。電力会社の 1 時間ごとの発電量[2] (1 年間分)から、差分と二重差分の分布を、図 2 に示す(図中の値は、世帯当たりの 1 日の平均電力消費量を 20kWh と想定して補正したもの)。差分の分布はオフセットがあるが、二重差分は 0 付近に分布しており、二重差分の方がより小さい値とみなすことができる。

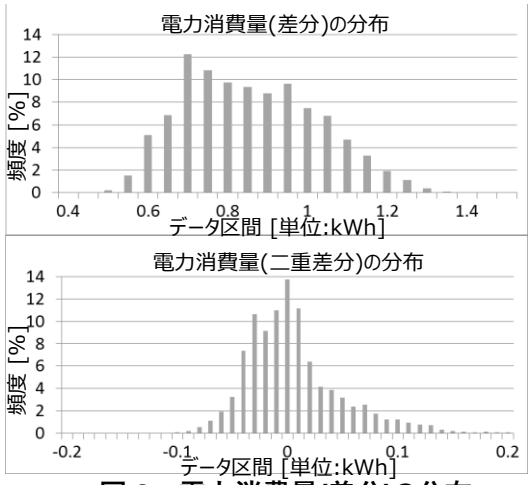

図 2：電力消費量(差分)の分布

◆データ格納方式

chunk のビット列表現として、図 3 に示す方式がある。

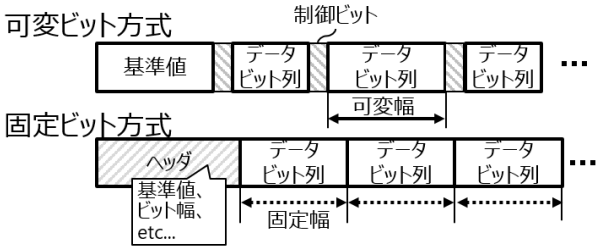

図 3：可変ビット方式と固定ビット方式

Storage method of time series data for meter data

† Yasushi OGAWA, Yohei MATSUURA, Information Technology R&D Center, Mitsubishi Electric Corporation.

可変ビット方式は、各差分値のビット幅を指定する制御ビットを設け、ビット幅を変更可能とする。固定ビット方式は、ヘッダにてビット幅を指定し、chunk 内は同一のビット幅でデータ列を表現する方式である。可変ビット方式がより高い圧縮率を実現できるが、データ位置を特定するためデータ列を前から順に走査する必要があり、データ参照時に処理コストが掛かることになる。

二重差分 ddx_n の導出式においても以下のように 2 つの式がある。

$$ddx_n = dx_n - d_{n-1} \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (1)$$
$$ddx_n = x_n - (x_0 + n \times dx) \cdots\cdots\cdots\cdots\cdots\cdots (2)$$

x_n: n番目のデータ, $dx_n = x_n - x_{n-1}$, dx: 基準となる差分値

式(1)は、2 つの差分値 dx_n、dx_{n-1} から導出するものである。式(2)は近似式であり、基準値 x_0、差分値 dx と要素の位置 n から導出する。二重差分 ddx_n から元データ x_n を求める場合、式(2)では基準値とする x_0、dx と位置 n から導出可能であるが、式(1)の場合、x_{n-1}、x_{n-2} など以前のデータを順次参照する必要がある。このため、式(1)は可変ビット方式、式 (2)は固定ビットとの組合せで利用される。

可変ビット方式+式(1)は高い圧縮率が期待できるが、chunk 内のデータ数が多いほどデータ参照時の演算回数が増える。特に、メーターデータにおいて発生する欠測補完に対して、chunk の再生成が必要となる。一方、固定ビット方式+式(2)は圧縮率が若干落ちるものの、データ参照が容易であり、欠測データに対して該当データの位置を空けておくことで、欠測補完時も容易に追記可能である。以上の点から、メーターデータに対しては固定ビット方式の方が適していると言える。

しかしながら、式(2)の近似はデータ変動が常に一定であることを想定しており、変動量が大きい状態が続くと累積して差分値が大きくなり、圧縮率の低下を招く。

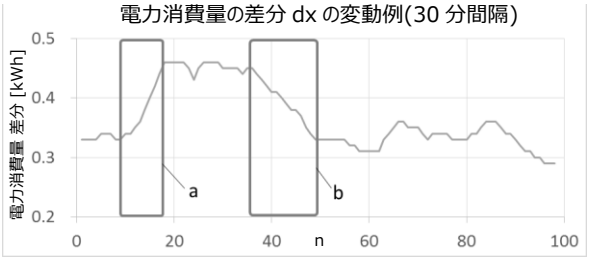

図 4：電力消費量の差分 変動例

図 4 に電力消費量の差分 dx の変動例を示す。dx が一定値の周辺で変動をしている区間は式(2)で表現できるが、差分 dx が増加する区間や減少する区間（図の a,b)ではズレが大きくなりやすい。ここで、差分の増加/減少量が一定と仮定すると等差級数の和から式(3)を導出できる。

$$ddx_n = x_n - \left[x_m + (n-m)\left(dx + \frac{ddx(n-m-1)}{2}\right)\right] \ldots\ldots (3)$$

m: 前区間の末尾位置, ddx: 前区間とのdxの差分

この式に対して、図 5 に示すように、差分 dx の変動が変わる区間で chunk 内にサブヘッダを設け、ビット幅や二重差分の導出式を切替える事で、変動パターンに対応させることができる。

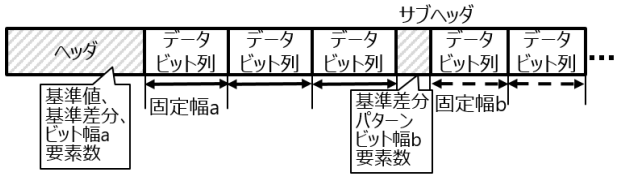

図 5：提案方式 の chunk 構造

この方式では、データ x_n を ddx_n から参照する際、分割した区間の数だけ計算を行う必要があるが、chunk 内の要素 n に比べると小さな回数で抑えられる。各区間においては固定ビット方式と同様にデータ位置を表現可能であるため、欠測補完にも対応可能である。

4. 評価

1 日分の 30 分間隔データ(48 データ)を 1 chunk とし、各格納方式でデータを格納した際のデータサイズを表 4-1 に示す。XOR 方式は、浮動小数点に対して差分符号化を用いたもの、それ以外は整数型データに対してそれぞれの方式で格納したものである。入力データには、前述の補正済みの平均電力消費量を用いた。

表 4-1：各方式の chunk 格納後のデータサイズ

方式	1 データ当たりのバイト数
XOR 差分[1]	3.26
二重差分(固定ビット)	1.34
提案方式	1.06

元データ：時刻データ 8 [B], 数値データ 8 [B]

提案方式は、固定ビット方式に対して、2 割程度データを小さくすることが可能である。

5. まとめ

本稿では、メーターデータに対して効率的に圧縮を行うため、二重差分として、差分が一定となる区間以外に、差分が増加/減少する区間を考慮した近似式を用いてデータを格納する方式を示した。提案方式は、固定ビット方式に比べてデータサイズの点で 2 割程度改善可能である。

参考文献

[1] Tuomas Pelkonen, Scott Franklin, Justin Teller,「Gorilla: A Fast, Scalable, InMemory Time Series Database」.2015.

[2] 電力需要 過去実績データ(リンクは中部電力) <http://denki-yoho.chuden.jp/denki_yoho_content_data/areajuyo_current.csv>.