大規模 JSON ファイルのデータ構造を保存した分割法

濱野 聖人
芝浦工業大学大学院 工学研究科 電気電子情報工学専攻

鈴木 徹也
芝浦工業大学 システム工学部 電子情報システム学科

1 はじめに

軽量なデータ記述言語 JSON(JavaScript Object Notation)[1] は，Web ブラウザ上で動作する JavaScript プログラムにおいて広く使われている．なぜなら，JSON は JavaScript の記述方法をベースとしており，JavaScript との親和性が高いからである．例えば，Ajax(Asynchronous JavaScript+XML)[2] において，データ記述言語に本来の XML の代わりに JSON が利用されることもある．

しかし，大規模な JSON データを Web ブラウザで読み込もうとすると，データの読み込み，解析時に CPU やメモリの使用量が増え，Web ブラウザがクラッシュする場合もある．

それに対し，我々はその問題を解決する手段として，JSON データの分割と参照方法を提案した [3][4].

本論では新しい分割方法を提案し，JSON データ分割の適切なパラメータを実験により求める．

2 関連研究

2.1 軽量データ記述言語 JSON

例を示して JSON の配列とハッシュを説明する．JSON では、配列とハッシュを入れ子にして，複雑な構造を持ったデータを記述できる．

```
{"key1":"value1", "key2":"value2"}
```

図 1 JSON におけるハッシュの記述例

図 1 は，JSON を用いたハッシュの簡単な例である．ハッシュは，順序付けされない名前と値の組の集合である．ハッシュの記述では，'{' と'}' の間に、キーと値の組を',' で区切って並べる．また，キーと値は':' で区切って並べる．図 1 では key1 と key2 がキーであり，value1 と value2 がそれぞれのキーに対する値である．

```
["array_value1", "array_value2"]
```

図 2 JSON における配列の記法例

図 2 は，JSON を用いた配列の例である．配列は，順序付けられた値の集まりである．配列の記述では，'[' と']' の間に',' で区切って値を並べる．

2.2 JSON データ分割記法 SJSON2

指摘した JSON の問題点に対して我々は JSON データ分割記法 SJSON2 を提案した [4]．それは SJSON[3] を基にした，以下のような手法である．

- 巨大な JSON ファイルを，そのデータ構造を反映しつつ，大きさがある閾値以下の複数のファイルに分割する．
- 分割で得られたファイルは，ハイパーリンクが埋められており，全体として木構造を成す
- 分割によって得られた複数のファイルを Web サーバ上に設置する．
- Web ブラウザ上の JavaScript から専用のライブラリを用いて分割ファイルを参照する．

この手法により，Web ブラウザからは巨大なデータの一部分だけを保持しながら，データ構造を自由に辿ることができるようになった．

記法について説明する．SJSON2 では，構造をもったデータとして，ハッシュ，分割されたハッシュ，配列，分割された配列の 4 つを提供する．そのいずれも配列を用いて表現する．

```
<SARRAY> := [5,<SARRAY_ELEMENT>]
<SARRAY_ELEMENT> :=
[<STRING>,<STRING>,0,<ARRAY_VALUE>] |
 [<STRING>,<STRING>,1,<URL>] | [6,<URL>]
```

図 3 SJSON2 における分割された配列の文法

```
[5,[0,499,1.1.js],[500,999,1,2.js]]
```

図 4 SJSON2 における分割された配列の例

図 3 に分割された配列の文法を，図 4 に分割された配列の例をそれぞれ示す．図 3 の <SARRAY> が分割された配列を示す非終端記号である．図 4 は，2 分割された配列の例である．そして，一番外側の配列の先頭要素 5 が，このデータ構造は分割された配列である事を示す．それに続く 2 つの配列が，分割の仕方を示している．例えば，1 つ目の配列は，添字 0 から 499 までの要素は別ファイル 1.js に記録されている事を表している．このように型，値，参照方法などを判別する特定の数値をその配列の特定の添字に記録する．そして，この数値より値が別ファイルにあるのか，即値として現在参照しているファイルにあるのか等を判別し，値を参照する．

現状の SJSON2 による分割方法の問題点として，以下の 2 つが挙げられる．

1. 分割方法が，二分木と線形リストだけである．
2. JSON データ分割の際，ファイルサイズの適切な閾値がどれほどなのか分からない．

1 番目は，2 分木では，どんなに閾値が大きくても中間ノードの中身が 2 つのファイルへのリンクのみになっている．そのため，中間ノードのファイルサイズが閾値よりもずっと小さく無駄が生じているという問題点がある．

3 提案手法

既存の SJSON2 の問題点に対して，以下の方法を提案する．

- n 分木で分割できるようにする．分割ファイルの大きさが閾値に近くなるように n の値を決定する．

n 分木では，現在の閾値から，いくつ下位ノードへのリンクを埋め込めるかを計算し，n を決定する．この手法により，各中間ノードの容量が，2 分木と比較して，n 分木では閾値により近くなると期待できる．

```
[5,[0,249,1.1.js],[250,499,1,2.js],
   [500,749,1,3.js],[750,999,1,4.js]]
```

図 5 SJSON2 における分割された配列の例

図 5 に n 分木での例を挙げる．n 分木では，2 分木の例である図 4 と比較して，下位ノードへのリンクが複数埋め込められている．

この n 分木を実装した SJSON2 を SJSON2n とここでは呼ぶこととする．

4 評価実験

4.1 方法

Web ブラウザにおいて，JSON，SJSON2，SJSON2n のファイルを読み込み，1000 回ランダムアクセスした場合における CPU 使用率，メモリ使用量，読み込み開始からアクセス終了までの時間を計測した．また，SJSON2 の場合，閾値を 1KByte，10KByte，100KByte，1MByte とした場合でそれぞれ計測した．

実験に使用したブラウザは，Firefox バージョン 3 と Internet Explorer バージョン 6 の 2 種類である．OS は，Windows XP Home Edition SP3 である．PC は，CPU PentiumM 1.6GHz，メモリ 480MByte である．

実験環境での回線速度は，およそ 6Mbps である．

また，実験に用いたデータは，日本全国の郵便番号と住所を記録したハッシュの配列である．その容量は，8MByte である．SJSON2 は，閾値を 10KByte とし，2 分木の形で分割した．その容量は，3998 ファイルで平均 2.8KByte，合計 24MByte ほどである．SJSON2n は，n 分木の形で閾値 10KByte として分割した場合，その容量は，2045 ファイルで平均 5.6KByte，合計 18MByte ほどである．

CPU 使用率は実行中での 1 秒おきの結果の平均を採用した．メモリ使用量は実行前のメモリ使用量と実行中における 1 秒おきの結果の差を取った平均を結果として採用した．

なお，JSON を IE で計測した場合，メモリ不足となり計測できなかったため，グラフではその部分を空欄とし，表ではハイフンとしている．

4.2 JSON-SJSON2-SJSON2n の CPU 使用率比較

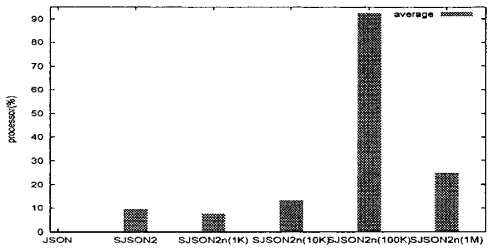

図 6 IE における CPU 使用率比較

図 6 が IE における CPU 使用率の比較である．SJSON2，SJSON2n 共に同じ閾値 (10K) で分割していれば，大きく CPU 使用率は変わらない事が分かる．閾値が，1KByte の場合，CPU 使用率が 10KByte の場合と比べて少し低く，100KByte，1MByte では，10KByte と比べて CPU 使用率が大きく増大する事が分かる．ブラウザが Firefox の場合においても JSON ファイルが読み込めた事を除いて，ほぼ同様の傾向が見られた．

4.3 JSON-SJSON2-SJSON2n のメモリ使用量比較

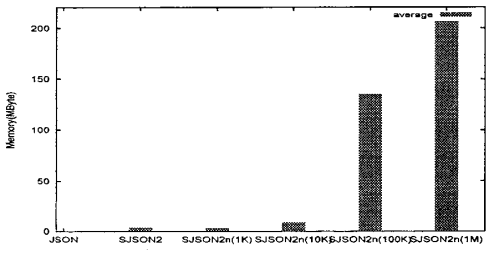

図 7 IE におけるメモリ使用量比較

図 7 が IE におけるメモリ使用量の比較である．SJSON2，SJSON2n 共に同じ閾値 (10KByte) で分割していれば，大きくメモリ使用量は変わらない事が分かる．閾値が 1K の場合，メモリ使用量が 10KByte の場合と比べて少し低く，100KByte，1MByte では，10K と比べてメモリ使用量が大きく増大する事が分かる．ブラウザが Firefox の場合においても JSON ファイルが読み込めた事を除いて，ほぼ同様の傾向が見られた．

4.4 JSON-SJSON2-SJSON2n の処理時間比較

表 1 実行時間比較

	読込開始	読込終了	アクセス開始	アクセス終了
JSON	0	39	-	-
SJSON2	0	0	0	249
SJSON2n-1K	0	0	0	138
SJSON2n-10K	0	0	0	119
SJSON2n-100K	0	0	0	497
SJSON2n-1M	0	0	0	56

表 1 は，処理時間の比較である．同じ閾値 (10KByte) における SJSON と SJSON2 を比較すると，SJSON2 と比べて SJSON2n-10K では 50% 以上処理時間を短縮できている事が分かる．SJSON2 で閾値を変えた場合，閾値が 100KByte の場合，大きく遅くなり，1M になると大きく高速になっている事が分かる．Firefox では，閾値が大きくなる程，高速に処理できるという結果となった．また，Firefox の JSON では JSON ファイルが読み込めて，JSON ファイルの読み込みからアクセス開始までに 19 秒かかり，実際のアクセスには，6 秒という結果となった．JSON ファイルが読み込めてしまえば高速にアクセスできている事が分かる．

4.5 評価

分割方法に 2 分木と n 分木を使用した場合を比較すると，平均の CPU 使用率，メモリ使用量共に同程度になるという結果になった．しかし，n 分木を使用した場合の処理速度は，2 分木を使用した場合と比べて，2 倍以上の高速化ができた．これは，n 分木で分割する事により 1 ファイルに情報が多く入り，HTTP アクセス回数が削減できたためだと考えられる．また，閾値は，10KByte から 100KByte の間とするのが適切であると考えられる．これは，10KByte 未満とするとアクセス回数が増大し，処理速度が遅くなる．一方で，100KByte 以上とすると IE での処理が遅くなり，また，回線速度が遅い環境でもデータのダウンロードの時間がかかってしまうため処理が遅くなると考えられるためである．IE で閾値を 100KByte とすると処理時間が遅くなる理由は，IE では JSON ファイルを評価する処理が遅く，大量のファイルを評価する 100KByte の場合で問題となっているためと考えられる．1MByte で問題となっていないのは，評価するファイル数が 100KByte の場合と比べて非常に少いからであると考えられる．

5 おわりに

本論では，JSON データ分割言語 SJSON2 による JSON データの分割方法に，n 分木を導入した．実験により，n 分木による分割の有用性と，適切なファイルサイズの閾値を示した．今後の課題は，キャッシュ処理を工夫することや，応用例で実験をして有用性を確認することが挙げられる．

参考文献

[1] JSON，http://www.json.org

[2] Michael Mahemoff，AJAX デザインパターン，株式会社オライリー・ジャパン，2007.

[3] 石川泰式，鈴木徹也：" 巨大 JSON データの分割と参照"，第 70 回情報処理学会全国大会講演論文集，3S-4，2008.

[4] 濱野聖人，鈴木徹也：" 大規模 JSON データのためのファイル分割と参照"，WI2-2008-47〜75，pp. 61 ー 66，2008.