データ転送方式と XML スキーマの改善によるガントＸの大規模データ転送への対応

Large Scale Data Transfer Technique for GanttX Through Use of Chunked Encoding and Improvement of XML Schema

内海　宏律　Utsumi Hironori
門司　太郎　Monji Taro

（株）日立東日本ソリューションズの製品 SynViz/PJ は，Web ブラウザ上で高機能なガントチャート画面が利用できるプロジェクト管理・工程管理システムである。SynViz ではガントチャートを Web ブラウザに表示するために，表示に必要なデータをサーバ側で生成し，XML 形式のデータとして HTML に埋め込んでクライアントへ転送している。しかし，長期間かつ大規模なプロジェクトになると，ガントチャート上に表示するオブジェクト数が数万以上になり，応答速度の著しい低下，場合によってはメモリ不足で表示できなくなるなどの問題が発生していた。

このためガントチャート画面の処理フローを見直し，HTTP/ 1.1 のチャンク転送によるメモリ使用量削減とクライアント待ち時間の短縮，および XML スキーマの改善によるデータ量削減の 2 つの手法により本問題を解決した。その結果，より大規模なガントチャートの表示が可能になり，かつ応答速度も 60% の高速化を実現できた。

１．はじめに

Web アプリケーションである SynViz では，高機能なガントチャート画面を実現するために ActiveX コントロールであるガント X を使用している。最近ではタスク数が数百から数千行，アクティビティやマイルストーンなどのオブジェクトの総数が数万以上となるような大規模プロジェクトで使用されるケースも出てきており，大規模なガントチャートを高速に表示したいという要求が高まってきている。非常に大きなガントチャートになると，画面表示までの応答速度が低下するだけでなく，メモリ不足で表示に失敗する場合があり，根本的な対策が必要になりつつあった。

こうした問題に対し，これまでは内部の処理ロジックの改善で対応してきたが，これ以上の性能対策は内部処理だけでは限界がある。また，ガントチャートのデータは XML 形式で転送しているが，XML のスキーマは元々 Web でのデータ受け渡しを前提に設計されたものではないため，データ量を抑える仕組みなどは備えていない。そのため，大規模なガントチャートではデータ量が数十から数百 MB と巨大になるため，データ転送に時間がかかり，結果として処理時間が延びていた。より大規模なガントチャートに対応するには，内部の処理だけでなく，データ転送やサーバ・クライアントでの処理フローなど周辺も含めた根本的な処理の見直しとデータ量の削減が必要である。

本稿では，ガントチャート画面を表示する際の処理フローを精査したうえで，メモリ使用量の削減および処理時間の短縮を実現可能な方法を検討する。そのうえで，チャンク転送を用いたデータ転送方式および XML スキーマの改善によるデータ量の削減の 2 つの手法を適用し，性能測定によりこれらの手法が有効であることを確認する。

２．ガントチャート画面の表示フロー

SynViz では，サーバサイドにアプリケーションサーバ（以下，SynViz サーバまたはサーバと表記）およびデータベース（DB）サーバを配置し，クライアントサイドに Internet Explorer(IE)を搭載した Windows PC を配置

する構成となっている。サーバとクライアント間はブラウザを介し，HTTP を使用してデータ通信を行う。ガントチャートのデータはXML形式のデータとしてHTMLに埋めこまれた形でクライアントへ転送される。ユーザがガントチャート画面を開く際の処理フローを図 1 に示す。

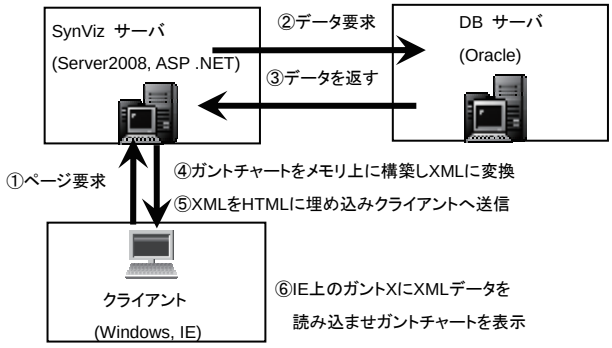


図 1　ガントチャート画面の表示フロー

初めに, ユーザが Web ページ上のリンクをクリックするとページ要求がサーバに送信される。サーバは該当するガントチャートのデータを DB サーバから取得し，このデータをクライアントに渡すために一旦サーバのメモリ上にガントチャートのイメージを構築する。その後，ガント X の機能を利用して XML 形式でガントチャートのイメージを出力（永続化）し，HTML に XML を埋め込んだうえでクライアントへ送信する。クライアントはページの読み込みを行い，画面に貼り付けられたガント X に XML 形式のデータを読み込ませることで，IE 上にガントチャート画面を表示する（逆永続化）。

３．大規模データ転送時の課題

大規模なガントチャートを表示する場合，メモリ使用量が増加し例外が発生する問題と処理時間が増加する 2 つの問題が発生している。

3.1　メモリ使用量の問題

サーバでガントチャートのイメージを XML 文字列に永続化する際に，OutOfMemoey 例外が発生する問題である。サーバサイドでは，DB サーバから取得したデータから，アクティビティなどのオブジェクトをメモリ上に構築し, XML 文字列に永続化する作業を行っている。ここで生成された XML 文字列は IIS によってクライアントへと転送され，転送が完了した後にメモリ上から削除される。サーバサイドでのメモリ使用量の変化を図 2 に示す。なお，32bit プロセスの場合，IIS のワーカプロセスが使用できる仮想メモリ空間は基本的に 2GB までとなる。

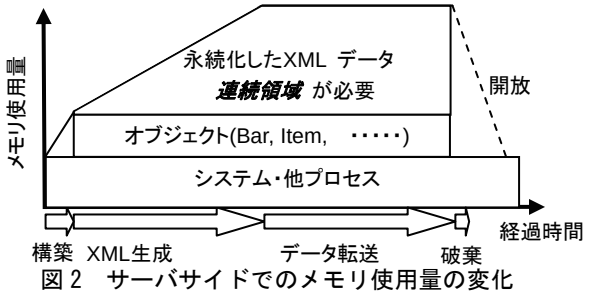


図 2　サーバサイドでのメモリ使用量の変化

初めに，サーバではガントチャートの構築を行うが，メモリ上に構築されたガントチャートに含まれる各オブジェクトは，オブジェクト単位でメモリ中に格納されるため，連続領域は必要としない。このステップでは理論上，仮想メモリ空間の限界までメモリを使用することができる。次のガントチャートを XML 文字列に永続化する処理では，数万オブジェクト分の XML 文字列全体を 1 つの連続領域に格納する必要ある。問題となっている OutOfMemory 例外は，この永続化の処理でメモリ上に連続した領域が必要になるが，メモリの断片化などにより連続領域が確保できなくなった場合に発生している [1)]。この例外は，物理メモリに余裕がある状態でも発生し，単純なメモリの増設のみでは対応できない特徴がある。この問題では例外が発生し，処理が途中で強制終了されるため，ガントチャートをまったく表示できなくなる。また，ガントチャートの規模を小さくする以外に有効な解決策は存在しない。

3.2　処理時間の問題

この問題は，単純にデータ量が増大したために各処理に時間がかかるようになったことが原因で発生している。SynViz では, ガントチャート画面の表示にかかる時間が他の画面と比較すると長いため，さらなる改善が必要となっている。これまでもガント X 内部の処理ロジックなどを見直すことで性能を改善してきたが，これ以上の性能改善を行う場合は，全体的な処理フローの見直しおよび転送されるデータ量の削減が必要となる。

現在の方式では，図 3 のように XML の生成やデータ転送などの処理がすべて直列的に行われており，クライアントはサーバサイドでの XML 生成処理がすべて完了するまで待ち状態となっている。また，サーバ・クライアントそれぞれに着目した場合でも，ネットワークや CPU などの同時に使用可能なリソースが互いに待ち状

態となっている。さらに，各処理間で受け渡すデータをすべてメモリなどの一時記憶領域に格納しておく必要があるため，メモリ使用量増大の原因ともなっている。

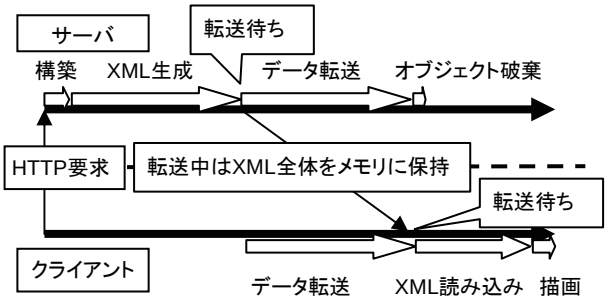


図 3　ガントチャート画面が表示されるまで

４．改善策の検討

4.1　メモリ使用量の削減

この問題は，XML 文字列をメモリ中に連続領域を必要としない形式で格納しておくことで解決できるが，XMLの生成が完了するまで巨大なXML全体をメモリ中に保持しておく方法はサーバリソースの有効活用とはいえない。この XML 自体はクライアントに送信した後は不要となるため，生成が完了した部分から逐次クライアントに送信し，送信が完了したものから順に捨てていく手法を提案する。これにより，図 4 のようにサーバのメモリ上に XML を保持せずに処理が可能となる。さらに，XML を格納する分の領域をオブジェクトの格納に使用可能となるため，転送可能なガントチャートのサイズも大きくなる。

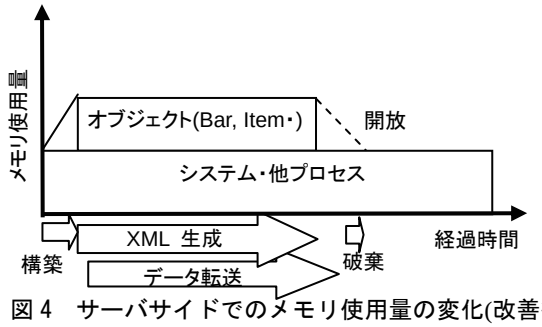


図 4　サーバサイドでのメモリ使用量の変化(改善後)

また，クライアントサイドでもサーバサイドと同様に XML の転送と読み込みを同時に行うことで，転送完了まで XML をメモリ中にバッファリングする必要はなくなる。これにより，クライアントサイドでのメモリ使用量の削減も可能であるため，同様の手法をクライアントにも適用する。

4.2　処理フローの改善

図 3 に示す現行の処理フローに対して同時実行可能な処理の洗い出しを行った。まずは，サーバサイドに着目すると，ガントチャートのイメージが完全にメモリ上に構築されるまで XML の生成は開始できない。このため，「オブジェクト構築」と「XML 生成」は同時実行不可能である。次に「XML 生成」と「データ転送」を見た場合，XML の生成はガントチャートに含まれるオブジェクトに対して順次実行され，生成が完了した XML 文字列に対する挿入・参照操作はない。そのため，生成が完了した部分はクライアントに送信するまではメモリ中にバッファリングされている状態に過ぎず，逐次クライアントに送信しても問題は発生しない。また，「XML 生成」では CPU が，「データ転送」ではネットワークが主に使用されるため，互いに干渉する要素は少ない。つまり，「XML 生成」と「データ転送」は同時実行可能であるといえる。処理の並列化により，生成が完了した XML 文字列をメモリ中にため込む必要がなくなり，メモリ消費量の削減にもつながる。

次に，クライアントサイドの処理フローを検討する。クライアントでは，XML を読み込む際に Simple API for XML(SAX) [2]を使用している。SAX は XML 文書を 1 つの木構造として扱う DOM と異なり，イベント駆動型の API となっており，XML 文書の先頭から順にアプリケーションへとイベントを発生させることにより XML 文書を読み込む。SAX では，一度読み込みが完了した部分を再度参照することはないため，XML 全体をメモリ中に保持しておく必要がない。そのため，巨大な XML 文書でもメモリに負担をかけずに読み込むことができるという特徴がある。SAX を使用する場合，XML 全体の転送が完了している必要はないため，その時点までにサーバから転送されたデータを逐次読み込むことが可能である。よって，「データ転送」と「XML 読み込み」は並列実行可能であるといえる。また，サーバの場合同様に，「データ転送」ではネットワークが，「XML 読み込み」では CPU が主に使用されるため，同時実行した場合でも処理時間に対して大きな影響はない。最後の「画面描画」については，オブジェクトの描画順の関係などにより，単純に読み込んだ差分のみを描画することはできず，毎回ガントチャート全体を再描画する必要がある。XML を一部読み込む度に再描画を行うと何度も全体を再描画することとなり，クライアントの CPU に負荷がかかる。そのため，今回は従来どおり XML の読み込み完了後に

1 度だけ実行することとする。

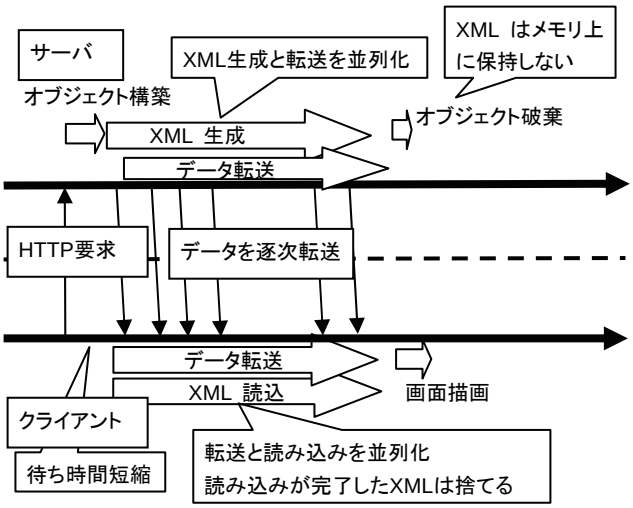


図 5　各処理を並列化した場合の処理フロー

これらの処理を並列実行することで，図 5 のように同時刻にサーバサイドの「XML 生成」と「データ転送」およびクライアントの「データ転送」と「XML 読み込み」を実行できるため，クライアントの待ち時間削減が可能となる。

これまでに述べた手法を実現するためには，サーバサイドでは動的生成するデータの生成完了を待たずに送信する技術，クライアントサイドでは到着したデータを順に受け取り，プログラムで処理する技術が必要になる。これらは HTTP プロトコル上でデータの逐次転送を可能とするチャンク転送を用いることで実現可能である。次章では,チャンク転送を用いてガントチャートの XML データを転送する手法を検討していく。

5．チャンク転送によるデータ転送方式の改善

チャンク転送は HTTP/1.1(RFC 2068 Hypertext Transfer Protocol HTTP/1.1 [3])で定義されており，データをチャンクと呼ばれる小さな塊に分割して逐次送信する手法である。HTTP プロトコルでは，送信されるデータ長をレスポンスヘッダの Content-Length で受信側へと通知している。受信側では Content-Length の値を基に，ダウンロード進捗状況の表示およびすべてのコンテンツのダウンロードが完了したかどうかのチェックなどを行っている。

動的コンテンツをチャンク転送する場合，送信開始時点ではデータの生成が完了していないため，Content-Length は未決定である。チャンク転送では，全体の長さではなく一度に送信するチャンクごとにそのチャンクの長さを送信する。受信側ではチャンクごとにデータ長を照合することができるため，データの受信確認を問題なく行うことができる。これにより，チャンク転送は静的コンテンツと動的コンテンツの双方に適用可能となっている。また，チャンク転送は HTTP/1.1 で定義されているが，ネットワークの経路上にあるプロキシサーバなどが HTTP/1.0 のみの対応となっていることも多い。HTTP/1.0 の場合はチャンクサイズが送信されないが，データの逐次転送自体は可能であるため HTTP/1.0 の場合でも逐次送信という意味でのチャンク転送は可能である。

6．XML スキーマの改善によるデータ量削減

本章では，ガントチャートのデータをクライアントへ送信する際に使用する XML のスキーマを改善することにより，通信時のデータ量を削減する方法を検討する。

現在，ガント X の XML は図 6 のように各オブジェクトのプロパティを XML の要素として出力している。

```
<Bar Start=“2011/02/01”　Finish=“2012/01/31”・・・>
  <Height>100</Height>
  <FillColor>#00FFFF</FillColor>
          :
</Bar>
```

図 6　ガント X の XML の構成

オブジェクトは複数のプロパティを持つため，数万オブジェクト分の情報が出力される XML 中には膨大な数の要素が出力される。現在のスキーマでは，プロパティ名が要素の開始タグと終了タグの両方に重複して出力されているため，データ長が長くなる原因となっている。特に,プロパティ名が長い場合にその影響は顕著となる。そこで，各プロパティの値を図 7 のように XML の属性として出力する手法を提案する。これにより，図 6 では "FillColor"が 2 回出現するが図 7 の提案手法では 1 回しか出現しなくなるため，データ量が削減できる。長いプロパティ名を多数使用する SynViz でも一層のデータ量削減が期待できる。

```
<Bar Start=“2011/02/01”　Finish=“2012/01/31”
Key=“”　Height=“100”　FillColor=“#00FFFF”・・・ />
```

図 7　プロパティを XML の属性で出力

チャンク転送を適用した場合，サーバとクライアントの CPU はほぼ 100%使用する状態となる。データ転送中のビットレートは約 10M から 50Mbps 程度で余裕があ

ることから，処理のボトルネックは CPU であるといえる。XML の生成および読み込み処理は，プログラム内部では文字列操作となるため，文字列長が短くなるほど CPU の占有時間が短縮される。また，SAX で XML を読み込む際は XML の要素ごとにイベント処理が実行されるため，属性とすることでイベントの発生回数を抑え，処理速度を向上させることができる。さらに，実際の環境では複数人が同時にアクセスし，ネットワークの負荷低減が重要な課題となるため，データ量削減は有効な手法であるといえる。

7．提案手法の評価

提案手法の有効性を確認するため，サーバとしてメモリを 6GB 搭載した Windows Server 2008 R2 を用いて性能評価を行った。

7.1　転送可能なガントチャートのサイズ

チャンク転送を用いた場合に転送可能となるガントチャートのサイズを画面に含まれるオブジェクト数として求めた。その結果を図 8 に示す。それぞれ上が IIS を 32bit プロセスとした場合，下が同 64bit とした場合の結果である。なお，IIS7.0 では 32bit 実行時のワーカプロセスのユーザ仮想メモリ空間は 4GB となる [4]。

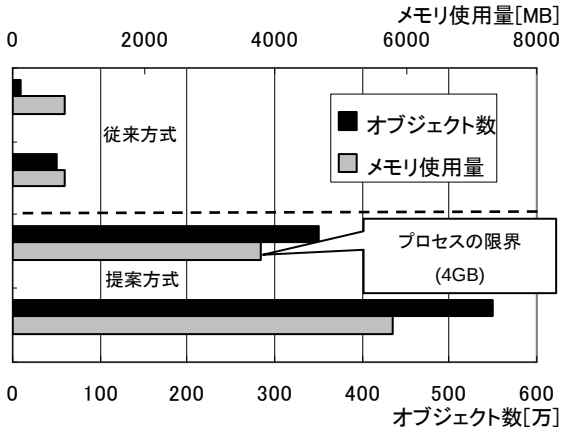


図 8　転送可能なガントチャートのサイズ

従来手法では 64bit プロセスとすることで，転送可能なオブジェクト数が 5000 オブジェクト程度に増加している。これは，64bit 化によりメモリ空間のアドレスが増加したため，物理メモリ上の連続した領域を取得しやすくなっているのが要因と推測されるが，最終的にはメモリを 800MB 程度使用した時点で OutOfMemory 例外が発生してしまう。

提案手法では，転送可能なサイズが大幅に拡大され，32bit プロセスの場合は 350 万オブジェクト，64bit の場合は 550 万オブジェクトが転送可能になっていることが分かる。550 万オブジェクトは SynViz の標準的なデータモデルで約 66,000 行に相当する規模であり，提案手法により大規模なデータが転送可能となったといえる。また，提案手法ではメモリをプロセスの限界まで有効に活用できていることが分かる。図 8 の 64bit プロセスの場合の 550 万オブジェクトという値は，物理メモリをすべて使い切った時点での値であり，実際にはスワップを使用することでこれ以上の規模のデータを転送可能である。

転送可能なサイズに関しては，XML スキーマ改善の効果は見られなかった。これは，チャンク転送により XML がメモリ上に格納されなくなったため，転送可能なサイズの上限に影響を与えなくなったためである。

7.2　画面表示までの時間

4 万オブジェクトを含むデータを使用し，ユーザがサーバに要求を送信する直前から，ガントチャートが完全に表示されるまでの時間を計測した。サーバの CPU は Intel Core i7 960 (2.93GHz)，クライアントの CPU は Intel Core2 Quad Q9650 (3.00GHz)を使用した。処理時間は図 9 のように，チャンク転送と XML スキーマ改善を併用した場合に最大で処理時間を 60%ほど削減できた。XML スキーマ改善によって処理すべきデータ量が削減され，ボトルネックである CPU の処理時間が短縮されたため，全体の処理時間が短縮されたと判断できる。

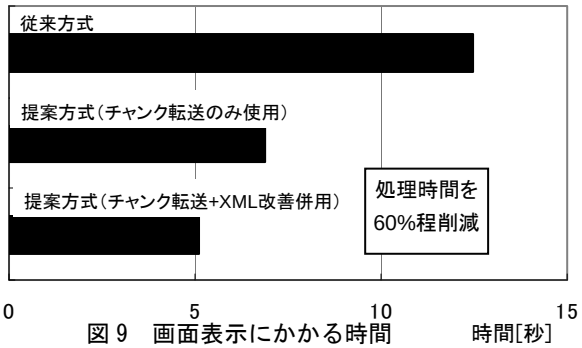


図 9　画面表示にかかる時間

8．おわりに

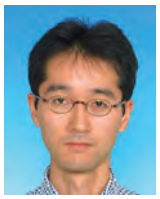
大規模なガントチャートのデータをサーバからクライアントに転送して表示する場合，メモリの連続領域不足により画面が表示できない問題と表示に時間がかかるという 2 つの問題が発生した。本稿では，チャンク転送によるデータ転送方式の改善および XML スキーマの改善

によるデータ量の削減を行った。その結果，表示にかかる時間の削減と表示可能なデータサイズが大幅に拡大されたことを確認した。今後，提案手法を SynViz に適用することで大規模データを短時間に表示することが可能となる見込みである。これにより，ガントチャート画面の操作性がさらに向上し，大規模データを扱う顧客への適用も可能となる。

参考文献

1) ASP.NET Web アプリケーションで System.OutOfMemoryException が発生する場合のトラブルシューティング，Microsoft, http://support.microsoft.com/kb/954830/ja, (Accessed 2010/10).
2) Simple API for XML，Wikipedia, http://ja.wikipedia.org/wiki/Simple_API_for_XML, (Accessed 2010/10).
3) RFC 2068 Hypertext Transfer Protocol HTTP/1.1，IETF, http://tools.ietf.org/html/rfc2068，(1997)，(Accessed 2010/11).
4) Mike Volodarsky 他，Microsoft Windows Server 2008 リソースキット IIS7.0 編 マイクロソフト公式解説書，日経 BP ソフトプレス，p659，(2009).

内海　宏律　　2006 年入社
研究開発部
自社パッケージソフトウェア, コントロールの設計，試作，開発
hironori.utsumi.01@hitachi-to.co.jp



門司　太郎　　1994 年入社
研究開発部
自社パッケージソフトウェア, コントロールの設計，試作，開発
monji@hitachi-to.co.jp