5Y-3

PCクラスターによる並列レンダリングの高速化に関する研究

長塩 智史† 渡部 雅人† 中嶋 正之†‡ 齋藤 豪†

†東京工業大学 大学院 情報理工学研究科 ‡国立情報学研究所

1 はじめに

近年，計算機の急速な性能向上に伴い，大規模な数値シミュレーション結果を可視化することに注目が集まっている．特に大規模な３次元仮想空間の可視化はゲーム，広告などの目的から期待を集めている．

しかし，この実現には，広大なシーンのリアルタイムレンダリングのための膨大な計算量を必要とするため，一般的には多くの時間を要する．高速にレンダリングを行いたい場合には，単純なレンダリングのアルゴリズムを利用したり，シーンの総ポリゴン数を少なくするなど，大きな制限が加えられる．また，より広大なシーンを処理する場合には，複数の計算機に処理を分散させるといった手法も多くとられている．

分散レンダリングはデータ転送に時間がかかるため，従来手法 [1] では高速な処理は難しい．本論文では Sort-lastImageCompositing 型 [2] の分散レンダリング手法を基に，サーバ間の転送画像の圧縮，OpenGL の PixelBufferObject 機能によってサーバ間通信の高速化を目指す．また，従来手法と比較することによって，本手法の有効性を検証する．

2 Sort-last 型分散レンダリングシステムの設計

提案手法は，Sort-lastImageCompositing 型の分散レンダリングを基に処理の高速化を目指す．分散レンダリングには，描画処理のどのプロセスを並列化するかによって複数の種類が存在し，多くの場合，サーバ-クライアント型の形態をとる．

Sort-lastImageCompositing 型もサーバ-クライアント型の分散処理を行う型の１つで，広大なシーンをオブジェクト空間において複数の領域に分割し，それぞれを複数の計算機に割り当てて分散処理を行う．以下，ユーザとの対話，及び結果となる画像の表示を行う計算機をクライアント，レンダリング等実際の計算処理を行う計算機をサーバと呼ぶ．

レンダリングは以下の手順で行う．まずレンダリング要求としてクライアントから各サーバへオブジェクト空間における視点位置情報を送信する．その位置情報を元に各サーバでは担当する領域内のレンダリングを行い，結果の画像及び深度値をクライアントに返す．最後に，クライアントが各サーバから受け取った画像を Z バッファ法を用いて合成し，これを表示する．

この手法の欠点としては，他の手法と比較してサーバの計算量が大きいこと，送信する画像のサイズが大きく，ネットワークの負荷が大きくなってしまっていること，領域分割を行っているため，最終的に表示する画像が単一の計算機でレンダリングを行った場合と異なってしまう可能性があることが挙げられる．

3 PBO を用いたサーバ内処理の高速化

PixelBufferObjects(以下，PBO) は OpenGL2.1 によって標準化されたライブラリで，主にテクスチャのダウンロード/アップロードの高速化に用いられる．通常はクライアントメモリに置くべきデータをビデオメモリに置いたまま処理を行うことができるため，CPU-GPU 間の転送負荷が減少する．巨大なデータを扱う場合や，頻繁にテクスチャを更新する場合などには特に効果が高い．この特性は Sort-last 型分散レンダリングのサーバ処理の負荷軽減に適していると考えられる．そこで，本稿では分散レンダリングに適したデータ転送手法について提案する．テクスチャへのデータ転送の詳細を図に示す．提案手法はファイルコピーが多く，高速化に適していないように見えるが，一番のボトルネックとされている CPU-GPU 間のデータ転送の負荷を抑えることができるため，パフォーマンスでは上回ることも考えられる．また，PBO を用いたデータ転送は CPU 負荷が少ないため，スレッドを並列化して複数スレッドによる処理を実装した場合，速度の低下が抑えられるというメリットも存在する．

提案手法の有効性を確かめるため，実際に実装を行い，画像のテクスチャへのダウンロード，及び表示画像の読み込みの 2 つのプロセスにおいて，PBO の有無による速度の違いを計測した．結果をそれぞれ表 1，表 2 に示す．

この結果から，テクスチャのダウンロード (CPU ⇒

Boosting Parallel Rendering using cluster of PCs

Tomofumi NAGASHIO†,
Masato WATANABE†,
Masayuki NAKAJIMA†‡,
and Suguru SAITOU†
†Graduate School of Information Science and Engineering, Tokyo Institute of Technology
‡National Institute of Informatic
†naga@img.cs.titech.ac.jp

画像サイズ () 内は容量	PBO 使用	PBO 未使用
640*480(921654)	261	238
320*240(230456)	1065	915

表 1: テクスチャダウンロード速度 (単位:FPS)

画像サイズ () 内は容量	PBO 使用	PBO 未使用
640*480(921654)	80	139
320*240(230456)	320	556

表 2: 画像の読み込み速度 (単位:FPS)

GPU) を行う場合には PBO を使用した方が FPS が上昇し，画像の読み込み (GPU ⇒ CPU) を行う場合は PBO を使用すると逆に FPS が下がることが判る．ただし，今回の実験ではスレッドの並列化は行っていないため，さらなる検証が必要である．

4 画質・速度を考慮したネットワーク負荷の軽減

Sort-lastImageComposition 型の分散レンダリングでは，画像をサーバ側からクライアント側へ送信するというプロセスが存在する．このため，画像サイズや接続サーバ数に起因するネットワークへの負荷が原因となり，システム全体の高速化に影響を与えている．これを解決するための手法として，画像の圧縮が考えられる．しかし，画像の圧縮を行うことにより，転送量を抑え，ネットワーク負荷の軽減は達成できるが，一方で，圧縮及び解凍に時間が掛かること，圧縮による画質の低下，という 2 つのデメリットが出てくる．また，画像の圧縮には多くの種類があり [3][4][5]，それぞれに長所及び短所が存在する．

そこで本稿では，ネットワークの負荷軽減を目的とした画像データサイズの縮小，圧縮・解凍に掛かる時間，画質の劣化度合いの比較を行うため，実際に画像の圧縮を行い，圧縮に要する時間と圧縮後のサイズを計測した．なお，従来手法 [6] では圧縮速度の優位性よりランレングス点順次圧縮法を用いることが多いが，場合によっては元のデータより容量が増えてしまうなどの欠点があり，また，近年のアーキテクチャの発展により他の圧縮方法との速度差も詰まっていると考え，本研究では対象から外した．結果を表 3 に示す．

LZO 圧縮に非常に時間が掛かっているが，GIF 画像には圧縮の前に減色処理を施しており，このプロセスにおよそ 2000msec の時間が掛かっているためこのような値となっている．

この結果から，圧縮時間，画像サイズ共に JPEG が非常に優れていることがわかる．しかし，JPEG 圧縮は非可逆圧縮のため，画質の劣化を検証する必要がある．

画像フォーマット	圧縮時間 (ms)	サイズ (byte)
無圧縮 (BMP)	0	921654
JPEG 圧縮	60	69449
PNG 圧縮	605	436295
LZO 圧縮 (GIF)	2500	171199

表 3: 画像圧縮手法の比較

5 まとめと今後の課題

本稿では，Sort-lastImageComposition 型の分散レンダリングの高速化を行うため，PBO によるデータ転送の有効性の検証と画像圧縮手法の比較を行った．今後はスレッドを並列化した場合の PBO の有効性の検証や圧縮した画像の画質の検証を行い，分散レンダリングシステムの実装を試みる予定である．

参考文献

[1] S.Molnar, M.Cox, D.Ellsworth, H.Fuchs: “A sorting classification of parallel rendering”, IEEE Computer Graphics and Applcations, vol.14(4), pp.59-68, 1994.

[2] M.Eldridge:Parallel graphics:Scalability and communication. In Proc. of SIGGRAPH 2001(37), pp.15-39, August 2001.

[3] ”Independent JPEG Group” http://www.ijg.org/

[4] ”Cover Sheet for the GIF89a Specification” World Wide Web Consortium. http://www.w3.org/

[5] ”Portable Network Graphics (PNG) Specificationand Extensions ” http://www.libpng.org/pub/png/spec/

[6] Nishikawa T.,Nishi T.,“ Realtime Rendering Application Development using Parallel Processing ”,UNISYS TECHNOLOGY REVIEW 第 80 号,FEB.2004