社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－CG－124 (9)
2006／8／18

GPUを用いた物理シミュレーションの高速化

菊川孝明† 緒方正人†
梶原景範† 土肥康孝††

実時間シミュレーションをめざして，可視化と計算を同時かつ高速に行う試みがなされている．この試みの一環として，大規模システムを経済的に構築できる理由から，数値計算にGPU(Graphics Processing Unit)を用いる試みがなされている．しかし，GPUは汎用CPUと異なり，条件分岐や広域変数の保持などにおいてストリームプロセッサ固有の制約を持ち，先達の英知の結晶である数値計算アルゴリズムを容易には実装できない．そこで，GPUへ各種の数値計算アルゴリズムを容易に実装できるような計算構造（仮想演算器とそのAPI）をGPU上に構築した．この計算構造の評価のため，有限要素法による構造解析を目的として，GPU上に共役勾配法を実装しCPUと性能の比較を実施した．CPUに実装した最適化共役勾配法と，GPUに実装した共役勾配法の計算速度の比較では，10万次元においてGPUでの実装が約3倍高速となった．

Speedup of Physics Simulation with GPU

Takaaki Kikukawa,† Masato Ogata,† Kagenori Kajihara†
and Yasunori Dohi††

To achieve real-time simulations, various attempts have been made on system architecture for speedup of simultaneous visualization and calculations. Because of the large system can be built economically, the active researches for substituting GPU for CPU in numerical calculation have arisen. However, the constraints on stream processor such as branch instruction or sustaining global variables make it difficult to implement sophisticated numerical algorithms directly on GPU. Therefore, we have built a virtual computing unit and its API that could help to implement various numerical calculation algorithm on GPU easily. As evaluation of this computation structure, we have implemented conjugate gradient method on GPU and have performed comparison with CPU. The experimental result shows that the CG method implemented in GPU is 3 times faster than CPU at calculation size of 100,000 dimensions.

1. はじめに

実時間での物理シミュレーションが夢物語でなくなりつつある．物理シミュレーションにおいては，現象の直感的把握のため，数値計算とその結果の可視化がインタラクティブに行えることが重要となる．このためには，大規模数値計算を高速に処理する価格性能比の良いアーキテクチャが必要となる．

近年の半導体技術の進歩によりシミュレーションを低価格システムでおこなうことが検討されている[1),2),9),11)]．特に，ゲーム用のグラフィックスカードに搭載されるGPU(Graphics Processing Unit)は性能向上がムーア則を上回り，浮動小数点演算性能においてはCPUを凌駕している．また，GPUは利用分野の広まりに伴い，開発環境，プログラム言語が整備されつつある[5)]．このような状況から，数値計算処理にGPUを適用することで，低価格で高速なシミュレーションの計算基盤として活用することが期待されている．

本論文では，3次元構造体の有限要素法を用いた変形シミュレーションの実現を最終目的に，多元連立一次方程式の解法である共役勾配法（CG法）を例にとりGPUへの実装手法を示す．また，計算速度をCPUに実装した同一手法と比較した．今回の評価は，CG法の精度に関しては対象外とし両者とも単精度での比較を行った．

2. 従来研究

問題のサイズにもよるが，これまで高価なスーパコンピュータの独壇場であったシミュレーション分野に，

† 三菱プレシジョン株式会社
Mitsubisi Precision Co., Ltd.
†† 横浜国立大学
Yokohama National University

PCクラスタが用いられるようになってきた．物理シミュレーションにおける数値計算量は膨大であり，市販デバイスの性能が向上したとはいえ十分ではない．そのため並列度を上げて計算能力を高める際に問題となる並列化率の低下を画像合成専用ハードウエアを用いて緩和し，対話的な可視化装置を実現する手法が提案された[7),8),10)]．

一方，GPUの進歩は著しく，これを計算に用いて性能向上を図る提案がなされている[1)~3)]．

中でもGPUクラスタ[1)]は，数値計算を加速するために，PCクラスタを構成する各PCの中にGPUを挿入してクラスタ化している．480×400×80格子のBoltzmannモデルのシミュレーションと可視化の同時処理を，30台のPCを用いて，0.32秒/ステップで達成している．

Jensら[2)]，またBolzら[3)]はGPUへ連立一次方程式の解法(CG法，ガウス-ザイデル法，MultiGrid法)を実装し，波動シミュレーション，流体シミュレーションの高速化を実施した[1)~3)]．何れの報告も，GPUの性能が高かったことが報告されているが，実装方式および条件が詳細には報告されていない．これを明確にすることが本報告の一つの目的でもある．

3. 対象とした問題と解法

本調査では，有限要素解析(FEM)による3次元構造体の変形シミュレーションの問題を対象とした．実験に用いたデータは，シミュレーション対象を単純な四面体要素で構成したモデルであり，構成要素数を変更することで，問題全体の次元を変化させる．図1にFEMによる変形シミュレーションの様子を示す．

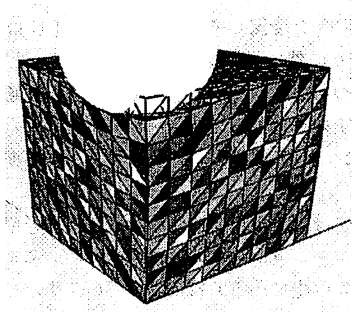

図1 FEMによる変形シミュレーションの様子
Fig. 1 Appearance of FEM Simulation.

3.1 有限要素法による構造解析

有限要素法を用いて，構造解析による変形シミュレーションを行う場合，大規模な連立一次方程式を解かなければならない．この連立一次方程式は，行列計算 $F = KU$ で表現さる．ここで，F：外力ベクトル，K：剛性行列，U：変位ベクトルである．構造解析の問題では，解析対象物のヤング率，ポアソン比等の物理量により，剛性行列Kが定義される．

3次元解析の場合，問題の次元数は，節点数の3倍となる．この多次元行列計算の解法には，共役勾配法を適用した．

3.2 共役勾配法

共役勾配法は連立一次方程式を解く反復法の一つであり，汎関数の極値を求めて間接的に方程式を解く変分法と同じ概念で解を求める[6)]．すなわち

$$Ax = b$$

を直接解く代わりに

$$f(x) = \frac{1}{2}x^t Ax - x^t b = \frac{1}{2}x^t Ax - b^t x$$

が極小値をとる x を探索することによって $Ax = b$ を満たす x を求める．

$Ax = b$ を解く共役勾配法のアルゴリズムを参考文献[6)]に従い図2に示す．

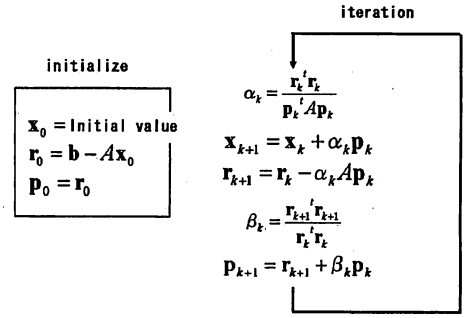

図2 共役勾配法のフロー
ここで，x:未知のベクトル，b:既知のベクトル，A:行列
r:残差ベクトル，P:修正ベクトル，k:反復回数
Fig. 2 Flow Chart of Conjugate Gradient Method.

4. GPUへの計算の実装

GPUへのプログラムの実装は，Cg言語[5)]を用いた．Cg言語はNVIDIAがMicrosoftと共同で開発した，C言語に類似した3Dグラフィック用言語であり，GPUのバーテックスシェイダー及びフラグメントシェイダーをプログラム可能である．

本調査ではフラグメントシェイダーに共役勾配法を実装した．その手法について以下に述べる．

4.1 ベクトル要素と画素

フラグメントシェイダーは，テクスチャデータを入力とし，フレームバッファの画素値を計算することを処理の単位としている．これを連立一次方程式 $F = KU$ を解く問題に適用すると，ベクトル F の1要素をフレームバッファの1画素に対応させることになる．すなわち，「問題の次元数」＝「計算画素数」となる．

4.2 セグメント

GPU へ実装するために，共役勾配法の計算を次元数回のループが必要なステップ単位に分解する．これを「セグメント」と呼ぶと，共役勾配法の1イタレーションは4つのセグメントからなる．また，初期化部は2セグメントからなる．この各セグメントで1回のフレーム描画処理を適用する．

図3は共役勾配法のフローをセグメント単位に再構成したものである．初期化を含み6セグメントからなる．

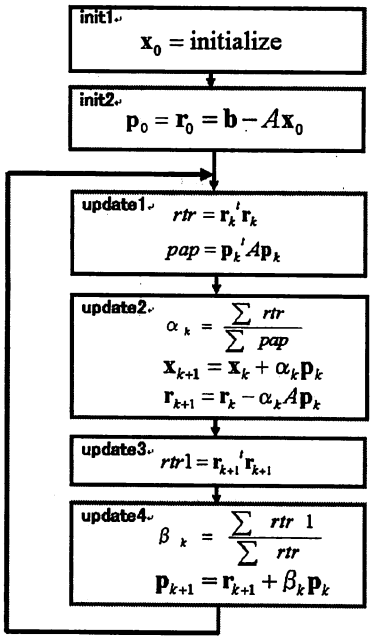

図3 共役勾配法の"セグメント"への分配
Fig. 3 Distribution of Conjugate Gradient Method to 6 "Segments".

4.3 Cg 言語による GPU への実装の問題

Cg 言語は C 言語に類似した言語ではあるが，CPU のプログラムのように，メモリをランダムに読み書きする記述はできない．これは GPU がストリームプロセッサ[4] の一種であるための制約であり，このことが GPU への汎用計算の実装の妨げとなる．

GPU へ計算を実装する場合の主な制限事項を次に示す．

(1) フラグメントシェイダーの出力先は，基本的にフレームバッファ(またはPバッファ) に限られている．テクスチャメモリに出力する場合は，直接的手法として Render-to-Texture，間接的手法として Copy-to-Texture がある．
(2) 出力先のメモリのアドレスは制御できない．
(3) グローバル変数，スタティック変数にデータを保持することができない．
(4) for 文等のループ回数に変数を指定できない．
(5) 中間データの受渡しの問題
各セグメントは，前段のセグメントの計算結果を入力する必要がある．これは Render-to-Texture によりセグメント出力をフィードバックすることで可能であるが，フィードバックできるデータは，pバッファのデータフォーマット R，G，B，A に対応する4種類のみである．
(6) ベクトルの内積計算の問題
ベクトルの内積を計算する場合，要素の乗算は可能であるが，乗算結果の積算はできない．図4参照．ここでは，アドレス"screen"を使って要素を読み出し，その積を出力先の p バッファのアドレス"screen"へ書き込む．このとき，隣り合う要素を積算することはできない．

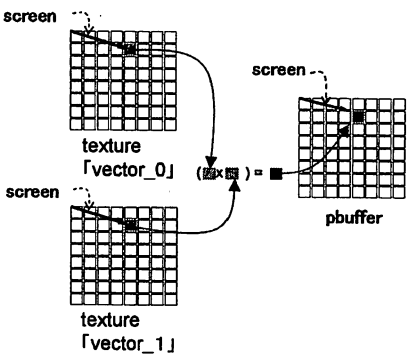

図4 GPU におけるベクトルの内積計算
Fig. 4 Inner Product Operation on GPU.

4.4 汎用計算のための仮想演算器

GPU へ汎用計算を実装するとき，問題となるのは，セグメントから次セグメントへ中間データを渡す方法である．この問題に対応するため，Pバッファを3つ

用いた1出力2入力のデータフィードバック構成をもつ仮想演算器をGPUに実装した．1つのPバッファを出力先として使用し，残り2つは入力テクスチャ$U0$及び$U1$に接続する．さらに，セグメントでの一連の処理が終了したとき，Pバッファ群を回転し別のPバッファに接続する．このとき，$U0$は前段セグメントの出力，$U1$はさらにその前段の出力となるように順序を守った制御をする．図5に概念図を示す．

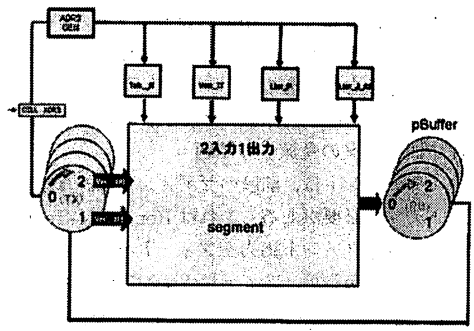

図5 GPUへ実装した仮想演算器
Fig. 5 Virtual Computing Unit on GPU.

5. 処理時間測定

5.1 各セグメントの処理時間の測定

上記仮想演算器を用いて共役勾配法をGPUに実装し，図3に示す各セグメントごとに処理時間の測定を行った．

このとき，問題の次元数は24から12,288まで変化させた．次元数と各セグメントの処理時間の関係を図6に示す．また，処理時間測定に用いた計算機環境を表1示す．

表1 実験1(セグメントの処理時間測定)を実施した環境
Table 1 Specifications of facility for the experiment 1.

No.	Item	Specifications
1	CPU	Pentium4 2.4GHz
2	GPU	Nvidia Geforce 6800Ultra
3	BUS	AGP 8x
4	OS	Windows 2000

5.2 最適化

図6に示す各セグメントの処理時間によると，次元が増加するに従って，セグメント間の処理時間の差が増大し，図3に示すセグメント，「update2」及び「update4」では，「update3」と比較して100倍の処理時間を要している．また，「init2」及び「update1」では，「update3」と比較して約10倍の処理時間を要している．これらの処理時間を短縮させるため，次の最適化を試みた．

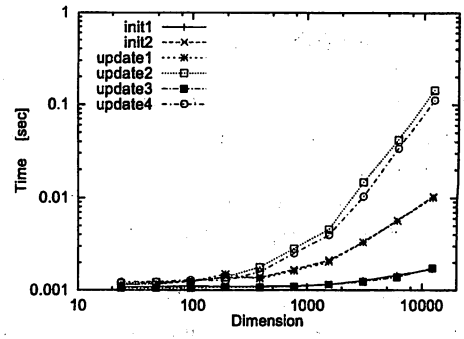

図6 セグメントの処理時間比較
Fig. 6 Comparison at Processing Time of Segments.

(1) 最適化1

「update2」では，rtr及びpapの要素の積算を次元計算毎に実施していた．

これを修正し，セグメントを分離することで，あらかじめ積算を実施した．「update4」も同様の変更を行った．図7参照．

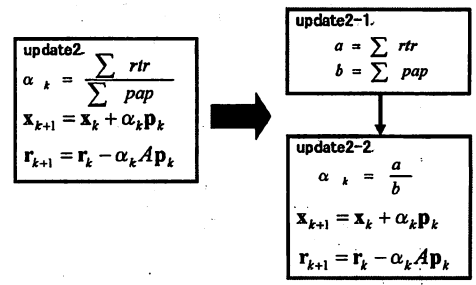

図7 最適化1のためのセグメント分割
Fig. 7 Division of the Segment as Optimize1.

(2) 最適化2

「init2」及び「update1」では，行列とベクトルの積を計算しているが，行列の行要素とベクトル要素の積和をfor文で記述していた．

これを修正し，行列の各行の非零要素を8×8の正方に，対応するベクトル要素へのインデックスと共に格納した．同時にセグメントは乗算用と積算用に分割した．図8参照．ここで，要素の積算はKrugerが示した手法[2]を用いた．

初期の実装，最適化1の実施後，および最適化2の実施後の測定結果を図9に示す．図は，初期化から全てのセグメントを1イタレーション実施したときの処理時間を測定した結果である．

セグメントを増加させ最適化を実施することで，低

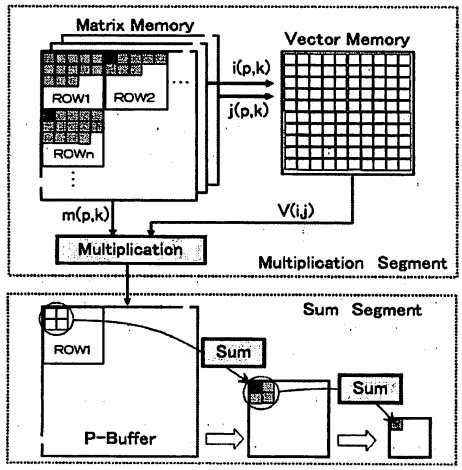

図 8 最適化 2 のための積和セグメントの分割
Fig. 8 Division of the Segment as Optimize2.

次元の領域では処理時間が増加するが，多次元領域では処理時間の増加が少なく，より高速となる．

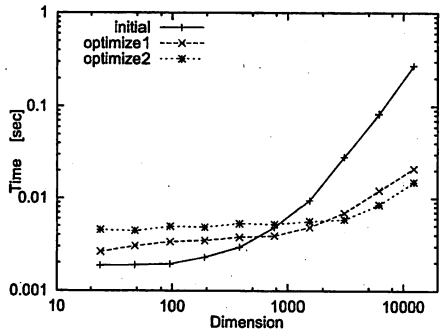

図 9 最適化によるセグメントの処理時間削減
Fig. 9 Decrease of Processing Time of Segments by Optimizations.

5.3 CPU とのパフォーマンス比較

GPU と CPU のパフォーマンスを比較する．ここでは，初期化から全てのセグメントを 1 イタレーション実施したときの処理時間を測定した．このとき，測定対象の次元数は 24 から 98,304 まで変化させた．問題の次元数と処理時間の関係を，図 10 に示す．また，このときの測定環境を表 2 に示す．演算回数と処理時間から，メモリアクセス時間等も含んだ実効演算性能は，7800GTX:540MFLOPS, 7900GTX:675MFLOPS, PentiumD:200MFLOPS となった．

5.4 他の実験との比較

GPU に実装した汎用計算の性能について明確な報告はなされていない．一方，Kruger[2] は，共役勾配

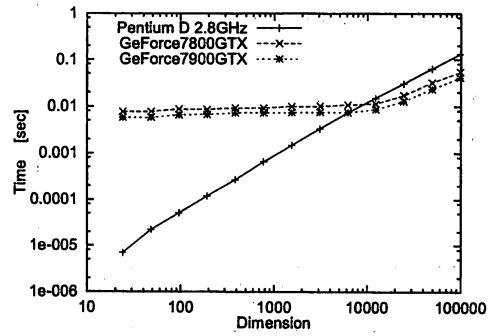

図 10 GPU と CPU の処理時間比較
Fig. 10 Comparison of Processing Time of GPU with CPU.

表 2 実験 2 (GPU と CPU の処理時間比較) を実施した環境
Table 2 Specifications of facility for the experiment 2.

No.	Item	Specifications
1	CPU	PentiumD 2.8GHz
2	GPU	nVIDIA Geforce 7800GTX or nVIDIA Geforce 7900GTX
3	Bus	PCI-Express
4	OS	Windows XP

法による 2 次元波動方程式の解法を GPU に実装し，公開した．この波動シミュレーションと本報告で示す FEM 解法との処理時間の比較を図 11 に示す．これらデータ測定は，表 1 の環境で行った．これによると，波動シミュレーションは，FEM に比較して約 10 倍高速に見える．

両者の性能の相違は，行列における非零要素の密度の差によると考えられる．波動方程式に比較して，我々が行った FEM の剛性行列には約 12 倍の非零要素があり，単純に 12 倍の演算量が必要となる．我々は一般的な行列計算への応用を目指し，行列要素の格納容量を大きくしている．この演算量の差を考慮すると，両者の演算性能は同等と考えられる．

6. 考　　察

CPU 及び GPU での処理時間測定結果を比較すると，CPU では次元の増加に比例して処理時間が増大するが，GPU の処理時間は，次元が小さい領域では一定であり，1 万次元付近から次元数に従って増加した．これは，次元の小さい領域ではセグメント起動のオーバーヘッドが影響することを示している．

セグメントの数を増加させることで，低次元領域での GPU のパフォーマンスは低下するが，各セグメントの処理時間を削減できれば，高次元領域で性能を得ることができる．また，次元数が増加するに従って，GPU の優位性が顕著になると考える．

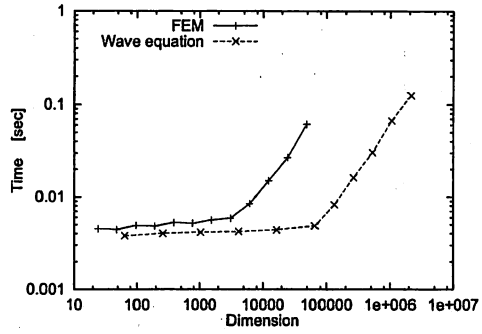

図 11 FEM と波動方程式の処理時間比較

Fig. 11 Comparison of Processing Time of FEM with Wave Equation.

GeForce7800GTX と GeForce7900GTX の処理速度を比較すると，7900GTX が 2 割程度高速であった．GPU のメモリバンド幅の性能が影響していると考えられる．GPU のスペックを表 3 に示す．

表 3 GPU の性能諸元

Table 3 Specification of GPU.

No.	item	Geforce 7800GTX	Geforce 7900GTX
1	core clock	430MHz	650MHz
2	mem clock	1200MHz	1600MHz
3	pixel pipeline	24	24

7. おわりに

本論文においては，有限要素法による 3 次元構造体の変形シミュレーションの実現を最終的な目的として，その計算基盤に関して報告した．GPU に各種数値計算手法を容易に実装可能な仕組みである仮想計算器を構成し，一例として共役勾配法を実装し，処理速度を CPU と比較した．結果，NVIDIA GeForce7900GTX では，Pentium D 2.8GHz への実装と比較して，10 万次元付近では約 3.2 倍の処理速度向上を達成した． また，計算次元数が大規模になる程，この傾向は続き，GPU に有利な結果を得ている．さらに，GeForce7800GTX と GeForce7900GTX を比較すると，GeForce7900GTX が 2 割程度高速となっている．将来的に開発されるであろう高性能な GPU を用いることで，計算性能はさらに向上させることができる．本報告では，GPU への処理の実装は特殊な方法（P バッファ）を用いたが，標準になりつつあるフレームバッファオブジェクトの利用により，将来的にはより高速化が得られると考える．今後，今回の成果を，複数の GPU に適用して，単一 PC 内に複数の GPU を実装し，この PC をクラスタ化して従来の VG クラスタの計算能力を向上させる予定である．

謝 辞

本研究の一部は独立行政法人 情報通信研究機構 (NICT) 民間基盤技術研究促進制度による研究助成で行われた．

参 考 文 献

1) Fan, Z., Qiu, F., Kaufman, A. and Yoakum-Stover, S.: GPU Cluster for High Performance Computing, *SuperComputing 2004*, ACM SIGARCH and IEEE Computer Societ, ACM Press (2004).
2) Kruger, J. and Westerman, R.: Linear Algebra Operators for GPU Implementation of Mumerical Algorithms, *SIGGRAPH2003*, ACM SIGGRAPH, ACM Press (2003).
3) Bolz, J., Farmer, I., Grinspun, E., and Schroeder, P.: Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid, *SIGGRAPH2003*, ACM SIGGRAPH, ACM Press (2003).
4) UJ Kapasi, S. Rixner, WJ Dally, B. Khailany, JH Ahn, P. Mattson, and JD Owens: Programmable stream processors, IEEE Computer, vol.36, no.8, pp.54–62(2003).
5) NVIDIA Corporation: Cg Toolkit User's Manual Release 1.2 (2004).
6) 戸川 隼人：共役勾配法, 教育出版, (1977).
7) Heirich, A. and Moll, L.: Scalable distibuted visualization using off-the-shelf components, *IEEE Symposium on Parallel Visualization and Graphics*, IEEE Computer Society, IEEE CS Press, pp. 55–118 (1999).
8) Lombeyda, S., Moll, L., Shand, M., Breen, D. and Heirich, A.: Scalable interactive volume rendering using off-the-shelf components, *IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, IEEE Computer Society, IEEE CS Press, pp. 115–1158 (2001).
9) Muraki, S., B.Lum, E., Ma, K.-L., Ogata, M. and Liu, X.: A PC Cluster System for Simultaneous Interactive Volumetirc Modeling and Visualization, *IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, IEEE Computer Society, IEEE CS Press, pp. 95–102 (2003).
10) Muraki, S., Ogata, M., Ma, K.-L., Koshizuka, K., Kajihara, K., Liu, X., Nagao, Y. and Shimokawa, K.: Net-Generation Visual Supercompusting using PC Clusters with Volume Graphics Harware Devices, *SuperComputing 2001*, ACM SIGARCH and IEEE Computer Society, ACM Press (2001).
11) Ogata, M., Kajihara, K., Kurita, T. and Fujishiro, I.: Volumetric Computing Graphics Cluster, *VSMM2004*, International Society on Virtual Systems and MultiMedia, IOS Press,Inc., pp. 220–224 (2004).