土木学会論文集B2（海岸工学）
Vol. 66, No.1, 2010, 056-060

数値波動水槽のための3D-MPS法のGPUによる高速化

GPU-accelerated 3D MPS Method for Numerical Wave Flume

堀 智恵実[1]・後藤仁志[2]・五十里洋行[3]・Khayyer Abbas[4]

Chiemi HORI, Hitoshi GOTOH, Hiroyuki IKARI and Abbas KHAYYER

The MPS method has been proven useful in simulating free-surface hydrodynamic flows. Despite its applicability, the MPS method suffers a high computational load. The main objective of this study is to develop a GPU-accelerated MPS code with using CUDA language. Several techniques have been briefly shown to optimize calculations. Some specific three dimensional calculations including a numerical wave flume have been carried out by the GPU-accelerated MPS method. The developed GPU-based code distinctly improves computational efficiency and shows comparable reliability to CPU-based codes.

1. はじめに

数値波動水槽は，海岸構造物の設計に革新をもたらすツールとして認知されつつあるが，砕波・越波・遡上といったviolent flowを安定して計算しようとすると，粒子法が計算エンジンとして有力な選択肢となる．しかし，高解像度3次元計算を実施しようとすれば，粒子法は高負荷であり，実務面での利用を普及するためには，高速並列計算を比較的廉価で実現できる計算環境が不可欠となる．

近年，高速な画像処理を目的として開発されてきたGPU（Graphics Processing Unit）を，科学計算の高速化の援用に利用する取り組みが活発である．粒子法においても，SPH（Smoothed Particle Hydrodynamics）法（Gingold・Monaghan, 1977）に関しては，陽解法型アルゴリズムのGPU計算への適用が容易なことから，既往の研究は多数存在する（例えばMcCabeら，2009）．しかしSPH法では，経験的な人工粘性を用いるため，粘性項の評価に十分な合理性がない．一方，MPS（Moving Particle Semi-implicit）法（Koshizuka・Oka, 1996）では，人工粘性は不要であるが，MPS法の陰解法型アルゴリズムへのGPU適用が必ずしも容易ではないため，GPUによる高速化に関する研究論文は，著者らの知る限りでは存在しない．そこで本稿では，MPS法高速化の手段として，CUDA（Compute Unified Device Architecture；NVIDIA, 2010-05-10 参照）を利用したGPU導入を検討し，MPS法コードをGPU対応コードへと移植するための鍵となる事項を検討し，その高速化に対する効果と計算結果の信頼性を検証する．

1 学生員 修(工) 京都大学大学院博士課程 社会基盤工学専攻
2 正会員 博(工) 京都大学教授 工学研究科社会基盤工学専攻
3 正会員 博(工) 京都大学助教 工学研究科社会基盤工学専攻
4 正会員 博(工) 京都大学講師 工学研究科社会基盤工学専攻

なお，本稿で扱うGPUは，NVIDIA社製GT200シリーズのGPUであり，2010年4月以降に発売されている同社製新世代GPUであるFermiシリーズに比べ，（特に倍精度演算の）処理速度が大きく劣る上，高速化する際の制約が多い．また，CUDAによって作成したコードに関して他社製のGPU上での完全な動作に保障がないこともあらかじめ記しておく．

2. 計算手法

（1）MPS法計算の概要

本稿で使用するMPS法計算コードの流れを，図-1に示す．計算の主要部分は，第一段階の陽的計算（重力・粘性項の計算）と，第二段階の陰的計算（圧力計算）である．圧力解は，連立一次方程式であるPoisson方程式を解くことで得られる．本稿では，GPU計算における実装の容易さを考え，CPUのみの計算の場合，GPUを併用した計算の場合の両者について，圧力計算にはSCG法（対角スケーリング付共役勾配法）を採用している．

（2）開発環境の概要

本稿ではNVIDIA社製Tesla C1060を使用した．最小単位の（単精度）演算ユニットであるストリーミングプロセッサ（SP）を8個搭載したマルチプロセッサ（MP）が，30個集まって1個のGPUが構成される．各MPの中には16KBの共有メモリがある．MP群の外には，4GBのグローバルメモリも搭載されている．Tesla C1060は前述したGT200シリーズのGPUを搭載しているが，ビデオ出力端子が省略されている．GT200シリーズには他に，Geforce GTX280やGeforce GTX285などがあり，これらは全て，グローバルメモリの容量が異なるだけで，プロセッサ数などの構造は，Tesla C1060とほぼ同一である．Geforceにはビデオ出力端子があり，グラフィックボード本来の機能も有し，価格の面でも個人レベルで調達しやすい．

このようにCPUより簡素な造りをした多数のSPを効

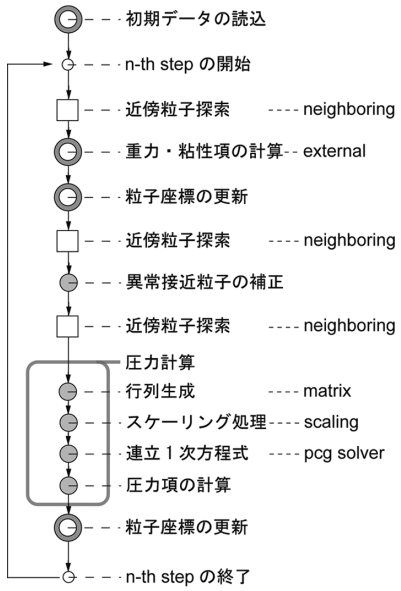

図-1　計算フロー

率的に活用して，多数のスレッドを操ることになる．ソフト構造（NVIDIA, 2009）としては，スレッドがブロックごとに管理され，ブロックの集合はグリッドと呼ばれる．CUDAプログラミング内では，C言語によるCPU用の通常のコードの中から，GPUの処理（カーネル関数）を呼び出す形になる．カーネル関数内の並列度（ブロックおよびスレッドの数）の設定やGPU-CPU間のデータのやり取り等もCPU側に記述する．次節では，移植上問題となる点と，それに対する具体的な対応策を論じる．

（3）CUDAへの移植

a）全体を通した留意点

多数スレッド配備を目的に，1スレッドに1粒子の計算を担当させる．また，アクセススピードがグローバルメモリの100倍以上である共有メモリが，同一ブロック内のスレッドからしかアクセスできないことに留意して，当該カーネル関数内で2回以上，同一ブロックのスレッドからアクセスされる変数は，共有メモリに格納してから計算中にアクセスするようにする．そして，グローバルメモリに保管する物理量，圧力などの1次配列，速度や位置などの多次元配列，および疎行列となる近傍粒子リストや圧力Poisson方程式の係数行列に関しては，スレッドからのアクセスが連続（コアレス・アクセス）となるように配置した．

ところで，本稿で使用するGPUに搭載されている倍精度浮動小数点実数演算ユニットは，各MPに1個ずつしかない．したがって，整数や単精度浮動小数点実数に比べて，倍精度浮動小数点実数の演算性能（処理速度）は格段に落ちる．しかし本稿では，元のCPU用逐次コードで倍精度浮動小数点実数を扱っている部分は全て，CUDAコードでも倍精度で対応している．また，計算実行中，GPU上ではスレッドは32個ずつ（ワープ）にまとめられて処理を行う．ワープ内で処理の分岐が生じると速度に影響するが，本稿では分岐命令において特別な改編は行わない．そしてコード全体は，軽負荷（処理時間が短い）カーネル関数の集合で構成し，グローバルメモリに対するブロック間同期を優先した．

b）近傍粒子探索

従来の逐次計算において効率的とされてきた格子登録法（Gotohら, 2005）を，CUDA版MPS法計算でも採用することとした．格子登録法を実現するためには，それぞれの格子が主体となって，粒子を探索し，その格子の位置に応じたリストに順次格納していくのが正攻法である．本稿では，1つの格子のための計算を，1ブロックに担当させた．したがって，演算量を軽減するため，格子数（＝ブロック数）を必要分だけ確保する（空の格子を極力作らない）処理を付加した．必要数格子の確保には，原田ら（2008）のスライスグリッドを参考にした．その後，格子ごとのリストに登録する作業を，1格子に割り当てられた1ブロックを構成するスレッド群が，その格子に該当する粒子を探索して行う．スレッド群の粒子検索範囲を狭めるため，粒子検索の前処理を2つ追加した．第1は，x座標が昇順になるように粒子番号を並べ替える処理（ソーティング）であるが，これには原田ら（2008）のブロックトランジションソートを適用した．第2は，x方向の各スライスに存在する粒子の番号を任意に各1個記憶しておく処理である．この2つの前処理によって，粒子番号検索が効率化される．

c）圧力のPoisson方程式

共役勾配法では，反復計算中に疎行列・ベクトル積が存在し，これが処理時間の大部分を占める．GPU計算でも同様に，この部分が反復計算のコアであり，その上，単純な移植だけでは，CPUの逐次処理に対して計算速度がそれほど上がらない．処理速度の足止め要因は，メモリアクセス遅延である．演算に用いる作業ベクトルおよび自由表面判定フラッグは，インデックスを介したアクセスなので，コアレス・アクセスが不可能である．さらには，粒子法のランダム性のため，共有メモリに格納しておくこともできない．したがって，グローバルメモリへのアクセス遅延が，計算速度に悪影響を及ぼす．その影響を少しでも抑えるために，この処理をするカーネル関数内では，この2つの変数へのアクセスにテクスチャキャッシュ（後述）を利用した（高井・永井，2009）．

d）テクスチャメモリの利用

GPUには，3Dグラフィックスで使うテクスチャの参照を高速化するために，テクスチャユニットという装置

表-1 計算環境

	only CPU	GPU+CPU
CPU	Intel Core i7 920 2.67GHz	
Main memory	2.49GB	
Graphics Card		NVIDIA Tesla C1060 4GB
Driver Version		185.85
OS	Microsoft Windows XP Professional Ver.2002 SP3	
Programming Language	Fortran	CUDA, C
Compiler	Intel Fortran Compiler 9.1	CUDA 2.1, Microsoft Visual Studio 2005

が実装されており，CUDAでも「テクスチャメモリ」として使用できる．このメモリは読み込み専用でGPU外にあるが，キャッシュされるためアクセスが比較的速い（NVIDIA, 2009）．テクスチャメモリの利用によって，グローバルメモリに対するコアレス・アクセスができない状況でも遅延が抑制されるので，粒子法のようなランダムアクセスが頻発する計算に有効である．コーディングの煩雑性を減じるため，本稿では，次の2つのカーネル関数でのみ，テクスチャメモリを使用した．第1は，前項で述べた圧力のPoisson方程式の反復計算内の疎行列・ベクトル積の演算を行うカーネル関数である．第2は，近傍粒子探索内の最後の処理，すなわち周囲格子を探索して近傍粒子をリストに格納する処理を行うカーネル関数である．この関数では，1粒子のための各スレッドに周囲27格子（3次元の場合）を確認させる命令に際して分岐の深度が大きくなるため，ワープ分岐も多発し，近傍粒子探索アルゴリズム全体の計算時間においてこの処理の比重が大きくなる．

3. 性能検証

（1）概要

本章では，前章での検討を経て作成したMPS法のCUDA対応計算コードと，従来の逐次（1スレッド）計算コードを用いて行ったシミュレーション結果について論じる．便宜上，これ以降，従来の逐次計算およびそれによる結果等を「only CPU」，本報告で作成したCUDA計算コードおよびそれによる結果等を「GPU+CPU」と呼称する．計算環境を**表-1**に示す．GPU+CPU用の特殊な処理を除けば，計算全体のアルゴリズム・計算手法は同一である．ゆえに，計算結果も全く同一になることが期待されるが，本計算結果は全く同一にはならなかった．その要因としては，1）ECC（Error-Correcting Codes）機能の未搭載，2）浮動小数点実数積和処理の自動最適化に際する中間結果の切り捨て（ただし，本稿では可能な限り最適化解除措置をとっている），3）コンパイラの性能，4）総和の方法（逐次足し合わせとリダクション）さらには5）人為的過誤の影響などが挙げられる．

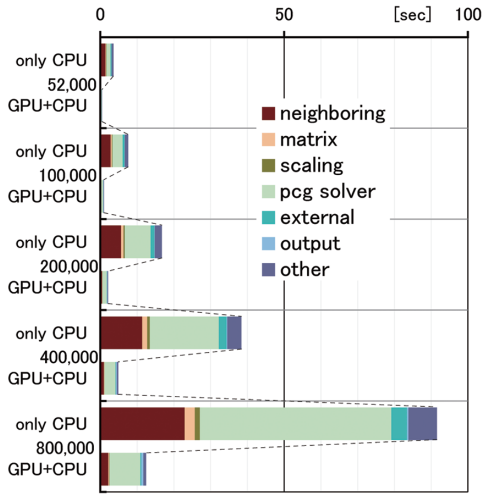

図-2 時間ステップあたりの計算時間（内訳）（3次元）

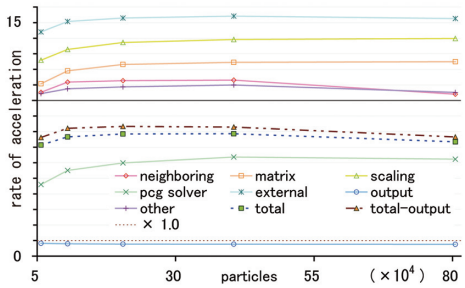

図-3 粒子数に伴う加速率の変化（3次元）

以上のような理由から，長時間積分を経るような計算では，演算エラーが顕在化する可能性がある．次節以降，計算速度の比較だけでなく，計算結果の精度についても重点を置いて議論する．これは，GPU+CPU計算が，only CPU計算と同程度の信頼性を保持し，実務計算に十分適うか否かを確認するために行うものである．

（2）計算結果

a）静水水槽

3次元矩形水槽に上端まで水を満たした静水状態で，100タイムステップを計算実行させた．その計算全体に要した実時間をステップ数100で除し，1ステップにおける計算時間を比較した．水槽のx方向の長さを変えて，総粒子数を52,000個から800,000個まで5段階に変化させた．y-z断面に含まれる粒子数は1,000個である．**図-2**は計算時間内訳の比較を示す．近傍粒子探索と反復計算が計算の大部分を占めることがわかる．**図-3**は，only CPUにおける計算時間をGPU+CPUにおける計算時間で除し

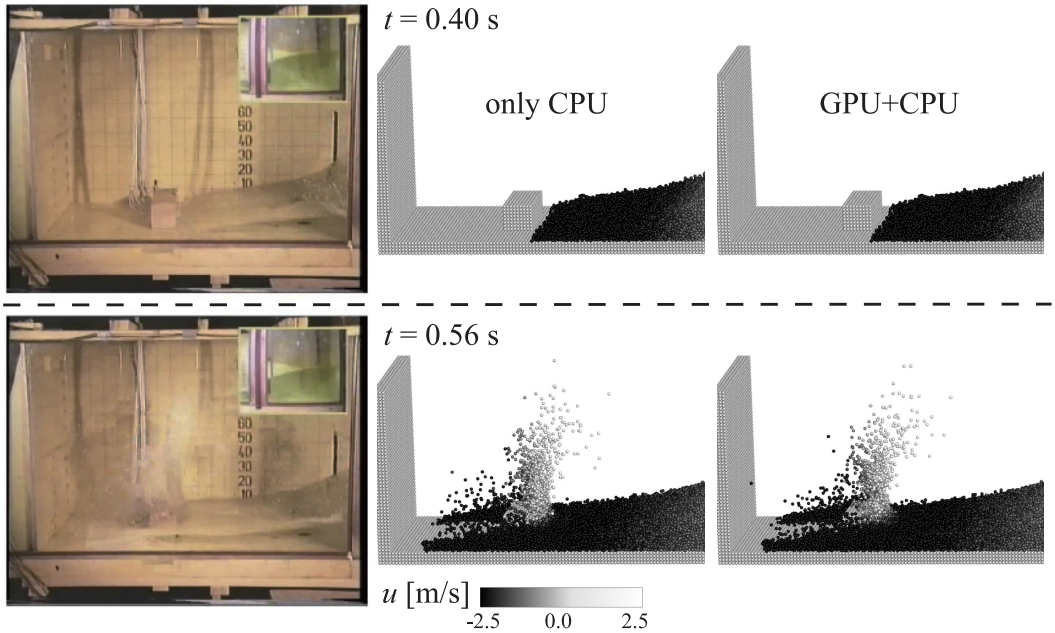

図-4　ダム崩壊流れ瞬間像（写真は，Kleefsmanら（2005）による実験）

た，加速率の変化を表す．200,000個以降はほぼ横ばい，もしくは逆に低下している項目もある．この傾向は，並列度の設定の問題あるいは単体GPUとしての限界などによるものと考えられる．

b）ダム崩壊流れ

水槽底面に突起（固定物体）のある場におけるダム崩壊流れのシミュレーションを行った．計算条件は，Kleefsmanら（2005）の実験条件と同様である．水平長，高さ，奥行がそれぞれ3.22，1.0，1.0mの水槽の右側に，高さ0.55mの水柱が待機した初期状態より，計算を開始し，実験で仕切り板を取り外す動作に対応させた．粒子径は2.0cm，初期状態での総粒子数は214,436個である．シミュレーションにおいても実験と同じく，水槽に天井は無く，側面は4面とも壁粒子を配置した．

図-4は，水槽左側における瞬間像の一例である．参考として，実験写真も挿入した．突起に衝突する前（t=0.40s）までは，only CPU，GPU+CPUの結果は同一であるが，突起に衝突して，進行を遮られた水塊がしぶきを上げながら鉛直上方に立ち上る際（t=0.56s），突起の右領域に散らばる粒子の挙動に差が見られる．実験と同様の地点（初期に水柱が配置されていた地点）で水深を計測した結果を図-5に示す．実験データと完全には一致しないのは，標準MPS法自体に圧力擾乱等の計算精度上の問題（例えば，Khayyer・後藤，2008）があるからである．only CPUとGPU+CPUについて見れば，シミュレーションで計測された値はほぼ同じであり，全体的な水面形状としては同じ結果が得られることを確認できた．

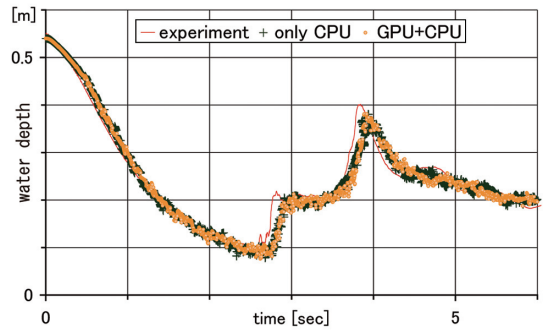

図-5　水深時系列の比較

本計算は物理時間6.0sの計算であり，総タイムステップ数は約12,000となった．計算実行総時間については，only CPUは57.31hrs.である一方，GPU+CPUは6.93hrs.であった．

c）一様勾配斜面遡上砕波

数値波動水槽を構築し，1/10勾配の一様斜面上の巻き波型砕波および遡上のシミュレーションを実施した．沖側の水路端に設けた造波板で，規則波（長尾ら（1997）による水理実験と同様）を造波した．粒子径は2.0mm，総粒子数は240,600個である．なお，側方を周期境界として側壁による抵抗の影響を除去した．図-6は．巻き波発生，水面への衝突および2次ジェットの発達の各瞬間のスナップショットを示す．計算開始後9波目の様子である．GPU+CPUの方が，多少，高速粒子の散らばりが目立ち，t=7.460sにおいては，x軸の正の方向に伸びる2

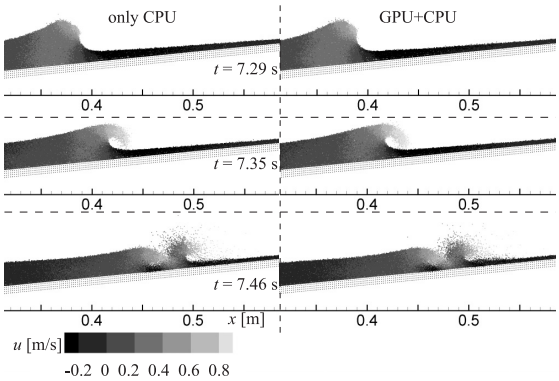

図-6　一様勾配斜面遡上砕波過程瞬間像

次ジェットの下の空気塊が確認できない．次に定量比較として，流速分布を求めた．シミュレーション結果に基づく流速の評価は，奥行き方向に平均して行っている．巻き波発生の瞬間での砕波地点の後方（x=0.37m）における流速分布を実験値と比較する（図-7）．プロットした値は実験値，計算値ともに位相平均値であり，特に計算値は，計算開始後5～9波目の波を対象に平均した．only CPUとGPU+CPUの結果はほぼ完全に一致しており，また，実験値にも良好に対応している．

本計算は物理時間8.0sの計算であり，総タイムステップ数は21,000強となった．計算実行総時間については，only CPUは7.11daysである一方，GPU+CPUは0.696daysであった．計算全体で10倍もの加速効果となったが，これは，x軸方向に伸びる側壁が取り外されたことにより，属性フラッグによる条件分岐回数が減少し，計算時間全体に占めるワープ分岐の発生による遅延の割合が減ったためと推測される．

4. おわりに

本稿では，計算負荷の高いMPS法を高速化する手段のひとつとして，GPU導入に関する基礎的な検討を行った．CUDAへ移植したMPS法計算によるシミュレーションを行い，従来のCPU 1スレッドのみによる逐次計算と，計算速度および精度の面で比較した．計算結果の精度については，流体が大きく変形する段階に至ると，粒子の飛散状況の相違が顕在化したが，大域的に視た水面変動および定量面では概ね一致した．

計算全体における本稿での加速結果は，計算条件に依存して8～10倍程度と，数十倍の加速結果が示されているものも少なくないSPH法を用いたGPU化の既往の研究と比較すれば，見劣りするかもしれない．しかし，MPS法はCPU計算でSPH法の10倍程度高速であり，このことを加味するとGPU-MPS法計算は十分に高速であると言える．

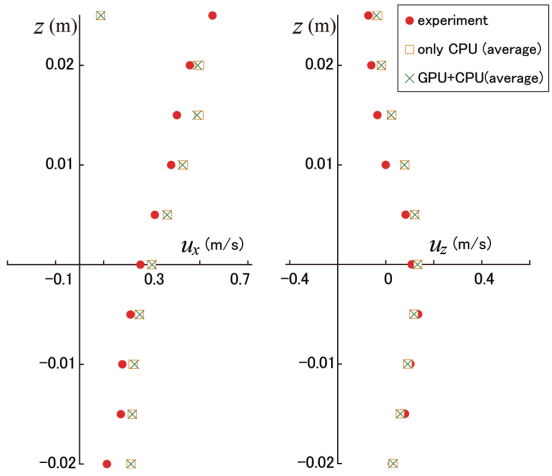

図-7　流速分布の比較

なお本稿では，標準MPS法のみを対象として，移植を行った．今後，高精度粒子法についてもGPU移植が進めば，実務において高い信頼性を有する数値波動水槽の構築につながると期待される．

参　考　文　献

Khayyer Abbas・後藤仁志（2008）：粒子法における圧力擾乱低減のためのCMPS-HS法の提案，海岸工学論文集，第55巻，pp.16-20.

高井陽介・永井学志（2009）：共役勾配法のGPUによる高速化，計算工学講演会論文集，Vol. 14，pp.283-284.

長尾昌朋・新井信一・上岡充男（1997）：PTVとPIVを組み合わせた砕波帯の流速分布測定，海岸工学論文集，第44巻，pp.116-120.

原田隆宏・政家一誠・越塚誠一・河口洋一郎（2008）：GPU上での粒子法シミュレーションの空間局所性を用いた高速化，日本計算工学会論文集，Vol. 2008，Paper No. 20080016, 10p.

Gingold, R. A. and J. J. Monaghan (1977): Smoothed particle hydrodynamics: theory and application to non-spherical stars, *Mon. Not. R. Astron. Soc.,* 181, pp.375-89.

Gotoh, H., H. Ikari, T. Memita and T. Sakai(2005): Lagrangian particle method for simulation of wave overtopping on a vertical seawall. *Coast. Eng. J.* 47(2 & 3), pp.157-181.

Kleefsman, K. M. T., G. Fekken, A. E. P. Veldman, B. Iwanowski and B. Buchner(2005): A volume-of-fluid based simulation method for wave impact problems, *J. Comp. Phys.,* 206, pp.363-393.

Koshizuka, S. and Y. Oka(1996): Moving particle semi-implicit method for fragmentation of incompressible fluid, Nuclear *Science and Engineering,* 123, pp.421-434.

McCabe, C., D. M. Causon and C. G. Mingham(2009): Graphics Processing Unit Accelerated Calculations of Free Surface Flows using Smoothed Particle Hydrodynamics, *proc.of 4th international SPHERIC workshop,* Nantes, France, May 27-29, 2009, pp.384-391.

NVIDIA (2009): NVIDIA CUDA Computed Unified Device Architecture Programming Guide, Version 2.3, 138p.

NVIDIA: CUDA ZONE, http://www.nvidia.com/object/cuda_home_new.html, 2010-05-10 参照