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

2006－EVA－16 (4)
2006／3／20

GPUを装備する計算機を計算資源とするグリッドにおける資源選択手法の検討

小谷裕基† 伊藤信悟†
伊野文彦† 萩原兼一†

本稿では，GPUを装備するPCを計算資源とするグリッドにおいて，資源提供者のアプリケーションに外乱を与えることなく計算資源を監視し，実効性能の高い計算資源を選択するperformance measures手法を提案する．提案手法は，(1) 低オーバヘッドな監視により計算資源が遊休状態か否かを判断し，(2) 遊休状態であると判断する場合のみその実効性能を評価する．提案手法は，(1) のために，スクリーンセーバとして動作した後，ビデオメモリの使用量を調べる．次に，提案手法は，(2) のために，ベンチマークプログラムを実行し，その結果の幾何平均値を計算資源の実行性能とする．実験の結果，提案手法は，高々100ミリ秒で計算資源が遊休状態か否かを判断できた．また，計算資源の実行性能は，実行するアプリケーションにおける処理の内訳を考慮して評価することが重要であることがわかった．

A Resource Selection Method for the GPU Grid

Yuki Kotani,† Shingo Itou,† Fumihiko Ino†
and Kenichi Hagihara †

This paper presents a resource selection method for the GPU Grid, which utilizes the GPU as computational resources of the Grid. Our method aims at selecting appropriate resources with less monitoring overhead to yield higher performance on the Grid. To achieve this less overhead, the method runs as a screen saver program. Once the program is invoked, it checks the VRAM usage to detect idle GPUs, and then measures their performance using benchmark programs. Thus, benchmarks are executed only on idle resources, enabling us to monitor GPU resources at a low overhead. The experimental results show that the monitoring overhead is at most 100 ms. Furthermore, we find that performance measures for the GPU Grid must consider the breakdown of the processing in the Grid application.

1. はじめに

GPU (Graphics Processing Unit)[1),2)]とは，3次元グラフィクス処理の高速化を目的とする演算装置である．近年，著しい性能向上を遂げており，単精度ではあるものの，浮動小数点演算に関してCPUを超える実効性能を達成している．さらに，プログラム可能なGPUが普及するにつれ，従来の描画処理のみならず，汎用処理への応用が注目されている．

我々は，GPUを装備するPCのうち遊休状態のものを計算資源とするGPUグリッドの実現を目指している．GPUグリッドの実現により，アプリケーションを従来より高速に実行できることが期待できる．本研究では，GPUグリッドにおける計算資源を監視し，実効性能の高い計算資源を選択する手法を考える．

計算資源としてGPUを用いる場合，次の3つの問題がある．(a) 資源提供者がGPUアプリケーションを実行している場合，資源提供者のアプリケーションに外乱を与える．(b) 遊休状態か否かを判断する方法が確立されていない．(c) 実効性能を評価する方法が確立されていない．(a) の理由は，GPUは，これまで，占有して使われることが前提となっており，複数のアプリケーションを同時に実行することに適していないためである．(b) の理由は，CPU使用率のような，GPUの負荷状態を表す指標がOSやGPUドライバに実装されていないためである．(c) の理由は，GPUは世代によりアーキテクチャおよび機能の違いが大きいため，および実効性能がCPUなどの状態により動的に変化するためである．

† 大阪大学大学院情報科学研究科コンピュータサイエンス専攻
Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

本稿では，問題 (a) ～ (c) を解決し，GPU グリッドにおける資源選択を実現するために，計算資源において (1) 低オーバヘッドな監視により計算資源が遊休状態か否かを判断し，(2) 遊休状態と判断する場合のみ計算資源の実効性能を評価する手法を提案する．提案手法は，(1) のために，計算資源においてスクリーンセーバとして動作した後，GPU のビデオメモリ (VRAM) 使用量を調べる．次に，(2) のために，遊休状態と判断する計算資源においてベンチマークプログラムを実行し，その結果の幾何平均値を計算資源の実効性能とする．なお，以降では，計算資源が遊休状態か否かを判断する手法を遊休状態判断手法，計算資源の実効性能を評価する手法を性能評価手法という．

以降では，まず 2 章で既存ツールが GPU グリッドにおける利用に適しているか検討する．次に 3 章で計算資源を選択するための手法について説明し，4 章で提案手法の実験の結果を示す．最後に 5 章で本稿をまとめる．

2. GPU グリッドにおける既存ツールの適用

本章では，まず GPU グリッドの構成について説明した後，GPU グリッドにおける計算資源の遊休状態について説明する．次に，既存ツールが GPU グリッドにおける利用に適しているか否かを検討する．

2.1 GPU グリッドの構成

図 1 に GPU グリッドの構成を示す．GPU グリッドでは，提供者，管理者および使用者が存在する．提供者は自らが所有する PC を計算資源としてグリッドに提供する．管理者は提供された計算資源の環境や性能などに関する情報を保持し，アプリケーションを高速に実行できる計算資源を選択し，その情報を使用者に提供する．使用者は管理者から提供された情報に従い，提供された計算資源を用いてアプリケーションを実行する．

なお，以降では，提供者が自らの PC において実行する GPU アプリケーションを PA (Provider Application)，使用者が提供者の PC を用いて実行する GPU アプリケーションを GA (Grid Application) とする．

2.2 計算資源の遊休状態

GPU グリッドにおいては，PA に外乱を与えることなく GA を高速に実行するために，次の状態にある計算資源を遊休状態とし，GA の実行に用いる．(I) 提供者が PA を実行しておらず，かつマウスなどのデバイス操作によりデスクトップ画面上に描画されているアプリケーションウインドウなどを再描画しない状態．

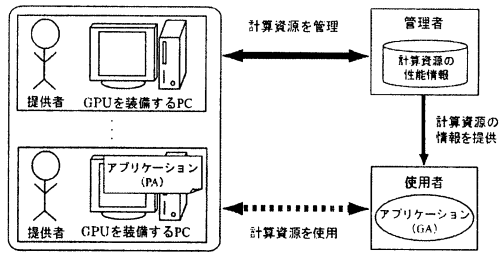


図 1 GPU グリッドの構成

表 1 提供者の作業状況

状況	ディスプレイ画面	PA	提供者の作業例
C1	使用	実行	PC ゲーム
C2	使用	不実行	Web ページの閲覧
C3	不使用	実行	汎用計算用 PA の実行
C4	不使用	不実行	音楽鑑賞

この理由は次の通りである．GPU はこれまで，占有して使われることが前提となっているため，CPU に比べ複数のアプリケーションを同時に実行することに適していない．よって，もし PA 実行中に GA を実行すれば，GA および PA の実行時間が増加する．また，もし GA 実行中に提供者がウィンドウなどを連続的に再描画すれば，GA の実行時間が増加する．したがって，GPU グリッドにおける遊休状態を (I) とする．

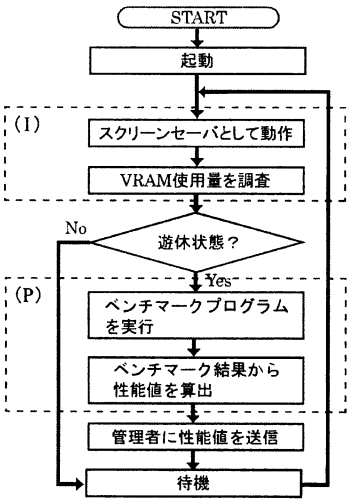
表 1 に，具体的な状況を明らかにするために提供者の作業状況を分類した表を示す．GPU グリッドにおける遊休状態は状況 C4 である．

2.3 既存ツールの検討

我々の知る限り，GPU グリッドにおける資源選択を対象とした研究はない．しかし，GPU が遊休状態か否かを判断するための指標を提供するツール，および GPU の実効性能を評価するための指標を提供するツールは存在する．

GPU の負荷状態をモニタできるツールとして，nVIDIA が提供する NVPerfKit[3] がある．このツールは，nVIDIA 製の GPU を対象とし，単位時間あたりにおける GPU のアイドル時間などを計測できる．しかし，このツールは，対象とする GPU が限られること，および GPU ドライバを変更しなければならない可能性があることなどから，第三者の PC を用いる GPU グリッドにおける利用には適していないといえる．

プログラムの実行により GPU の性能を計測するツールとして，ベンチマークツールが挙げられる．例えば，3 次元グラフィクス処理に関する 3DMark06[4] や，汎用計算に関する gpubench[5] などがある．ベン



(I) 遊休状態判断手法
(P) 性能評価手法

図 2　計算資源における提案手法の流れ

チマークツールは一般に複数の性能を計測する．それぞれのベンチマーク結果は CPU などの状態による実効性能の変化を反映しているため，計算資源における実効性能の評価に用いることができる．しかし，GPU グリッドにおける資源選択に用いるためには，計算資源の実効性能を決定するために必要な性能を選択，計測し，その結果を用いて実効性能を一意に表す必要がある．また，既存のベンチマークプログラムはオーバヘッドが大きいため，PA を実行している状況での実行は，提供者のアプリケーションに外乱を与える．

3. 資源選択手法

本章では，提案する資源選択手法について述べる．**図 2** に計算資源における提案手法の流れを示す．提案手法は，計算資源において起動し，動作する．次に，計算資源が遊休状態か否かを判断し，遊休状態である場合，計算資源の実効性能を評価する．最後に性能値を管理者に送信し，GA の実行に備えて待機する．また，遊休状態でないと判断すれば，遊休状態になるまで待機する．

以降では，遊休状態判断手法，性能評価手法についてそれぞれ詳しく説明する．

3.1 遊休状態か否かの判断

遊休状態判断手法は以下に示す 2 つのステップからなる．

(1) スクリーンセーバとして動作
(2) VRAM 使用量を調査

提案手法は，これらを用いることにより，低オーバヘッドで計算資源が遊休状態か否かを判断できる．

(1) は，提案手法が，提供者がデバイスの操作によりデスクトップ画面上に描画されているアプリケーションウィンドウなどを再描画しない状況を検知することを目的としている．スクリーンセーバを用いる理由は，スクリーンセーバがデバイスからの入力のない場合にのみ動作するからである．

(2) は提供者が PA を実行しているか否かを判断することを目的とする．提供者が PA を実行しているか否かの判断に VRAM 使用量を用いることには次の 3 つの利点がある．(a) 調べる際のオーバヘッドが小さい，(b) 対象とする GPU を限定しない，および (c) Windows にあらかじめ実装されている DirectX の API を用いて調べることができる．(a) ～ (c) より，VRAM 使用量を用いることは，2 章で述べた NVPerfKit を用いる場合に比べ，GPU グリッドに適しているといえる．

VRAM 使用量を調査することで提供者が PA を実行しているか否かを判断できる理由は次の通りである．GPU アプリケーションを実行していないときの VRAM は，ディスプレイ画面の描画のためにのみ用いられている．このときの VRAM 使用量は，画面解像度および色深度など，計算資源の環境によって決まる特定の量となる．したがって，提案手法は，調査した VRAM 使用量と提供者が PA を実行していないときの VRAM 使用量を比較し，両者が等しければ提供者が PA を実行していないと判断できる．

3.2 実効性能の評価

性能評価手法は，計算資源の実効性能を性能値 P として一意に表すために，遊休状態の計算資源においてベンチマークプログラムを実行し，その結果の幾何平均を P とする．

ベンチマークプログラムは，計測の際のオーバヘッドを小さくし，かつその結果から計算資源の実効性能を表せることを目指し，GPU の演算性能および CPU − GPU 間の通信性能を計測対象とする．GPU を用いる汎用計算目的のアプリケーションにおいては，GPU の演算性能は最も重要である．また，CPU − GPU 間の通信はボトルネックになることが多く，重要である．具体的には次の 4 つを計測する．演算性能として，

(B1) ピクセル描画性能
(B2) ジオメトリ処理性能

通信性能として，

(B3) CPU から GPU への通信性能
(B4) GPU から CPU への通信性能

表 2　実験環境

	PC1	PC2	PC3
CPU	Pentium 4 3.4 GHz	Pentium 4 3.0 GHz	Pentium 4 2.8 GHz
GPU	nVIDIA GeForce 7800 GTX	nVIDIA GeForce 6800 GTO	nVIDIA Quadro FX 3400
Core clock	430 MHz	350 MHz	350 MHz
Bus	PCI Express	PCI Express	PCI Express

R1，R2，R3 および R4 をそれぞれ B1，B2，B3 および B4 の結果とする．

P は実行したベンチマーク結果の幾何平均の値とする．幾何平均を取ることで，個々のベンチマーク結果を P に均等に反映する．提案手法は P を次の式 (1) により決定する．

$$P = (R1 \cdot R2 \cdot R3 \cdot R4)^{1/4} \qquad (1)$$

4. 実　　験

提案手法を評価するために 2 つの実験を行った．それぞれについて以下に説明する．

(実験 1)　提案手法が提供者のアプリケーションに外乱を与えないことを検証するために，次の 3 つを比較する．1) 提案手法のオーバヘッド，2) ベンチマークプログラムのオーバヘッド，および 3) PA の実行時間．

(実験 2)　提案手法による性能値 P が資源選択の基準となるか検証するために，P と GA を実行したときのスループットを比較する．なお，ここでいうスループットとは，計算資源が 1 秒間当たりに GA を実行できる回数である．

また，実験結果から，性能評価手法についての課題を述べる．

表 2 に実験環境を示す．実験には次の 3 つの GPU アプリケーションを用いた．係数行列 2048 の LU 分解[6]，係数行列 64 の共役勾配法[7]，および 2 次元／3 次元剛体位置合わせ[8]．これらのアプリケーションを実験 1 では PA として用い，実験 2 では GA として用いた．

4.1　遊休状態判断手法のオーバヘッド

PC1 において，提案手法で監視する状況，ベンチマークプログラムで監視する状況および監視しない状況を想定し，PA の実行時間，提案手法およびベンチマークプログラムのオーバヘッドを計測した．

表 3 に計測結果を示す．提案手法は PA に外乱を与えない低オーバヘッドな手法であるといえる．

まず，提案手法およびベンチマークプログラムのオーバヘッドを比較する．提案手法は高々0.1 秒であるのに対して，ベンチマークプログラムは最大 4.9 秒であることから，提案手法は低オーバヘッドな監視を実現しているといえる．提案手法のオーバヘッドがベンチマークプログラムのオーバヘッドに比べて小さい理由は，提案手法は GPU および CPU などのリソースを消費する時間が短いためである．ベンチマークプログラムは，正しい計測結果を得るために，提案手法に比べて多くの時間をリソースを消費する．

表 3　PC1 での PA 実行中における提案手法およびベンチマークプログラムのオーバヘッド時間

PA	監視なし	提案手法		ベンチマーク	
	T	T	O	T	O
LU 分解	2.5	2.6	0.1	7.4	4.9
共役勾配法	1.7	1.7	0.0	4.4	2.7
位置合わせ	14.2	14.3	0.1	16.5	2.3

T：PA の実行時間（秒）
O：オーバヘッド（秒）

次に，オーバヘッドと監視しない状況における PA の実行時間を比較する．提案手法のオーバヘッドは PA の実行時間に比べて十分小さいといえ，PA に外乱を与えない程度である．一方，ベンチマークプログラムのオーバヘッドは，LU 分解および共役勾配法においては実行時間より大きく，PA に外乱を与える大きさである．

4.2　性能評価手法の性能値

PA として GPU を用いるものおよび用いないものを想定し，CPU 使用率を 0%から 20%刻みに変動させて P およびスループットを得た．

図 3 に CPU 使用率を変動させたときの P を示す．また，**図 4** に CPU 使用率を変動させたときの 3 つの GA のスループットを示す．

CPU 使用率が同じである状況において，P（図 3）および GA のスループット（図 4）を比較すると，実験環境の順序は同じである．したがって，各計算資源の CPU 使用率が同じである状況では，P の高い順に計算資源を選択すれば GA を高速に実行できることから，P を資源選択の基準にできるといえる．

一方，CPU 使用率に関係なく P（図 3）および GA のスループット（図 4）を比較すると，CPU 使用率と実験環境における組み合わせの順序は異なる．例えば，4 番目に大きい値に着目すると，P では CPU 使用率 0%の PC2 であるが，位置合わせのスループット（**図 4(c)**）では CPU 使用率 60%の PC1 である．P および各 GA におけるスループットの間で，CPU 使用率と実験環境における組み合わせの順序がどの程

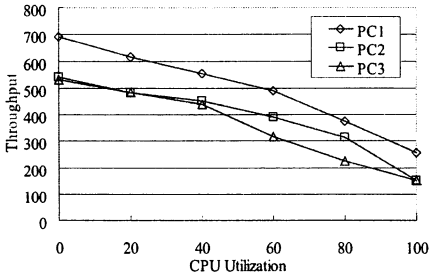


図 3 CPU 使用率を変動させたときの性能値 P

度合致しているかを明らかにするために，P およびスループットにおける順序の一致率を考える．一致率は，各 GA について，スループットにおける順序が P における順序と全て一致するときを 100% とする．図 3 および図 4 より，一致率は，LU 分解が 33%，共役勾配法が 28%，位置合わせが 33%である．P における順序は，どのスループットにおける順序とも 30%前後しか合致していない．したがって，CPU 使用率が関係のない状況では，P の高い順に計算資源を選択しても GA を最も高速に実行できる計算資源が得られないことがあることから，P は資源選択の基準として不十分であるといえる．

4.3 性能評価手法の課題

実験 2 の結果から，P を資源選択の基準とするためには，GA における処理の内訳を P に反映する必要がある．以下に理由を述べる．

GA を高速に実行できる資源を選択するためには，GA について，CPU 使用率の変動によるスループットにおける変化の傾向を知ることが必要である．図 4 から各 GA についてスループットの変化を比較すると，CPU 使用率の変動による変化の傾向が異なる．具体的には，LU 分解（**図 4(a)**）では各環境におけるスループットに差があるのに対し，位置合わせ（図 4(c)）では，PC2 と PC3 の差が小さい．また，LU 分解では各環境とも線形に低下しているのに対して，位置合わせでは低下しているものの線形でない．この理由は，実行時に CPU および GPU の担当する処理量の割合が異なるためであると考えられる．LU 分解は GPU より CPU の担当する処理量の割合が大きいため，CPU 使用率の増加がスループットの低下に結びついている．逆に，位置合わせは CPU よりも GPU の担当する処理量の割合が大きいため，CPU 使用率の増加がスループットの低下に結びつきにくい．したがって，スループットの傾向を知るためには，GA において CPU および GPU の担当する処理量の割合を明らかにし，P に反映する必要があると考えられる．

P への反映は，次の方法などが考えられる．まず，計算資源において，それぞれの処理の内容に対応する実効性能を計測する．次に，その計測結果に対して，対応する処理の全処理に占める割合に従った重みを付ける．最後に，重み付きの計測結果を P の算出に用いる．

まとめると，P を資源選択の基準とするための課題は以下のようになる．

- 実行する GA において CPU および GPU の担当する処理の内容およびその処理の全処理に占める割合を明らかにすること
- 処理の内容それぞれについて対応する実効性能を計測するためのベンチマークプログラムを選定すること
- 計測結果それぞれに対して対応する処理の全処理に占める割合に従った重みを付け，P の算出に用いること

5. おわりに

本稿では，GPU グリッドにおいて (1) 低オーバヘッドな監視により計算資源が遊休状態か否かを判断し，(2) 遊休状態であると判断する場合のみ計算資源の実効性能を評価する手法を提案した．提案手法は，(1) のために，スクリーンセーバとして動作した後，VRAM 使用量を調べる．次に，提案手法は，(2) のために，4 つのベンチマークプログラムを実行し，その結果の幾何平均値を計算資源の実効性能とする．

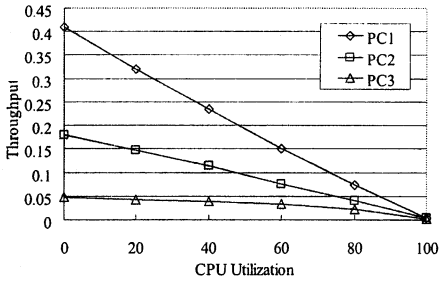
遊休状態か否かを判断する手法のオーバヘッドを測定した結果，高々100 ミリ秒であった．これはベンチマークプログラムのオーバヘッドおよび GPU アプリケーションの実行時間に対して小さく，提案手法は提供者のアプリケーションに外乱を与えない手法であるといえる．また，実効性能を評価する手法における性能値を GPU アプリケーションのスループットと比較した結果，アプリケーションにおける処理の内訳を考慮することが重要であることが分かった．

今後は，資源選択の基準となり得る，計算資源の実効性能を評価する方法の考案に取り組んで行きたい．

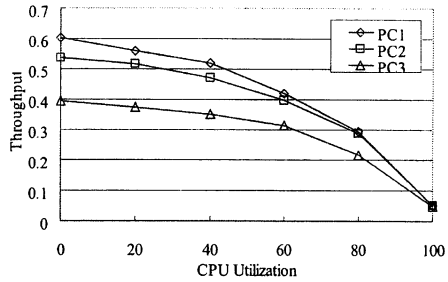
謝辞 本研究の一部は，科学研究費補助金基盤研究 (B) (2) (16300006) および特定領域研究 (16016254) の補助による．

参考文献

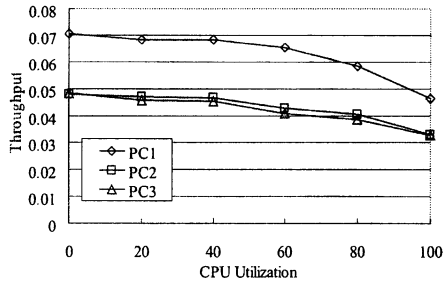
1) Fernando, R.(ed.): *GPU Gems: Programming Techniques, Tips and Tricks for Real-*



(a) LU 分解



(b) 共役勾配法



(c) 位置合わせ

図 4 CPU 使用率を変動させたときの GA のスループット

Time Graphics, Addison-Wesley, Reading, MA (2004).

2) Pharr, M. and Fernando, R.(eds.): *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, Addison-Wesley, Reading, MA (2005).

3) nVIDIA: NVPerfKit (2005). `http://www.developer.nvidia.com/object/nvperfkit_home.html`.

4) Futuremark: 3DMark06 (2006). `http://www.futuremark.com/products/3dmark06/`.

5) Buck, I., Fatahalian, K. and Hanrahan, P.: GPUBench: Evaluating GPU Performance for Numerical and Scientific Application, *Proc. 1st ACM Workshop General-Purpose Computing on Graphics Processors (GP^2'04)*, pp. C–20 (2004).

6) Ino, F., Matsui, M. and Hagihara, K.: Performance Study of LU Decomposition on the Programmable GPU, *Proc. 12th Int'l Conf. High Performance Computing (HiPC'05)*, pp.83–94 (2005).

7) Corrigan, A.: Implementation of Conjugate Gradients on Programmable Graphics Hardware (2004). `http://www.cs.stevens.edu/~acorriga/other_undergrad_projects/gpu_conjugate_gradients/`.

8) 五味田遵，合田圭吾，川崎康博，伊野文彦，萩原兼一：汎用グラフィクスハードウェアを用いた 2 次元／3 次元剛体位置合わせの高速化，情処研報，2005-HPC-104, pp.25–30 (2005).