様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　２７　年　　５　月　２９　日現在

機関番号：　１２６０８
研究種目：　若手研究(B)
研究期間：　2013～2014
課題番号：　２５８７０２２３

研究課題名（和文）低消費エネルギー型ＧＰＵベース次世代気象計算コードの開発

研究課題名（英文）Development of low-power GPU-based next-generation weather prediction code

研究代表者
下川辺　隆史（Shimokawabe, Takashi）

東京工業大学・学術国際情報センター・助教

研究者番号：４０６３６０４９

交付決定額（研究期間全体）：（直接経費）　　3,100,000円

研究成果の概要（和文）：気象計算はスパコンで実行される重要なアプリケーションの一つである。気象計算を最新のGPUスパコンで効率的に実行するためには、複雑な最適化手法を導入する必要がある。本研究では、これらをアプリケーションに簡単に導入できるフレームワークを開発した。ユーザーは、フレームワークを用いることで、従来通りのコードを記述するだけで、GPUスパコンに最適化されたコードを生成できる。これを用い気象庁で開発されている非静力気象モデルASUCAをGPUスパコンへ実装した。これを東京工業大学のGPUスパコンTSUBAME2.5で実行し、高い生産性・可搬性を実現しながら、高い実行性能で高い空間解像度の気象計算を実現した。

研究成果の概要（英文）：Numerical weather prediction is one of the major applications in high-performance computing and is accelerated on recent GPU supercomputers. Skillful programming techniques are required for obtaining good parallel efficiency on these supercomputers. To apply these complicated optimizations to various mesh-based applications easily, we have developed a high-productivity and high-portability framework for multi-GPU computation of mesh-based applications. By using this framework, the programmer can write user code just in a standard language and generate program code optimized for the GPU supercomputers. We have implemented the weather prediction code ASUCA on a multi-GPU platform by using this proposed framework; ASUCA is a high-resolution meso-scale atmospheric model being developed by the Japan Meteorological Agency. The framework-based ASUCA performed on TSUBAME2.5 supercomputer at Tokyo Institute of Technology has achieved high-resolution simulation results with high performance.

研究分野：　格子法に基づいた大規模物理計算

キーワード：　気象計算　高性能計算　GPU　大規模計算　高生産フレームワーク　スーパーコンピュータ

１．研究開始当初の背景

研究代表者らは気象庁が次期気象予報に向けて開発を進めている次世代気象シミュレーションコード ASUCA 全体を GPU で実行する研究に取り組んできた。GPU は元々は画像処理用のプロセッサ（ビデオカード）であったが、ここ数年、これを汎用計算に適用する研究が盛んに行われている。 GPU は従来のプロセッサである CPU と比べて計算の処理能力が非常に高く、消費電力当たりの演算性能が高いため、ペタスケールのスパコンでは GPU を大規模に搭載してきている。

この GPU を使い気象計算できれば、CPU で計算する場合と比べて圧倒的に高速に広域の計算をすることができるため、短時間に精度良く予報（計算）しなければならない天気予報にとって非常に大きなメリットがあり、気象分野では GPU 計算の導入は高い注目を集めている。

２．研究の目的

研究代表者は、気象庁が次期天気予報の現業コードとして開発している ASUCA の完全版をフル GPU 化する。GPU は画像処理を目的に開発されたプロセッサであり低消費電力で高速演算できる。これを汎用計算である気象計算 ASUCA に適用し、(1) 高速かつ高精細な格子で計算することにより気象計算精度が格段に向上することを目的とする。(2) 一般のスパコンはエネルギー消費が大きく、東京工業大学の GPU スパコン TSUBAME で計算することにより従来の CPU を搭載したスパコンの 1/5 の消費エネルギーで同じ予測を可能にする。開発にはディレクティブ（指示文）を挿入するだけで GPU を利用できる技術を積極的に用い、GPU 版 ASUCA の開発を通し、(3) 実アプリケーションでの高い生産性を達成できる GPU 利用技術を確立する。

３．研究の方法

本研究では、次期気象計算コード ASUCA の完成版をフル GPU 化する。GPU 化にはディレクティブを利用する予定であったが、高性能と高生産性を両立できないことが判明したため、 まず (1) 格子計算に GPU スパコン向け最適化手法を簡便に導入できるフレームワークを開発する。(2) このフレームワークを用い、GPU スパコンに最適化された現業運用できる GPU 版 ASUCA を完成させる。そして (3) 東京工業大学にある GPU スパコン TSUBAME 2.0 の 4000 台以上の GPU を用い、高速に高精細な格子を用いた高解像度の気象予測計算を行う。

４．研究成果

気象計算コード ASUCA は気象庁か次期の気象予報のための現業コードとして開発を進めている次世代高分解能局地モデルである。気象計算 ASUCA の GPU 化では、GPU 上で性能を出すために配列の次元を変更し、GPU アーキテクチャに向けた最適化手法を導入する必要がある。また、大規模計算では、通信コストを隠蔽する通信と計算のオーバーラップ手法などの計算手法を導入が必要である。

本研究では、これらの最適化手法を簡便に導入し、高い生産性を実現するフレームワークを開発し、このフレームワークを用いて気象計算 ASUCA を実装した。ASUCA を東京工業大学の TSUBAME2.5 スパコンで大規模高性能に実行することに成功した。これらの成果をスパコン分野の最高峰の国際会議 SC14 に投稿し、採択され、発表した。以下では、具体的な研究成果について述べる。

(1) 研究の主な成果

① フレームワークの概要

本研究で開発したフレームワークは、直交格子型の気象計算を対象とし、各格子点上で定義される物理変数（プログラム上は配列となる）の時間変化を計算する。また、当該物理変数の時間ステップ更新は陽的であり、ステンシル計算によって行われる。TSUBAME に導入されている NVIDIA 社製 GPU で実行することを目指し、実装には、ホストコードは C/C++言語、デバイスコードは CUDA を用いる。

フレームワーク設計における主な目標を述べる。

- プログラマは格子点上での計算についてのみ記述する。格子全体の処理はフレームワークが行う。格子全体の処理がユーザコードからフレームワークへ分離され GPU に合った最適化手法を隠蔽する。また、バックエンドとして様々なプロセッサを採用することができ、拡張性と高い生産性を持つ。現在、フレームワークでは、GPU および CPU で実行可能である。
- フレームワークが提供するテンプレートを用いた C++クラスを用い格子点上の計算を記述する。言語拡張や標準的でないプログラミングモデルを利用すると、既存コードからの乖離も大きく利用しにくい。また、拡張部分がフレームワークを他の環境へ移植する妨げになりうる。その点、C++のテンプレートやクラスは広く使われており、基盤とできる。
- 複数ノードでの GPU 計算に対応するため、ノード間およびノード内通信を簡便に記述するクラスを提供する。このクラスを用いることで、プログラマは MPI や OpenMP を明示的に使用した通信を記述すること必要がなくなる。

フレームワークの詳細について述べる。

② ステンシル計算関数の定義

本フレームワークでは、ステンシル計算は、フレームワークの提供する `ArrayIndex3D` 等

を用い、C++ファンクタとして定義し、ステンシル計算関数と呼ぶ。3 次元の拡散計算では、次のように関数を定義できる。

```
struct Diffusion3d {
__host__ __device__
void operator()(const ArrayIndex3D &idx,
       float ce, float cw, float cn, float cs,
       float ct, float cb, float cc,
       const float *f, float *fn) {
fn[idx.ix()] =
  cc*f[idx.ix()]
+ ce*f[idx.ix<1,0,0>()] + cw*f[idx.ix<-1,0,0>()]
+ cn*f[idx.ix<0,1,0>()] + cs*f[idx.ix<0,-1,0>()]
+ ct*f[idx.ix<0,0,1>()] + cb*f[idx.ix<0,0,-1>()];
}
};
```

ArrayIndex3D は、対象とする配列のサイズ *(nx,ny,nz)* を保持し、ある特定の格子点を表すインデックス *(i, j, k)* を設定できる。対象とする配列が f であるとき、idx を ArrayIndex3D のオブジェクトとすると f[idx.ix()]として使われ、これは配列 f の *(i, j, k)* 点の値を返す。ArrayIndex3D はテンプレートを用いたメンバ関数が定義されており、例えば、idx.ix<+1,0,0>()、idx.ix<-1,-2,0>()とすると、*(i+1, j, k)*、*(i-1, j-2, k)* を表すインデックスを返す。テンプレートを用いることで、GPU および CPU でのインデックス計算の高速化を図っている。

ステンシル計算関数の第一引数は固定で、計算対象となる格子のインデックス情報を持つ idx を受け取らなければならない。関数実行時には、格子点 *(i, j, k)* の値が設定されているため、*(i, j, k)* を中心としたステンシル計算を関数内に記述する。f, fn は配列へのポインタであり、これに対しステンシルアクセスすることとなる。

③ ステンシル計算関数の実行

本フレームワークは、全ての格子点に計算を行う Loop3D 等のクラスを提供する。これを用い、ステンシル計算関数の実行は以下のように行う。

```
Loop3D loop3d(nx+2*mgnx, mgnx, mgnx,
              ny+2*mgny, mgny, mgny,
              nz+2*mgnz, mgnz, mgnz);
loop3d.run(Diffusion3d(), ce, cw, cn, cs,
           ct, cb, cc, f, fn);
```

Loop3D はステンシル関数を適用する範囲指定するパラメータで初期化する。Loop3D::run()は任意個の異なる型を引数にとるテンプレート関数として定義されている。C++のテンプレート関数の型推論を利用し、Loop3D::run()は、与えられた全ての引数を第二引数以降に持つファンクタ Diffusion3d()を呼び出す。ファンクタは、NVIDIA CUDA の__host__、__device__で定義することができ、ホスト、デバイス両方へコンパイルされており、Loop3D::run()が CPU 上のデータに対しても、GPU 上のデータに対しても同じ関数を実行する。Loop3D 内部の実装としては、CPU 上で実行する場合はファンクタを全格子点に対して for 文で実行し、GPU 上で実行する場合は内部で生成される CUDA のグローバル関数に包み、適当な CUDA の block 数、thread 数が渡され全格子点に対して実行される。第二引数以降で初めて出てくるポインタが GPU メモリの配列を指すか CPU メモリの配列を指すかを判定し、GPU で実行するか CPU で実行するかを自動的に決定する。

④ 変数の GPU 間通信

複数 GPU による格子を用いた計算では、図 1 に示す領域分割法を用いる。領域分割法では、隣接領域間の境界領域の交換が必要であり、配列として確保された変数は隣接 GPU から送信されたデータを格納する境界領域を持つ。本フレームワークでは、この隣接領域間の境界領域を交換するための GPU 間通信を簡単に記述するためのクラス BoundaryExchange を提供する。複数 GPU における計算を効率的に行うため、フレームワークは、ノード内並列では OpenMP で並列化し、GPU 間の通信を効率的に行うため GPUDirect による peer-to-peer 通信を用いる。一方、ノード間並列では MPI を利用する。MPI によるノード間通信では、転送のための一時領域をホストメモリに確保している。

クラス BoundaryExchange を用いて、以下のように GPU 間通信を行う。

```
BoundaryExchange *exchange = domain.exchange();
exchange->append(array1);
exchange->append(array2);
exchange->append(array3);
exchange->transfer();
```

domain はクラス Domain のオブジェクトで、計算領域サイズ、隣接ノード番号等を保持している。クラス BoundaryExchange は、Domain を参照することで計算領域サイズ等を取得している。これらの情報を用いることで、exchange->append()で登録された変数 array1, array2, array3 に関して exchange->transfer()で境界領域を転送する。この関数内で、OpenMP, MPI を用いた転送が行われる。

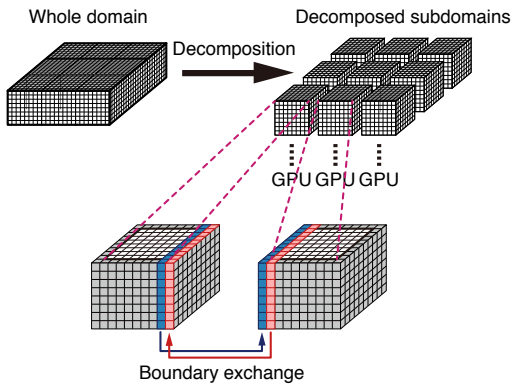

図 1 複数 GPU による格子に基づいた計算

⑤ 通信と計算のオーバーラップ手法

大規模計算では、GPU 間の通信時間は全実行時間に対して無視できない。通信コストを計算で隠蔽するオーバーラップ手法の導入を大規模計算における性能向上にとって重要となる。

本フレームワークは、カーネル分割によるオーバーラップ手法を提供する。この手法は、ある物理変数内でのデータの非依存性を利用する。ある変数のそれぞれの要素は他の要素の計算と独立に計算できるため、1 つの GPU が担当する計算領域を境界領域と残りの中心領域に分割して、独立に計算することが可能である。図 2 にオーバーラップ手法の計算方法を示す。オーバーラップ手法では、中心領域の計算を開始し、それと同時に境界領域の計算に必要なデータを取得するための通信を行う。通信が終了した後、境界領域の計算を行う。これによって、通信の隠蔽を行う。

本フレームワークでは、ユーザコードにカーネル分割によるオーバーラップ手法を適用するために、クラス CompCommBinder を提供する。これを用い、拡散計算では、次のようにオーバーラップ手法を記述できる。

```
BoundaryExchange *exchange = domain.exchange();
exchange->append(f);
CompCommBinder<Loop3D> ccbinder(exchange);
ccbinder.set_post_func(&loop,
    create_funcholder<Loop3D>(Diffusion3d(),
       ce, cw, cn, cs ,ct, cb, cc, f, fn));
    ccbinder.set_use_overlapping();
    ccbinder.run();
```

CompCommBinder は、Loop3D による計算範囲ステンシル関数と GPU 間通信を制御する BoundaryExchange オブジェクトによって初期化される。CompCommBinder は、Loop3D を境界領域と中心領域に対応する複数の Loop3D へ分割し、これらの Loop3D でステンシル関数を実行する。BoundaryExchange による通信を適当なタイミングで行い、この通信を隠蔽する。

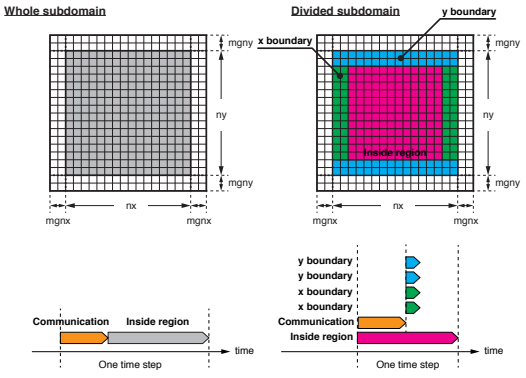

図 2　カーネル分割による計算と通信のオーバーラップ手法

⑥ フレームワークを用いた気象計算 ASUCA の性能評価

本研究では、気象計算 ASUCA を本フレームワークを用いて実装した。東京工業大学の TSUBAME2.5 に搭載された NVIDIA Tesla K20X GPU を複数用い、性能測定を行う。

ASUCA (Asuca is a System based on a Unified Concept for Atmosphere) は気象庁か次期の気象予報のための現業コードとして開発を進めている次世代高分解能局地モデルである。ASUCA は一般座標系を採用し、力学過程の方程式系はフラックス形式の完全圧縮・非静力学方程式系である。ASUCA では鉛直方向の音波関連項のみを陰的に扱う HEVI (Horizontally explicit-Vertically implicit) 法を採用している。風速などに比べて伝播速度の速い音波、重力波に関係する項はサブステップを用いて小さい時間刻みで計算し、移流計算や物理過程など、その他の項は大きい時間刻みで計算する。小さい時間刻み、大きい時間刻みともに 3 段階 Runge-Kutta 法を用いている。現在では、気象庁で運用中の気象モデル JMA-NHM と同等の雲物理過程が導入されている。

図 3 に現在の数値予報で使用されている初期値データと境界値データを用いた台風の気象計算を行った例を示す。TSUBAME2.5 の 672 GPU を用い計算格子 5,376×4,800×57 (実際の格子間隔は水平 500m) で計算している。

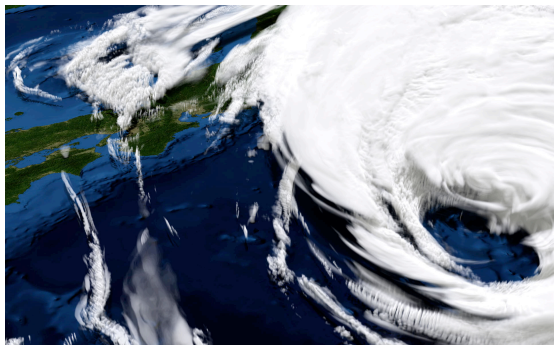

図 3　TSUBAME2.5 の 672GPU を用いた台風の気象計算の例。計算格子 5,376×4,800×57 を用い水平解像度 500m で計算している。

図 4 に強スケーリングの結果を示す。オーバーラップ手法を用いた場合、用いない場合の性能を示す。参考として 1 つの GPU を 1MPI プロセスで計算する Flat-MPI の結果も合わせて示す。計算格子 1536×1280×60 および計算格子 3072×2560×60 を単精度計算した。TSUBAME の各ノードには 3GPU 搭載されているが、この計算ではこのうち GPUDirect で通信可能な 2GPU を用いている。オーバーラップ手法を用いたことによる性能向上がみられ、計算格子 3072×2560×60 の 512GPU 計算では、18.9 TFlops を達成した。

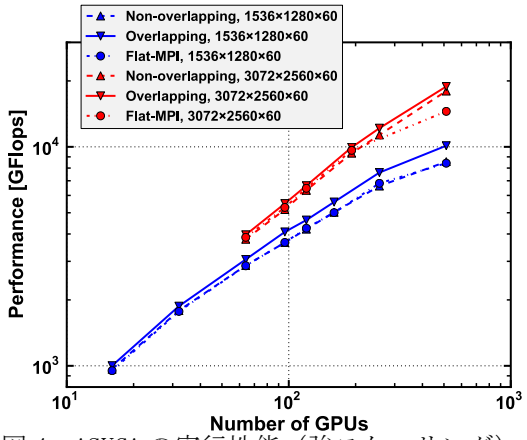

図 4　ASUCA の実行性能（強スケーリング）

図 5 に弱スケーリングの結果を示す。1 GPU あたり計算格子 768×128×60 を用い単精度計算した。強スケーリングの測定と同様、オーバーラップ手法と用いた場合と用いない場合 の測定結果を示す。強スケーリングの測定とは異なり、最大の実行性能を得るため、各ノードに搭載された 3GPU 全てを使用した。オーバーラップ手法を用いることで性能向上がみられ、4,108 GPU で 209.6 TFlops を達成した。

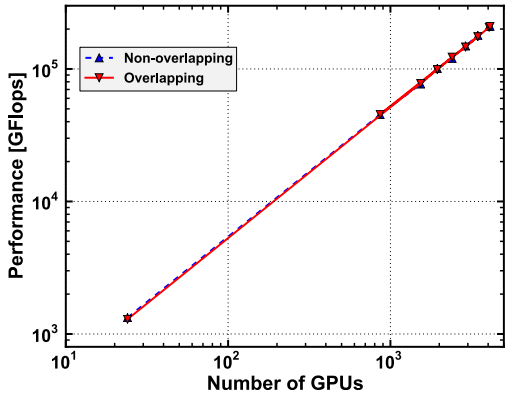

図 5　ASUCA の実行性能（弱スケーリング）

(2) 得られた成果の国内外の位置づけとインパクト

本研究では、格子計算用のフレームワークを完成させ、これを用い GPU 版 ASUCA を完成させ、東京工業大学のスパコン TSUBAME2.5 で大規模実行に成功した。

構築したフレームワークを用いると、通常のプログラムコードを記述するだけで、GPU スパコンに必須な最適化をアプリケーションへ簡便に導入することができる。また、大規模 GPU 計算で重要となるノード間の通信を隠蔽する計算手法に対応した。従来の手法を拡張し、ノード内の通信では GPU 同士で直接通信する手法を新たに開発・導入し、性能を向上させることに成功した。

構築したフレームワークを用い、GPU 版 ASUCA を実装した。大気の流体計算（力学過程）のコード全体と物理過程モジュールの一部を実装し、東京工業大学のスパコン TSUBAME2.5 へ最適化した。フレームワークに基づいた GPU 版 ASUCA によって、TSUBAME2.5 の 4108 GPU を利用した大規模気象計算を高性能に実行することに成功した。また、気象庁で用いられている実際の入力データを用い、TSUBAME2.5 上で水平解像度 500m という極めて高い空間解像度の気象計算を実現した。

これらの成果をスパコン分野の最高峰の国際会議 SC14 に投稿し、採択され、発表した。本研究で開発したフレームワークは、高性能アプリケーションを高生産に開発できる技術として注目されている。本フレームワークは、気象計算以外の格子に基づいたシミュレーションに適用することが可能となっている。フレームワークの構築とこれを用いた ASUCA の開発を通して、GPU アプリケーションを高生産に開発する方法について有益な知見が得られた。

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 2 件）

① Takashi Shimokawabe, Takayuki Aoki, and Naoyuki Onodera, ″High-productivity Framework on GPU-rich Supercomputers for Operational Weather Prediction Code ASUCA,” in Proceedings of the 2014 ACM/IEEE conference on Supercomputing (SC’14), New Orleans, LA, USA, Nov 2014, pp. 1-11. 査読有
http://dx.doi.org/10.1109/SC.2014.26

② 下川辺隆史, 青木尊之, 小野寺直幸, ″複数 GPU による格子に基づいたシミュレーションのためのマルチ GPU コンピューティング・フレームワーク,″ 2014 年ハイパフォーマンスコンピューティングと計算科学シンポジウム HPCS2014, 東京, 2014 年 1 月. 査読有
http://id.nii.ac.jp/1001/00096905/

〔学会発表〕（計 9 件）

① Takashi Shimokawabe, ″A High-Productivity Framework for Multi-GPU Computing of Weather Prediction Code ASUCA,″ GTC 2015, San Jose, CA, USA, Mar 17, 2015. (Poster, GTC Poster Award finalist)

② 下川辺隆史, 青木尊之, 小野寺直幸, ″GPU スパコンによる格子に基づいたシミュレーションのための GPU コンピューティング・フレームワーク″, AXIES 大学 ICT 推進協議会 2014 年度 年次大会, AER（アエル）ビル（宮城県・仙台）, 2014 年 12 月 12 日.

③ 下川辺隆史, 青木尊之, 小野寺直幸, ″ステンシル計算のための GPU コンピューティング・フレームワークのチューニング高度化″, 第 205 回 ARC・第 147 回 HPC 合同研究発表会 (HOKKE-22), 小樽経済センターホール (北海道・小樽), 2014 年 12 月 10 日.

④ Takashi Shimokawabe, ″High-productivity Framework on GPU-rich Supercomputers for Weather Prediction Code,″ CREST-ECRC workshop, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia, Sep. 4, 2014.

⑤ Takashi Shimokawabe, “ A High-productivity Framework on GPU-rich Supercomputers for Mesh-based applications,” Workshop on HPC and Cloud Accelerators, 理化学研究所計算科学研究機構 (兵庫県・神戸)、2014 年 8 月 26 日.

⑥ 下川辺隆史, 青木尊之, 小野寺直幸, ″GPU コンピューティング・フレームワークを用いた気象計算コードの開発″, 日本計算工学会 第 19 回計算工学講演会論文集, 広島国際会議場(広島県・広島), 2014 年 6 月 11 日.

⑦ Takashi Shimokawabe, Takayuki Aoki and Naoyuki Onodera, ″A High-productivity Framework for Multi-GPU computation of Mesh-based applications,″ First International Workshop on High-Performance Stencil Computations (HiStencils), Vienna, Austria, Jan 21, 2014.

⑧ Takashi Shimokawabe, Takayuki Aoki and Naoyuki Onodera, ″A High-productivity Framework for Weather Prediction Code on Multi-GPU Computing,″ the 5th Asia Pacific Congress on Computational Mechanics (APCOM2013), Singapore, Singapore, Dec 11, 2013.

⑨ 下川辺隆史, 青木尊之, 小野寺直幸, ″気象計算コードのための GPU コンピューティング・フレームワーク,″ 日本機械学会第 26 回計算力学講演会, 佐賀大学 (佐賀県・佐賀), 2013 年 11 月 2 日.

〔その他〕
ホームページ等
http://www.sim.gsic.titech.ac.jp/Japanese/Member/shimokawabe/index-ja.html

６．研究組織
(1)研究代表者
下川辺　隆史 (SHIMOKAWABE, Takashi)
東京工業大学・学術国際情報センター・助教
研究者番号：40636049