Vol.2019-HPC-170 No.22
2019/7/25

# GPU・FPGA 複合演算加速による輻射流体シミュレーションコード ARGOT の実装

中道安祐未[1]　藤田典久[2]　小林諒平[2,1]　朴泰祐[2,1]　吉川耕司[2,3]　梅村雅之[2,3]

**概要**：近年，高性能コンピューティング ( HPC : High Performance Computing ) の分野において，アクセラレータを搭載した大規模計算クラスタが主流の 1 つとなっている．アクセラレータには，主に Graphics Processing Unit (GPU) が用いられているが，HPC 分野では処理の柔軟性や電力効率の高さから Field Programmable Gate Array (FPGA) が注目されつつある．そこで，GPU が不得意な計算を FPGA に行わせる GPU+FPGA の複合システムにより実アプリケーションのさらなる高性能化を目指す．前回の発表では，GPU と FPGA の両方を搭載した計算機で GPU+FPGA のハイブリッドアクセラレーションを実現するプログラムの開発手法と環境について議論した．GPU・FPGA の両デバイスを協調する方法を確立したため，本研究では，その方法を用いて輻射流体シミュレーションコード ARGOT の実装を行う．従来は CPU・GPU を用いて高速化が行われていたが，アルゴリズムの特性より，本研究では FPGA を用いた方がより高速化できるアルゴリズムに対して OpenCL による実装を用いたソースコードを組み込んだ．実装にはまだ至ってはいないが，実装に対する議論を行う．

## 1. はじめに

近年, HPC 分野では GPU などのアクセラレータを搭載した大規模計算クラスタが主流の 1 つとなっている．世界のスーパーコンピュータの性能ランキングである TOP500[1] を参照すると，上位 10 位のうち 5 つのシステムがアクセラレータを搭載している．

GPU は CPU よりも高い電力効率と演算性能を有しているが，条件分岐により性能が大幅に低下する，データレベルの並列性が低いと GPU の性能を発揮できない，並列化をしたときノードをまたぐ GPU 間の通信コストが高いなどのデメリットを抱えている．

その一方で，近年，FPGA の HPC 分野へのアクセラレータとしての応用が注目されている．FPGA は，内部の論理回路の構造を何度も繰り返し再構成可能なハードウェアで回路規模が数百万ゲートの大規模なものも存在している．特に近年の高性能 FPGA 間は CPU を介さずにデバイスからのダイレクトな高速光通信を可能とするインタフェースを備えたものもあり，FPGA が主体的に高速通信を行うことができる．しかし，アプリケーションの実行時には，動的再構成を取り入れなければ計算リソースが限られてしまう．これらの問題を解決し，FPGA と GPU を相補的に活用するため，我々は図 1 のように GPU と FPGA が PCI express (PCIe) で接続された GPU+FPGA 複合システムに関するフレームワークを開発した [2].

本研究では, ARGOT の高速化について考える．ARGOT は，ARGOT 法と ART 法の 2 種類によって計算される [3]. 現在，その 2 つの手法は CPU と GPU を用いて高速化することで実行されている．先行研究では，ART 法の計算に FPGA を用いることで高速化が実現している [4]．そこで，従来通り ARGOT 法は CPU・GPU で計算を行い，ART 法は FPGA で計算を行うことで ARGOT の高速化を図る．

我々は，これまでの研究で，GPU と FPGA という異なるデバイスが共存する環境上で，2 つの異なるコンパイル環境を使い分けて 2 つの異なる言語を協調させる方法を見つけ，動作を確認している [2]．本研究では，この環境を輻射流体シミュレーションコード ARGOT に実装する．

本稿の構成を以下に示す．第 2 章では関連研究，第 3 章では輻射流体シミュレーションコード ARGOT，第 4 章では実装方法について述べる．第 5 章では評価環境，最後に，第 6 章にて本研究をまとめ、今後の課題とする．

## 2. 関連研究

筑波大学計算科学研究センターでは，CPU を介さない GPU 間直接通信機構として TCA (Tightly Coupled Accel-erators) アーキテクチャを提唱し，その実装とし

[1] 筑波大学 システム情報工学研究科
[2] 筑波大学 計算科学研究センター
[3] 筑波大学 数理物質科学研究科

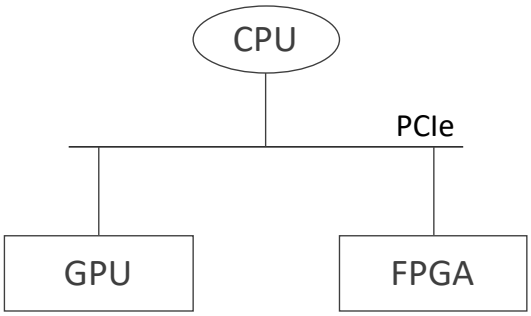


図 1　ハードウェアの構成

て PEACH2 (PCI Express Adaptive Communication Hub version 2) を FPGA により開発した [5]．PEACH2 を経由して通信を行うことで，GPU 同士の通信を高速に行えるが，GPU 間の通信途中に CPU による演算が必要になると，PEACH2 の性能を生かすことができない．本来通信機構としてしか用いられていない PEACH2 に追加要素として演算機構を取り付け，ノード間通信を行う際に CPU で行う処理をオフロードする．

Axel クラスタは，各ノードに FPGA，GPU など複数の種類のアクセラレータを含んだシステムである [6]．ノード内・ノード間通信にはシステムバスが用いられている．Axel クラスタのための Map-Reduce フレームワークによっ て，異なるタイプの処理要素と通信チャネルを通して空間的および時間的局所性を利用した結果，N 体シミュレー ションの性能が 4.4 倍から 22.7 倍向上した．

## 3. 輻射流体シミュレーションコード ARGOT

ARGOT (Accelerated Radiation transfer on Grids using Oct-Tree) [3] とは，筑波大学 計算科学研究センター (CCS) で開発されている宇宙輻射輸送を解くプログラムである．光子の輻射輸送 ( RT : Radiative transfer ) は銀河，星，ブラックホールなどの天体の形成に重要な問題である．ARGOT プログラムは，2 つのアルゴリズムである ARGOT 法と ART 法を組み合わせて解く．以下で，ARGOT 法と ART 法について説明する．

### 3.1 ARGOT 法

ARGOT 法は，点光源からの輻射輸送を計算する．多くの光源からレイトレーシングを計算するアルゴリズムである．ARGOT は光源の分布を八分木のデータ構造で扱う．離れたツリーノード内の光源は単一の光源として扱うことができるため，計算を行う光源の数を $N$ から $logN$ に減らすことができる．メッシュの各グリッドで各放射線源から来る光子束は以下の式 1 で表すことができる．

先行研究では、ARGOT 法を GPU で実行する場合、図 2 のような処理手順となる。また、CPU を介さない GPU 間直接通信機構として TCA (Tightly Coupled Accel-erators)

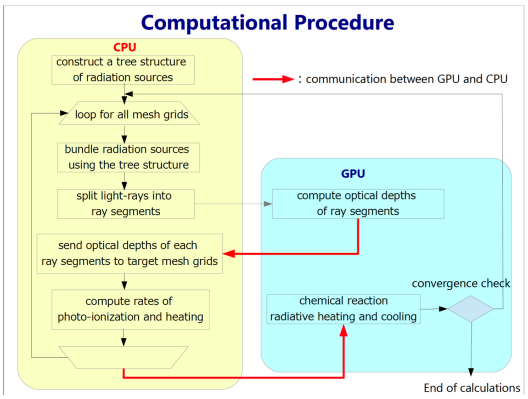


図 2　ARGOT 法の処理手順

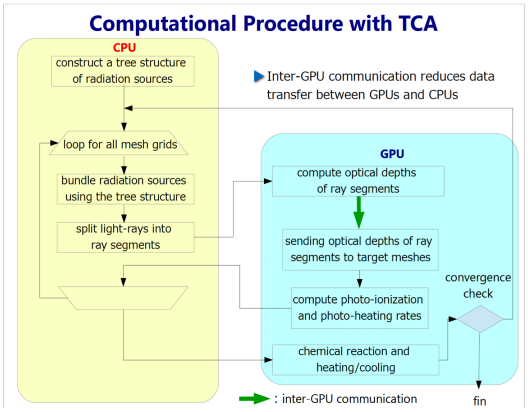


図 3　ARGOT 法の TCA を使った処理手順

アーキテクチャを用いる場合は図 3 のような処理手順となる。TCA を用いると GPU でレイセグメントの光学的深さを計算したあと、CPU にデータを戻して光イオン化と加熱率を計算する必要がなくなり、GPU で行うことができる。

$$f(\nu) = \frac{L(\nu)e^{-\tau(\nu)}}{4\pi r^2} \tag{1}$$

このとき，

- $L(\nu)$：振動数 $\nu$ での光源の光度
- $\tau(\nu)$：振動数 $\nu$ での光学的距離
- $n(\boldsymbol{x})$：光を吸収するガス分子の数密度

である．$\tau(\nu)$ は，以下で求められる (式 2)．

$$\tau(\nu) = \sigma(\nu)\int n(\boldsymbol{x})dl \simeq \sigma(\nu)\sum_i n(\boldsymbol{x}_i)\Delta l \tag{2}$$

ノード並列化では，シミュレーション空間を各次元に均等に分割する．複数のノードにまたがる光線は，ノード間の境界で「レイセグメント」分割し，各セグメントの計算が異なるノードで並列して行われる．そして，各セグメントの光学的厚みの計算結果の和を求めて全体の計算結果を求める．

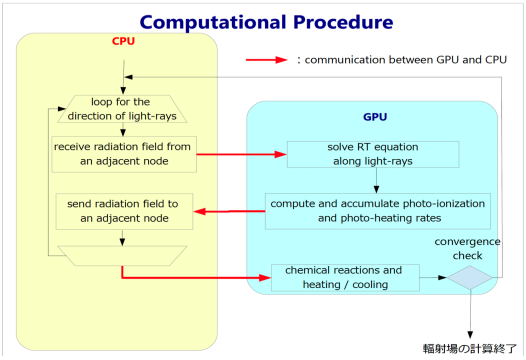


図 4　ART 法の処理手順

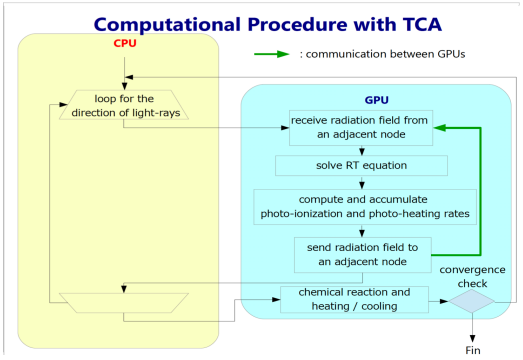


図 5　ART 法の TCA を使った処理手順

### 3.2　ART 法

ART 法は，空間に広がる光源からの輻射輸送を計算する．この計算時間は ARGOT の全体の計算時間の 90%以上を占めている．そのため，この部分を FPGA を用いて高速化を図ることで全体時間の高速化につながる．問題空間の一方の端から他方の端までの平行光線に沿って放射伝達方程式を解く．計算された放射場に基づいて，各メッシュグリッドについて反応速度および光加熱速度を計算する．複数の光線がメッシュを通過するので，それらのメッシュ内の物理量はアトミックな方法で増加されなければならない．ART 法ではレイトレーシングを行っており，各レイが進行方向に向かって順々に計算される．その一方で，異なるレイの間に計算の依存関係がないので，どの順序で計算を行っても問題はないため，並列化が容易である．しかし，レイが進む角度によってメッシュデータへのアクセスパターンが変化すること，レイの角度が全球に均一に分布していることから CPU や GPU のような SIMD 型の計算機で高速に計算することが難しい．FPGA は，内蔵メモリに低レイテンシ高バンド幅でアクセスでき，そのアクセスパターンも自由にプログラムが可能となっている．

先行研究では、ART 法を GPU で実行する場合、図 4 のような処理手順となる。また、CPU を介さない GPU 間直接通信機構として TCA (Tightly Coupled Accel-erators) アーキテクチャを用いる場合は図 5 のような処理手順となる。TCA を用いると、隣接ノードから放射線場のデータを受け取り、レイトレーシングを計算し、光イオン化と加熱率の累積を行い、その結果を隣接ノードの放射線場に送る処理を CPU を介さずに GPU のみで行うことが可能となる。

また，ART 法の計算式は以下の通りである (式 3).

$$I_\nu^{out}(\hat{\boldsymbol{n}}) = I_\nu^{in}(\hat{\boldsymbol{n}})e^{-\Delta\tau_\nu} + S_\nu(1 - e^{-\Delta\tau_\nu}) \qquad (3)$$

このとき，

$\Delta\tau_\nu = \sigma(\nu)n\Delta l$：光路長 $\Delta l$ に対する光学的深さ

である．

## 4.　実装方法

従来は，ARGOT 法も ART 法も CPU・GPU で計算が行われていた．しかし，ART 法はアルゴリズム特性により，CPU で計算を行うよりも FPGA で計算を行った方がより高速化につながることが先行研究により実証されている [4]．また，他の研究では次のことがわかっている．GPU 実行では問題サイズに十分な並列性があることが優れた性能を発揮するための条件となるが，FPGA 実行ではすべての問題サイズで優れた性能を発揮することができる．大きな問題サイズの場合の FPGA 実行は GPU 実行とほぼ同じ性能を発揮することも実証されている [7]．したがって，FPGA 実行の基本性能は GPU と比較して十分であると考えられる．

FPGA 実行のための ART 法は OpenCL が使用されている．そのため，ART 法の GPU での演算部分を FPGA で行うことができるようにソースコードを一部書き換えた．このとき，ARGOT は図 6 のように構成されており，先行研究のソースコードを ART 法を計算している部分に組み込んだ．先行研究では，ART 法はホストコードが C++で記述されていたが，ARGOT は C 言語で記述されているため C 言語に書き換えた．組み込んだ部分は，図 7 に示した calc_difuse_photon_radiation.c 内の ray_tracing_loop_st 関数内である．ray_tracing_loop_st 関数は argot_mpi.c 内の step_radiation_tree 関数からさらに step_radiation_tree.c 内の calc_diffuse_photon_radiation 関数内で呼ばれている．

従来では，CPU・GPU を用いたプログラムであったが，今回は FPGA も用いるプログラムとなっている．FPGA は OpenCL で記述されたソースコードを使用しているため，CUDA と OpenCL の 2 つの言語とコンパイル環境を統合する必要がある．

組み込み後のコンパイル・リンク時には以下のオプションを設定した [2]．このオプションを設定しないと OpenCL で記述されたソースコードをコンパイル・リンクすることができない．このときのコンパイルフローを図 8 に示す．

- -I/opt/share/intelFPGA_pro/17.1.2.304/hld/host

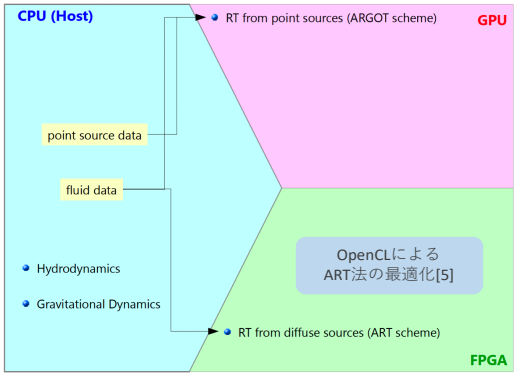


図 6　ソースコードの構成

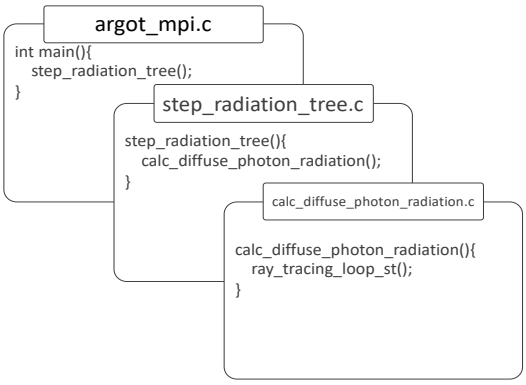


図 7　関数の依存関係

/include ( compile-config : OpenCL ホストコードのコンパイルに必要なヘッダーファイル群 )

- -L/opt/share/intelFPGA_pro/17.1.2.304/hld/board /bittware /a10pl4_4/linux64/lib -L/opt/share /intelFPGA_pro /17.1.2.304/hld/host/linux64/lib -Wl,–no-as-needed -lalteracl -lbittware_a10pl4_mmd -lelf ( link-config : OpenCL ホストコードのリンクに必要な実行ランタイムライブラリ群 )

また，先行研究より，次のことが実証されている．FPGA の性能向上に加えて，OpenCL ベースの FPGA 開発環境が FPGA ベンダーによって提供されているため，従来に比べてプログラミングコストが低くなっている．このような改善によって，低レイテンシのデータ移動を実行しながら，CPU / GPU の FPGA への性能が低いオンザフライのオフロード計算を可能にするという概念を実現した．この概念を実現するために OpenCL と Verilog HDL の混合プログラミングを用いた高性能の GPU-FPGA データ通信を提案し，両者を円滑に連携させた [8]．その結果，toy program を用いた実験より， 提案手法が GPU と FPGA 間で 0.6 [$\mu$ s] のレイテンシ，最大 6.9 [GB/s] を達成し，提案手法が高性能な GPU-FPGA 協調演算の実現に有効であることを示している．これを ARGOT 法と ART 法に適用する．

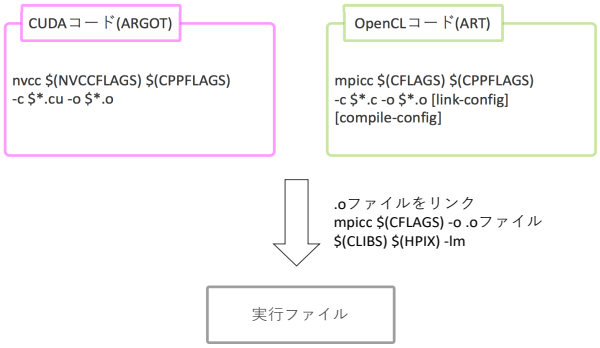


図 8　コンパイルフロー

表 1　本研究の実行環境

| | |
|---|---|
| CPU | Intel(R) Xeon(R) CPU E5-2660 v4 x2 |
| GPU | NVIDIA P100 x2 (PCIe Gen3 x16) |
| FPGA | BittWare A10PL4 (PCIe Gen3 x8) |
| OS | CentOS 7.3 |
| Host compiler | gcc 4.8.5 , openmpi 2.1.3 |
| GPU compiler | CUDA 9.2.148 |
| FPGA compiler | quartus 17.1.2.304 , aocl 17.1.2 |

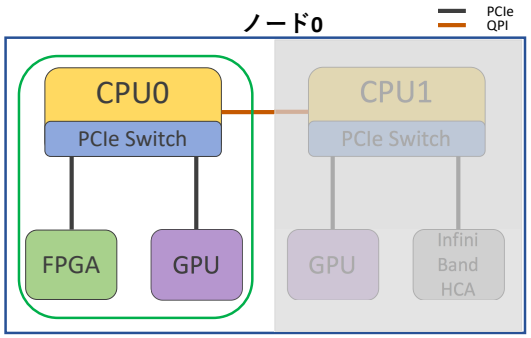


図 9　ノードの構成

## 5. 実装環境

今回，GPU･FPGA 複合演算加速による 輻射流体シミュレーションコード ARGOT の実装をし，実行するにあたって筑波大学 計算科学研究センター ( CCS ) が所有している PPX (Pre-PACS version X) を使用する．実装環境については表 1 に示す．ノード構成を図 9 に示す．本研究で用いたノード部分は緑の枠で囲まれている部分である．今回は実アプリケーション 輻射流体シミュレーションコード ARGOT への実装であり，1CPU･1GPU･1FPGA を対象としているためグレーの部分は使用しない．

## 6. 実装状況

ARGOT プログラムの ART 法を計算している GPU カーネルを OpenCL 版の FPGA コードに差し替える。具体的には、ホストプログラム以下のように書き換える。

- GPU カーネルの呼び出し -> FPGA カーネルの呼び出し

- CPU-GPU 間のデータの送受信 -> CPU-FPGA 間のデータの送受信
- OpenCL の初期化 ( OpenCL を使うときに新たに追加 )

しかし、現在、FPGA カーネルの呼び出し部分は実装中であり、実装後は第 5 章の環境で性能評価を行う。そして、最後には筑波大学 計算科学研究センター ( CCS ) で所有しているスーパーコンピュータ Cygnus に移植して最終評価を行う予定である。また、現在はホストプログラム上で GPU-FPGA 間のデータ交換を行っているが、最終評価では GPU-FPGA 間のデータ交換を DMA 転送 ( 第 4 章参照 ) に切り替えることで、性能の向上を図る。

## 7. まとめ

GPU-FPGA 実装はまだ実装途中である．そのため，実装できた場合の予測に対する考えを次に示す．

CPU 単体実行の場合と CPU・GPU 実行の場合の性能差の原因と考えられるのは，以下のとおりである．CPU 単体実行の場合よりも GPU を使用した場合の方が計算時間が大幅に短い．GPU を使用した場合，分解された計算ドメインの表面を通過する光線の数に比例する．また，$N_m$ が小さい場合は，全体の Wallclock time の多くの割合を通信オーバーヘッドが占めている． $N_m$ に関するスケーリングは，計算コストよりも $N_m$ への依存度が低い．そのため，CPU・GPU 間の通信オーバーヘッドは十分な数のメッシュグリッドに対して隠蔽することができ，$N_m$(メッシュグリッド) / $N_{node}$(使用する計算ノード数) が大きいほど並列効率が良くなるからである [9].

しかし，GPU は HPC では最も一般的に使用されるアクセラレータであるが，場合によっては性能のボトルネックとなることもある．その一方で，回路設計が柔軟な FPGA を利用する機会が増えてきている．しかし，アプリケーション開発者がアプリケーションやアルゴリズム用に FPGA の論理回路を実装することは容易ではない．そのため，近年の FPGA の開発環境の進歩により，OpenCL 言語を用いた高位合成 (HLS) 開発環境が普及してきている．そこで，OpenCL を用いて FPGA 上での Authentic Radiation Transfer (ART) 法の最適化を行った [4]．その結果，OpenMP を使用した CPU 実装と比較して 6.9 倍高速な性能を達成した．複数の FPGA を使用し並列化された実装は，GPU よりも高い性能を達成すると考えられる．このことより，GPU・FPGA の強調計算が実現すると，ART 法の計算時間は ARGOT の全体の計算時間の 90%以上を占めているため，CPU・GPU 実行よりも高速化できると予想される．

実装が完成して，GPU・FPGA 協調計算を実現できると，CPU・GPU 実行のときの性能に比べ，より高い性能を出すことができると考えている．

**謝辞** 本研究は，文部科学省「次世代領域研究開発」(高性能汎用計算機高度利用事業費補助金) 次世代演算通信融合型スーパーコンピュータの開発の一環として実施したものである．

### 参考文献

[1] June 2019 — TOP500 Supercomputer Sites
https://www.top500.org/lists/2019/06/
[2] 中道安祐未, 小林諒平, 藤田典久, 朴泰祐, ”GPU・FPGA 混載ノードにおけるヘテロ演算加速プログラム環境に関する研究”, 研究報告ハイパフォーマンスコンピューティング (HPC), 2019-HPC-168(10)
[3] Okamoto T.，Yoshikawa K.，Umemura M.，”ARGOT: Accelerated radiative transfer on grids using oct-tree”，Monthly Notices of the Royal Astronomical Society/419/p.2855-2866, 2012-01
[4] Norihisa Fujita, Ryohei Kobayashi, Taisuke Boku, Yuma Oobata, Yoshiki Yamaguchi, Kohji Yoshikawa, Makino Abe, Masayuki Umemura, ”Accelerating Space Radiative Transfer on FPGA using OpenCL”, Proc. of HEART2018 (Int. Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies), Toronto, Jun. 21st 2018.
[5] Y. kodama, T. Hanawa, T. Boku, M. Sato, ”PEACH2: FPGA based PCIe network device for Tightly Coupled Accelerators”, Proc. of HEART2014, Sendai, 2014. (receiving Best Paper Award of HEART2014)
[6] Kuen Hung Tsoi, Wayne Luk: Axel: a heterogeneous cluster with FPGAs and GPUs. FPGA 2010: 115-124
[7] Accelerating Space Radiate Transfer on FPGA using OpenCL Fujita, Norihisa;Kobayashi, Ryohei;Yamaguchi, Yoshiki;Oobata, Yuma;Boku, Taisuke;Abe, Makito;Yoshikawa, Kohji;Umemura, Masayuki HEART 2018 Proceedings of the 9th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies Article No. 6 6:1-6:7 Jun 2018
[8] Ryohei Kobayashi, Norihisa Fujita, Yoshiki Yamaguchi, Taisuke Boku, “OpenCL-enabled high performance direct memory access for GPU-FPGA cooperative computation”, Proc. of IXPUG Workshop Asia 2019 (in HPC Asia 2019), Guangzhou, Jan. 14th, 2019.
[9] Tanaka, S., Yoshikawa, K., Okamoto, T. and Hasegawa, K.: A new ray-tracing scheme for 3D diffuse radiation transfer on highly parallel architectures, Publications of the Astronomical Society of Japan, Vol. 67, No. 4, p. 62 (online), DOI: 10.1093/pasj/psv027 (2015).