

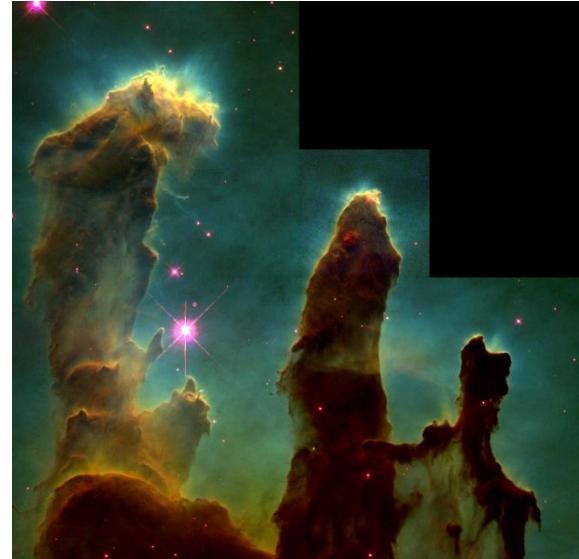
Diffuse Photonを考慮した 3次元輻射輸送計算 on GPUs

筑波大学大学院
数理物質科学研究所
博士後期課程3年
田中 賢

- Introduction
- Method
 - ART
 - ARGOT
- Result
- Implementation to the GPUs
- Summary

Introduction

- HI領域とHII領域には多くの不規則な構造が見られる。
- 電離波面(IF)の不安定性の研究はVandervoort(1962)を始め古くから解析的に行われているが、3次元かつ電離非平衡での計算はあまり行われていない。->Whalen(2008)
- これまでに開発した3次元輻射輸送計算コードを用いて銀河内に存在する星間ガス雲内に星が出来た場合に波面がどのように振る舞うかを調べる。
- GPUを用いて計算を加速することにより現実的な時間で再結合光子の輸送も正確に解くことができるのでその影響も取り入れる。



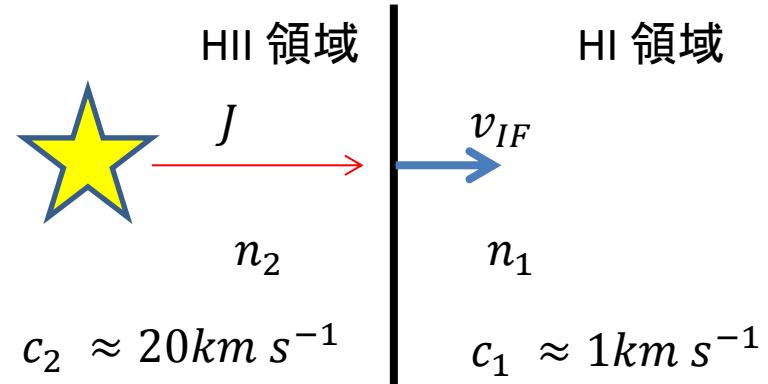
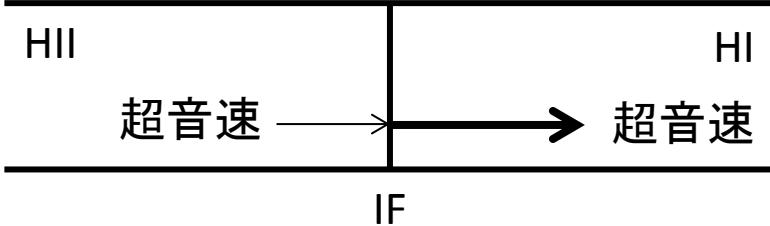
Eagle Nebula



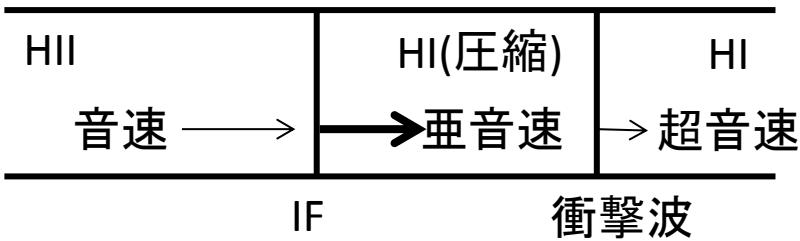
Elephant's trunk

電離波面のタイプ

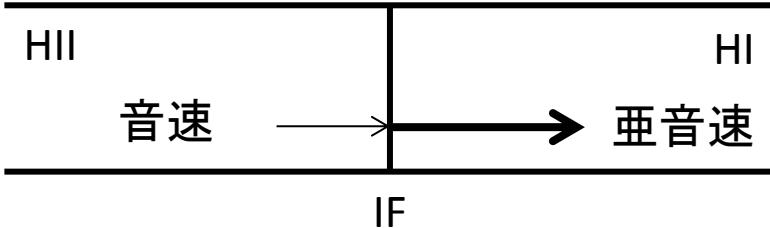
(i) R-type ($v_{IF} \geq v_R$)



(ii) M-type ($v_R \geq v_{IF} \geq v_D$)



(iii) D-type ($v_{IF} \leq v_D$)



$J = n_1 v_{IF}$ の保存則より

$v_{IF} \geq v_R$ or $v_{IF} \leq v_D$ で実数解を持つ。

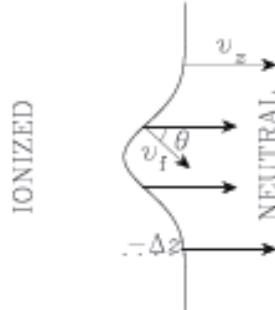
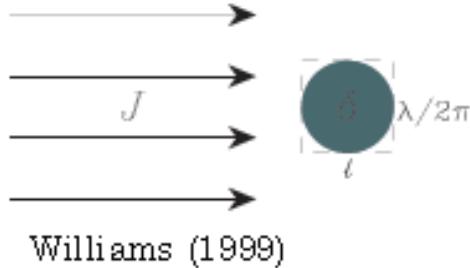
典型的な星間雲では

$$v_R \approx 2c_2 \approx 20 \text{ km s}^{-1}$$

$$v_D \approx \frac{c_1^2}{2c_2} \approx 0.05 \text{ km s}^{-1}$$

IF Instability

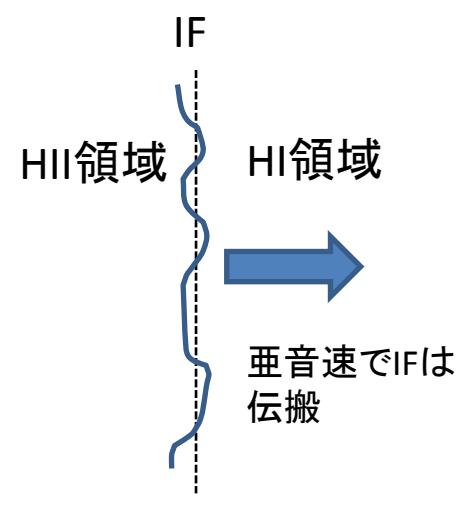
- R-type shadowing instability



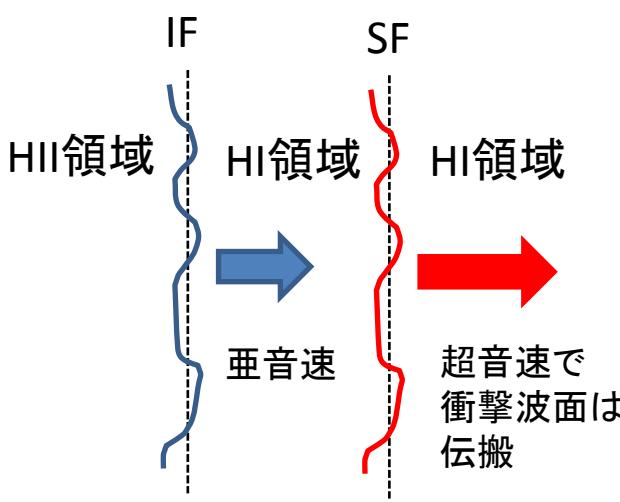
電離波面がR-typeで進行する場合
HI領域の密度分布の非一様性により、波面に不安定性が生じる。

- D-type instability

IF不安定

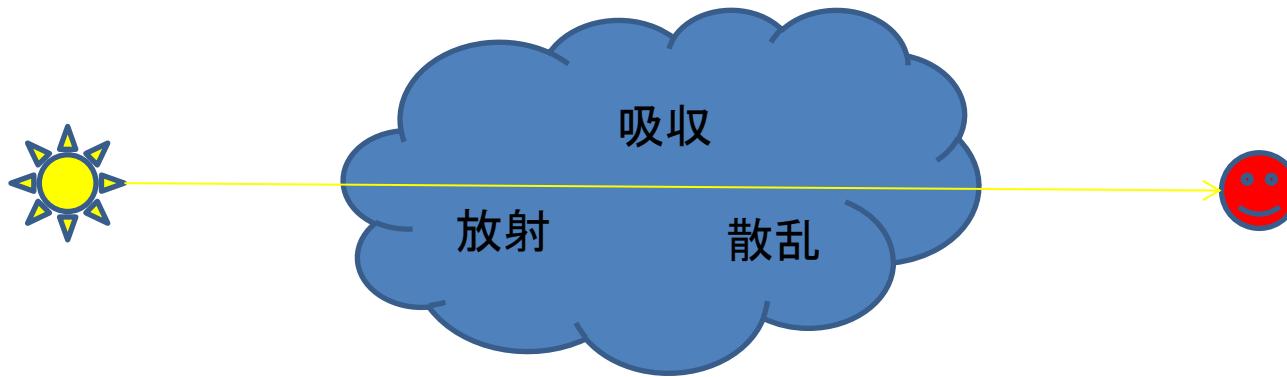


IS不安定



電離波面がD-typeまたはM-typeで
伝搬するときにIFの微小な形状が
流体不安定性によって増大する。
IFのみの場合と
IS(電離波面-衝撃波面)
不安定性を考える場合もある。

輻射輸送方程式



ガスなどの星間媒質中を光が進むときの増減を記述する式

$$\frac{dI_\nu}{ds} = -\kappa_\nu I_\nu + \varepsilon_\nu$$

I [erg/s/cm²/Hz/str] : 輻射強度

κ [cm⁻¹] : 吸收係数

ε [erg/s/cm³/Hz/str] : 放射係数

輻射輸送方程式

- $\frac{dI_\nu}{ds} = -\kappa_\nu I_\nu + \varepsilon_\nu$
- $\frac{1}{\kappa_\nu} \frac{dI_\nu}{ds} = -I_\nu + \frac{\varepsilon_\nu}{\kappa_\nu}$

光学的厚み

$$\tau_\nu = \int_0^r \kappa_\nu dr'$$

τ が大きければ光は通りにくく
小さければ通りやすい

$$S = \frac{\varepsilon}{\kappa}: \text{源泉関数}$$

$$\frac{dI_\nu}{d\tau_\nu} = -I_\nu + S_\nu$$

輻射強度

$$I_\nu(\tau) = I_\nu(0)e^{-\tau} + \int_0^\tau e^{-(\tau-\tau')} S_\nu(\tau') d\tau'$$

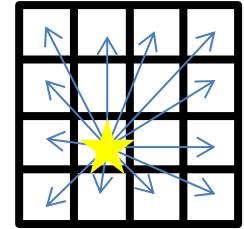
源泉関数が場所によらず一定なら

$$I_\nu(\tau) = I_\nu(0)e^{-\tau} + S_\nu(1 - e^{-\tau_\nu})$$

輻射源となる光源のタイプ[°]

➤ 点光源からの輻射

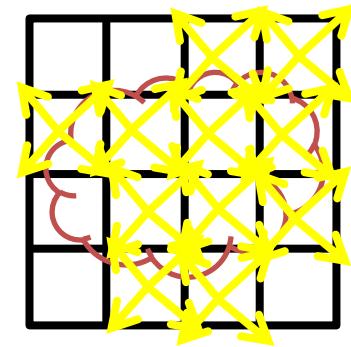
- ✓ 各点光源から光線を飛ばすことで計算可能
- ✓ 再結合光子の輸送を考慮しない**on-the-spot近似**を使うと計算量は一点光源当たり $O(N^{4/3})$ (N : 3次元メッシュ数)
- ✓ **ARGOT法**(Okamoto et al. 2012) を使うことにより点光源数が増えた場合も高速に計算可能



➤ ガス雲などの広がった天体からの輻射 (diffuse radiation)

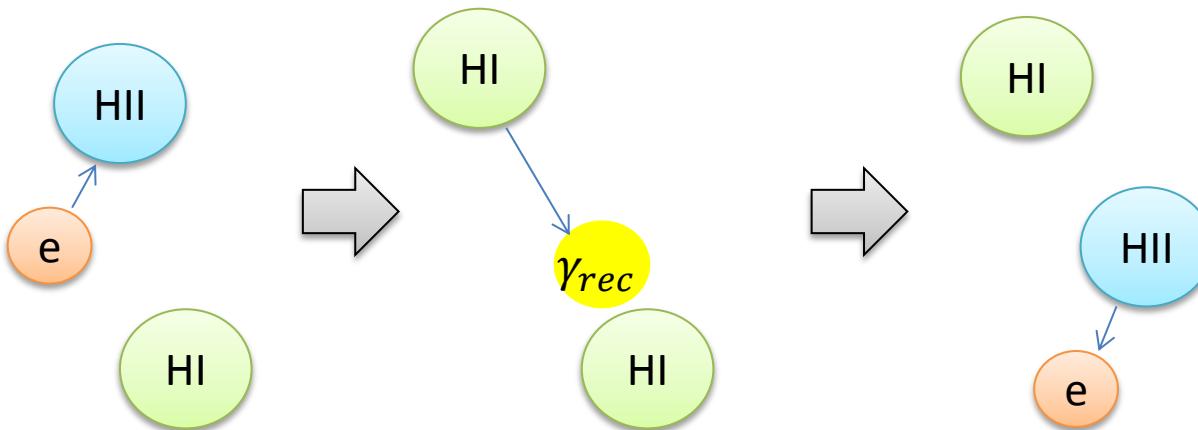
- ✓ 広い領域からの輻射輸送を計算をする必要がある
- ✓ 計算量が膨大である
- ✓ 再結合光子やdustの輻射輸送はこちらのタイプ
- ✓ long characteristics法で解くと $O(N^{6/3})$ 。 short characteristics法では $O(N^{5/3})$ 。
- ✓ 本研究では**ART法(Accelerated Ray Tracing)**を使う

➤ 点光源からの輻射輸送はARGOT法で、再結合光子はART法を用いて実装。

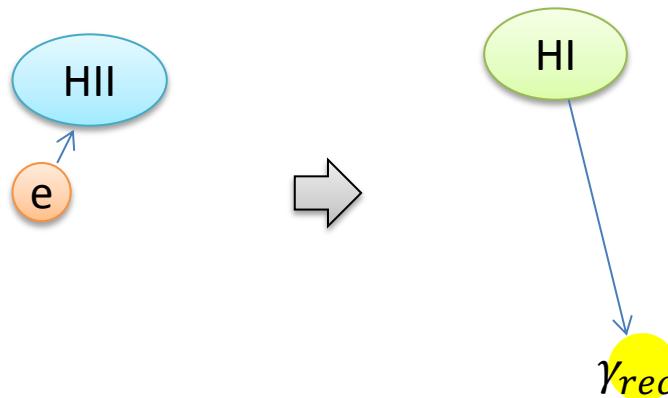


on-the-spot近似

- 多くの輻射輸送数値計算法では計算量を削減するためon-the-spot近似を用いて再結合光子が生成されてもすぐに周りのHIに吸収され、電離に使われるためそこでは何も生成されなかつたと仮定できる。光学的に厚い場合は非常に有効。



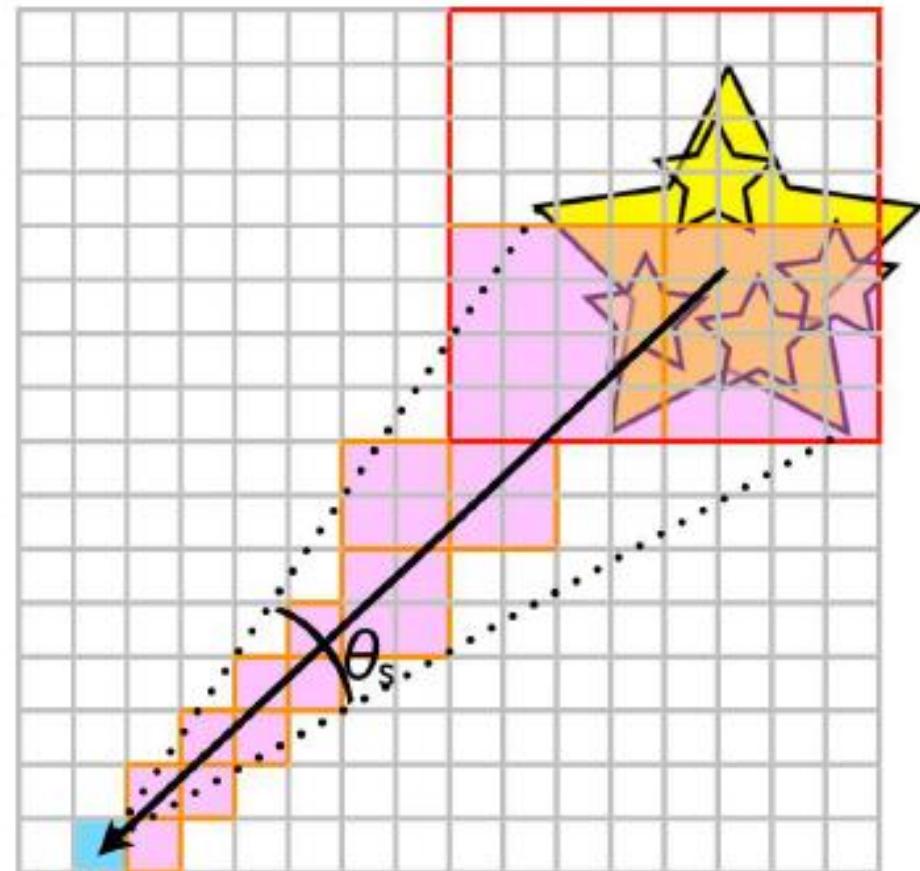
- 光学的に薄い場合はon-the-spot近似を使うのは不適切。
- 再結合光子は遠くまで飛んでいけるため、光源として振る舞う。



ARGOT

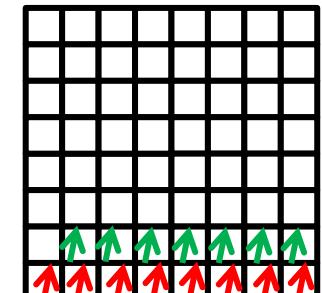
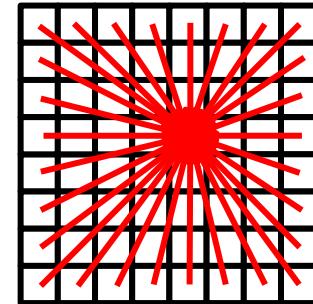
Accelerated radiative transfer on grids using oct-tree

- 遠くにある複数の点光源を1つの点光源のように取り扱い高速化を行う方法(Okamoto et al. 2012)
- 光源が N_s 個あるときの計算量は、振動数当たり $N_m^{4/3} \log(N_s)$
(普通の点光源long法では振動数当たり $N_m^{4/3} N_s$)
- $\theta_s = 0$ でpoint source long法と同等



Diffuse Sourceの数値計算法

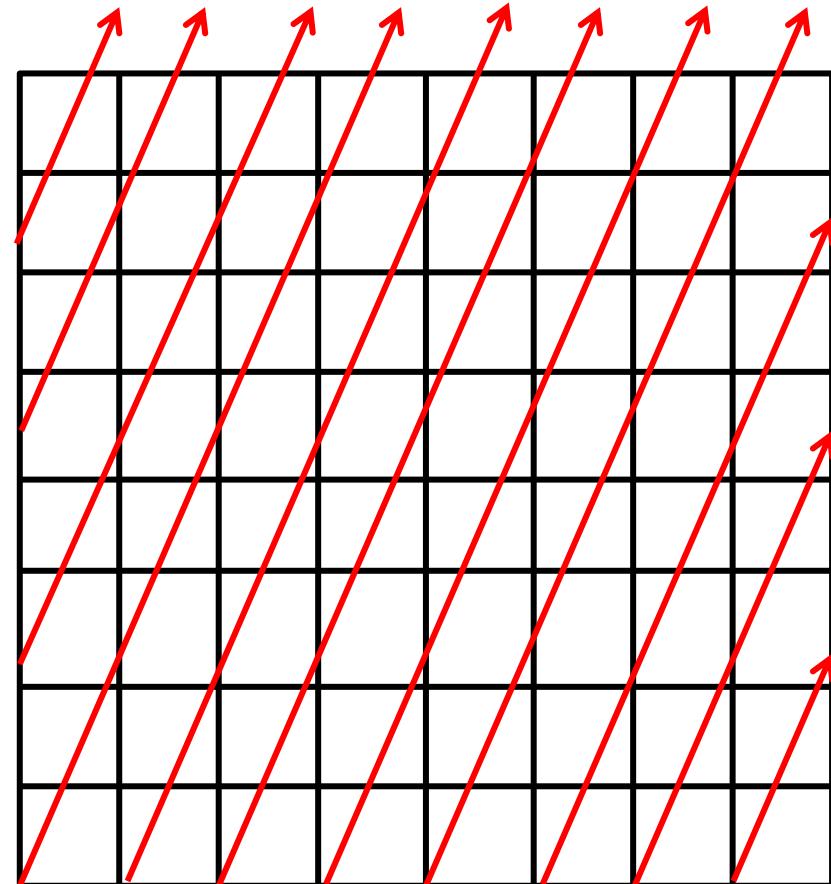
- 再結合光子の輸送も解ける代表的なものにlong characteristics(long法)、short characteristics (short 法) がある。
- long法
 - メリット 他の方法より正確に解ける。実装も難しくない。
 - デメリット 全メッシュから光線を出すため計算量が膨大。
計算領域が N メッシュであるとすれば計算量が
$$N \times N_{path} \times N_\theta N_\phi \times N_\nu \times N_{itr} \sim O(N^2)$$
 (N_ν, N_{itr} が N に比べて小さい場合)
$$O(N^{1/3}) \quad O(N^{2/3})$$
- short法
 - メリット 平行光線を飛ばしlong法より計算量を削減。
$$N^{2/3} \times N_{path} \times N_\theta N_\phi \times N_\nu \times N_{itr} \sim O(N^{5/3})$$
 (N_ν, N_{itr} が N に比べて小さい場合)
 - デメリット 補間を繰り返すため光線を一本飛ばしても拡散する。



ART

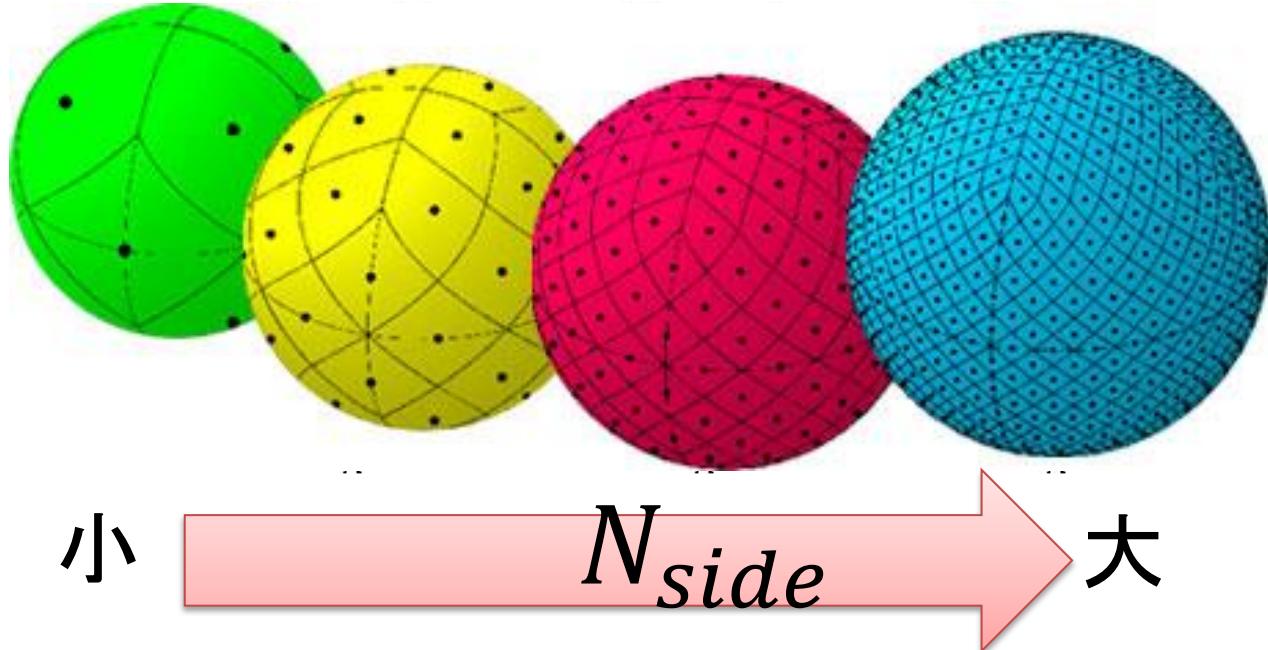
authentic RT

- **ART法**とは計算量はshort法、計算精度はlong法と同等であるそれぞれのメリットを合わせたような方法である。
- short法のような平行光線を飛ばすのだが1メッシュ進むごとに補間をせずにlong法のように境界まで光線を飛ばす。
- 各光線は独立に計算できるため並列計算が可能。基本的に平行光線は同じ演算量なのでSIMD化が容易。
- 計算量は
$$N^{2/3} \times N_{path} \times N_\theta N_\phi \times N_\nu \times N_{itr} \sim O(N^{5/3})$$



輻射輸送の方向解像度

方向数: $N_\theta N_\phi = 12 \times N_{side}^2$



- 光線の方向はHEALPixという手法を用いて決める。
- N_{side} : 方向の解像度を決めるパラメータ
- 方向解像度が高ければ、より遠くのメッシュを光線が一様に通過することができる
→ $R(l)$
- $N_{side} = 4 : R(l) = \lambda_{\text{mfp}}$
 $N_{side} = 1 : R(l) = 0.25\lambda_{\text{mfp}}$
- 光線の届く範囲が“mean free path”より極端に小さくなければ良い

流体計算

- ASUM+ (Advection Upstream Splitting Method) (Liou 1996)で実装
- 空間二次精度
- Heating と Cooling の項を考慮する

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \quad (\text{連続の式})$$

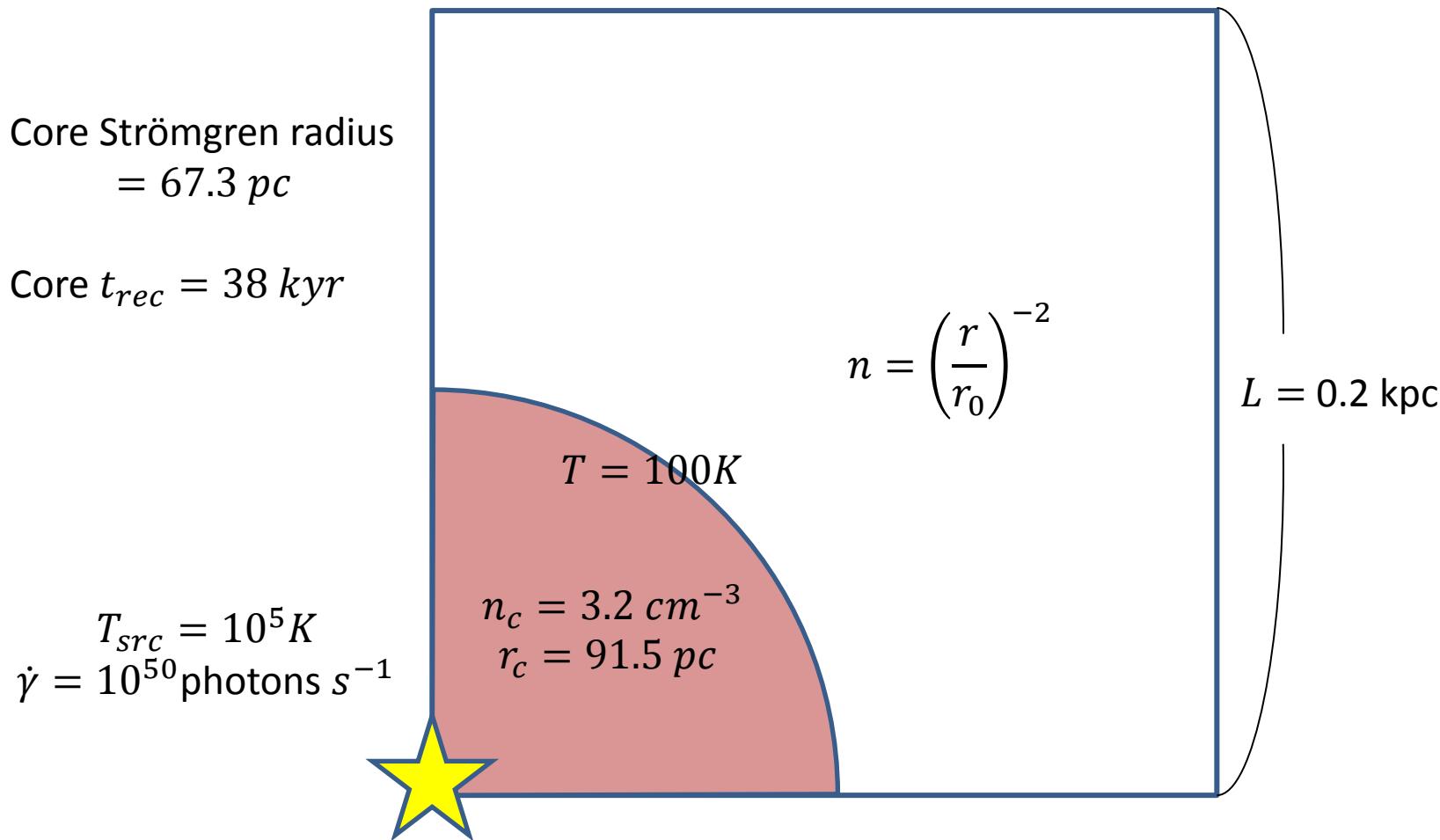
$$\rho \left\{ \frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla) \mathbf{v} \right\} = -\nabla p + \mathbf{F} \quad (\text{運動方程式})$$

$$\frac{\partial e}{\partial t} + \nabla \cdot \{ \mathbf{v}(e + p) \} = \mathcal{H} - \Lambda + \mathbf{F} \cdot \mathbf{v} \quad (\text{エネルギー式})$$

(\mathbf{F} : 外力, \mathcal{H} : エネルギーの増加率 (加熱), Λ : エネルギーの損失率 (冷却))

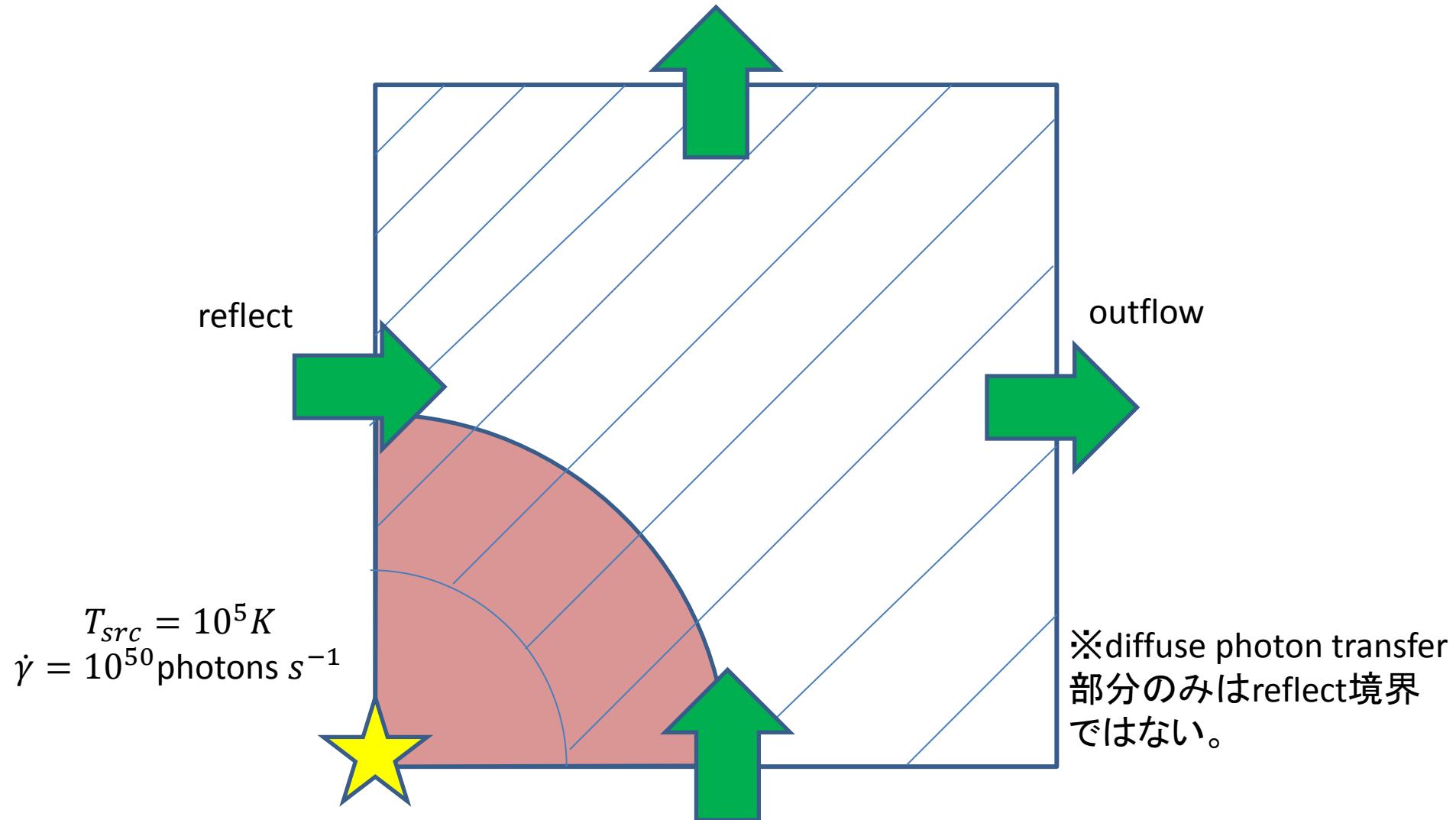
IF Instability Test

- Iliev(2009) のTest 6 をベースに設定
- 中心に星があるnHの数密度 $\sim 1.0 \text{ cm}^{-3}$ 程度のコアを持つISMを想定



IF Instability Test

- 光源から $r > 0.22 L$ の各meshの密度に **±50%の擾動**を与える
- R-type不安定性がでないように光源近くには擾動を与えない



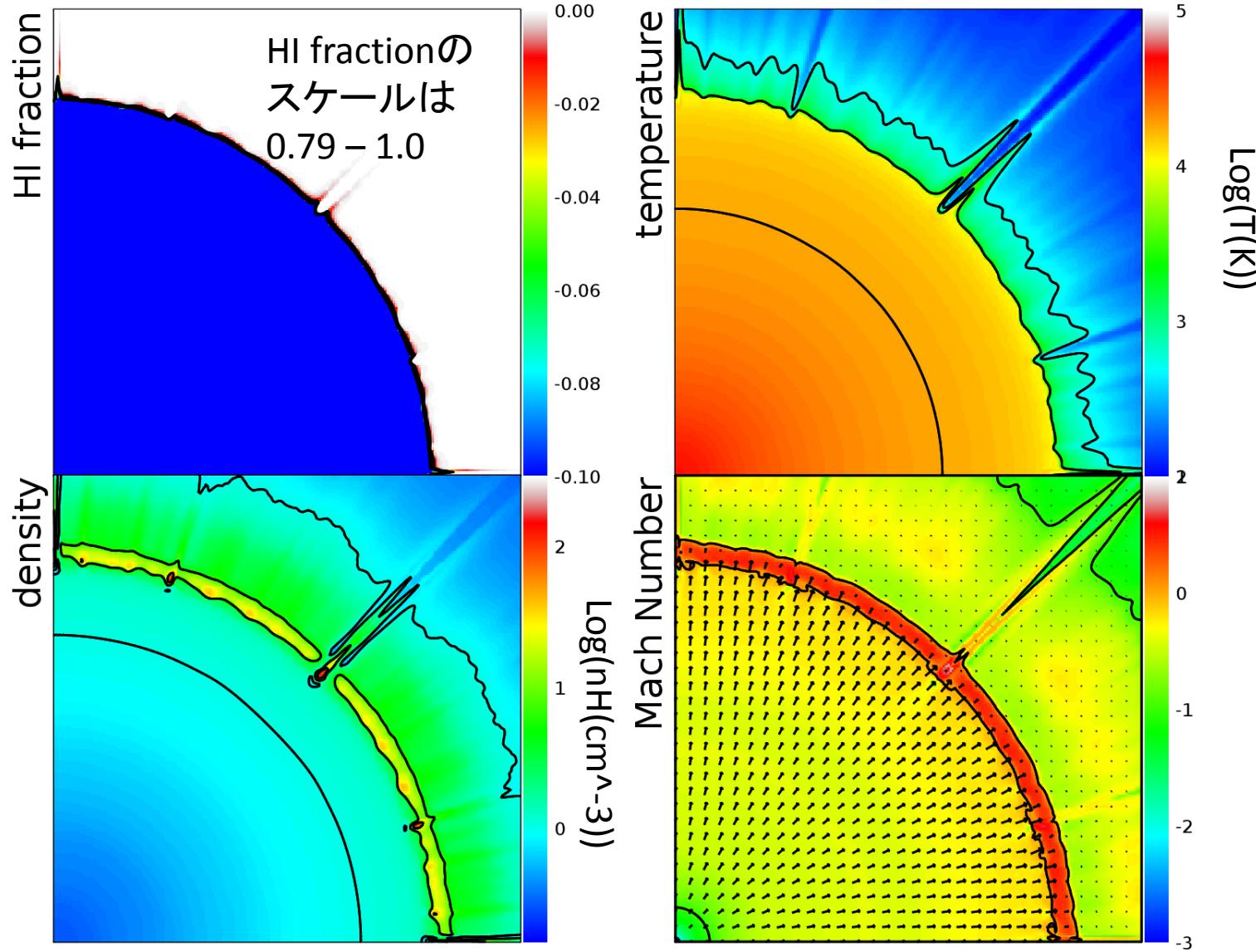
Simulation setting

- ✓ $N_{mesh} = 256^3$
- ✓ $N_{side} = 4$ ($\lambda_{mfp} = 0.016 pc$, mesh size = $0.78 pc$)
- ✓ Time step $t_{step} = \min(t_{chem}, t_{hydro})$
 $t_{hc} = 0.1 \times t_{chem}$ のsub-cycle
- ✓ Run time $t = 5 Myr$ ($\sim 130 t_{rec}$)
- ✓ Iteration の収束条件は Ionization rate および 内部エネルギーの前のステップとの相対差が1%以下

- ✓ 摂動の有無、再結合光子の有無の4パターンで比較
- ✓ 摂動有りの場合のRandom seed は全て同じ

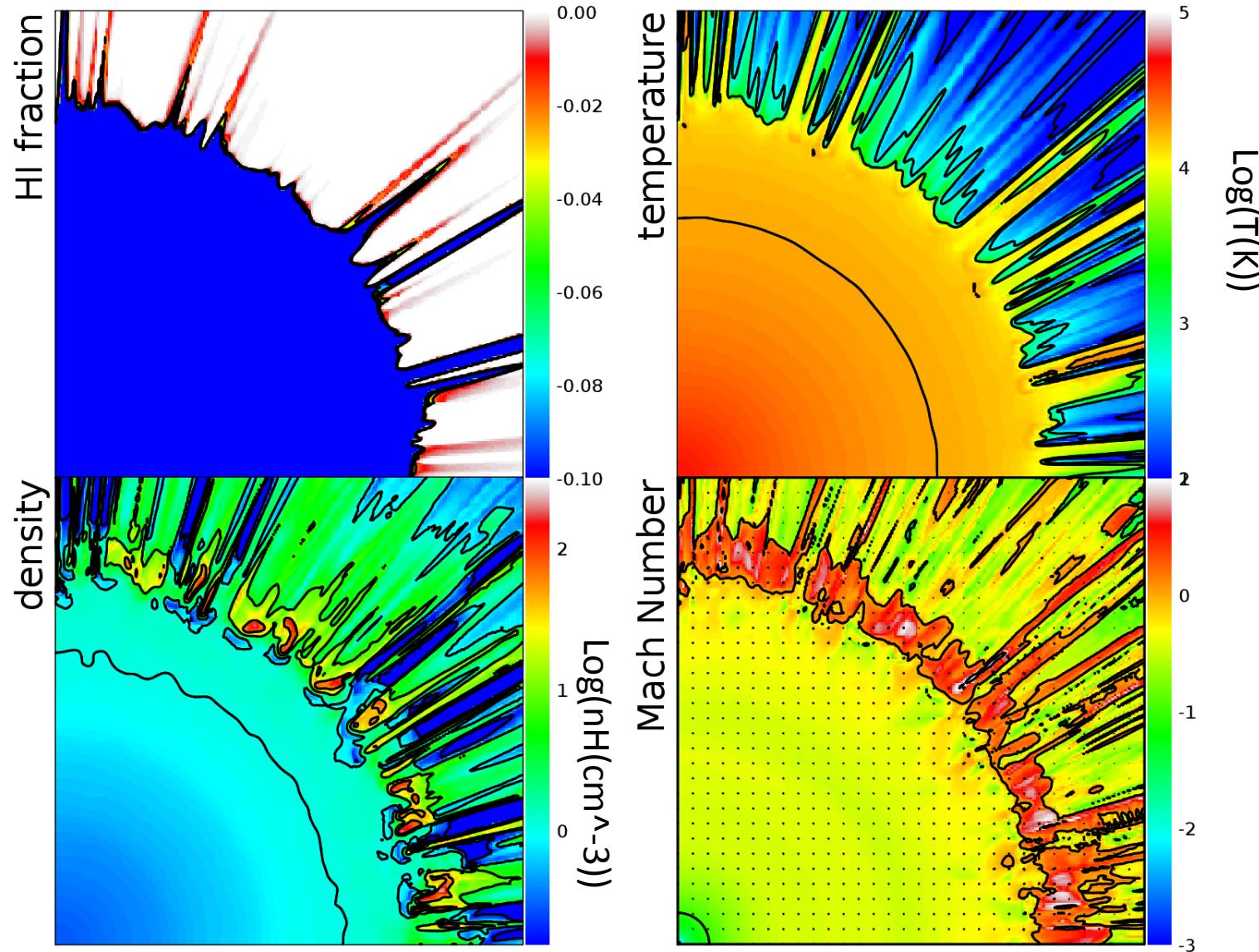
摂動なし 再結合光子なし

$t = 5\text{Myr}$



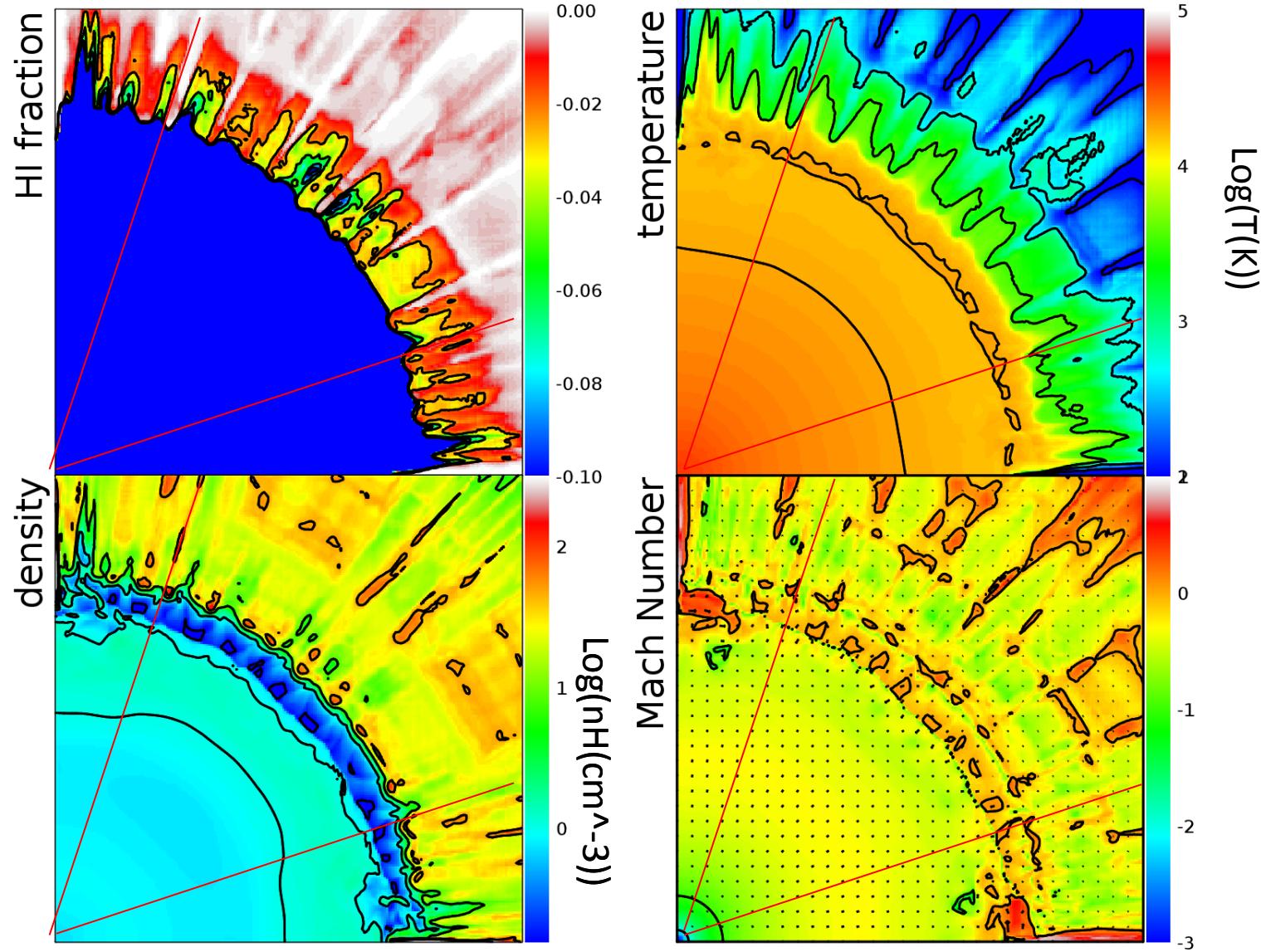
摂動あり 再結合光子なし

$t = 5\text{Myr}$



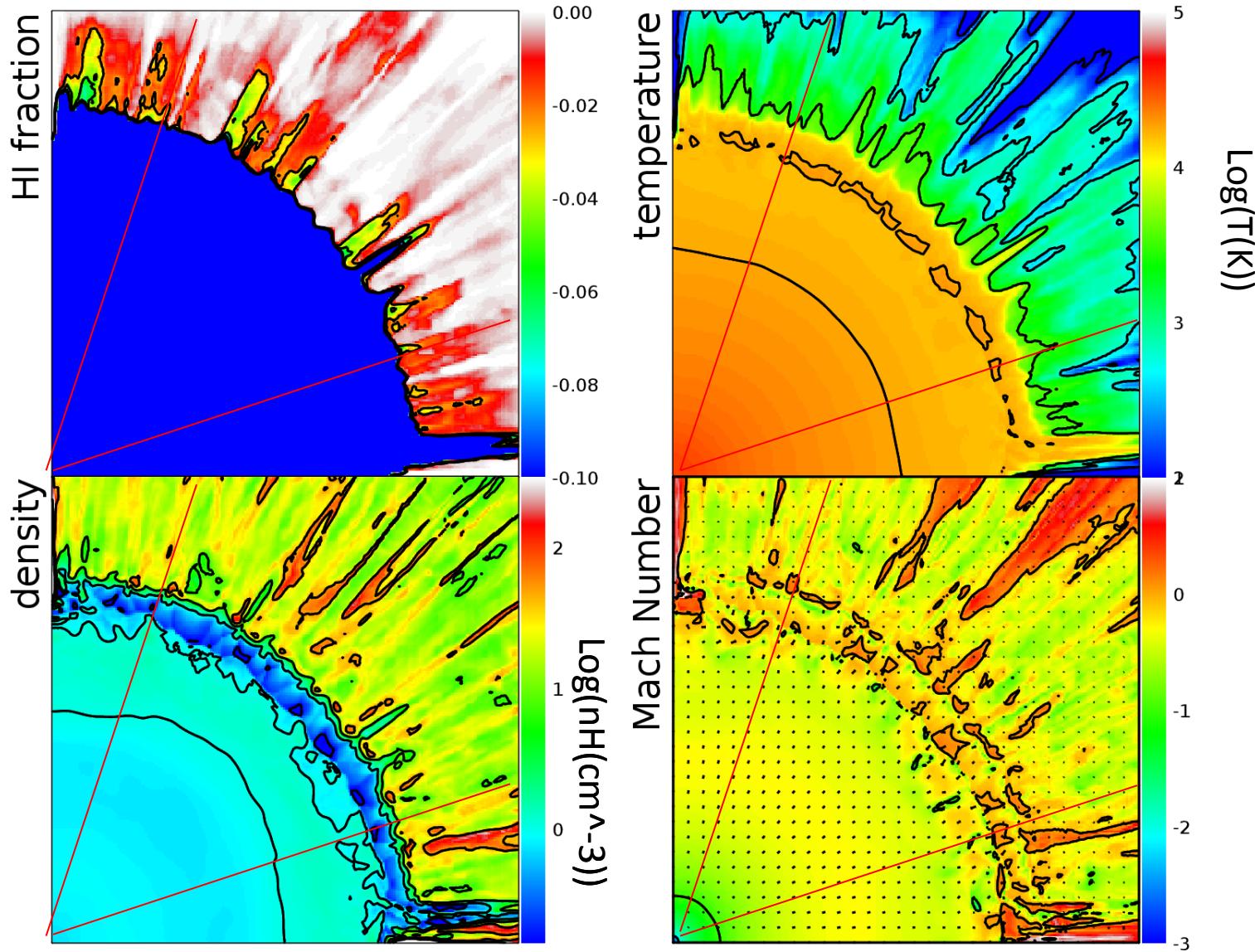
摂動なし 再結合光子あり

$t = 5\text{Myr}$

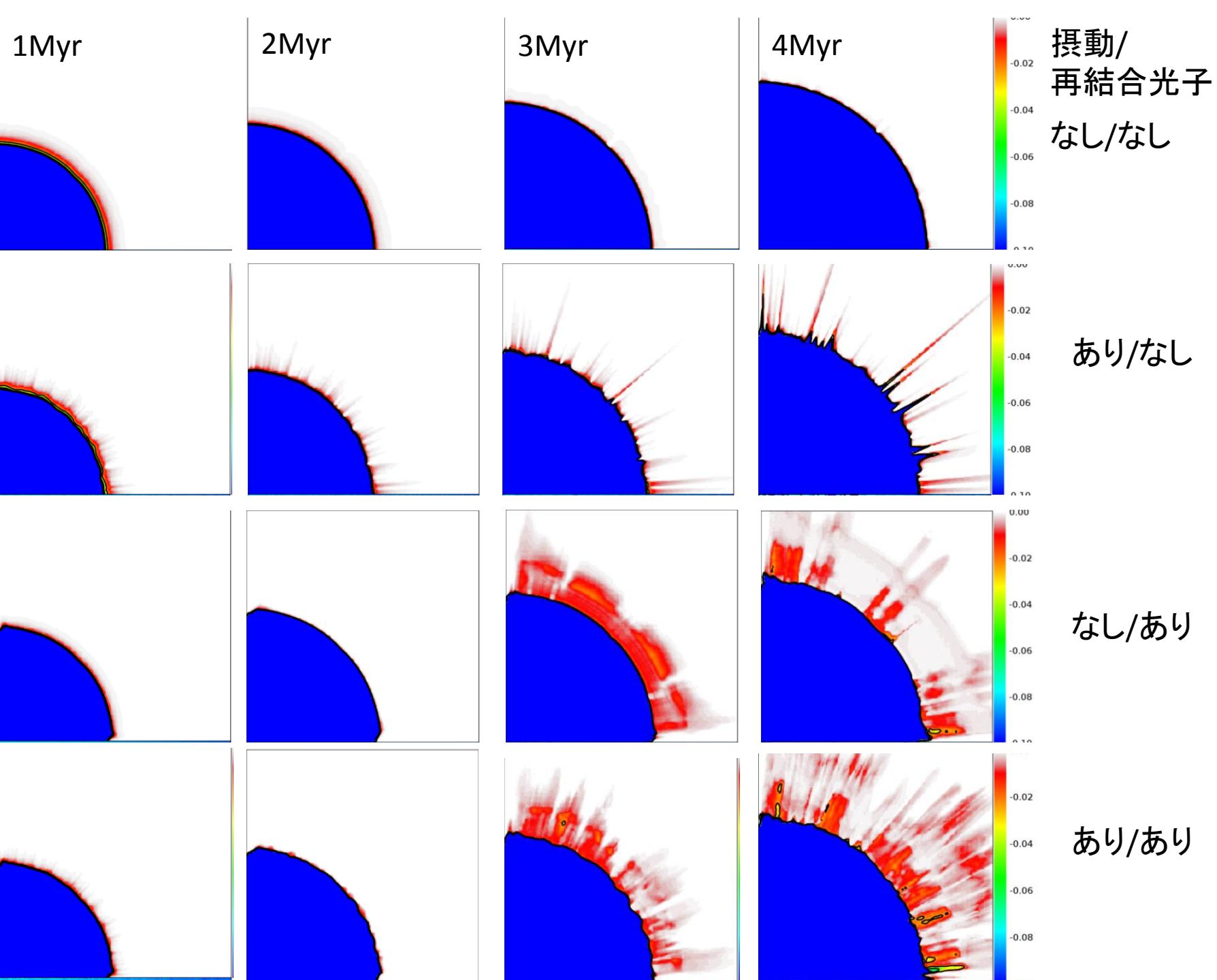


摂動あり 再結合光子あり

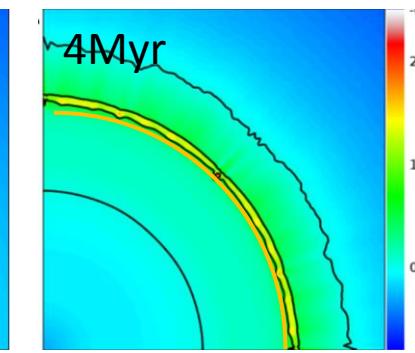
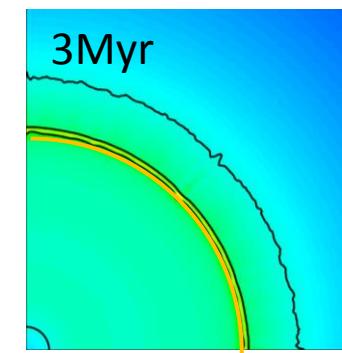
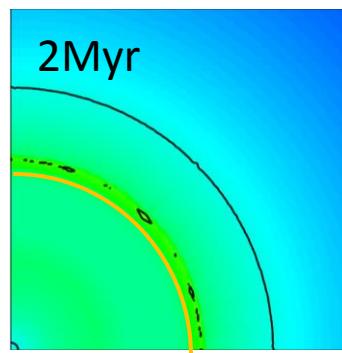
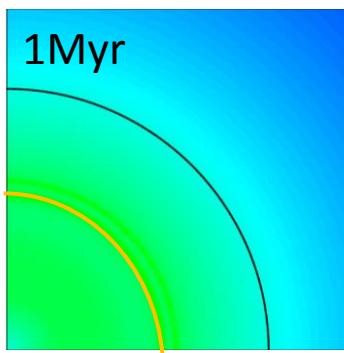
$t = 5\text{Myr}$



HI fraction Map

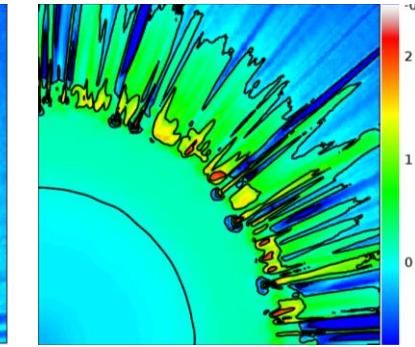
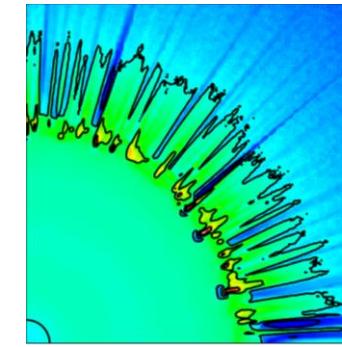
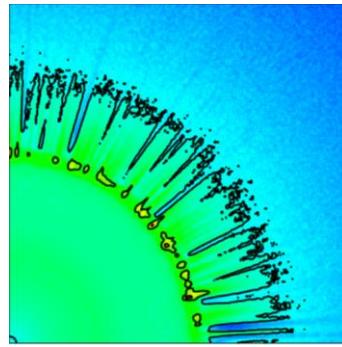
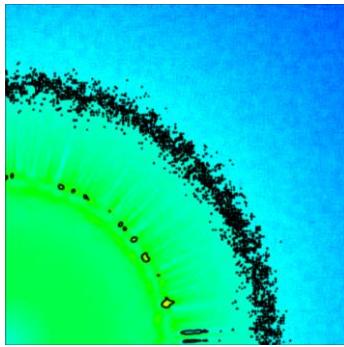


Density Map

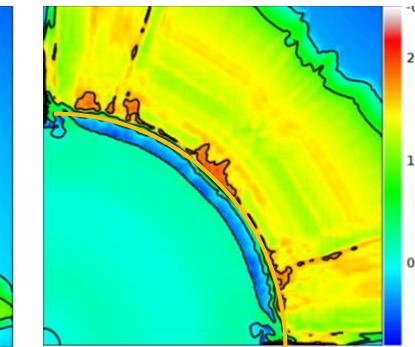
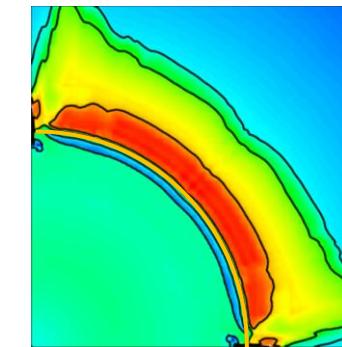
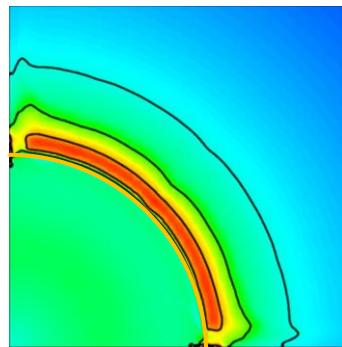
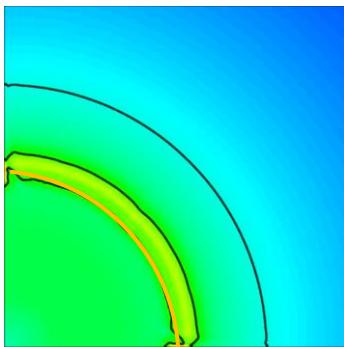


摂動/
再結合光子
なし/なし

黄色の実線:
IF front

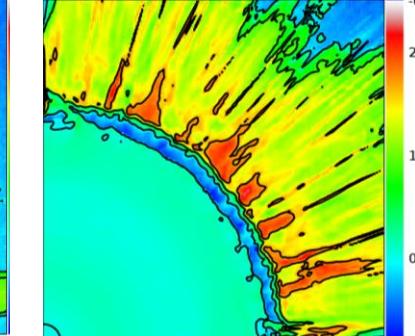
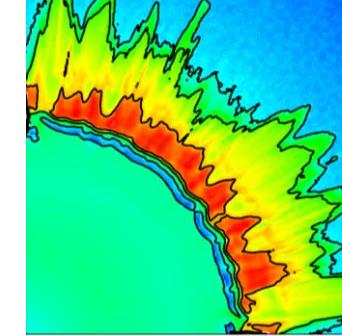
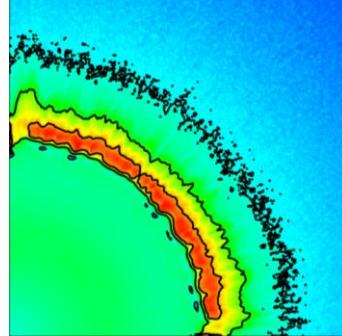
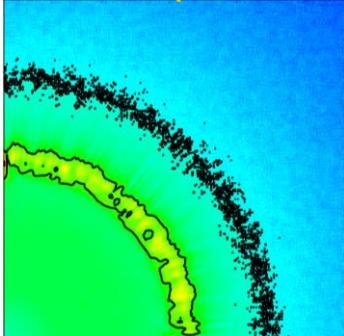


あり/なし



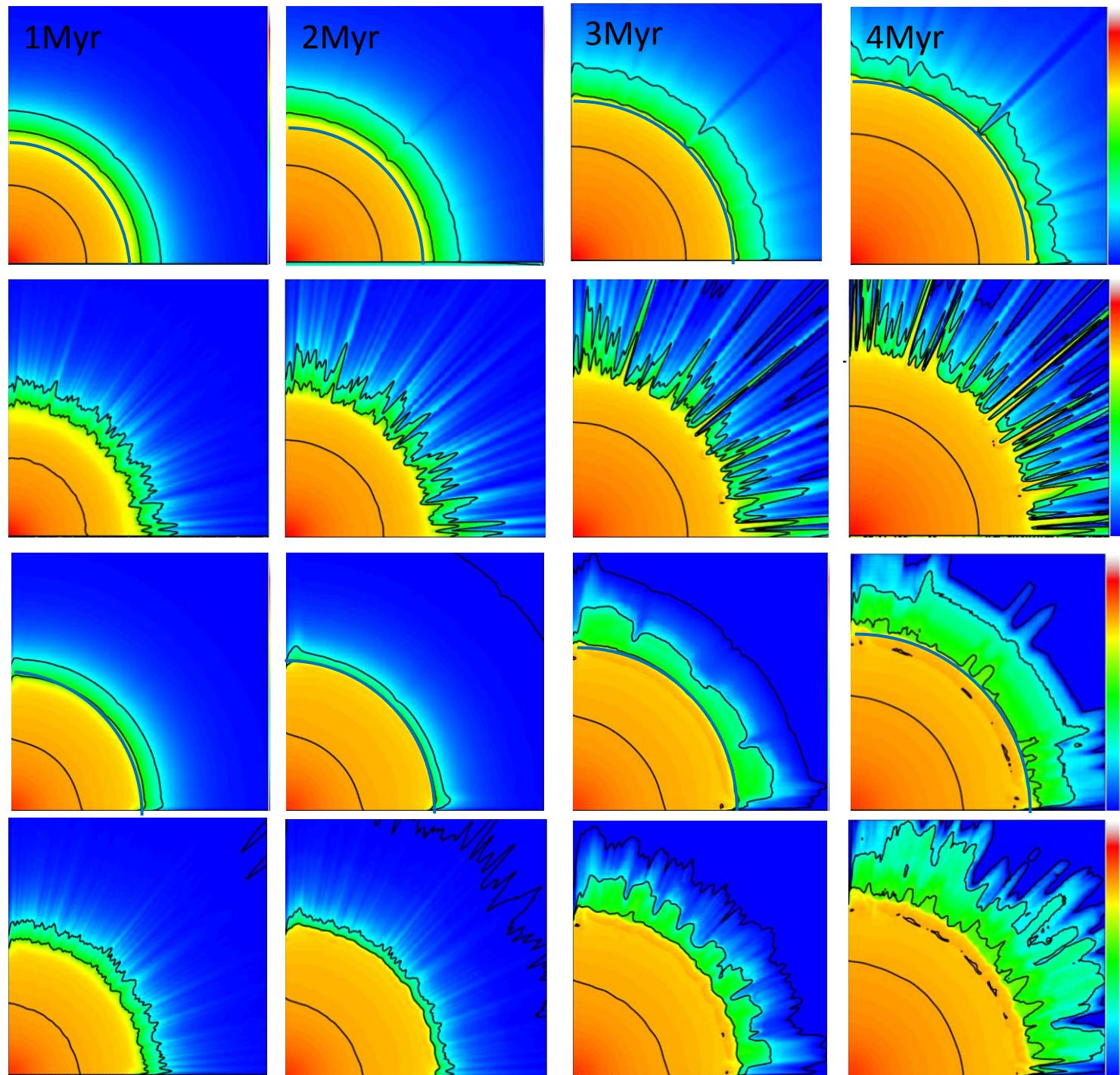
なし/あり

黄色の実線:
IF front



あり/あり

Temperature Map



摂動/
再結合光子
なし/なし

青色の実線:
IF front

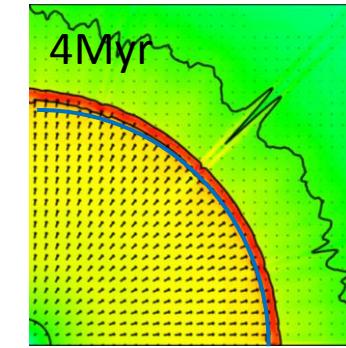
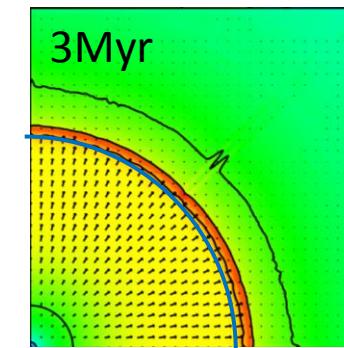
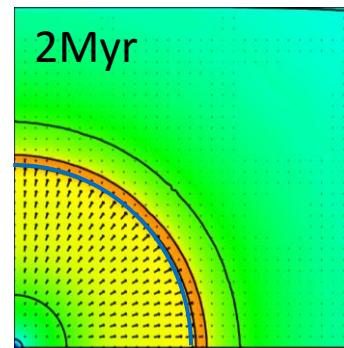
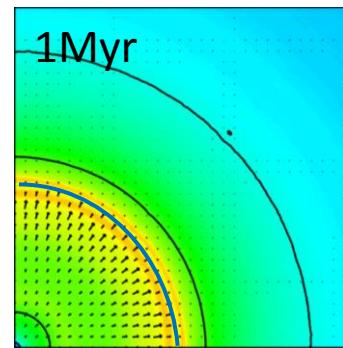
あり/なし

なし/あり

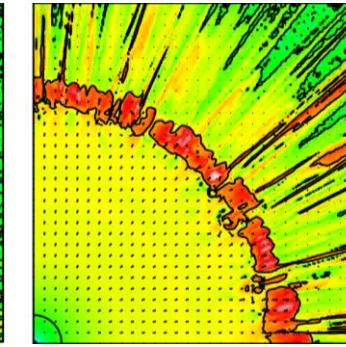
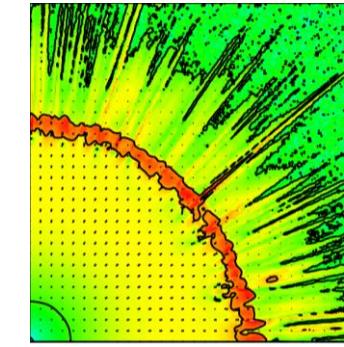
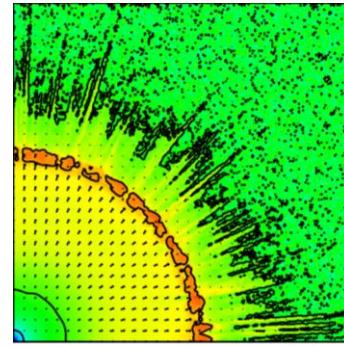
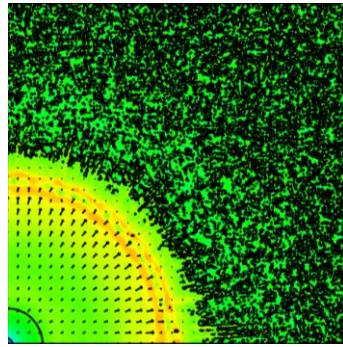
青色の実線:
IF front

あり/あり

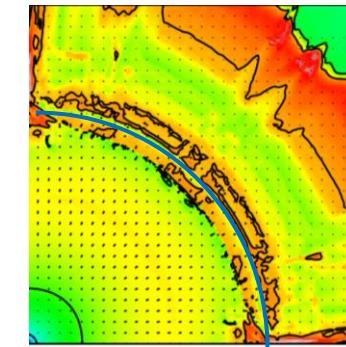
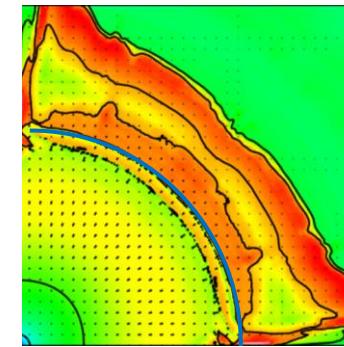
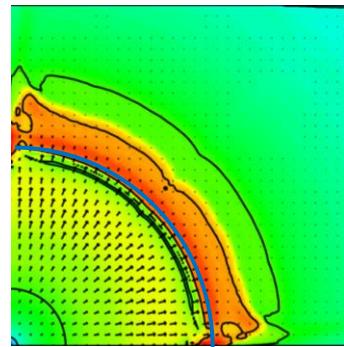
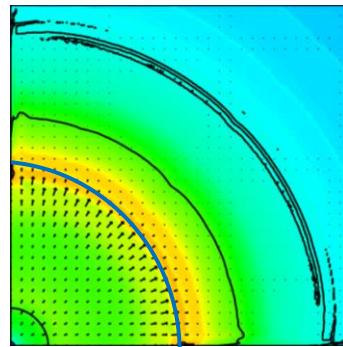
Mach Number & velocity vector



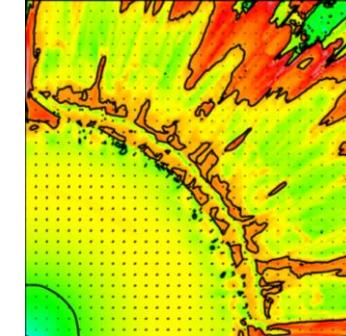
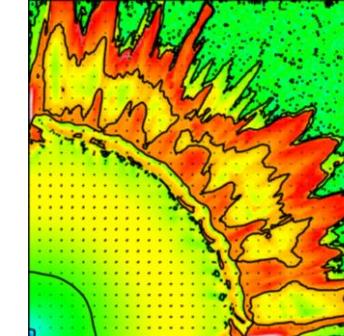
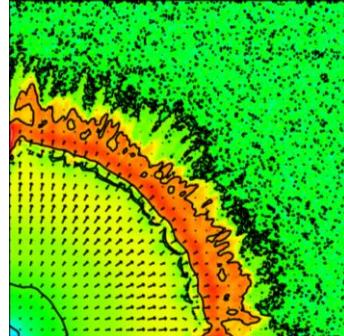
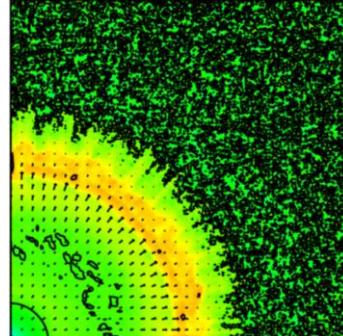
擾動/
再結合光子
なし/なし



あり/なし



なし/あり



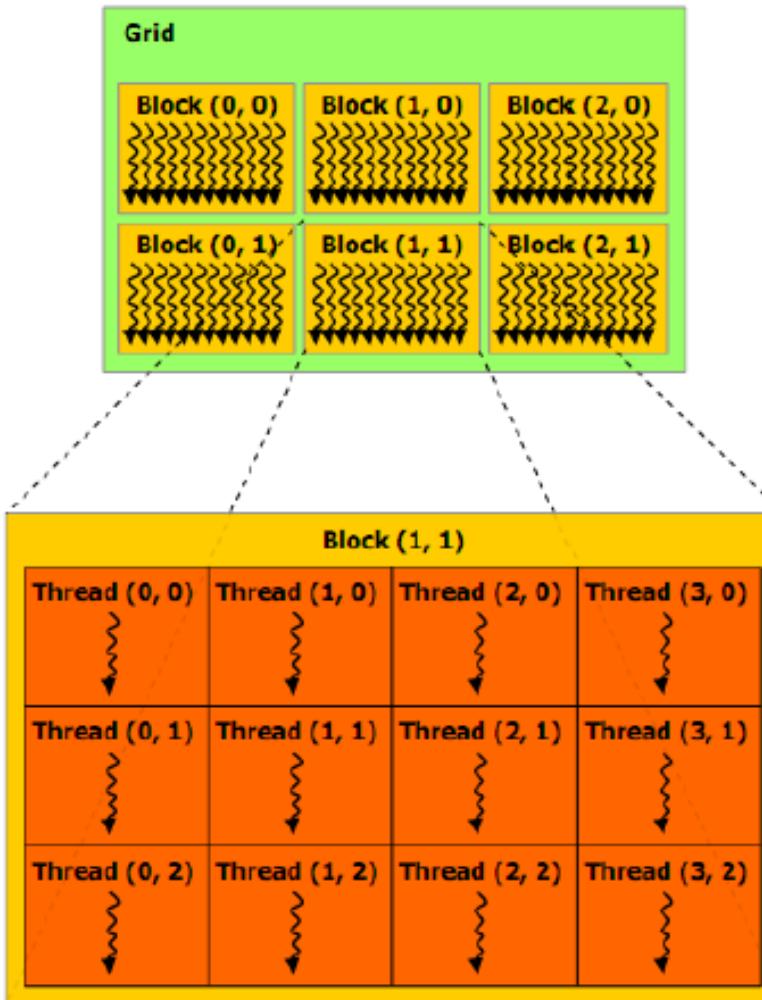
あり/あり

ここまでまとめ

- 3次元輻射流体を行うことにより銀河間の星間ガス(渦状腕HI領域を過程)を対象とした電離-衝撃波不安定性の計算を行った。
- 摂動を入れた場合、波面付近の濃い領域と薄い領域の密度分布が生まれ、特に薄い部分では光学的厚みが小さくなり、中心星からの輻射の影響がより遠くまで到達するため、特徴的な細いスパイク構造が多数見られた。
- 再結合光子を入れた場合は同様に密度分布からなる不安定性が見られたが、再結合光子の影響により全体的にスパイクがなまされる構造となった。また、密度の薄い領域だけでなく密度のある領域も電離されるため中心星の輻射も再結合光子がない場合に比べて遠くまで届きやすくなっている。
- 今後は再結合光子を入れた場合、波面前方に高密度領域が形成される理由がいまいち理解できていないため詳しく調べる。
- 今回の計算ではまだ不安定性ができる初期の段階までしかみていないため、計算領域を広げ、もっと時間が経った状態の不安定性がどうなるかを確認していきたい。

Compute Unified Device Architecture

- NVIDIA社のGPUを対象としたプログラミング環境
 - C / C++ の拡張
- 多数のコアを有効に使うために多くのスレッドを生成
- ブロック：スレッドの集合
 - 典型的には128-256スレッド
 - 同期はブロック単位で実行
 - 最も遅いスレッドが全体を律速



| CUDA C Programming guide 4.0 より

ARGOT並列化

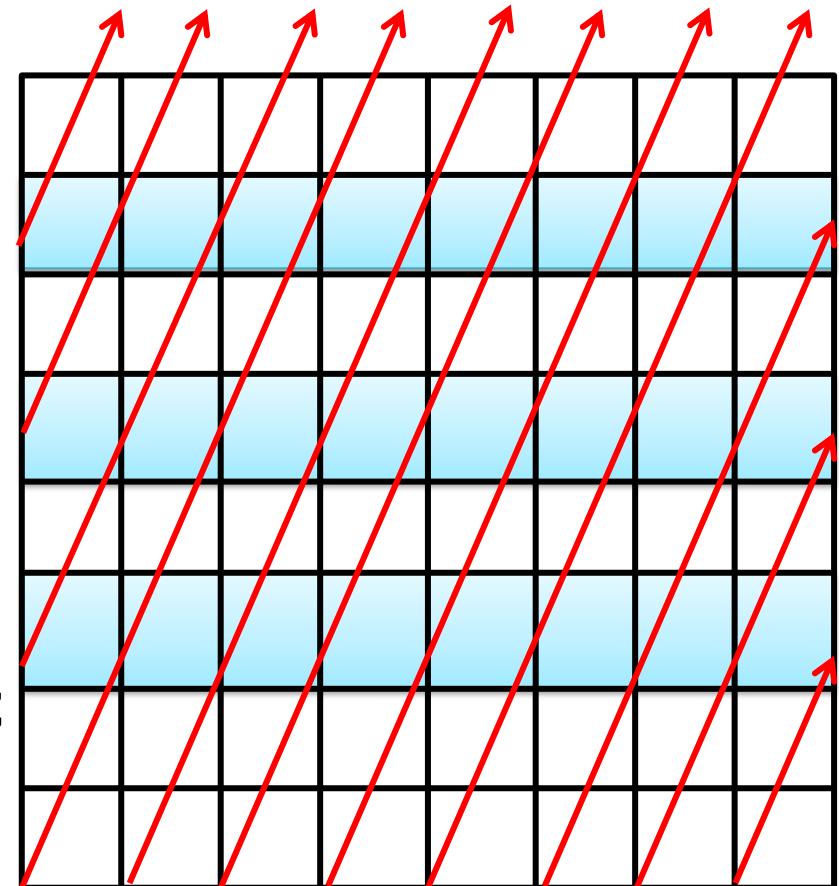
- GPU並列化
 - 光線1本1本の計算をGPUの多数あるスレッドに割り当てる。
- Multi GPU並列化
 - 光線index分割をし、複数のGPUに割り当てる。
- Multi Node並列化
 - 領域分割をし、複数ノードに割り当てる。

ART並列化

- GPU並列化
 - 光線1本1本の計算をGPUの多数あるスレッドに割り当て。
 - atomic演算の回避
- Multi GPU並列化
 - HEALPixで作った方向分割をし、複数のGPUに割り当て。
- Multi Node並列化
 - 領域分割をし、複数ノードに割り当て。
 - MWF(Multiple Wave Front)法で実装。

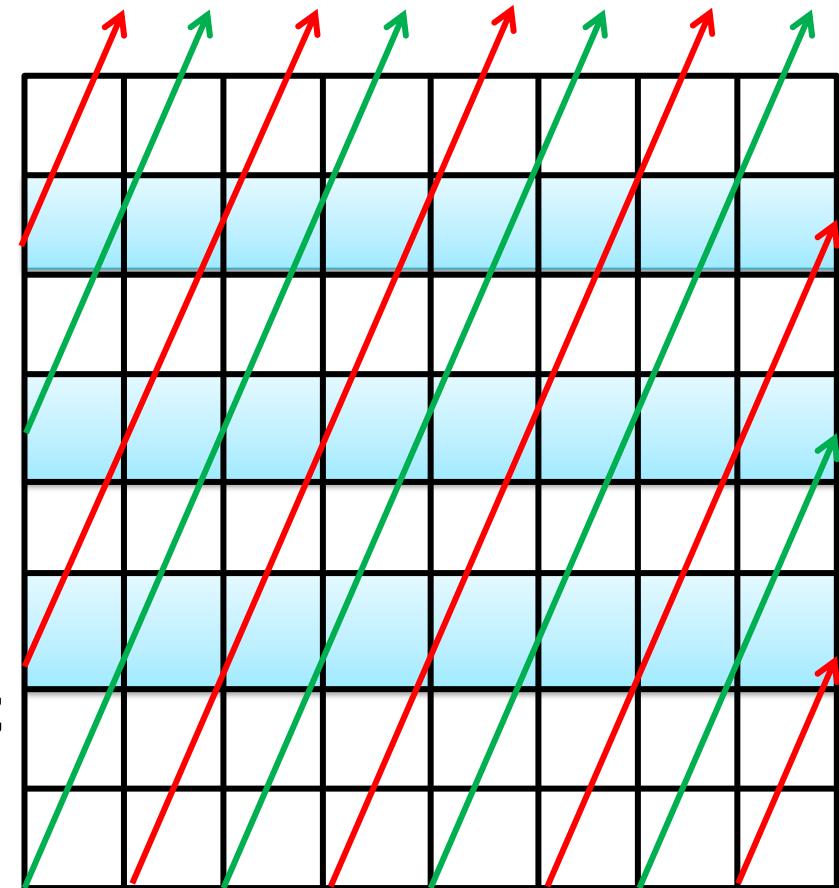
GPU並列化

- 単純に光線1本1本を順番にスレッドに割り当てるなら、同じメッシュを通過する光線でatomic演算が必要。
- 同じメッシュを通過する可能性がある隣合う光線を同時に計算しないでずらして計算。
- 3次元の場合は2次元メッシュから平行光線が出てくるため4グループ作り、順番にカーネル関数を立ち上げ。
- Fermi世代のGPUでは2倍くらいの性能向上。(Kepler世代ではatomic演算の性能が向上したおかげであまり性能は向上しなかった。)



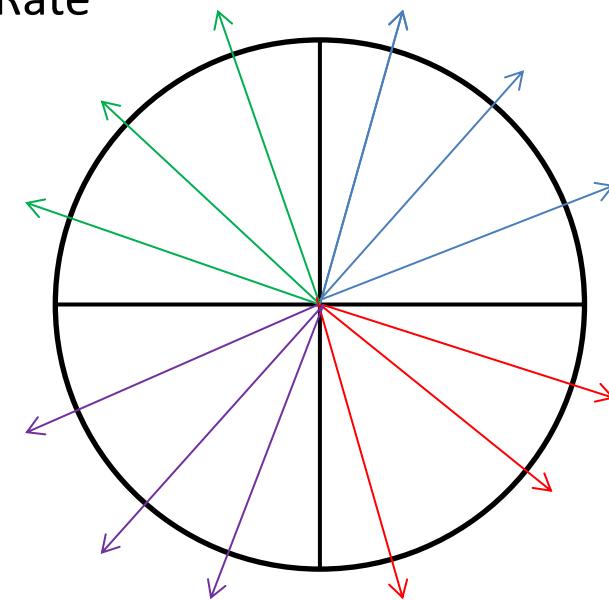
GPU並列化

- 単純に光線1本1本を順番にスレッドに割り当てるなら、同じメッシュを通過する光線でatomic演算が必要。
- 同じメッシュを通過する可能性がある隣合う光線を同時に計算しないでずらして計算。
- 3次元の場合は2次元メッシュから平行光線が出てくるため4グループ作り、順番にカーネル関数を立ち上げ。
- Fermi世代のGPUでは2倍くらいの性能向上。(Kepler世代ではatomic演算の性能が向上したおかげであまり性能は向上しなかった。)

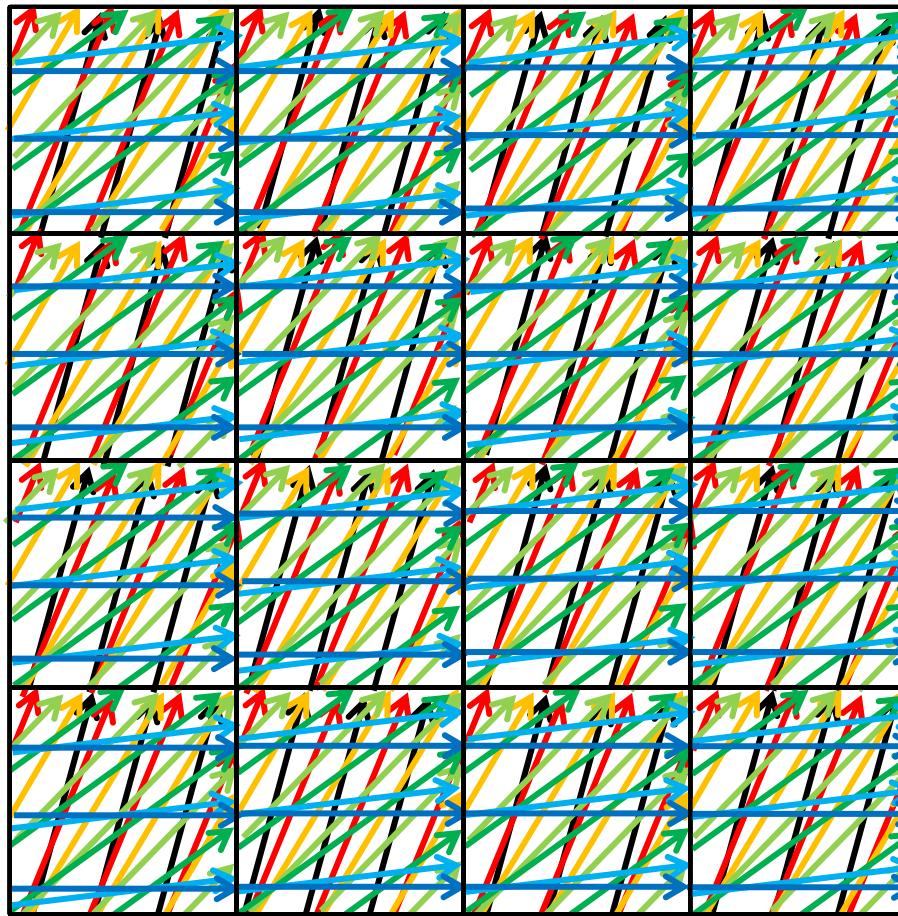


Multi GPU並列化

- HEALPixで作った方向をノード当たりのGPUの数で分割。
- OpenMPを使い、それぞれのGPUが担当する方向を並列計算。
- 最後にそれぞれのGPUのPhotoionization Rate を足し合わせ。



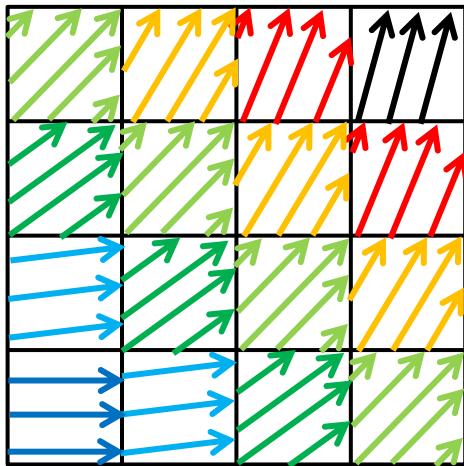
Multi Node並列化



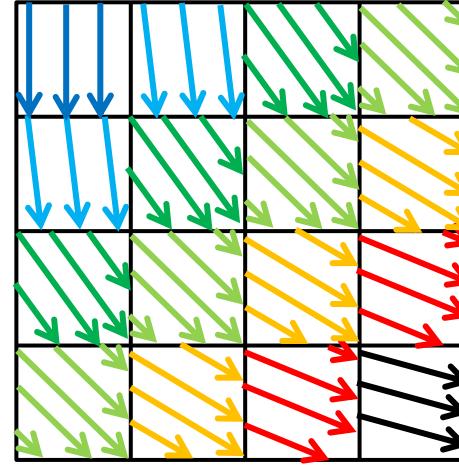
- Multiple Wave Front (WMF)法(Nakamoto 1998)を使用
- 3D box では平行光線の方向によって8グループの向き
- 最初と最後に働くないノードがあるが同じグループの方向数が十分多ければあまり問題にならない

Multi Node + Multi GPU並列化

GPU 0

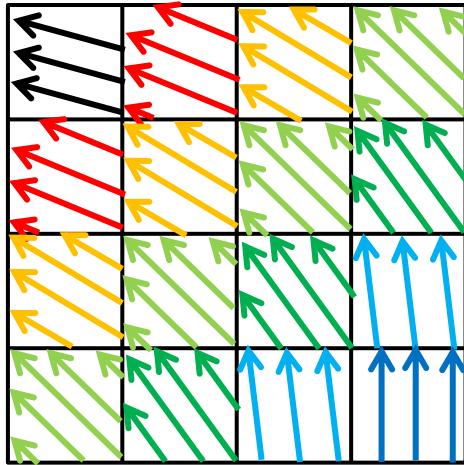


GPU 1

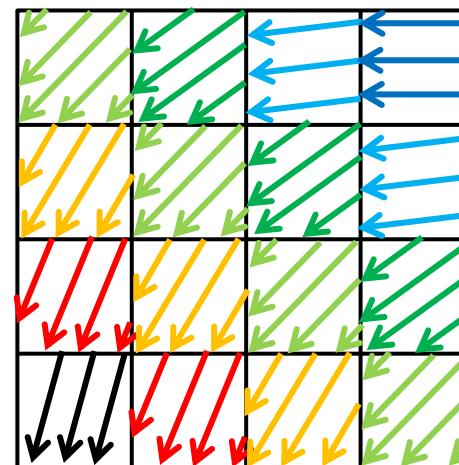


今回使う計算機にはノード当たりにGPUが4枚あるため8グループの内2グループづつを各GPUに割り当てる。

GPU 2

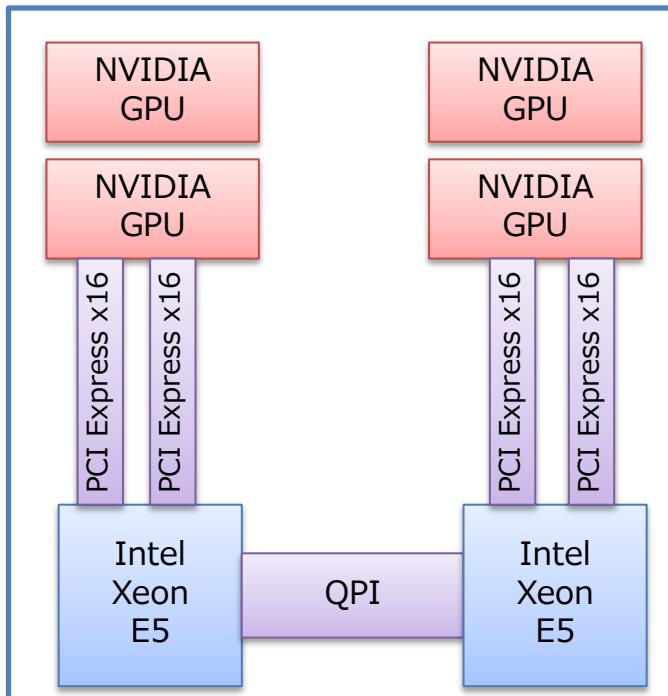


GPU 3

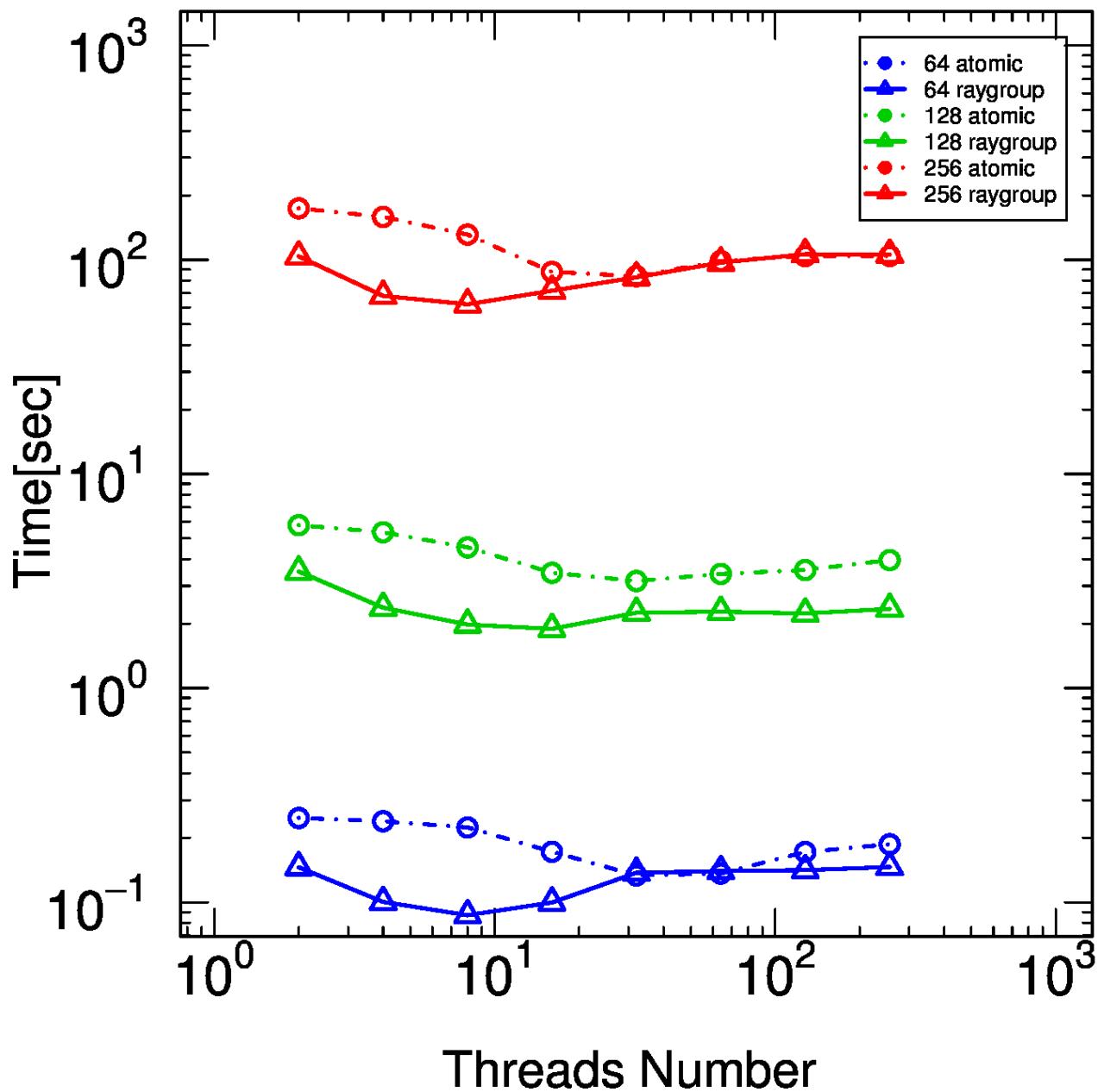


ベースクラスタ

- CPU : Intel Xeon E5-2670 @ 2.60GHz
 - 1ノード当たり8コア×2使用可能(16core×20.8GFLOPS)
- GPU : NVIDIA M2090
 - 1ノード当たり4GPU使用可能(4GPU×1330GFLOPS(单精度))
 - CUDAコア 512コア / 1枚
- ノード数 : 268



ブロックスレッド数の関係



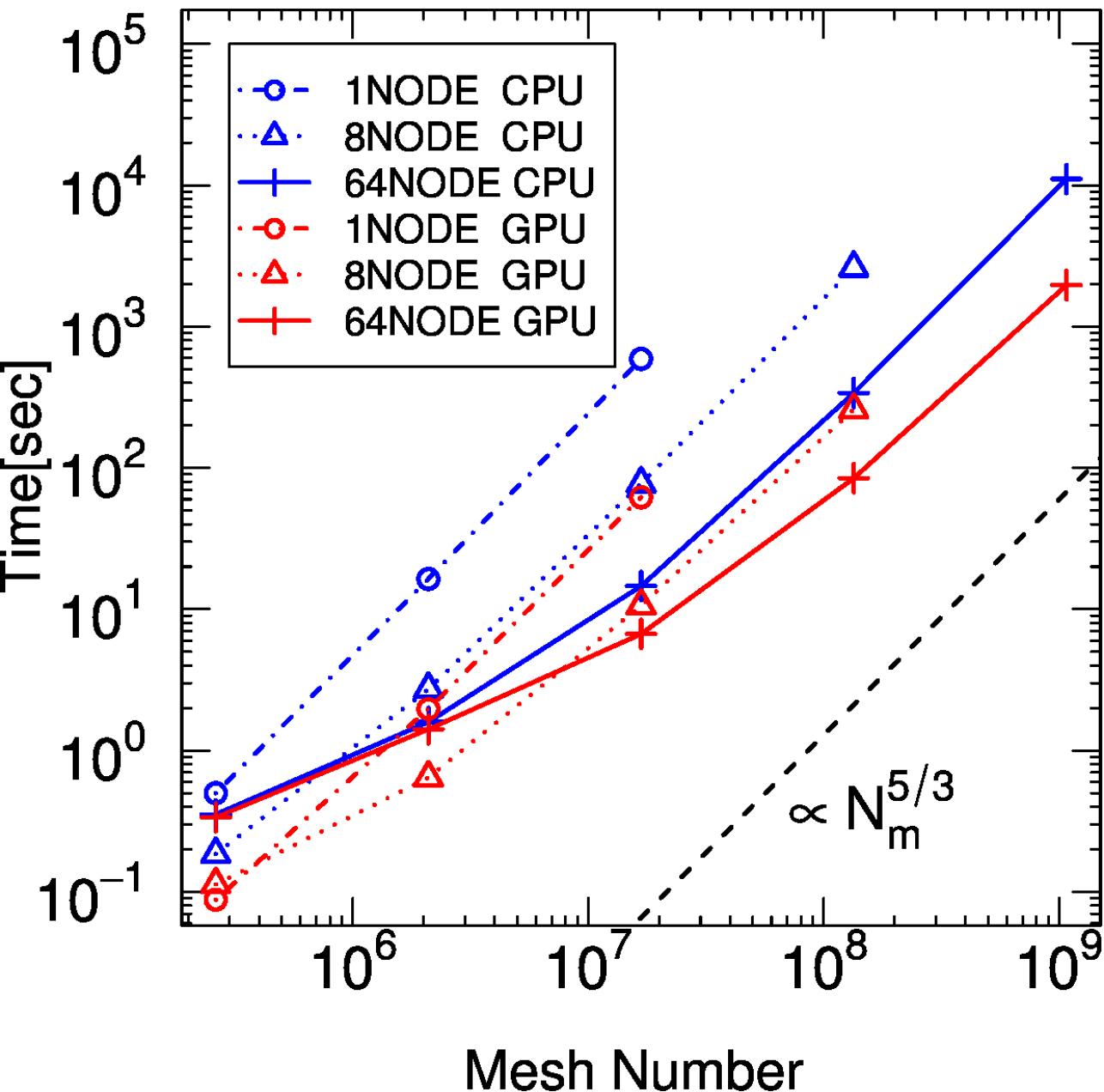
GPUにおける1ブロック当たりのスレッドとdiffuse photon計算時間の関係。
1node計算。

実線はray groupingでatomic演算を削除。点線はatomic演算使用。

最も計算時間が短くなるのはスレッド数が 8 or 16。

GPUは内部的に32スレッドが同時に動くため、それ以下に設定しても意味は無いと思っていたがかなり速くなった。
→ キャッシュの関係？

Diffuse Photon計算時間



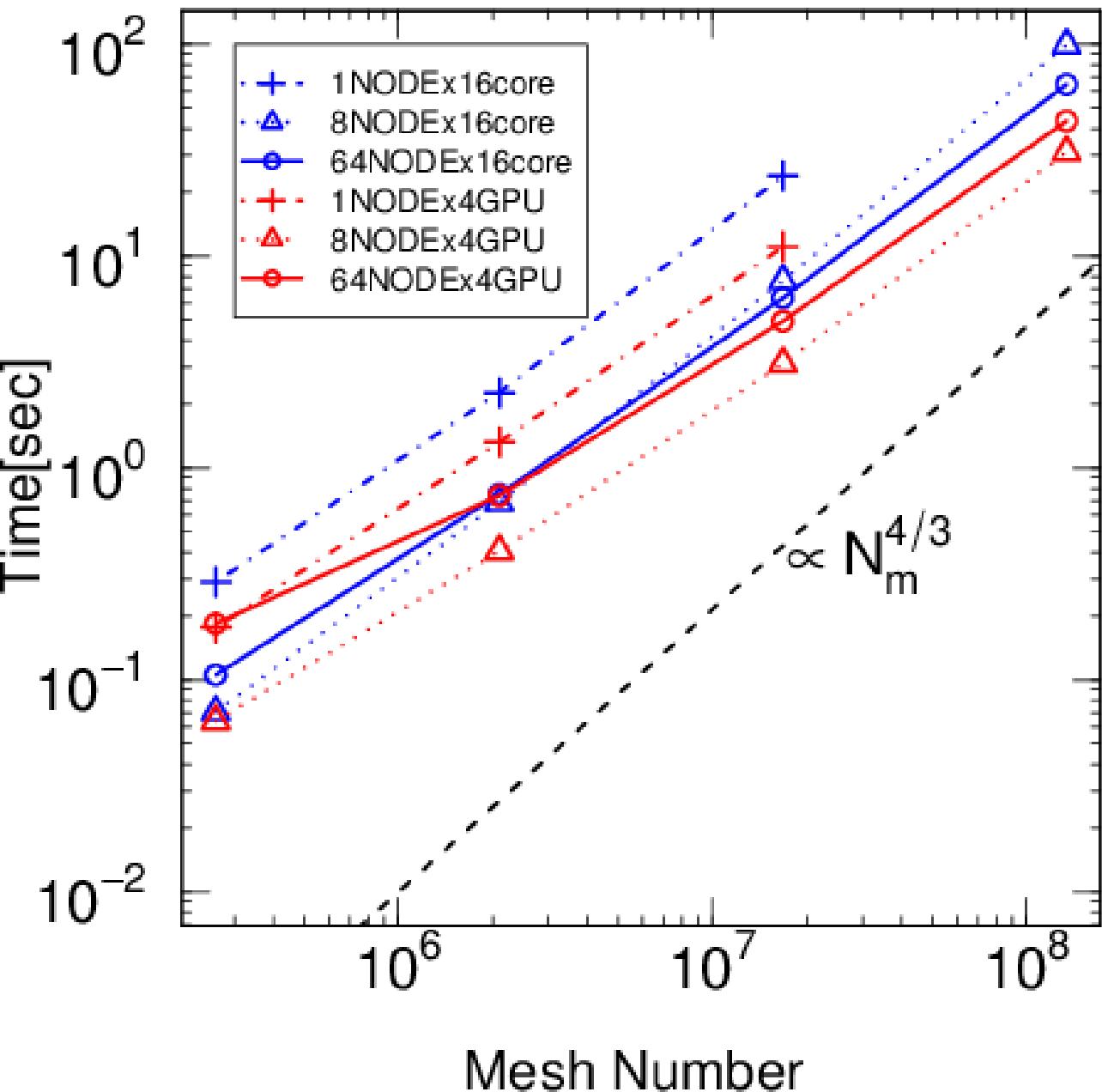
CPUは8x2core, GPUは4枚。

64NODE GPUの計算時間
はまだスレッド数を最適に
できていない。

8NODE , 512^3 mesh では
スレッド数最適化で
 $468\text{ s} \rightarrow 259\text{ s}$ になるので
64 NODE 1024^3 でも2倍
近くの高速化が期待できる。

CPUの10倍以上の高速化を
実現。

ARGOT部分計算時間①

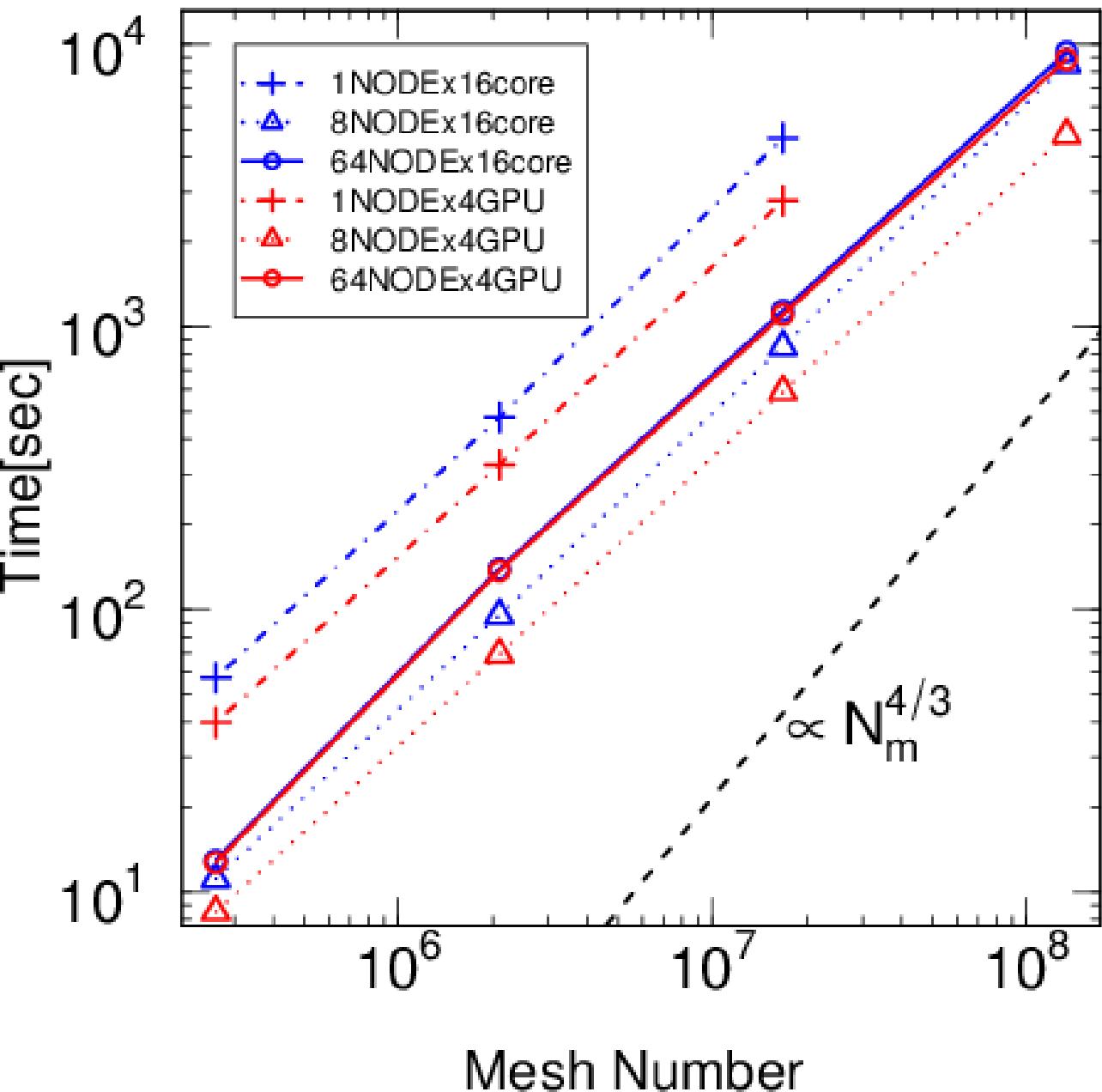


Point source数 : 1体

GPUではray-tracingの演算量に比べ、MPI通信量の方が影響が大きいためNODEを増やしすぎると性能が低下。

CPUではまだ演算量の方が影響が大きいためメッシュ数が大きいところではNODEを増やした方がいいが加速率は8倍が理想だが1.5倍くらい。

ARGOT部分計算時間②

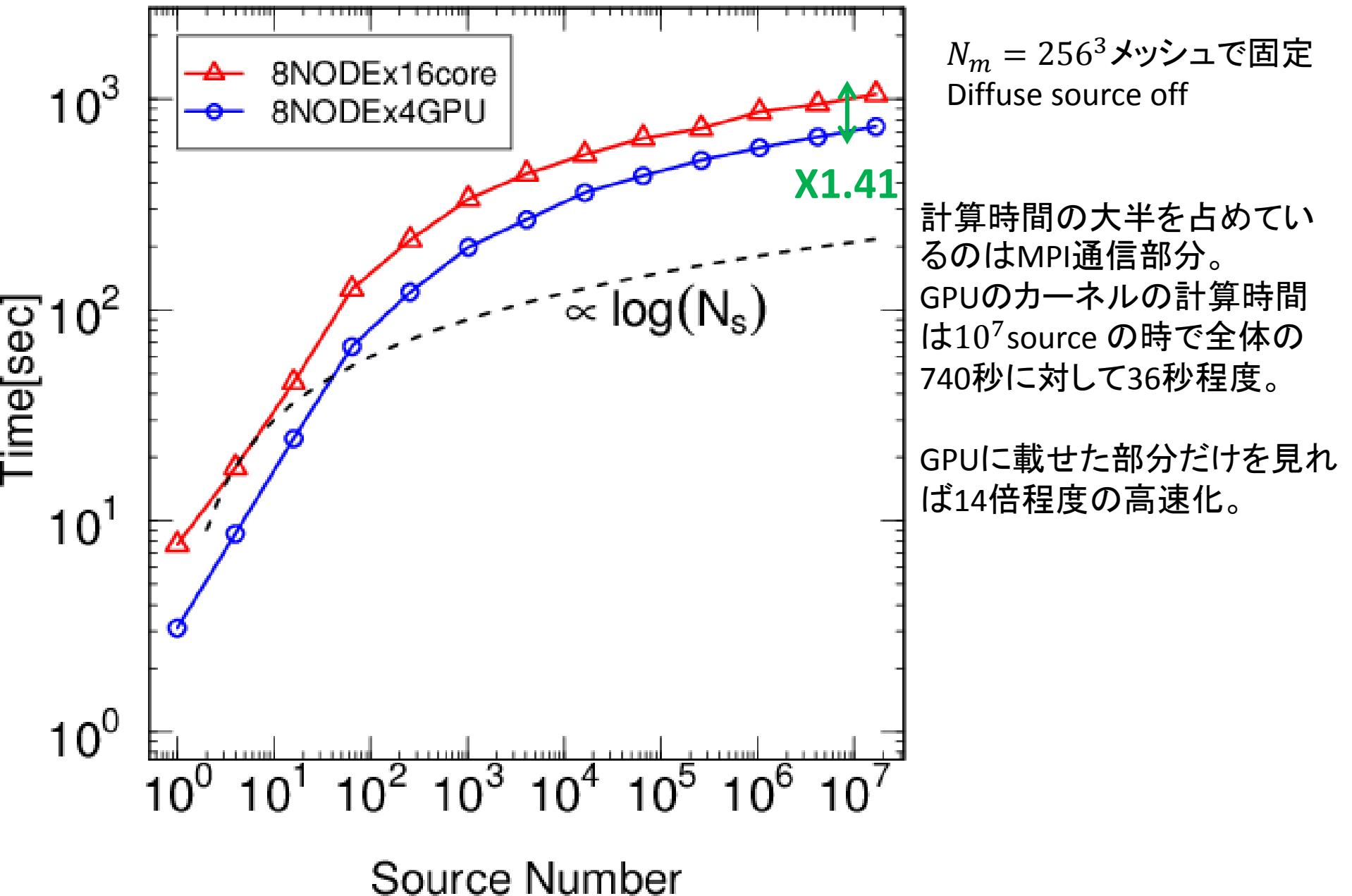


Point source数 : 4^{10}
 ≈ 100 万体

CPU、GPU共にray-tracingの演算量に比べ、MPI通信量の方が影響が大きいためNODEを増やしすぎると性能が低下。

メッシュ数の大きいところでは8NODEx4GPUが最も効率が良い。

Point source数依存性



まとめ

- 最適なブロックスレッドを設定することにより、十分メッシュ数が多ければ今までの計算の4-8割程度の高速化に成功した。
特に条件分岐が多いカーネルではwarpが有効に働くためブロックスレッド数を少なくすることにより無駄な分岐が最小限に抑えられ、今まで以上にキャッシュが効いたのではないかと思われる。
- 複数の点光源があるARGOTではMPI通信部分で時間がかかりすぎるため、いかにまとめてデータ通信ができるかが課題。
一つのメッシュに複数の光源が入っているような場合にうまくまとめる等。