統合ダイバータコードSONIC における MPMD システムの開発

清水勝宏

原子力機構　那珂

第20回NEXT研究会プログラム

2015年1月13日～14日　京都テルサ

内　容

1. 統合コード開発の難しさ

2. 今回開発したMPMDシステムの概略

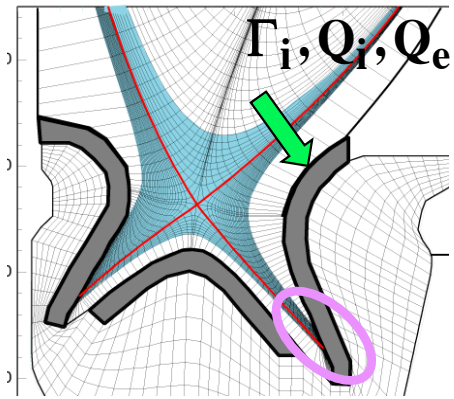
3. MPMDシステムの構成

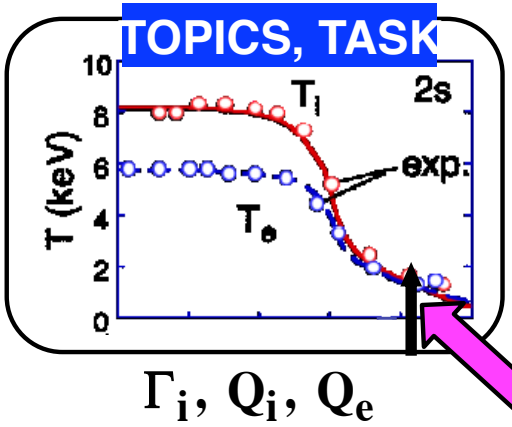
・コードスペックとは　・データ送受信、並列処理時の同期は、
・コミュニケータの定義　・グローバル変数の考え方
・派生型データ　　・実行方法

4. 統合コードのMPMDシステム化への手順

5. 具体的な適用例

原子力機構における統合ダイバータコード



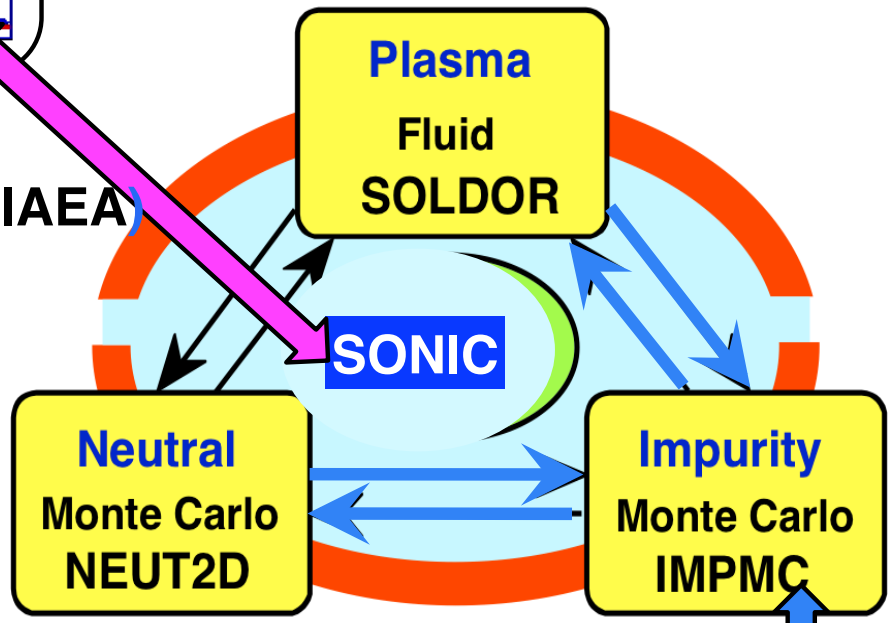


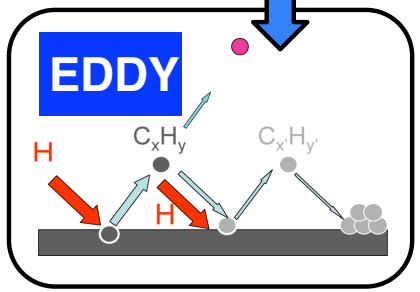
The divertor code is indispensable in divertor design work.

- **coupling of an MC impurity code (2008 IAEA)**
- **Interaction with core edge (2010 IAEA)**

core transport F_i, Q_i, Q_e

S_n, W_{cx}, W_{rad} SOL/divertor





SONIC has been consistently coupled to a 1.5 D tokamak transport code (TOPICS or TASK).

同様の統合コード計画：JINTRAC (EU) and FACETS (US)

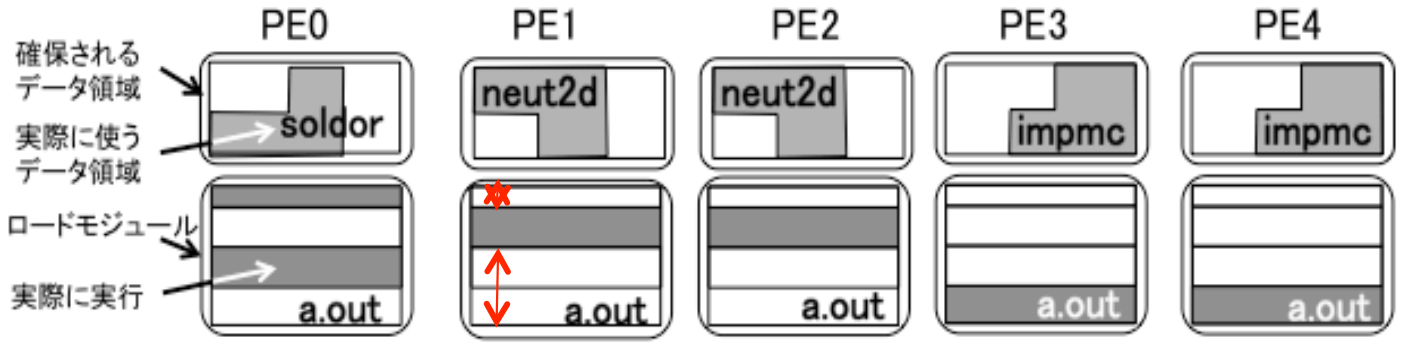
統合コード開発で問題となる点

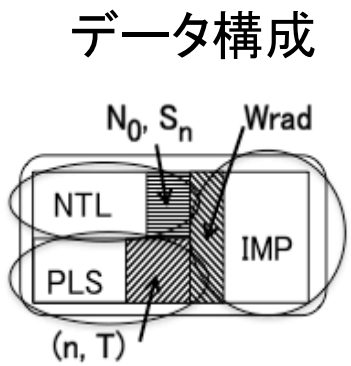
各コードは複数の開発者により同時進行で改良されているため、
各コードの独自性を保ちながら結合する必要が有る。

相互干渉をできるだけ限定する

並列計算機ではSPMDからMPMDシステムへ

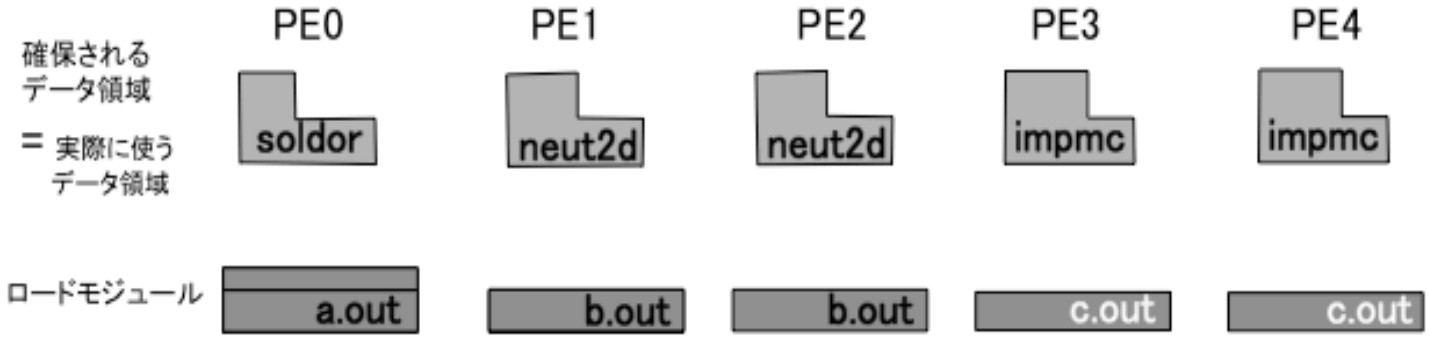
Single Program Multiple Data　全PEで実行するプログラムは同一





↔ 担当する計算以外はif文で飛ばす。

Multiple Program Multiple Data　各PEで実行するプログラムは異なる



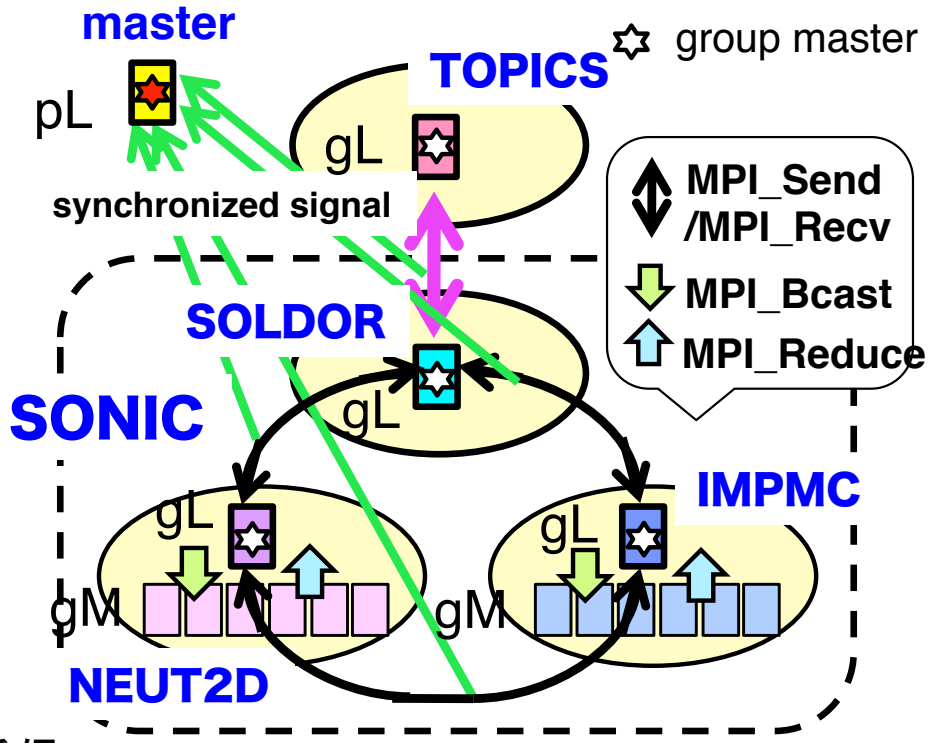
コード間のデータ転送は、Message Passing Interface(MPI)で行う。

今回開発したMPMDシステム

全てのPEは担当するプログラム毎にグループ分け

1. gL（グループリーダー）が、計算に必要なデータを gM に送信する (MPI_Bcast)。
2. gMのモンテ計算の結果は、gLが受け取り集計する（MPI_Reduce）
3. プログラム間のデータ転送（派生データ）は、各gLが自分で判断し、実行(MPI_Send/MPI_Recv)。
4. 各gLは処理が終われば データ転送終了／計算終了 のメッセイジをpLへ送信。
5. pLの役割は、終了メッセイジを受け取り、データ送受信と並列処理の計算ステップの同期を取ることである。

派生データ：メモリ上に不連続に存在するデータを1つの型として定義することで、1回の呼び出しで不連続なデータの通信を可能とする。



grand master => pL (project Leader)
group master => gL (group Leader)
slave => gM (group Member)

特徴：計算制御をpLが行うのではなく、外部ファイルが規定し、各gLが制御する。

今回開発したMPMDシステムの特徴

コードの結合にあたって、オリジナルソースに加えるべき修正点／追加点は最小限に。

各コードの実行制御は（どのステップで計算）各gLが判断し、pLが決定しない。

時間制御も指定したコードが行い、pLは行わない。

pLの役割は、データ転送／計算終了の同期をとる事である。

2008年開発のMPMDシステムと異なり、並列処理可能。入力、出力データが複数個可能。

ユーザが行うべき事

(1) 各コードのドライバールーチン作成し、

(2) コード間でのデータやり取りの派生データを定義し、

(3) 計算の流れを記述したファイル（コードスペック）を用意するだけで、

MPMDシステムが自動的に、必要な時に必要なデータをコード間で転送し、協調して計算を進める事ができる。

各コードのドライバールーチンは，

以下の5つのステップから成る。

1．init　：配列の割り付け、変数の初期化。
2．parm：入力データの読み込み。
3．prof　：初期分布。
4．exec：1ステップ（dtim）だけ計算を進める。
5．term　：計算終了時の後処理。

（注）このステップの考え方は、SONIC、TOPICSでも同じであるが、その表現はTASKにならっている。

コードスペック


```
comt : define Data Flow between Codes of TOPICS/SONIC
code : master  topics  soldor neut2d  impmc  ofmc
time : soldor
```

コメント
コード名のリストアップ

時間は、soldorが制御

派生型データのリストアップ

```
dtyp : TOK    topics
dtyp : DIV    soldor
dtyp : NTL    neut2d
dtyp : IMP    impmc
dtyp : FLX    impmc
dtyp : NBI    ofmc
```

各ステップでの実行順序

```
init 1 : topics
init 1 : soldor
init 1 : ofmc
init 1 : neut2d
init 1 : impmc

parm 1 : topics   => TOK
parm 1 : soldor   => DIV
parm 1 : ofmc     => NBI
parm 1 : neut2d   => NTL
parm 1 : impmc    => IMP
```

数字は実行の順序（順位）

step

```
prof 1 : topics    => TOK
prof 2 : ofmc      TOK                  => NBI
prof 3 : neut2d    TOK NBI              => NTL
prof 4 : impmc     TOK DIV NTL NBI      => IMP
prof 5 : soldor    TOK IMP NTL          => DIV
prof 6 : topics    DIV NTL IMP NBI      => TOK
```

step

```
exec 1 : ofmc      TOK                  => NBI
exec 1 : neut2d    TOK NBI              => NTL
exec 1 : impmc     TOK DIV NTL NBI      => IMP FLX
exec 2 : soldor    TOK IMP NTL          => DIV
exec 3 : topics    DIV NTL IMP NBI      => TOK
```

```
term 1 : soldor
term 1 : ofmc
term 1 : neut2d
term 1 : impmc
term 2 : topics
```

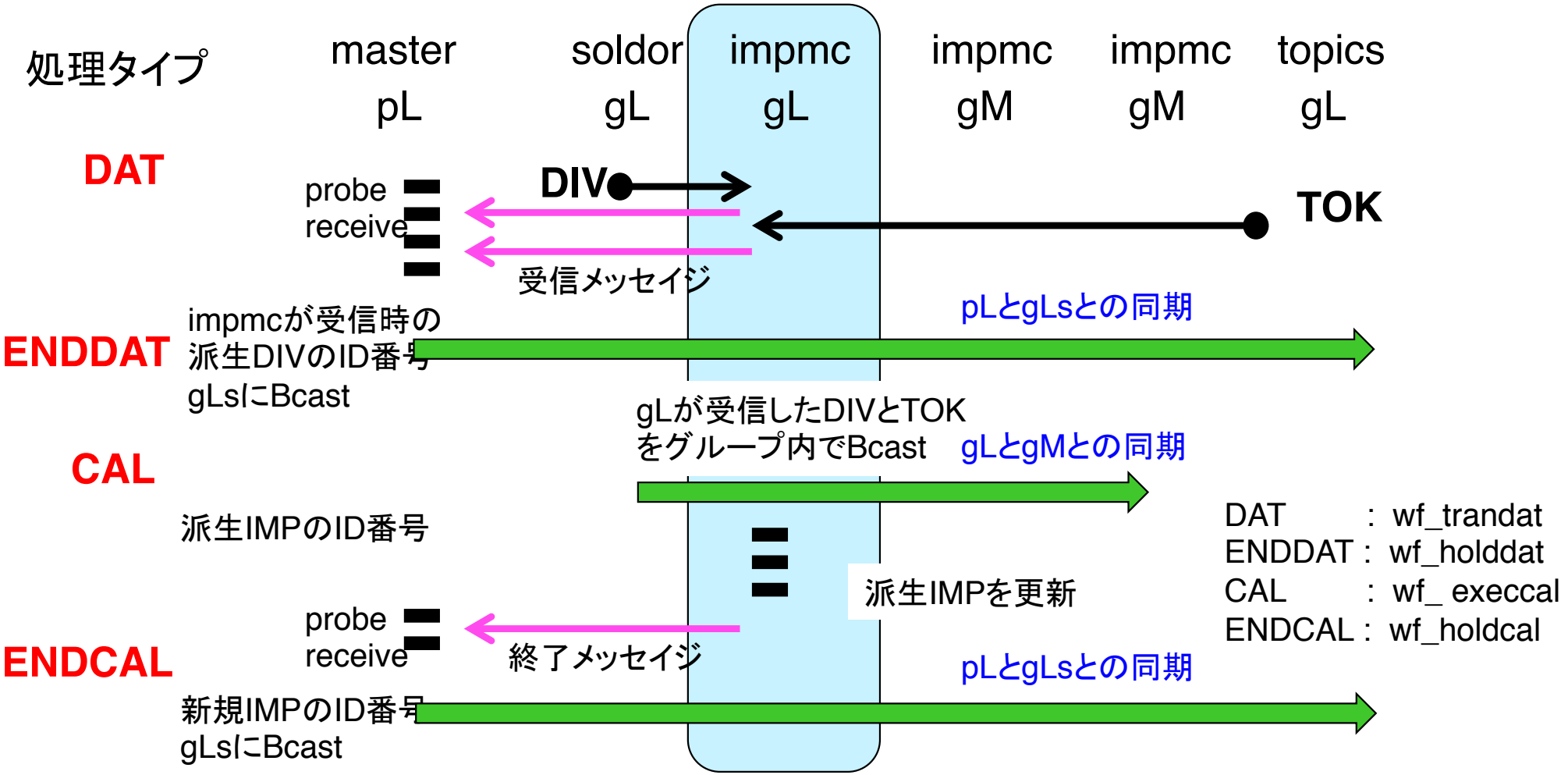
neut2dコードはTOKとNBIのデータを受信し、計算してNTLのデータを更新する。

ofmc, neut2d, impmcは、同順位なので並列処理

プロセス内での同期

MPIを用いたコード開発で重要な事は、データの受信と送信のタイミングを如何に合わせる（同期）かである。開発したMPMDシステムでは、この同期はシステムが行い、ユーザが意識する必要は無い。

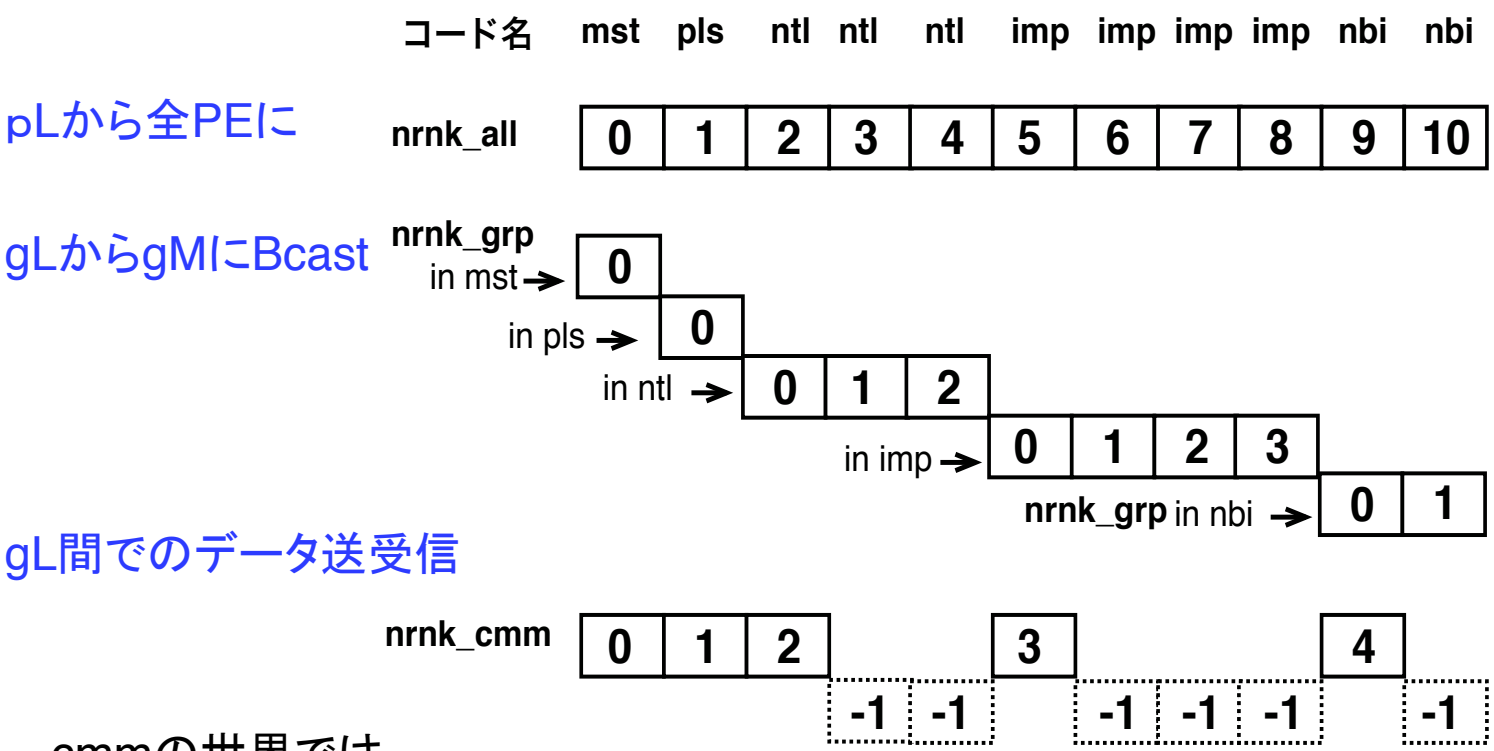
```
exec 1 : impmc    DIV TOK   => IMP
```



コミュニケーターの定義

コミュニケータは3つ定義される。all（全PEを対象）、grp（グループ毎に定義、gLとgMから成る）、そして、グループ間でのデータ転送を行うcmm（pLとgLから成る）である。

mpirun –np 1 lod_mst : -np 1 lod_pls : -np 3 lod_ntl : -np 4 lod_imp : -np 2 lod_nbi



cmmの世界では、
gLのランクはgroup番号、gMには-1

派生データの型番の定義

モジュール**mod_dtypdef.f** を用いて、派生データの型番を定義する。

A. Takayama et al., J. Plasma Fusion Res. SERIES, Vol. 9 (2010)

```
!*************************************
 subroutine dtypdef_IMP(ktag,kerr)
!*************************************
 use mod_glb_imp
                          派生型データの名前定義と初期化
 call dtyp_init( "IMP", ctg_imp )
 call dtyp_addv( Ghd_imp, 1 )
 call dtyp_addv( Git_imp, 1 )
 call dtyp_addv( Gtm_imp, 1 )

 call dtyp_addv( G_wrad,  1 )   登録する変数
 call dtyp_addv( G_wpuf,  1 )
       配列の場合、1の替わりにsize(G_xxx)とする。

 call dtyp_term( idtag, ierr ) 登録終了   idtagは派生データの型番
 itg_imp = idtag                モジュール変数として型番定義

end subroutine dtypdef_IMP
```

integer, real, characterの混在可能

派生データの最初と型番の指定で送受信可能

派生データIMPをsoldor(grp:1)に送る時は、

```
if( nrnk_cmm == 3 ) then
 call MPI_Send( Ghd_imp, 1, itg_imp, rpe, 125,
                                    nwld_cmm, ierr )

elseif( nrnk_cmm == 1 ) then
 call MPI_Recv( Ghd_imp, 1, itg_imp, spe, 125,
                                nwld_cmm, istat, ierr )

endif
```

ここで注意すべきは、型番idtagは正とは限らないこと、
計算機によって、gLとgMで番号が異なる事もある。

MPMDシステム化

(1) 各プログラムのドライバールーチン作成し、

subroutine prg_init（配列の割り付け), prg_parm（入力データの読み込み）、
prg_prof(初期分布）、prg_exec（時間発展）、prg_term（後処理）

(2) コード間でのデータやり取りの派生データを定義し、

ライブラリー dtyp_init, dtyp_addv, dtyp_term を用いて
サブルーチン　subroutine dtypdef_IMP(ktag,kerr) を作成

(3) 計算の流れを記述したファイル（コードスペック）を用意する

```
comt : define Data Flow between Codes of TOPICS/SONIC
code : master  topics  soldor neut2d  impmc  ofmc
time : topics
dtyp : TOK    topics
dtyp : DIV    soldor
dtyp : NTL    neut2d
init 1 : topics
init 1 : soldorinit 1 : ofmc
parm 1 : topics   => TOK
parm 1 : soldor   => DIV
```

独断と偏見!?

他のシステム(Kepler)との比較

・IMASでは、汎用ソフトKepler(ワークフロー)を用いて、統合コードの開発と結果のデータベース化
　データフローをGUIで定義し、実行可能モデルを構築　IMAS :The ITER Integrated Modelling & Analysis Suite)

・機構(CCSE)では、グリッドコンピューティングの技術の下に、統合化ツールSOAFを開発
　ファイルの存在がコードの実行をトリガー

MPMDシステムが統合化開発の問題点を解決

(1) 各コードの計算に必要なデータは、ファイルより取得するのでIOが大きく計算効率が悪くなり、大規模な統合コードのプログラミングには不適。

⇨ MPIによるデータ交換なので高速である。

(2) ディレクター(gLに相当)は、アクター(gMでの計算)での計算のトリガーを制御する。
間引き計算(各コード入力データ)の条件がディレクターの判断には必要。

⇨ コードスペックにより、各gLが自律的に判断し(計算にも参加)、データ交換、計算(間引き)を行い、pLは同期を取る。データ交換、同期等の制御はシステムが行う。

(3) 複雑な統合コードのデータ構成をGUIによるデータフローの指定は困難。

⇨ コードスペックの記述は簡単。

デモプログラム　TOPICS/SONIC

TOPICSモデル

$$\frac{dN}{dt} = -\frac{N}{\tau_N} + \Gamma_{recy} + F_{puffM} + F_{nbi}$$

$$\frac{dW}{dt} = -\frac{W}{\tau_w} + W_{oh} + W_{nbi} - W_{rad}$$

$$N = \int n_e(r)dV, \quad W = \int \frac{3}{2}\left(n_e(r)T_e(r) + n_i(r)T_i(r)\right)dV$$

$$(N,W) \quad => \quad (n_{e0}, T_{e0})$$

$$n_e(x) = n_{e0} \cdot \left\{(1-b_n)(1-x^2)^{0.5} + b_n\right\},$$

$$T_e(x) = T_{e0} \cdot \left\{(1-b_{T_e})(1-x^2)^{1.25} + b_{T_e}\right\},$$

$$\tau_W = 0.85\ \mathrm{s\ (OH)}, 0.25\mathrm{s\ (L)}, 0.80\mathrm{s\ (H-mode)},$$

$$\tau_N \sim 2 \cdot \tau_W \ \mathrm{(OH/L/H)}$$

H=>L遷移　Te=0.6 keV at x=0.95

SOLDOR (2点ダイバータモデル)

b_n, b_{T_e}, b_{T_i}　at mid plane

NEUT2D

$$\Gamma_{recy} = \gamma \cdot \frac{N}{\tau_N} \quad (N, \tau_N, \gamma) \text{ from topics}$$

Γ_{puffM} = input data &

density control($t > 4$ sec)

OFMC

$P_{nbi}(t)$ = input data

$P_{abs} = P_{nbi} \cdot \eta(n_{e0})$

n_{e0} from topics

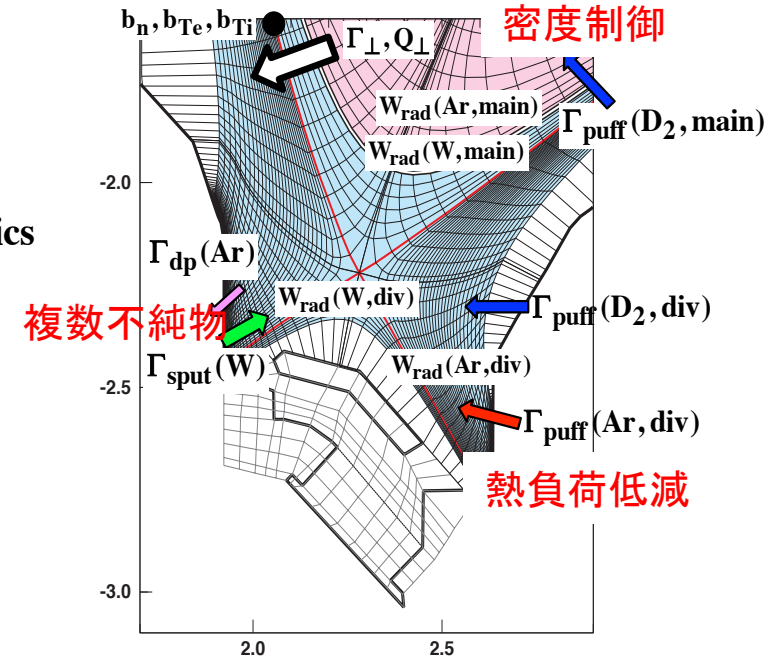
$F_{nbi} = P_{abs} / 80.0\mathrm{keV}$

$W_{nbi} = P_{abs}$

IMPMC_1/2

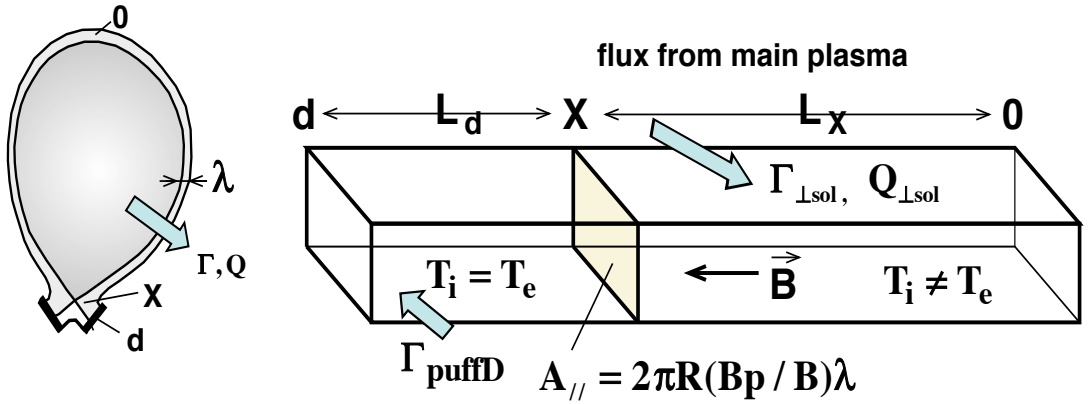
$$W_{rad} = W_{Ar} + W_W$$

JT-60SAのシミュレーションを模擬



SOLDOR：2点ダイバータモデル

清水勝宏、滝塚知典、J. Plasma Fusion Res. **80** (2004) 183.



$$f_{//x} = \Gamma_{\perp sol} / 2A_{//n} = \Gamma_{\perp sol} / 4\pi R\lambda_n (B_p / B)$$

$$q_{//x} = Q_{\perp sol} / 4\pi R\lambda_{q//} (B_p / B)$$

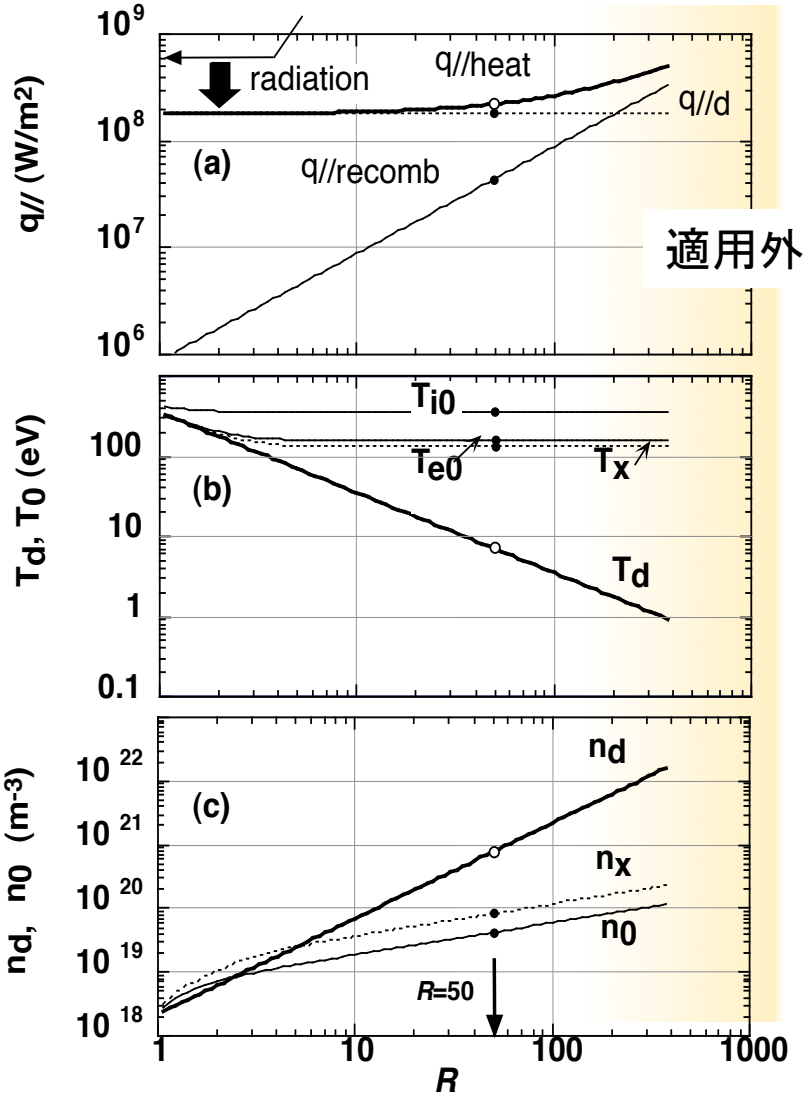
2点ダイバータモデルとは、

X点およびダイバータ板での物理量は、粒子束と熱流束（$f_{//x}$ $q_{//x}$）とフラックス増倍率 R によって表す事ができる。

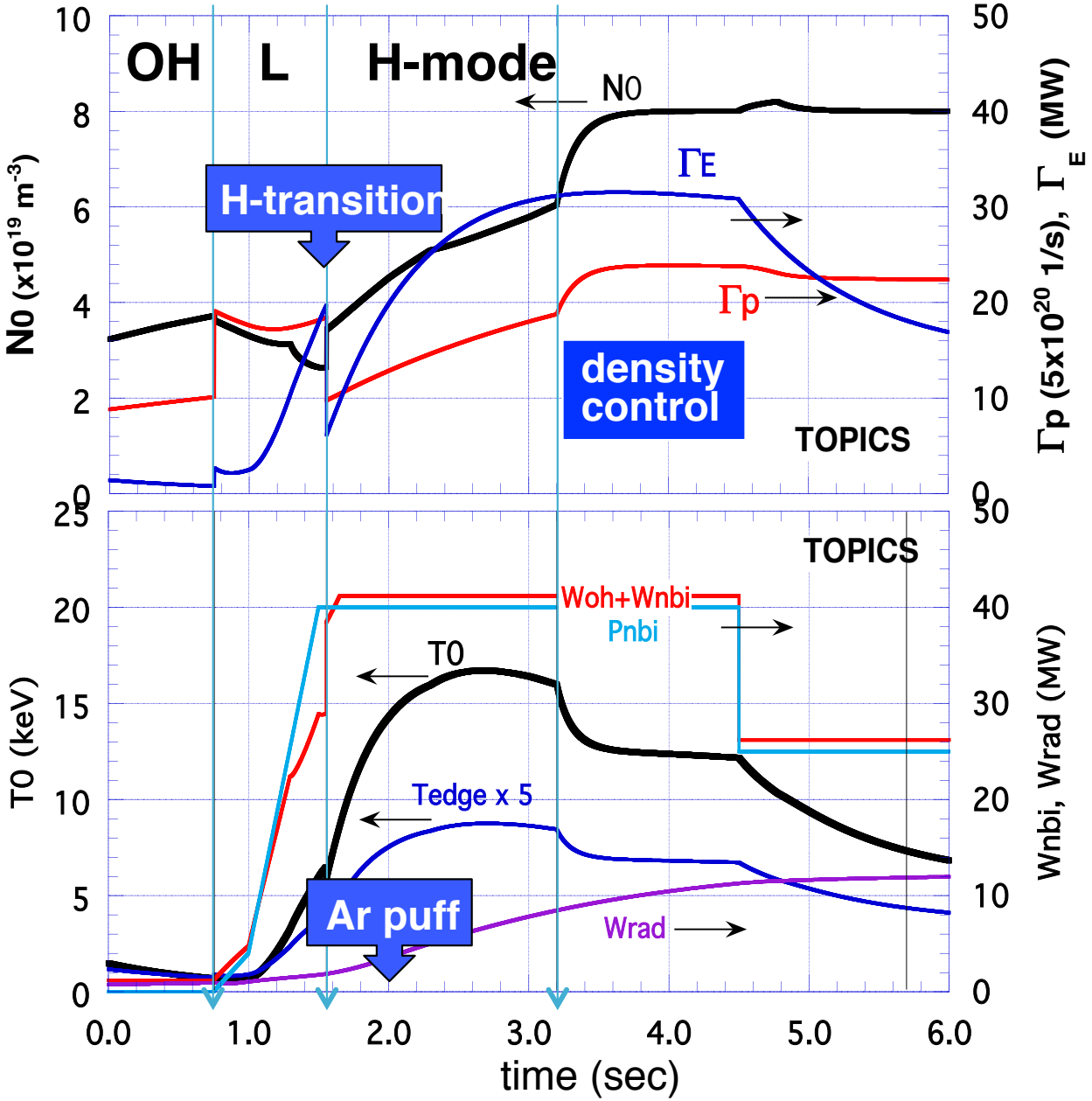
ITERのパラメータ

$$\Gamma_{\perp sol} = 1.5\times10^{23}\ s^{-1},\ Q_{\perp sol} = 80\ MW$$

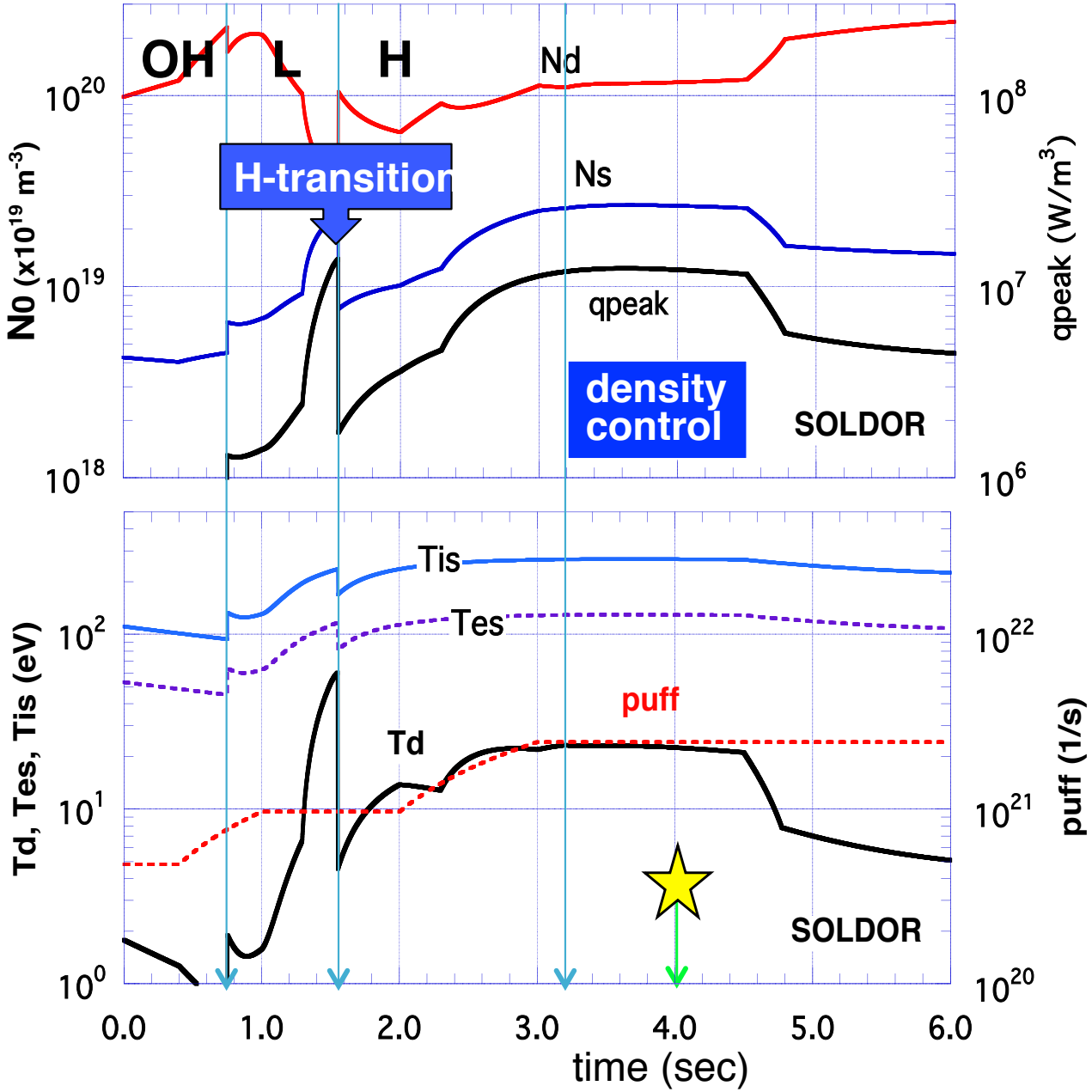
$$F_{rad} = 0.7,\quad \lambda_n = 4.7cm,\quad \lambda_{q//} = 1.7cm$$



計算結果 –TOPICS-


OH L H-mode
N0
ΓE
Γp
H-transition
density control
TOPICS
N0 (x10^19 m^-3)
Γp (5x10^20 1/s), Γ_E (MW)
TOPICS
Woh+Wnbi
Pnbi
T0
Tedge x 5
Ar puff
Wrad
T0 (keV)
Wnbi, Wrad (MW)
time (sec)

計算結果 –SOLDOR-



$T_d = 20$ eV

$T_{es} = 130$ eV

$T_{is} = 270$ eV

$N_d = 1.2 \times 10^{20}$ m^{-3}

$N_s = 2.7 \times 10^{19}$ m^{-3}

$q_{peak} = 12$ MW / m^2

/home/shimizu/MPMD5/
exe/drout_0223C

ま と め

- 独立な複数の物理モジュールがMPIによりデータを必要に応じて交換し、協調して計算を進めるシステム（MPMD)の開発を行った。

- 開発者が行うべき事は、3点
  各コードのドライバールーチン作成し、
  コード間でのデータやり取りの派生データを定義し、
  計算の流れを記述したファイルを作成

- 簡単なモデルであるが、TOPICS、SONICを模した6個のモジュールを本システムで統合し、その有効性を実証した。

- 今後SONICを解体し、この新しいMPMDシステムで統合化し直す。
  複数不純物計算、コアとの結合（TOPICS & IMPACT）へと進める。