ステンシル系プログラムによるFX100の性能評価と高速化チューニング

高木　亮治
宇宙航空研究開発機構
宇宙科学研究所

内容

- JSS2の紹介
  - SORA-MA（富士通FX100）
- SORA-MAの性能評価
  - STREAM(TRIAD)
  - ステンシル系プログラム(UPACS-Lite)
- 高速化チューニング
  - ステンシル系プログラム(UPACS-Lite)
- まとめ

JSS2 (JAXA Supercomputer System Generation 2)

新スパコン 宙 (SORA: Supercomputer for earth Observation, Rockets, and Aeronautics) システム概要図

調布事業所

計算システム (SORA-MA (MAin))

第二期 2015年4月～
理論演算性能：1.31 PFLOPS
総メモリ量：40 TiB
総ノード数：1296
（32コア/1CPU）

→ 第三期 2016年4月～
理論演算性能：3 PFLOPS 以上
総メモリ量：90 TiB 以上
総ノード数：3000 以上
（32コア/1CPU）

プレポストシステム (SORA-PP (PrePost))

理論演算性能：53.7 TFLOPS
総コア数：1920 コア
総メモリ量：10 TiB
2CPU, 64 GiB ×160 ノード

大メモリ計算システム (SORA-LM (Large Memory))

理論演算性能：2.10 TFLOPS
総コア数：80 コア
総メモリ量：3.5 TiB
2CPU, 1.0 TiB ×2 ノード
2CPU, 0.5 TiB ×3 ノード

ログインシステム (SORA-LI (LogIn))

理論演算性能：1.34 TFLOPS
総コア数：48 コア
総メモリ量：1.5 TiB
2CPU, 0.37 TiB ×4 ノード

IO 結合スイッチ

Ethernet

管理・制御部

ファイルシステム部 (SORA-FS)

第一期 2014年10月～ 磁気ディスク：1 PB
→ 第二期 2015年4月～ 磁気ディスク：5 PB

遠隔部

所内 LAN and SINET4

つくば事業所	プレポストノード (SORA-TPP) 8.40 TFLOPS 総コア数：300 コア（2CPU, 64 GiB ×25 ノード） ログインノード (SORA-TLI) 0.336 TFLOPS 総コア数：12 コア （2CPU, 0.37 TiB ×1 ノード） ファイルシステム (SORA-TFS) 200 TB	
角田事業所	ファイルシステム (SORA-KFS) 100 TB	
相模原事業所	ファイルシステム (SORA-SFS) 100 TB	

J-SPACE

Jaxa's Storage Platform for Archiving, Computing, and Exploring

アーカイバ部

ディスクキャッシュ
容量：0.7 PB

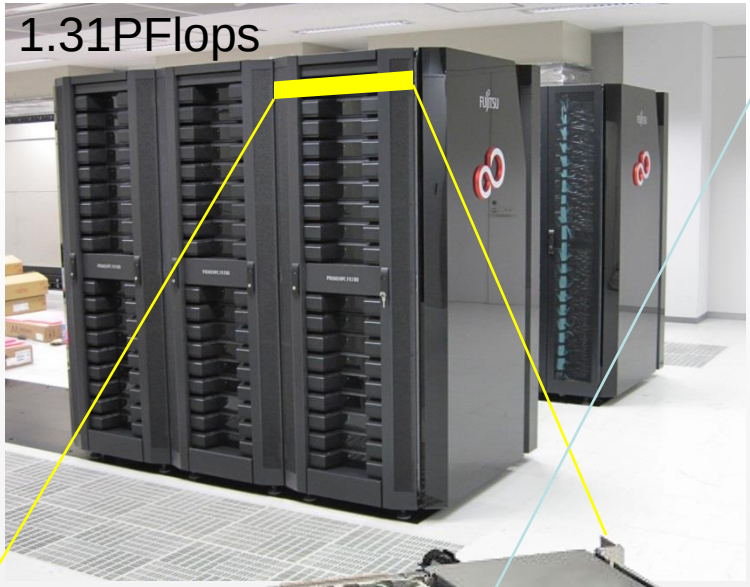
テープ容量：20 PB

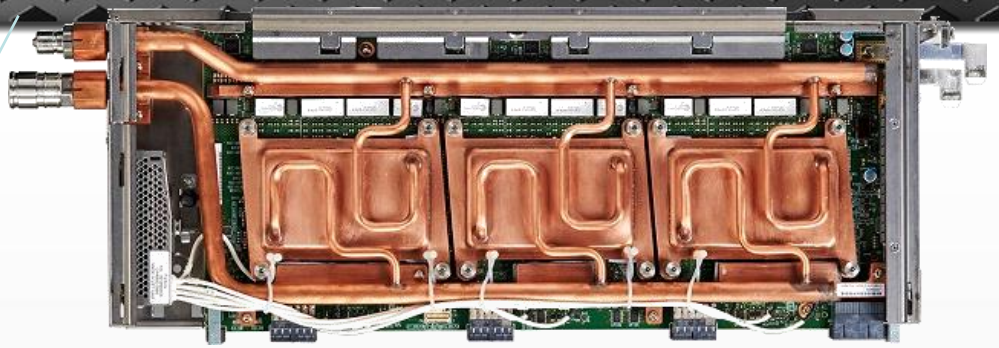
JSS2システム概要

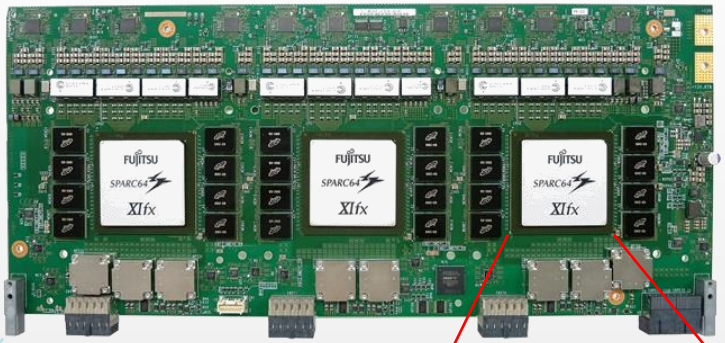
- SORA：**S**upercomputer for earth **O**bservation, **R**ockets and **A**eronautics
- J-SPACE：**J**AXA's **S**torage **P**latform for **A**rchiving, **C**omputing and **E**xploring (HPSS)

呼称 SORA-XX	主な用途	特徴
MA (MAin)	計算サーバ	1.31PFLOPS, 40TB (→ 3+PFLOPS, 90TB)
PP (PrePost)	前後処理	53.7TFLOPS, 10TB, 160ノード
LM (LargeMemory)	大メモリ	1TBノード, 512GBノード
LI (LogIn)	ログイン	4台の冗長構成
FS (FileSystem)	ファイルシステム	5PBのRAID6ディスク
TPP (TsukubaPP)	筑波計算サーバ	SORA-PPと同一構成が25ノード
TFS, KFS, SFS Tsubuka, Kakuda, SagamiharaFS)	筑波、角田、相模原のファイルシステム	データー時保管、高速データ転送

SORA-MA（富士通FX100）







3TFlops

12TFlops

1TFlops

SORA-MA,PP

- SORA-MA（FX100）
  - Fujitsu SPARC64 Xifx 1.975GHz
  - 1ノード：1CPU、32コア
  - 16コア/CMG × 2CMG/CPU × 1 = 32コア
  - 1.011TFLOPS、431GB/s → B/F=0.43
- SORA-PP（PRIMERGY RX350 S8）
  - Intel Xeon E5-2643V2 3.50GHz
  - 1ノード：2CPU、12コア
  - 6コア/CPU × 2CPU
  - 336GFLOPS、119.4GB/s → B/F=0.36

SORA-MAの性能評価

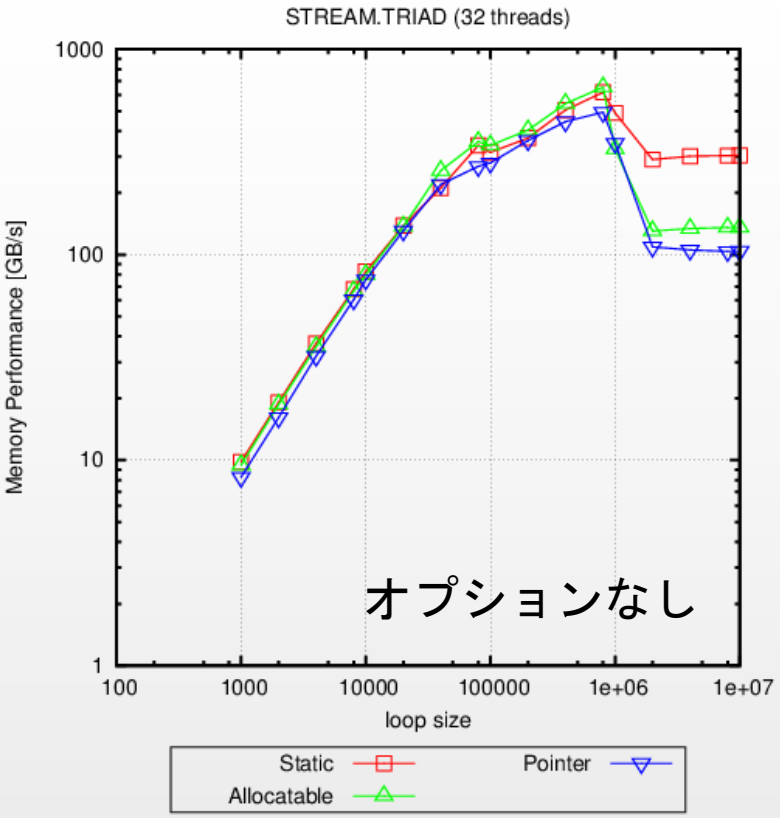
STREAM（TRIAD）

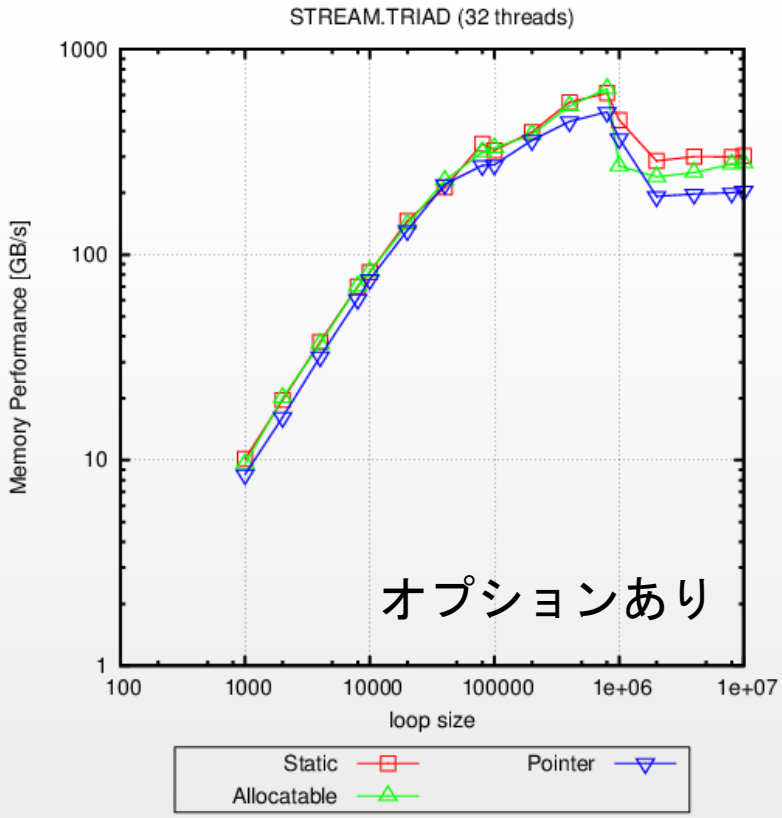
ステンシル系プログラム（UPACS-Lite）

STREAM(TRIAD)

```
do i=1,n
  a(i) = b(i) + S * c(i)
enddo
```

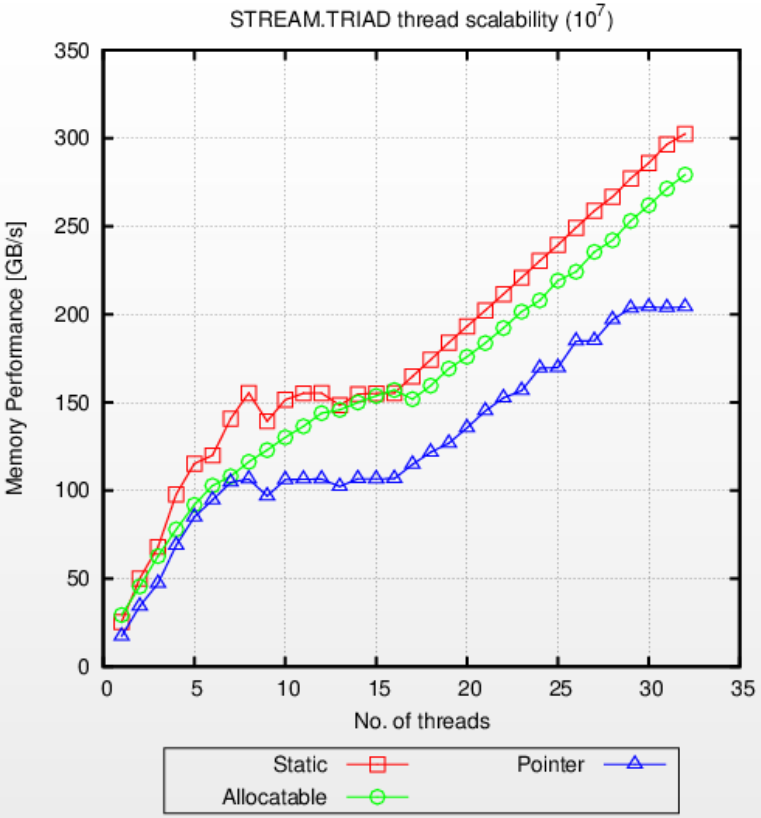
TRIAD@FX100

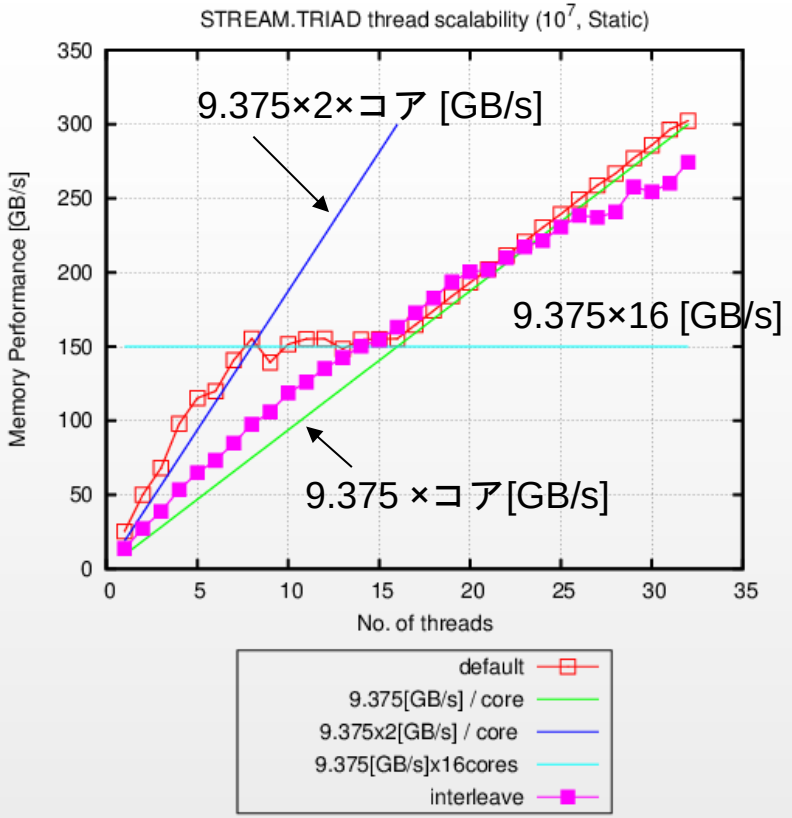




- 配列の種類によって性能が変化
  - 静的（302GB/s）～動的（278GB/s）> ポインタ（206GB/s）
  - 実行時オプション（lpgparm –l demand）の指定が必要

TRIAD@FX100





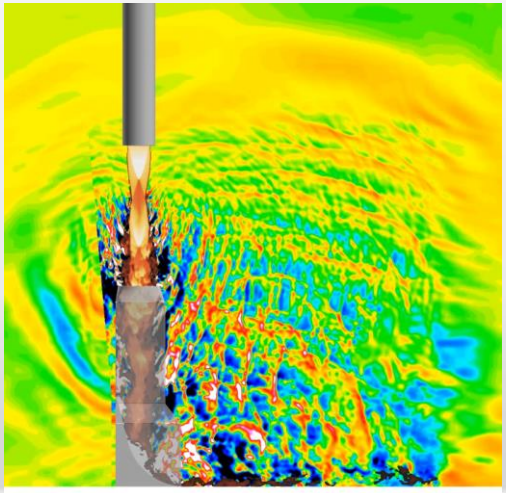
- CMG×2による複雑な挙動
- numactl –interleave=allでほぼ線形な挙動
  - 1スレッド：25.28GB/s（default）、13.66GB/s（interleave）

UPACS-Lite

ステンシル系プログラム

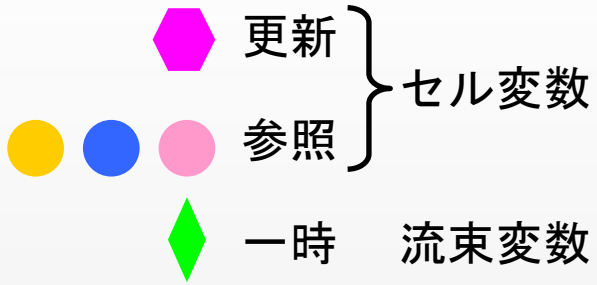
UPACS-Lite

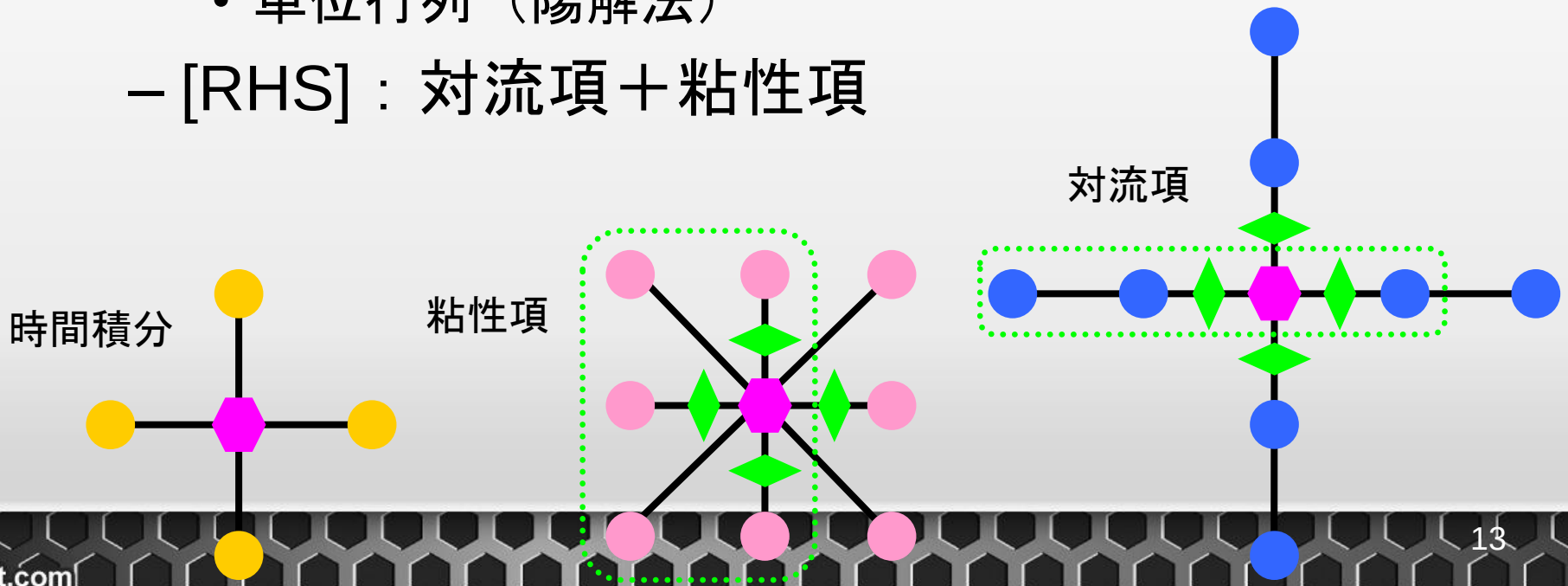
- UPACS-Lite：圧縮性流体解析プログラム UPACSのサブセット
  - ステンシル系プログラム
- マルチブロック構造格子
- 陰解法（Block Red-Black、2nd Euler、内部反復は2回）
- MUSCL+SHUS
- 乱流モデルはなし
- MPI + OpenMP



UPACS-Lite

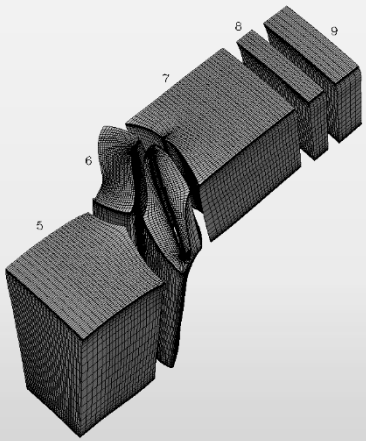
- 離散方程式：[時間積分] ΔQ = [RHS]
  - ΔQ：更新（未知）ベクトル
  - [時間積分]：
    - 疎行列（陰解法）
    - 単位行列（陽解法）
  - [RHS]：対流項＋粘性項

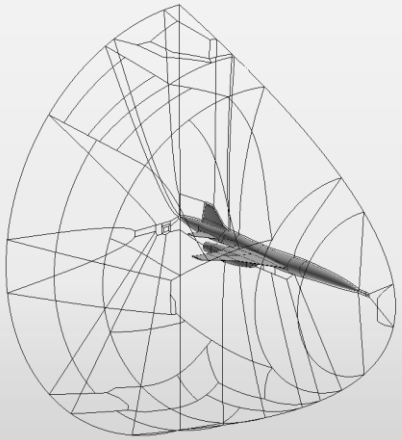


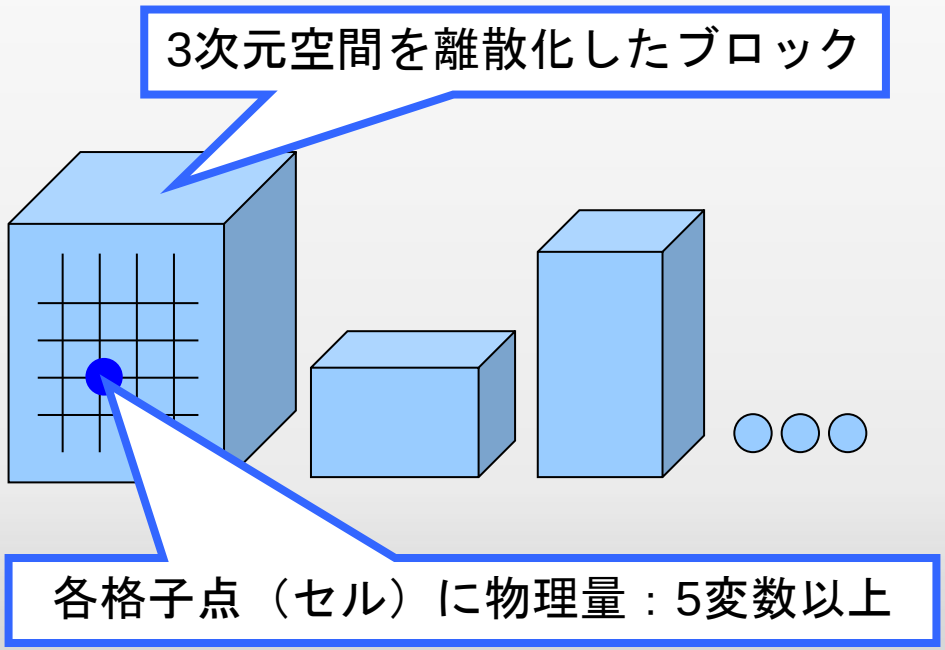


UPACS-Lite

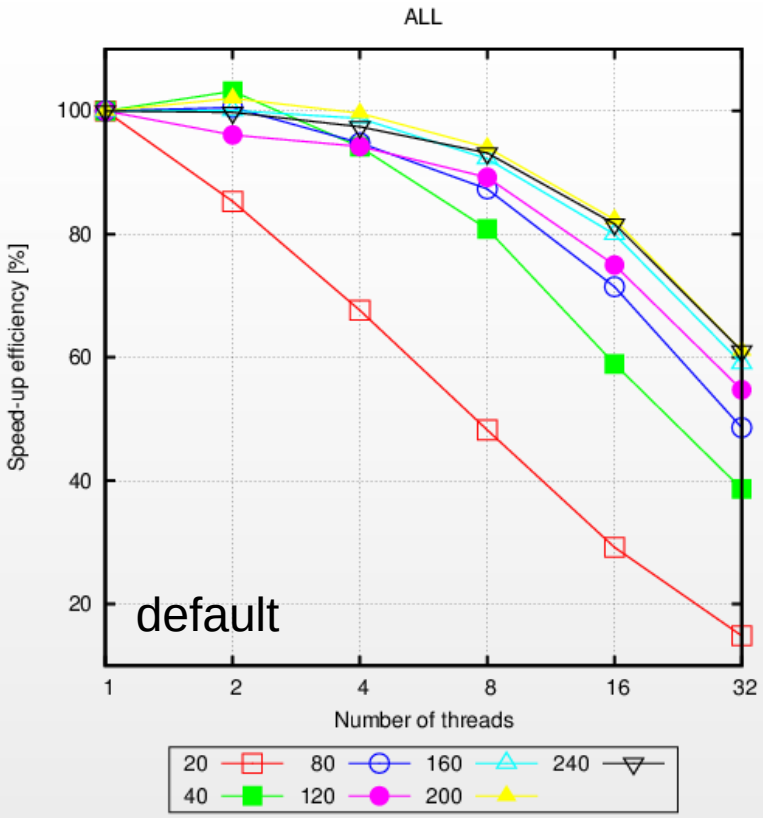
- マルチブロック構造格子
  - ブロックの大きさ、アスペクト比はまちまち
    ⇒ループ長がまちまち
  - 階層構造をそのまま
    データ構造に反映

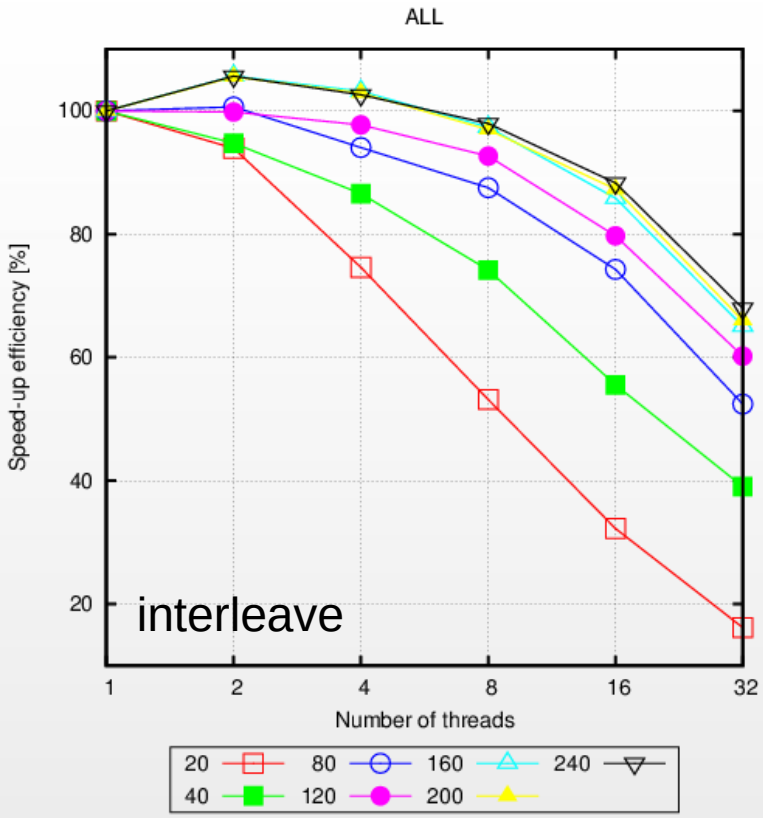






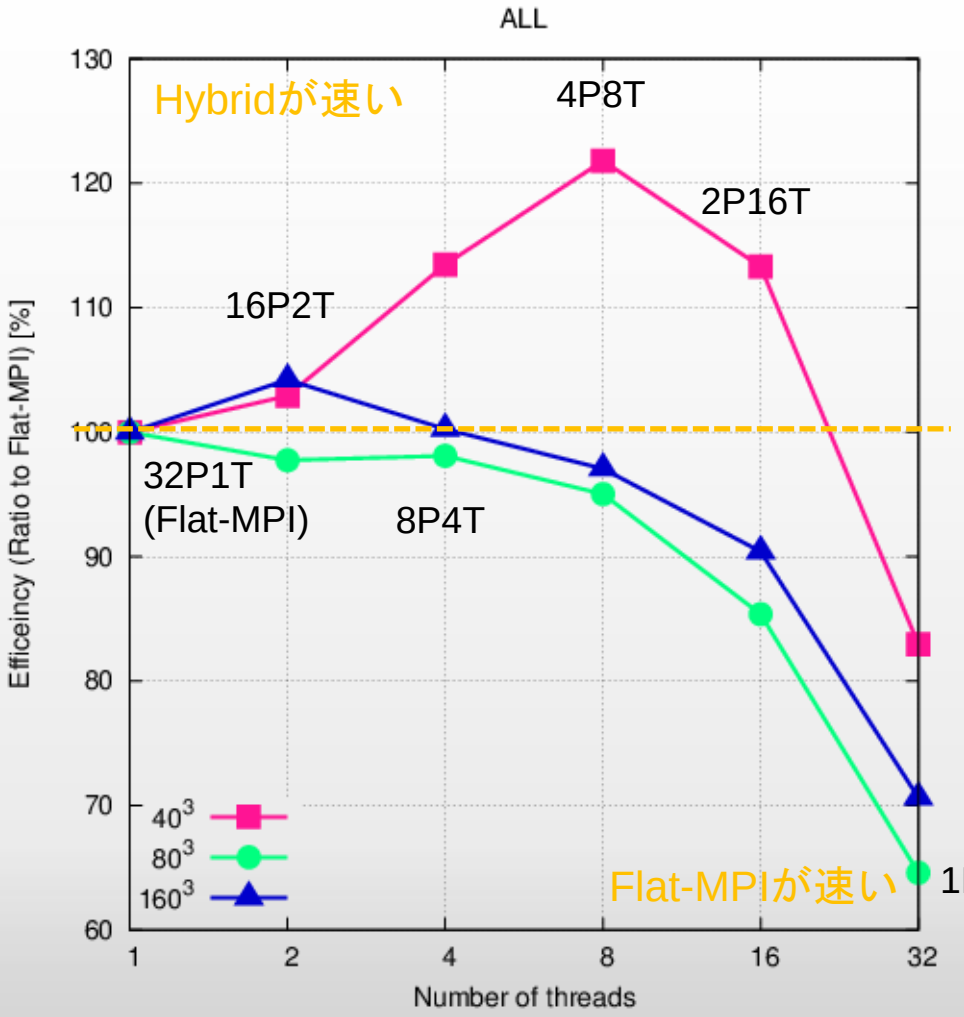
Thread Scalability





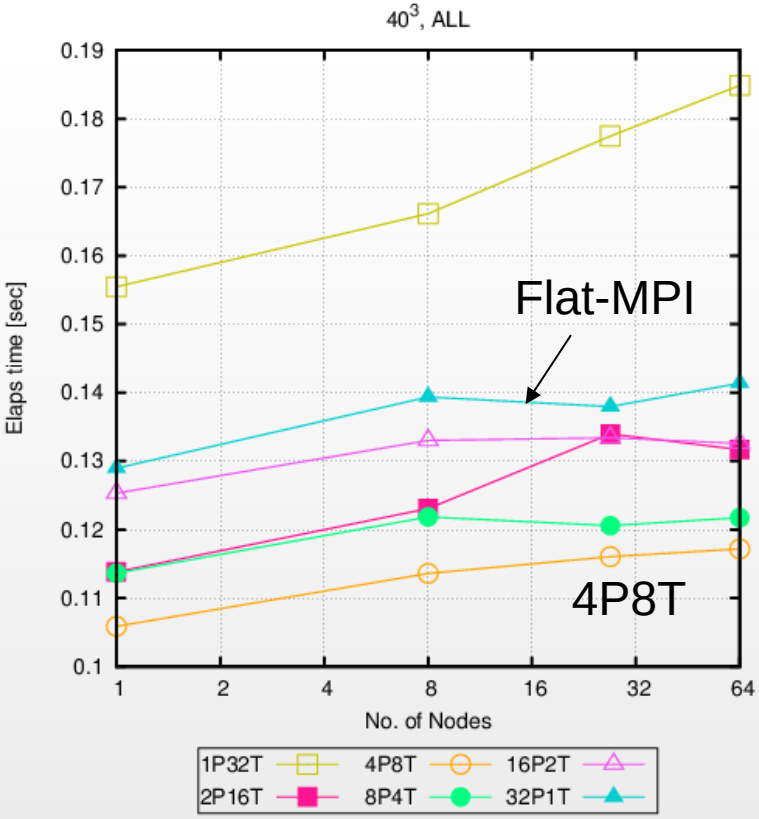
- 8スレッドまでは良い性能（>80%）
- Interleaveを使うことでスケーラビリティは若干UP
  - 1スレッドの性能が低下するため

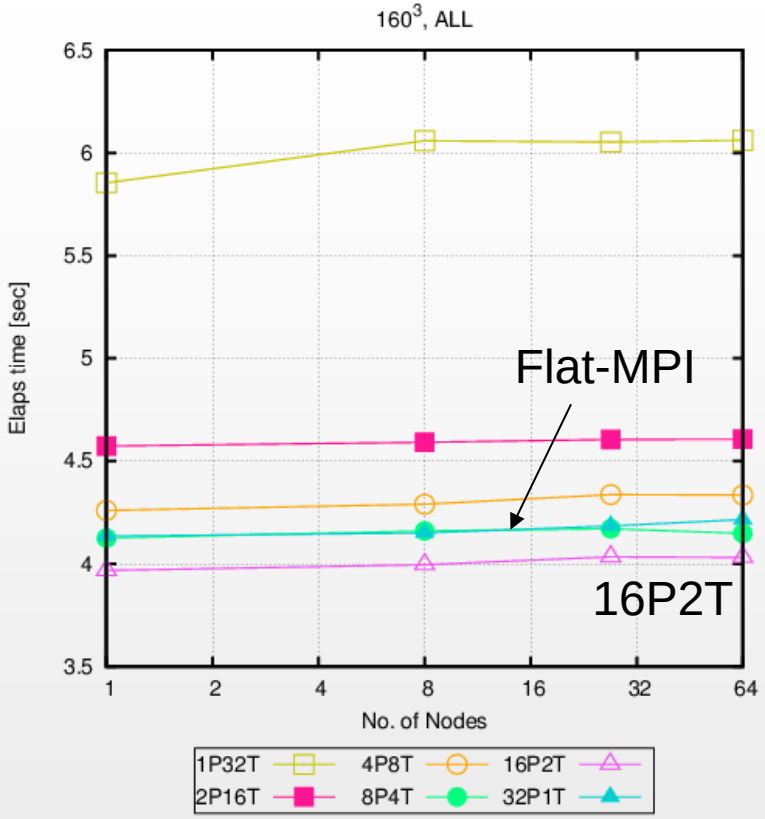
Hybrid vs Flat-MPI@1ノード



- 同一規模@ノード
  - 40^3,80^3,160^3
- Hybridが高速の場合もある
  - ブロックが小(40^3)
  - 160^3では16P2T, 8P4T
- 80^3はFlat-MPIが高速

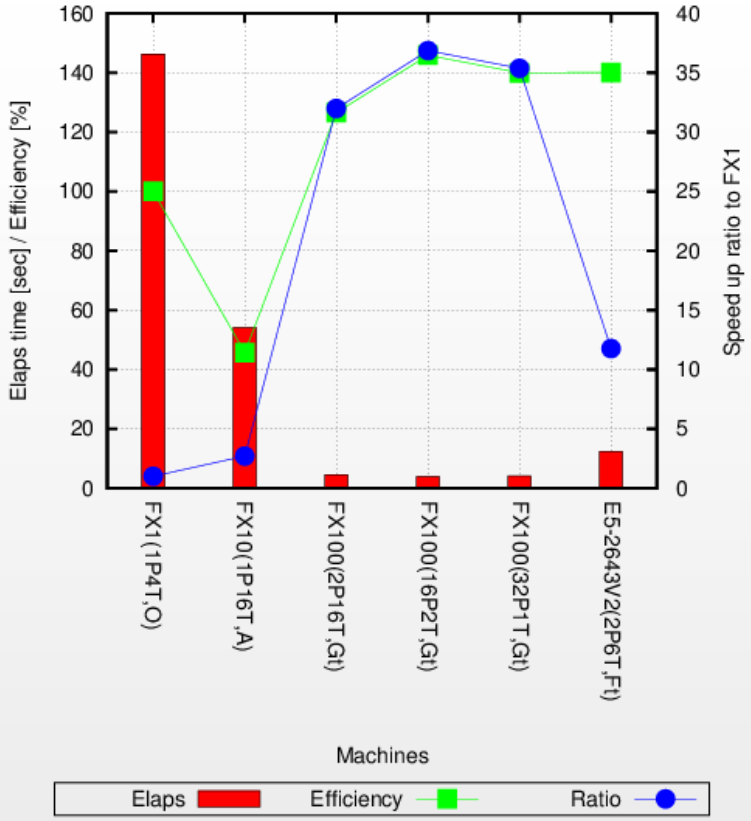
Hybrid vs Flat-MPI@多ノード

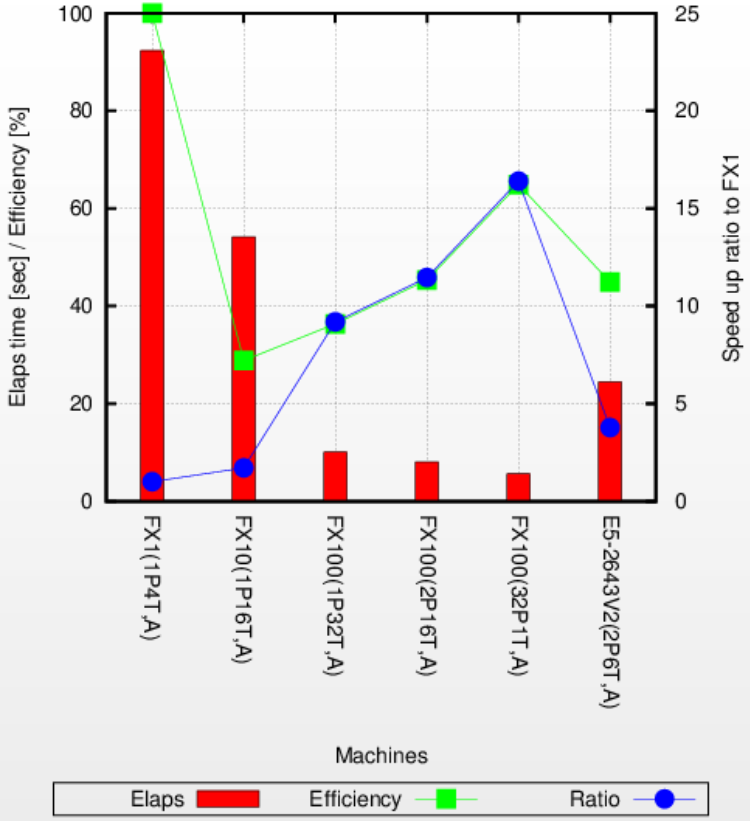




- 64ノード（weak scaling）までは傾向は大きく変わらず。

FX1, FX10, FX100 and Intel





- FX100(Gt, 2P16T)はFX1(O, 1P4T)の約32倍(S/Wチューニングを含む)
  - 最速(16P2T)は約37倍
- 同一S/W(A)では約11倍(2P16T)、約16倍(32P1T)

Intel:Xeon E5-2643V2,3.5GHz

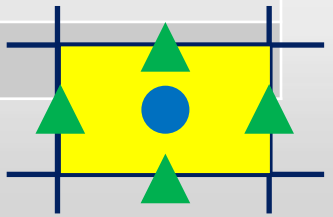
高速化チューニング

UPACS-Lite

高速化チューニングの概要

Version	Tuning
O	オリジナル
A	flux配列の構造体の変更＋SIMD化促進@FX10
B	j,kループの手動融合（スレッド並列数の確保）
C	MFGSの書き下し+OCL（依存関係無視）
D	flux配列のインデックス変更：(i,j,k,:) → (:,i,j,k)
E	flux配列のインデックスの変更（CとDの比較で高速版を選択） 対流項：(i,j,k,:)、粘性項：(:,i,j,k)
F	手動アンローリング 初期化のベクトル記述：dq=0 → OpenMPで並列化 保存量ループの展開（do n=1,nPhysを削除）
G	諸々（後で紹介）
?t	?のcell配列インデックスの変更：(i,j,k,:) → (:,i,j,k)

主な配列：cell配列、flux配列



A（flux配列の構造体）

O	A

type cellFaceType
  real(8) :: area,nt
  real(8), dimension(3) :: nv
  real(8), dimension(5) :: q_l,q_r
  real(8), dimension(5) :: flux
end type
type visCellFaceType
  real(8) :: area, mu
  real(8), dimension(3) :: nv, dTdx, u
  real(8), dimension(5) :: flux
  real(8), dimension(3,3) :: dudx
end type
	type cellFaceType
	  real(8), pointer, dimension(:,:,:) :: area,nt
	  real(8),pointer, dimension(:,:,:,:) :: nv
	  real(8), pointer, dimension(:,:,:,:) :: q_l,q_r
	  real(8), pointer, dimension(:,:,:,:) :: flux
	end type
	type visCellFaceType
	  real(8), pointer, dimension(:,:,:) :: area, mu
	  real(8), pointer, dimension(:,:,:,:) :: nv, dTdx, u
	  real(8), pointer, dimension(:,:,:,:) :: flux
	  real(8), pointer, dimension(:,:,:,:,:) :: dudx
	end type

type(*), pointer, dimension(:,:,:) :: cface
 	
do n ; do k ; do j ; do i
  cface(i,j,k)%flux(n) …
enddo
	type(*) :: cface
	
	do n ; do k ; do j ; do i
	  cface%flux(i,j,k,n) …
	enddo

A（SIMD化@FX10）

SIMD化を促進するための修正（粘性項）

- FX10向けチューニングで実施
- 一時配列のスカラー化：a(3,3)→a_11,a_12,…
  - viscous_cfacev:0%→99%@FX10
- 組み込み関数の手動展開
  - matmul, dot_productなど
  - viscous_flux:10.64%→99.11%@FX10

A（SIMD化@FX10）

- SIMD化は最内ループに適用される。
  - ループボディに配列のベクトル記述があると、そこがSIMD化され、残りはSIMD化されない。

SIMD ×	SIMD ○

allocate(a(imax,5),b(imax,5),c(imax,5))
 	
do i=1,imax
  u = a(i,2)/a(i,1)
  v = a(i,3)/a(i,1)
  w = a(i,4)/a(i,1)
  a(i,:) = b(i,:) + c(i,:) ←ここだけSIMD化
Enddo
	allocate(a(imax,5),b(imax,5),c(imax,5))
	
	do i=1,imax
	  u = a(i,2)/a(i,1)
	  v = a(i,3)/a(i,1)
	  w = a(i,4)/a(i,1)
	  a(i,1) = b(i,1) + c(i,1)
	  a(i,2) = b(i,2) + c(i,2)
	  a(i,3) = b(i,3) + c(i,3)
	  a(i,4) = b(i,4) + c(i,4)
	  a(i,5) = b(i,5) + c(i,5)
	enddo

B（OpenMPループの一重化）

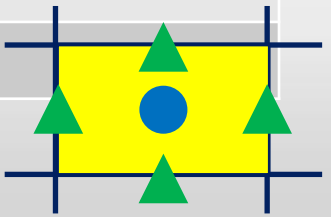
A	B	G
do n=1,nPhys !$omp parallel do private(i,j,k) **do k=1,kmax** do j=1,jmax do i=1,imax … enddo enddo enddo !$omp end parallel do enddo	jkmax = jmax*kmax do n=1,nPhys !$omp parallel do private(jk,i,j,k) **do jk=1,jkmax** **k = (jk-1)/jmax+1** **j = jk-jmax*(k-1)** do i=1,imax … enddo enddo !$omp end parallel do enddo	do n=1,nPhys !$omp parallel do private(i,j,k) & !$omp **collapse(2)** do k=1,kmax do j=1,jmax do i=1,imax … enddo; enddo; enddo !$omp end parallel do enddo

- 外側ループの一重化でスレッド並列の粒度を確保
- B（手動一重化）とG（collapseで指定）でほぼ性能差はなし
- ほとんど効果はなかった
  - スレッド数 > ループ長の場合は効果があったが

高速化チューニングの概要

Version	Tuning
O	オリジナル
A	flux配列の構造体の変更＋SIMD化促進@FX10
B	j,kループの手動融合（スレッド並列数の確保）
C	MFGSの書き下し+OCL（依存関係無視）
D	flux配列のインデックス変更：(i,j,k,:) → (:,i,j,k)
E	flux配列のインデックスの変更（CとDの比較で高速版を選択） 対流項：(i,j,k,:)、粘性項：(:,i,j,k)
F	手動アンローリング 初期化のベクトル記述：dq=0 → OpenMPで並列化 保存量ループの展開（do n=1,nPhysを削除）
G	諸々（後で紹介）
?t	?のcell配列インデックスの変更：(i,j,k,:) → (:,i,j,k)

主な配列：cell配列、flux配列

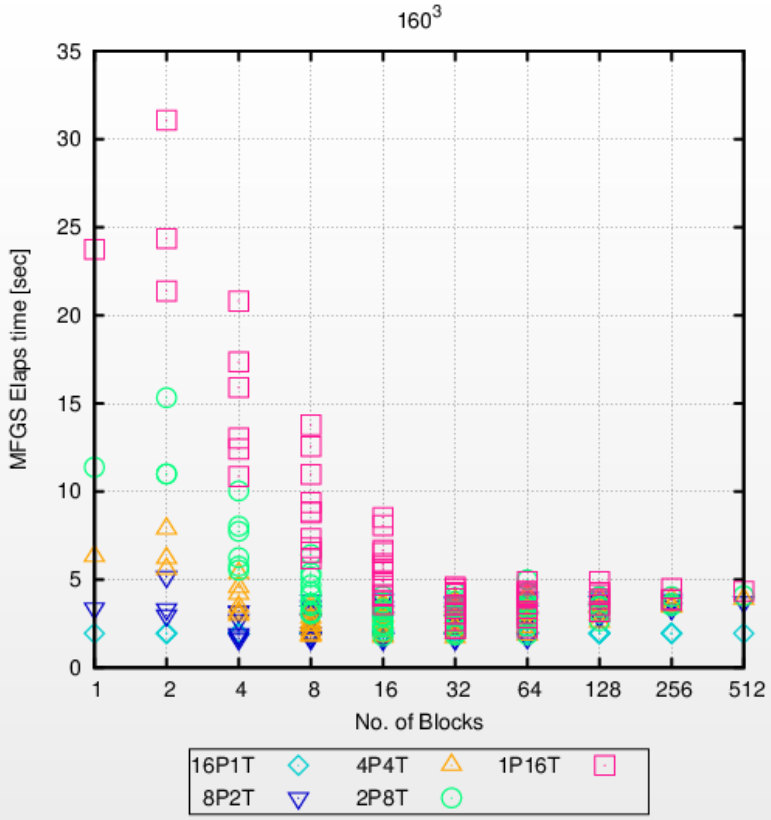


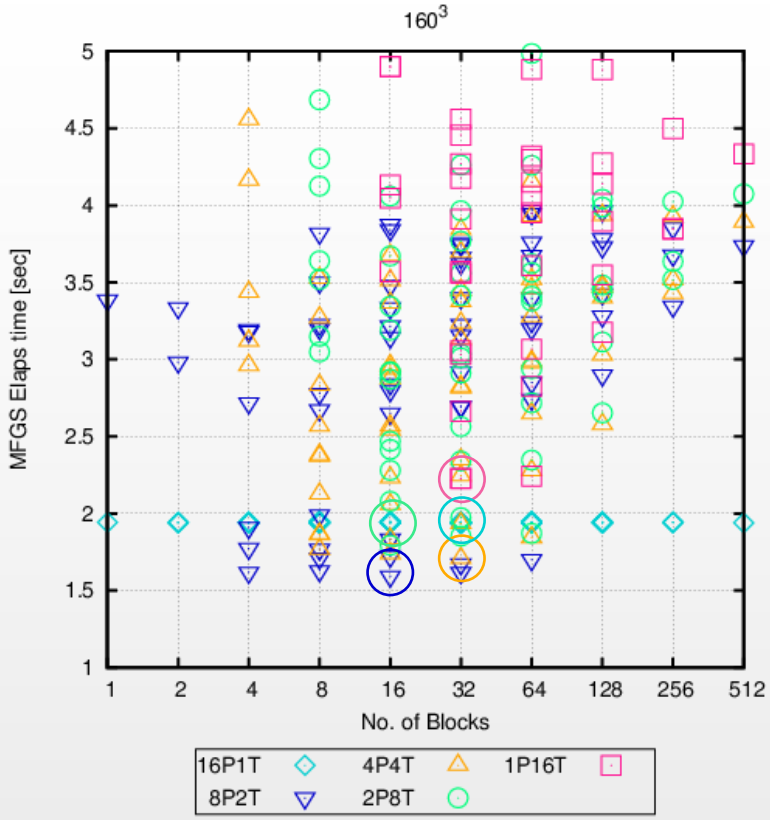
F（保存量ループの展開）

E	F

jkmax = jmax*kmax
do n=1,nPhys
!$omp parallel do private(jk,i,j,k)
  do jk=1,jkmax
  k = (jk-1)/jmax+1
  j = jk-jmax*(k-1)
  do i=1,imax
  q(i,j,k,n) = …
  …
  enddo
  enddo
!$omp end parallel do
enddo
	jkmax = jmax*kmax
	
	!$omp parallel do private(jk,i,j,k)
	  do jk=1,jkmax
	  k = (jk-1)/jmax+1
	  j = jk-jmax*(k-1)
	  do i=1,imax
	  q(i,j,k,1) = …
	  q(i,j,k,2) = …
	  …
	  enddo
	  enddo
	!$omp end parallel do

G





- MFGS（時間積分）で使われているBlock Red-Blackのブロック分割の最適化

G

- 2重ループの一重化をやめてOpenMPのcollapse(2)を利用
- データ通信の前後処理部のOpenMP化
- nPhysループの位置の変更

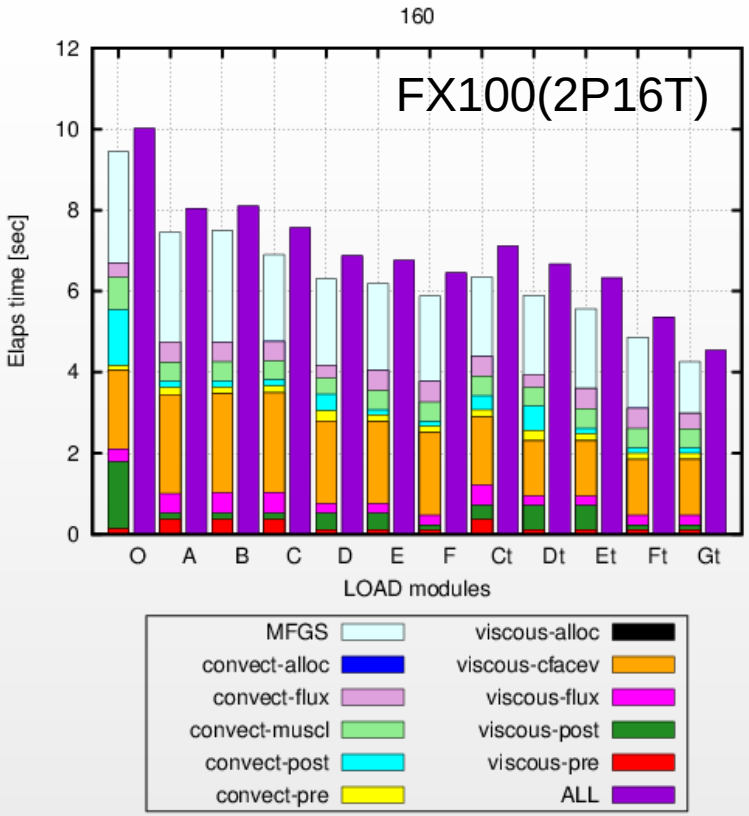
F	G

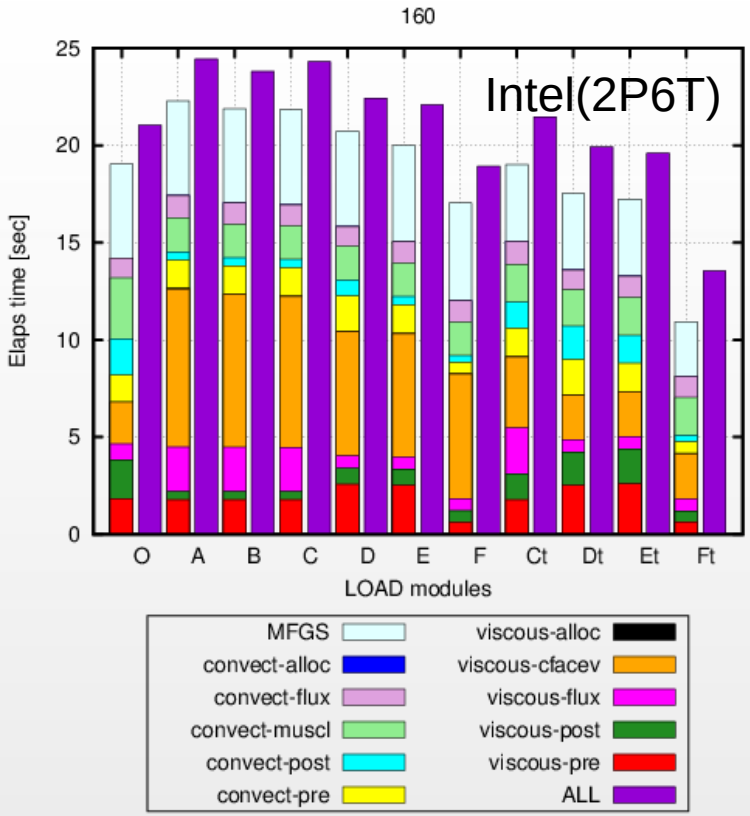
!$omp parallel do
do k=1,kmax
do j=1,jmax
do i=1,imax
  dq(**1**,i,j,k) = ..
  dq(**2**,i,j,k) = ..
…
enddo; enddo; edndo
!$end parallel do
	!$omp parallel do collapse(2)
	do k=1,kmax
	do j=1,jmax
	do n=1,nPhys
	do i=1,imax
	  dq(**n**,i,j,k) = ..
	enddo; enddo; enddo; enddo
	!$omp end parallel do

G（nPhyループの位置）

convect-post	cell配列	Elaps
do n=1,nPhys !$omp parallel do do k=1,kmax ; do j=1,jmax do i=1,imax dQ(:,:,:,:) = dQ(:,:,:,:)+V(i,j,k)*(cface%f(i,j,k,**n**)-cface%f(i,jm,km,**n**)) enddo; enddo ; enddo !$omp end parallel do enddo	dQ(i,j,k,**n**)	0.320 (E)
	dQ(**n**,i,j,k)	0.703 (Et)
!$omp parallel do do k=1,kmax ; do j=1,jmax do i=1,imax dQ(:,:,:,:) = dQ(:,:,:,:)+V(i,j,k)*(cface%f(i,j,k,**1**)-cface%f(im,jm,km,**1**) … enddo ; enddo ; enddo !$omp end parallel do	dQ(i,j,k,**1**)	0.247 (F)
	dQ(**1**,i,j,k)	**0.246** **(Ft)**
!$omp parallel do do k=1,kmax ; do j=1,jmax **do n=1,nPhys** do i=1,imax dQ(:,:,:,:) = dQ(:,:,:,:)+V(i,j,k)*(cface%f(i,j,k,**n**)-cface%f(i,jm,km,**n**)) enddo ; enddo ; enddo ; enddo !$omp end parallel do	dQ(i,j,k,**n**)	**0.250** **(G)**
	dQ(**n**,i,j,k)	**0.268** **(Gt)**

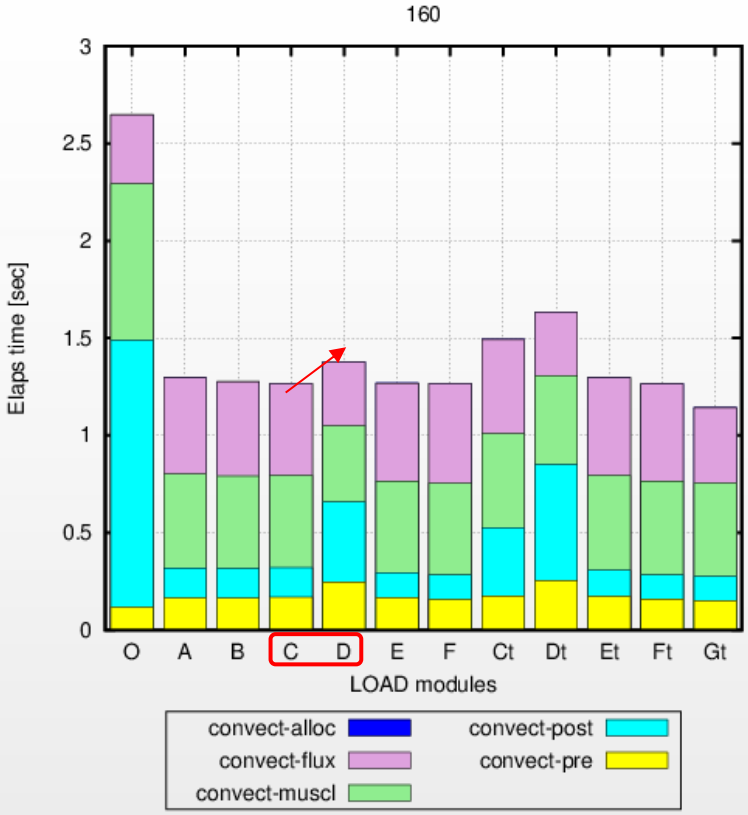
ALL（160^3）

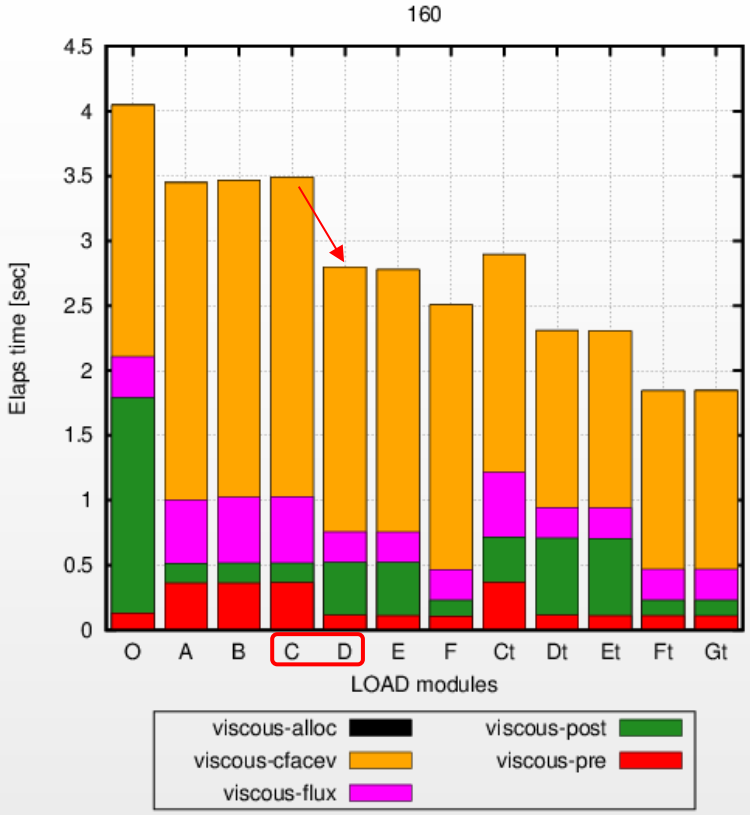




- FX100とIntelでO→A（flux配列の構造体変更）の傾向が逆
- A以後のチューニング傾向は同じ

対流項と粘性項（160^3）



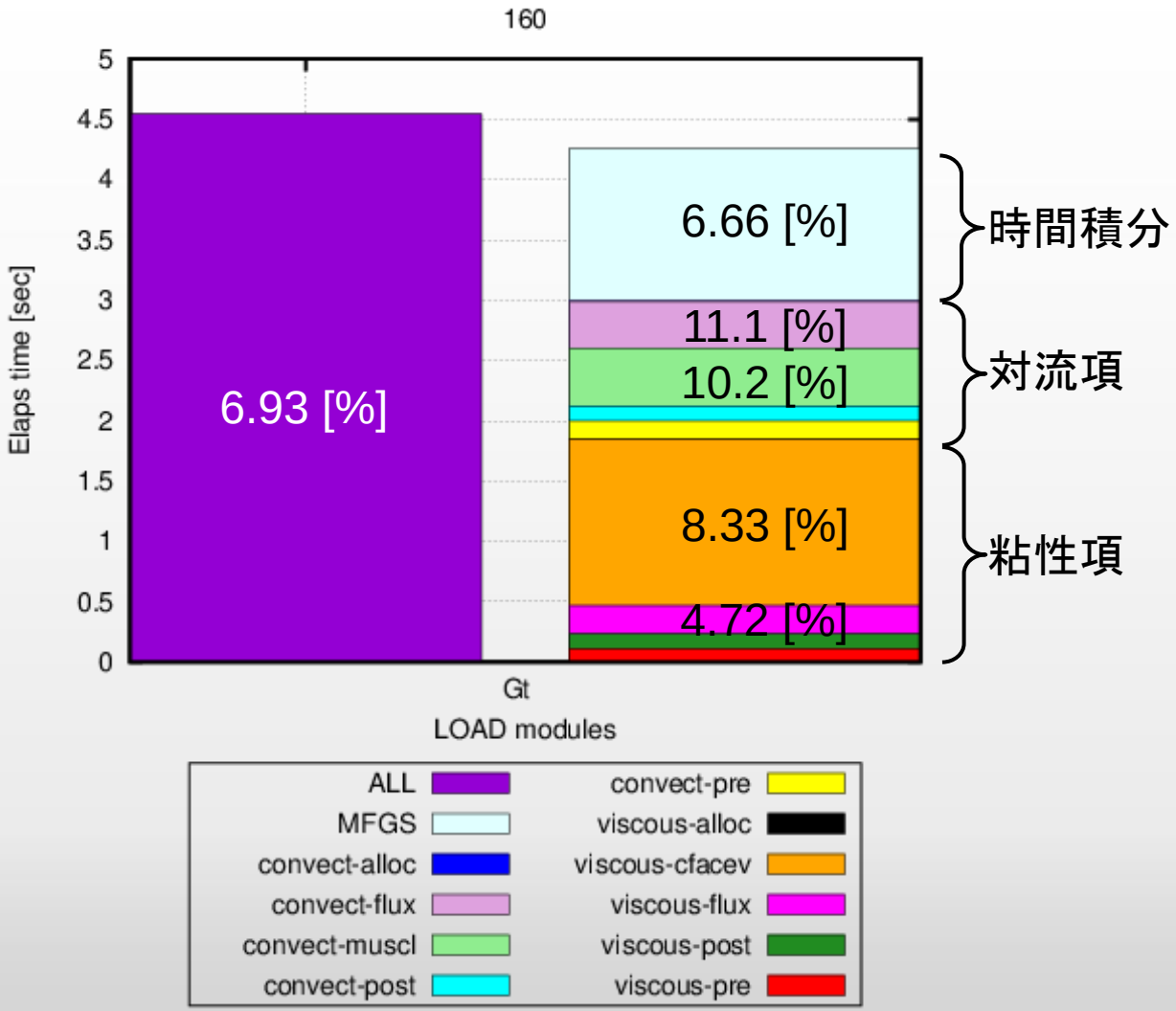


- E：対流項←C(i,j,k,:)、粘性項←D(:,i,j,k)
  - 粘性項は対流項よりもflux配列の使いまわしが多いから？

まとめ

- JAXAに新しく導入されたJSS2のシステム概要を紹介すると同時に、中核システムであるSORA-MA（富士通FX100）の性能評価結果を紹介した。
- ステンシル系プログラムを対象とした高速化チューニングを通じて、利用のための知見が得られた。
  - 配列インデックスによる影響
  - SIMD化を促進するために必要な事
  - SPARC系とIntel系の違い
  - …
- 更なる高速化、利用ノウハウの蓄積を目指す。

実行効率（Gt, 160^3, 2P16T）



項目		実行効率[%]
ALL		6.93
convect (対流項)	muscl	10.2
	flux	11.1
	post	2.75
viscous (粘性項)	cfacev	8.33
	flux	4.72
	post	2.78
MFGS(時間積分)		6.66