第３回計算科学技術特論B(2026)
スレッド並列＋コア単体性能最適化概要

株式会社 waveZ
南一生

2026年4月23日（木）13:00－14:30

主催：高度情報科学技術研究機構(RIST)
次世代HPC・AI研究開発支援センター(HAIRDESC)
共催：東京大学物性研究所
後援：理化学研究所計算科学研究センター、
計算物質科学人材育成コンソーシアム(PCoMS)

計算科学技術 特論B

第3回
スレッド並列・スレッド性能最適化概要

2026年4月

株式会社 waveZ
テクノロジーディレクター
南 一生

minami@wave-z.com

講義全体の概要

waveZ

- スーパーコンピュータとアプリケーションの性能
- アプリケーションの性能最適化１（高並列性能最適化）
- スレッド並列・スレッド性能最適化概要
- スレッド並列・スレッド性能最適化詳細（1）
- スレッド並列・スレッド性能最適化詳細（2）

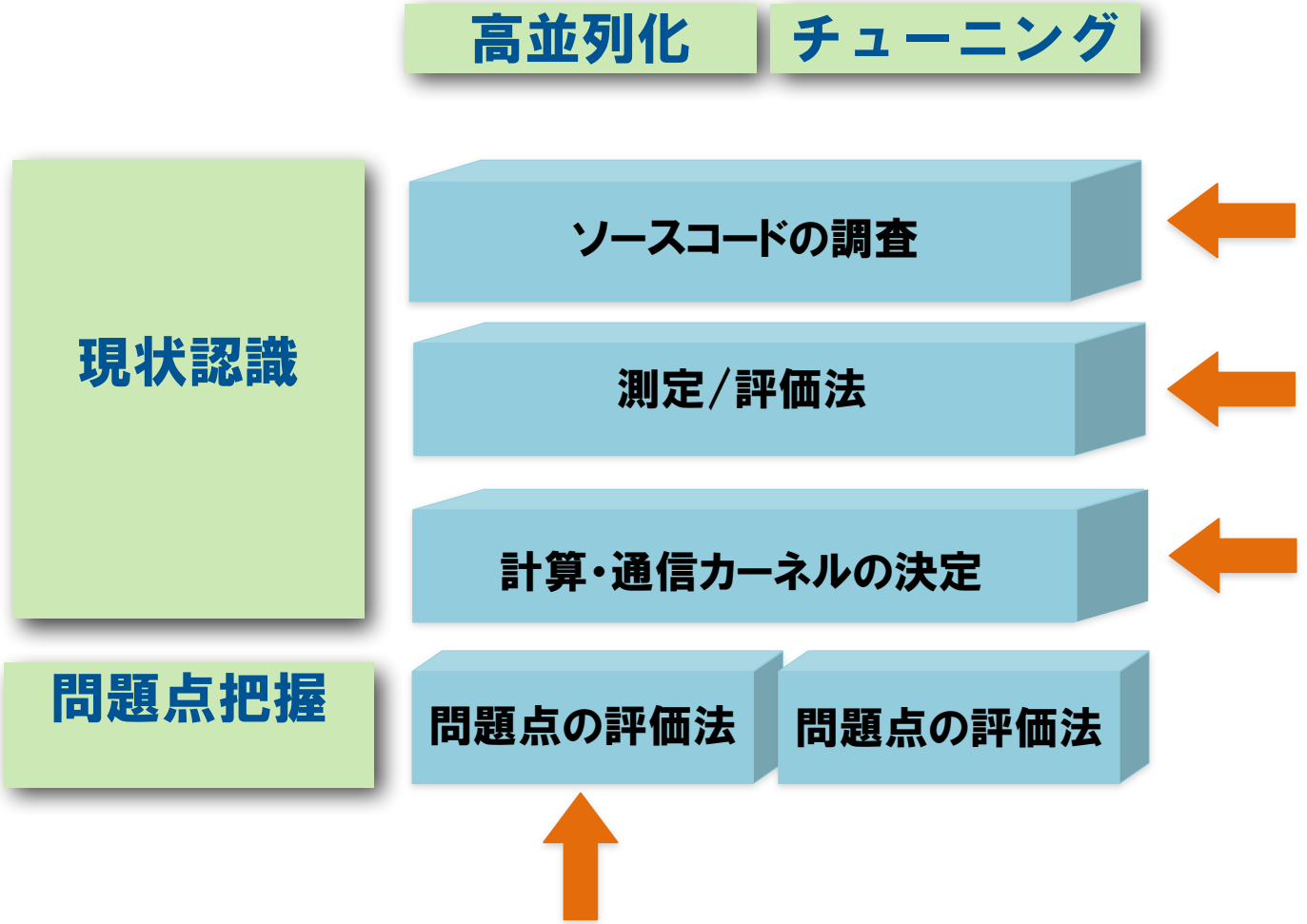
今回の講義内容

waveZ

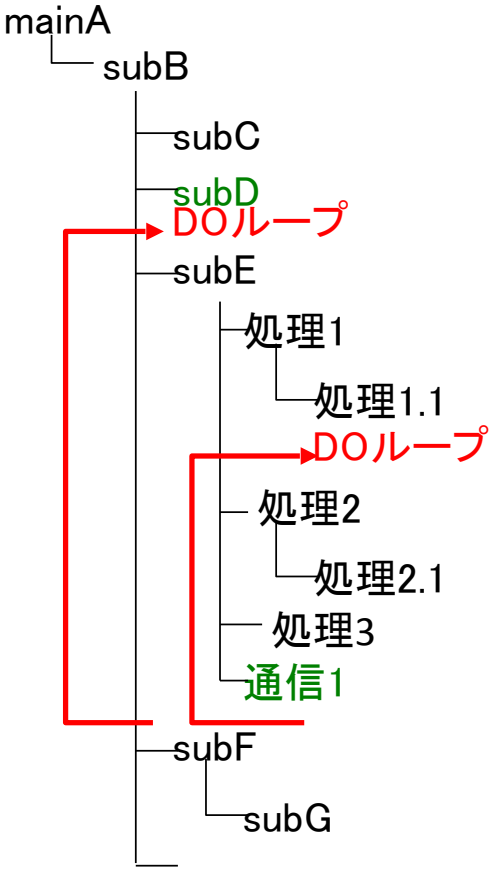
- 高並列化手法の手順（簡単な実例）
- 高並列化手法の手順（RSDFTの事例）
- スレッド並列・スレッド性能

高並列化の手順
（CPUノードを使った
評価の簡単な実例）

簡単な実例



ソースコードの調査
測定/評価法



	実行時間・スケーラビリティ	物理的処理内容	演算・通信特性	演算・通信見積り	カーネル
ブロック1（計算）			部分並列	Nに比例	
ブロック2（計算）			完全並列	N**3に比例	○
（通信）			隣接通信	隣接面に比例	○
ブロック3					

計算・通信カーネルの決定

CCA/EBT ← 検索

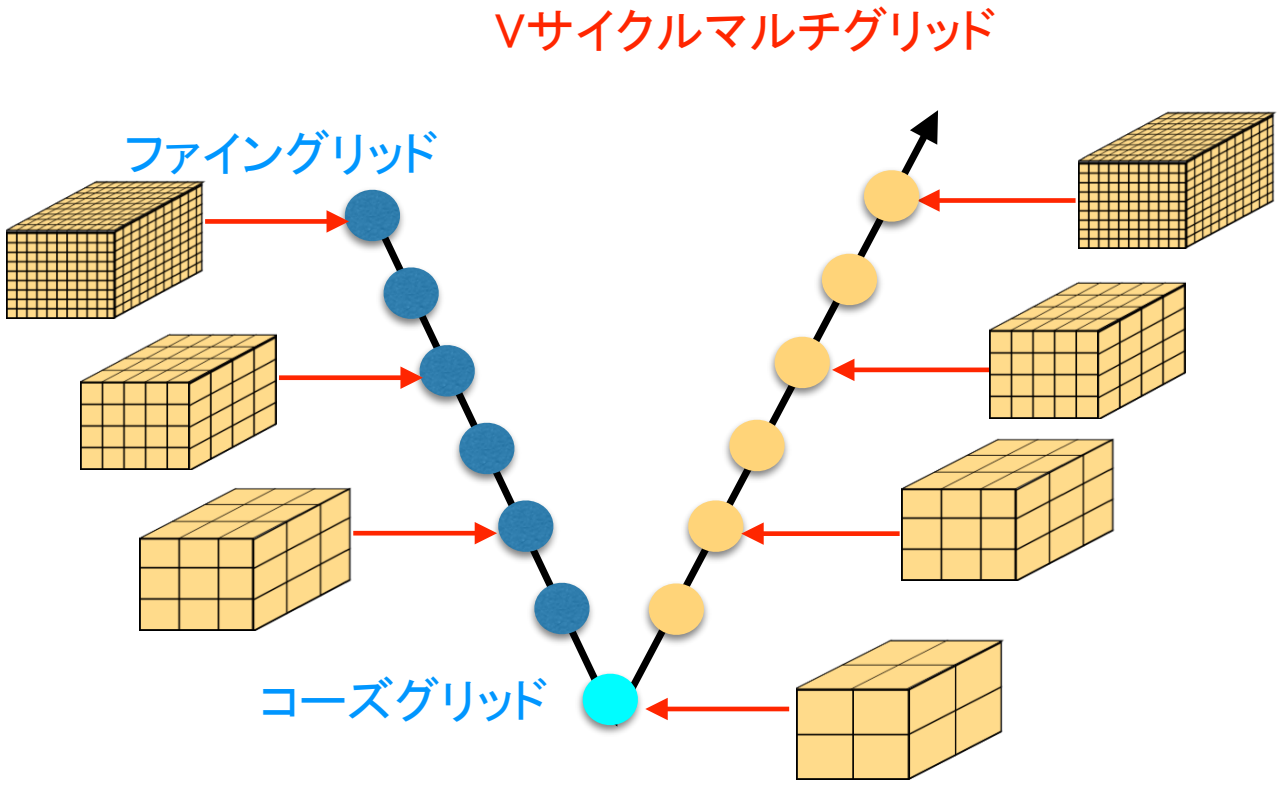
(Code Comprehension Assistance for Evidence-Based performancTuning)

waveZ

- プログラム構造を俯瞰的にみて理解を助けるツール.
- Fortran及びCが対象.
- プロシージャ呼び出し,ループ,分岐に代表されるプログラムの構造の可視化を行う.
- プログラム構造解析支援ツール.
- 多数のFortranプログラムの解析実績あり.

NPB MGを使用した解析例

NPB MG (マルチグリッド法により連立一次方程式を解くベンチマークテストプログラム)

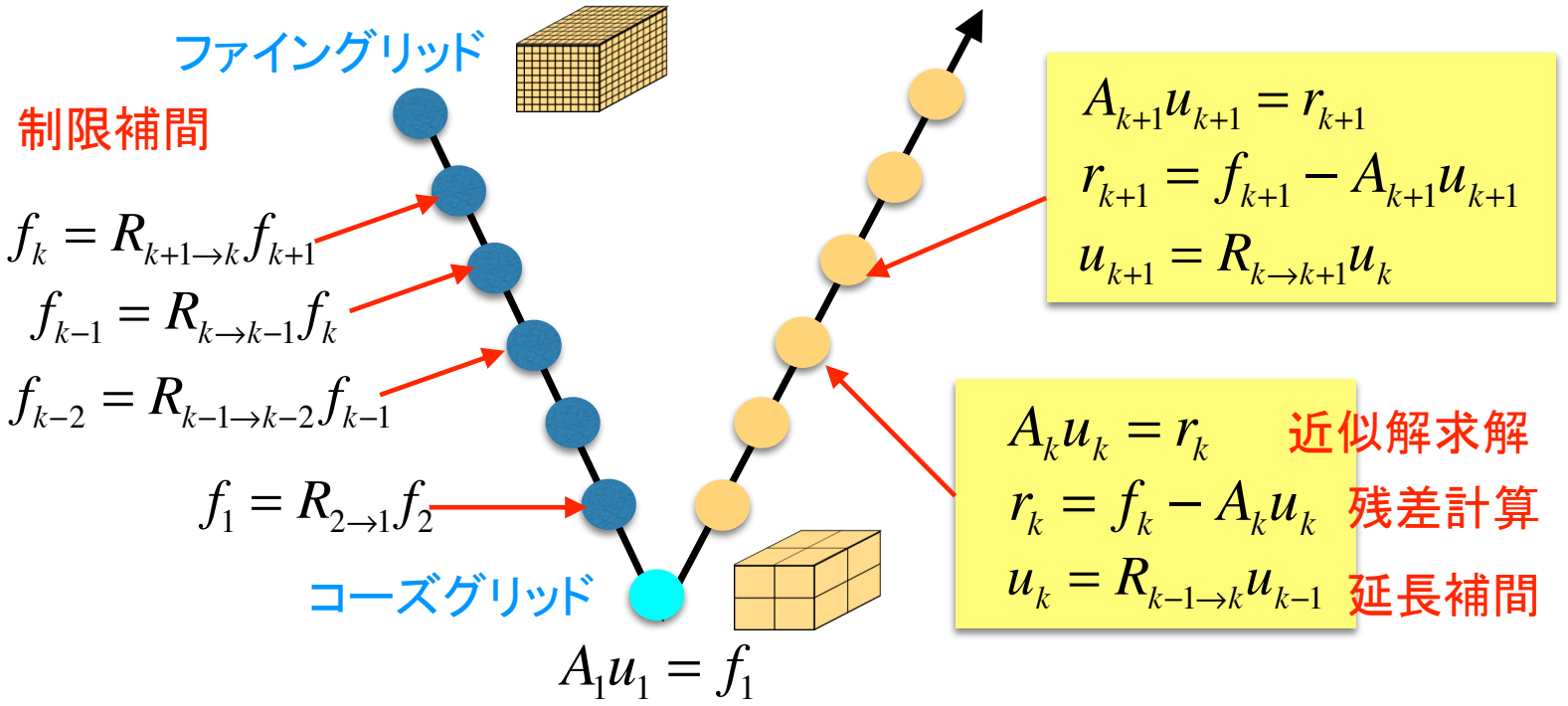


NPB MGを使用した解析例

NPB MG (マルチグリッド法により連立一次方程式を解くベンチマークテストプログラム)

MGのアルゴリズム
$Au = f$ を解く

Vサイクルマルチグリッド



アプリケーションの構造の調査（概要）

(1) 計測のスタートとストップに注目

(2) 繰り返しループに注目

(3) mg3pサブルーチンに注目

(4) マイナス方向のループでダウンサイクルと予測

(5) 制限補間ルーチンと予測

(6) ダウンサイクル後の近似解求解と予測

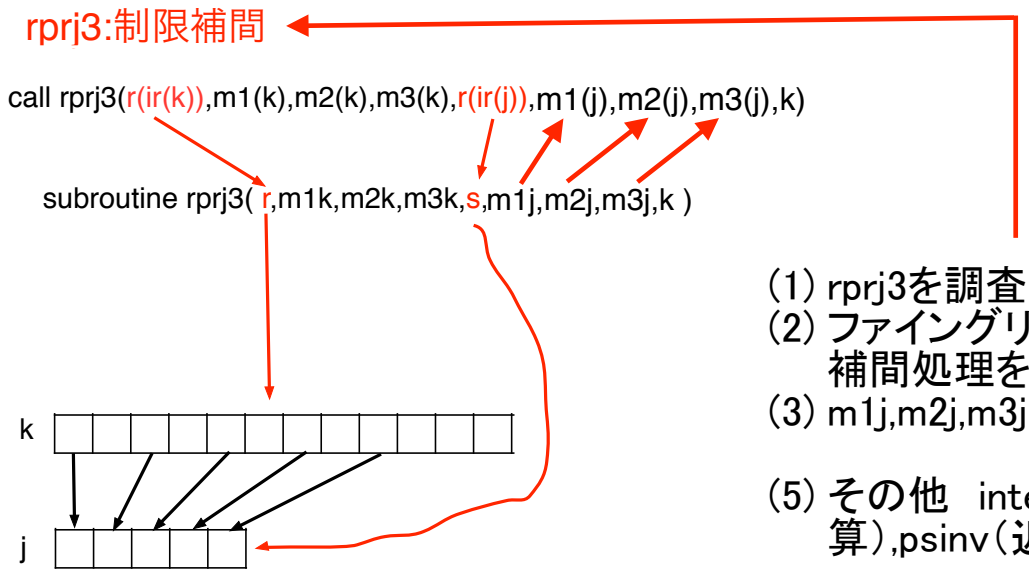
(7) プラス方向のループでアップサイクルと予測

(8) 延長補間と予測

(9) 残差計算と予測

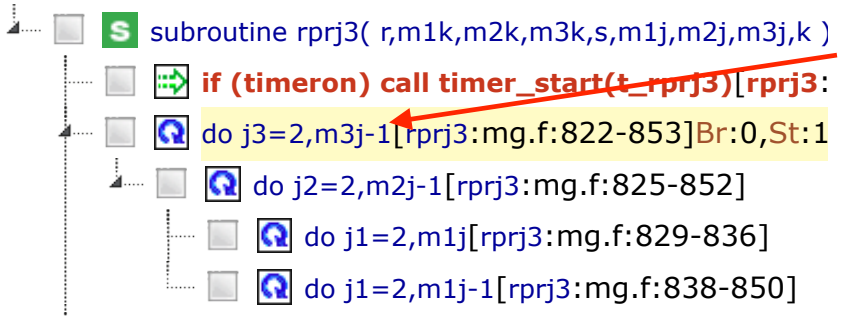
(10)アップサイクル内の近似解求解

アプリケーションの構造の調査（確認）



(1) rprj3を調査の結果制限補間処理と判明
(2) ファイングリッドからコーズグリッドへの補間処理を確認した
(3) m1j,m2j,m3jはm1,m2,m3に結合

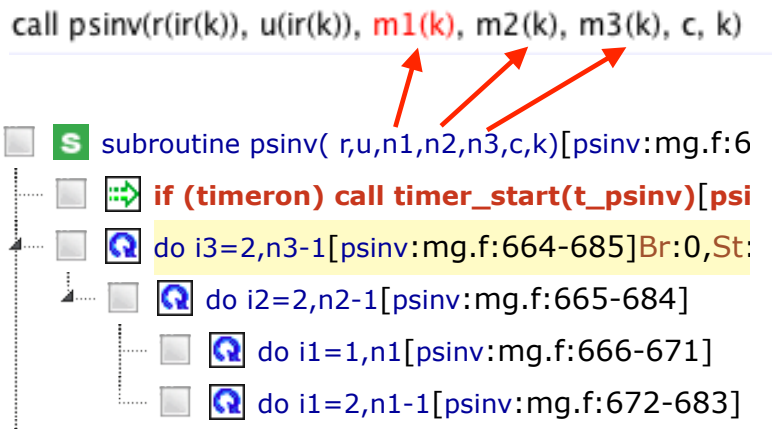
(5) その他　interp(延長補間),resid(残差計算),psinv(近似解求解)と確認



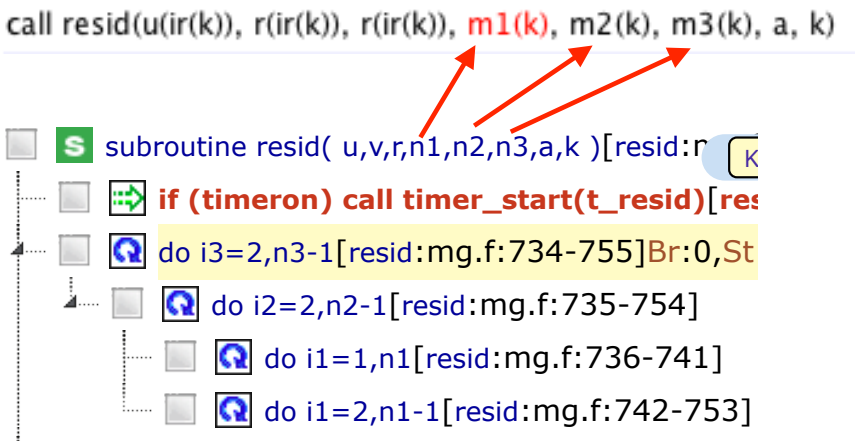
(6) rprj3のループ構造調査
(7) m3j,m2j,m1jの３重ループ構造

アプリケーションの構造の調査（確認）

(1) psinvのループ構造調査
(2) n3,n2,n1の３重ループ構造
(3) n1,n2,n3はm1,m2,m3に結合



(1) residのループ構造調査
(2) n3,n2,n1の３重ループ構造
(3) n1,n2,n3はm1,m2,m3に結合

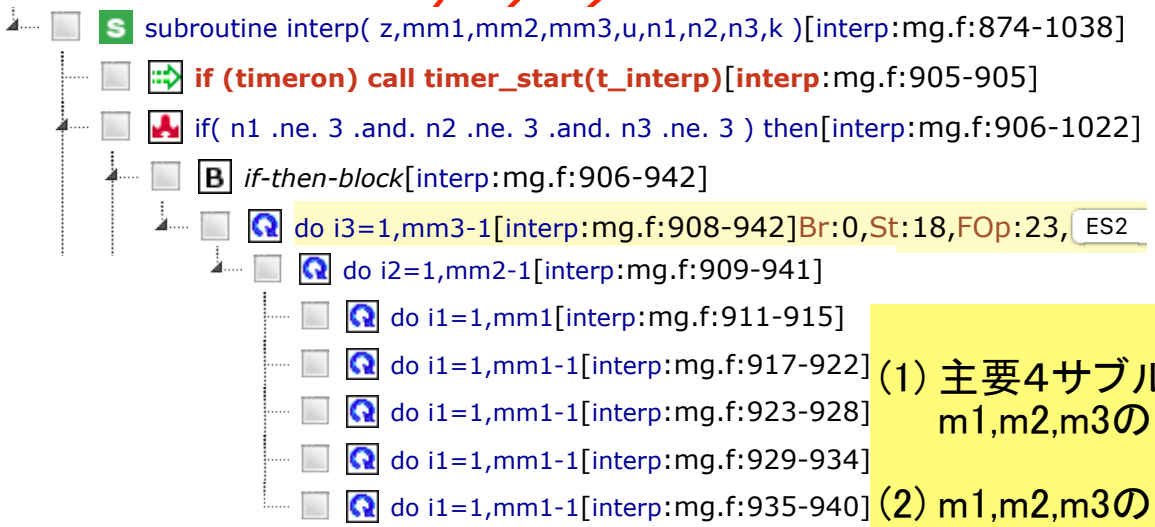


アプリケーションの構造の調査（確認）

waveZ

(1) interpのループ構造調査
(2) mm3,mm2,mm1の３重ループ構造
(3) mm1,mm2,mm3はm1,m2,m3に結合

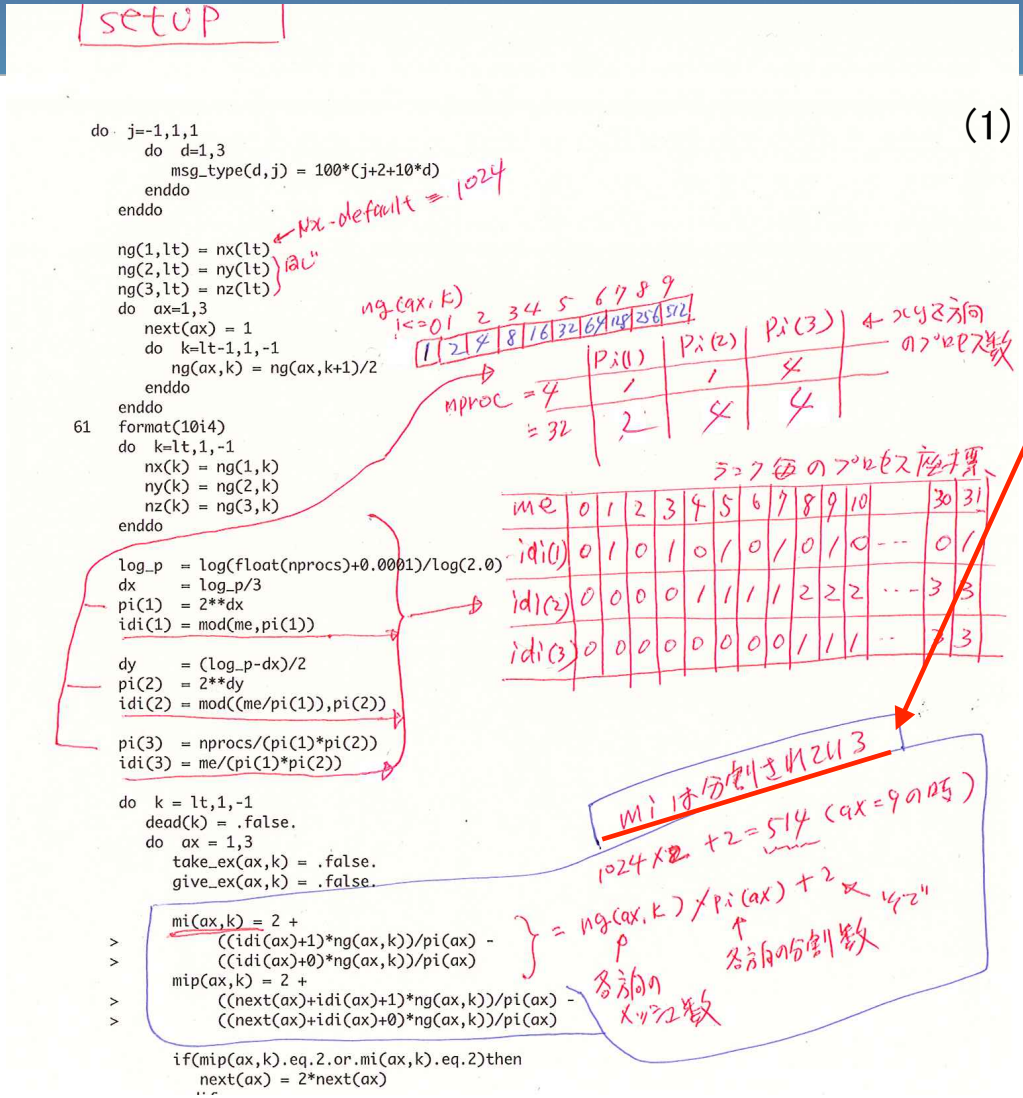
```
call interp(u(ir(j)), m1(j), m2(j), m3(j), u(ir(k)), m1(k), m2(k), m3(k), k)
```



(1) 主要4サブルーチンとも処理量は
m1,m2,m3の大きさに依存している.

(2) m1,m2,m3の分割はどうなっているか？

データ分割の調査



(1) setup内の配列：miを調べた結果，プロセス分割されている事が判明

スケーラビリティの調査

(1) 4プロセスの測定．ウィークスケーリング測定

```
4プロセス
Basic profile
Performance monitor : Performance
    ********************************************************************************
        Process 0
    ********************************************************************************

          Elapsed(s)      User(s)    System(s)         Call
    ------------------------------------------------------------
             17.3529     133.6500       0.2200            1    all 0
              8.5085      68.0400       0.0000          191    resid 1
              1.5350      12.3600       0.0100          168    interp 1
              4.2610      34.0600       0.0000          189    psinv 1
              1.4298      11.4000       0.0000          168    rprj3 1
```

主要処理のelaps時間
全体で17sec

```
Performance monitor event
    ------------------------------------------------------------------------------------------
    ********************************************************************************
        Application - performance monitors
    ********************************************************************************

        Elapsed(s)       MFLOPS MFLOPS/PEAK(%)         MIPS    MIPS/PEAK(%)
    ------------------------------------------------------------------------------
        16.6556     10361.0515         2.0236   18605.9895          7.2680    Application
    ------------------------------------------------------------------------------
        16.6556      2590.2629         2.0236    4651.6009          7.2681    Process       0
        16.6052      2598.1206         2.0298    4668.2023          7.2941    Process       2
        16.5407      2608.2634         2.0377    4683.4974          7.3180    Process       1
        16.5213      2611.3249         2.0401    4686.9329          7.3233    Process       3

                  Mem throughput   Mem throughput
        Elapsed(s)    _chip(GB/S)       /PEAK(%)         SIMD(%)
    ------------------------------------------------------------
        16.6556         32.3398         12.6327         48.7947    Application
    ------------------------------------------------------------
        16.6556          8.0885         12.6382         48.7936    Process       0
        16.6052          8.1070         12.6672         48.7676    Process       2
        16.5407          8.1357         12.7120         48.7981    Process       1
        16.5213          8.1551         12.7423         48.8196    Process       3
```

プロセスのロード
インバランスは
発生していない
ピーク性能比2%

スケーラビリティの調査

(1) 32プロセスの測定．ウィークスケーリング測定

32プロセス
Basic profile
Performance monitor : Performance

```
    ********************************************************************************
        Process 0
    ********************************************************************************

          Elapsed(s)      User(s)    System(s)          Call
    ------------------------------------------------------------
             39.6560     311.7600       0.3100             1    all 0
             20.2033     161.5600       0.0000           512    resid 1
              3.8802      31.3600       0.0000           459    interp 1
             10.3244      82.6200       0.0000           510    psinv 1
              3.4813      28.0800       0.0100           459    rprj3 1

    ------------------------------------------------------------------------------------------
Performance monitor event

 ********************************************************************************************
      Application - performance monitors
  ********************************************************************************************

     Elapsed(s)        MFLOPS MFLOPS/PEAK(%)          MIPS    MIPS/PEAK(%)
  ------------------------------------------------------------------------------
     39.3654      83044.5238         2.0275     149003.2766          7.2756    Applicati
  --
     39.3654       2595.1412         2.0275       4654.6126          7.2728    Process      0
     39.3575       2595.6627         2.0279       4650.9112          7.2670    Process      7
     39.3546       2595.8520         2.0280       4656.6897          7.2761    Process      8
                                          :


     Elapsed(s)  Mem throughput _chip(GB/S)  Mem throughput /PEAK(%)      SIMD(%)
  ------------------------------------------------------------------------------
     39.3654              258.7562                  12.6346              49.3192    Application
  ------------------------------------------------------------------------------
     39.3654                8.0773                  12.6208              49.3303    Process      0
     39.3575                8.0573                  12.5895              49.3647    Process      7
     39.3546                8.0544                  12.5850              49.3071    Process      8
                                          :
```

主要処理のelaps時間
全体で40sec

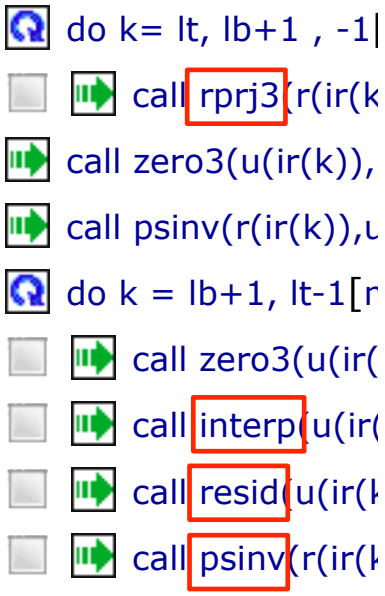
プロセスのロード
インバランスは
発生していない
ピーク性能比2%

スケーラビリティの調査

waveZ

- ウィークスケーリングで4プロセスから32プロセスで全体で17secから40secへ実行時間が増大．約2.5倍の増大．
- 主要4サブルーチンも同様の傾向．
- しかしピーク性能比は、2%で変わらず．
- 1プロセスの演算量が同じなら実行時間が延びる事でピーク性能比も落ちるはず？
- 調査の結果ウィークスケール測定であったが32プロセスでは全体のイタレーション回数が2.5倍に設定されていた事が判明．
- スケーラビリティとしては良好である事が分かった．
- またロードインバランスも良好であることが分かった．
- 並列化の観点では問題なし．

高並列特性評価（結果）



実行時間・スケーラビリティ	物理的処理内容	演算・通信特性	演算・通信見積り	カーネル
良好	制限補間	完全並列	Nに比例	○
良好	延長補間	完全並列	Nに比例	○
良好	残差計算	完全並列	Nに比例	○
良好	簡易求解	完全並列	Nに比例	○

高並列化の手順
(RSDFTの事例)

RSDFTの並列特性分析（処理・演算量）

waveZ

ルーチン	処理内容		演算量	高並列化性能	単体性能
DTCG	ML×ML対称行列の固有値，固有ベクトルを共役勾配法で固有値の小さいものから順にMB本求める．	レイリー商 $\frac{\langle \psi_n \vert H_{KS} \vert \psi_n \rangle}{\langle \psi_n \vert \psi_n \rangle}$ ⟶ minimize	O(ML×ML)→O(N^2) O(N^2)		
GramSchmidt	規格直交化	$H_{m,n} = \langle \psi_m \vert H_{KS} \vert \psi_n \rangle$	O(ML×MB^2)→O(N^3) O(N^3)		
DIAG	ML次元の部分空間に限ってハミルトニアンの対角化をする．				
	行列要素生成（MatE）	$\psi'_n = \psi_n - \sum_{m=1}^{n-1} \psi_m \langle \psi_m \vert \psi_n \rangle$	O(ML×MB^2)→O(N^3) O(N^3)		
	固有値求解（pdsyevd）	$\begin{pmatrix} H_{N\times N} \end{pmatrix} \begin{pmatrix} \vec{c}_n \end{pmatrix} = \varepsilon \begin{pmatrix} \vec{c}_n \end{pmatrix}$	O(MB^3)→O(N^3) O(N^3)		
	回転（RotV）	$\psi'_n(r) = \sum_{m=1}^{N} c_{n,m} \psi_m(r)$	O(ML×MB^2)→O(N^3) O(N^3)		

ML:格子数，MB:バンド数

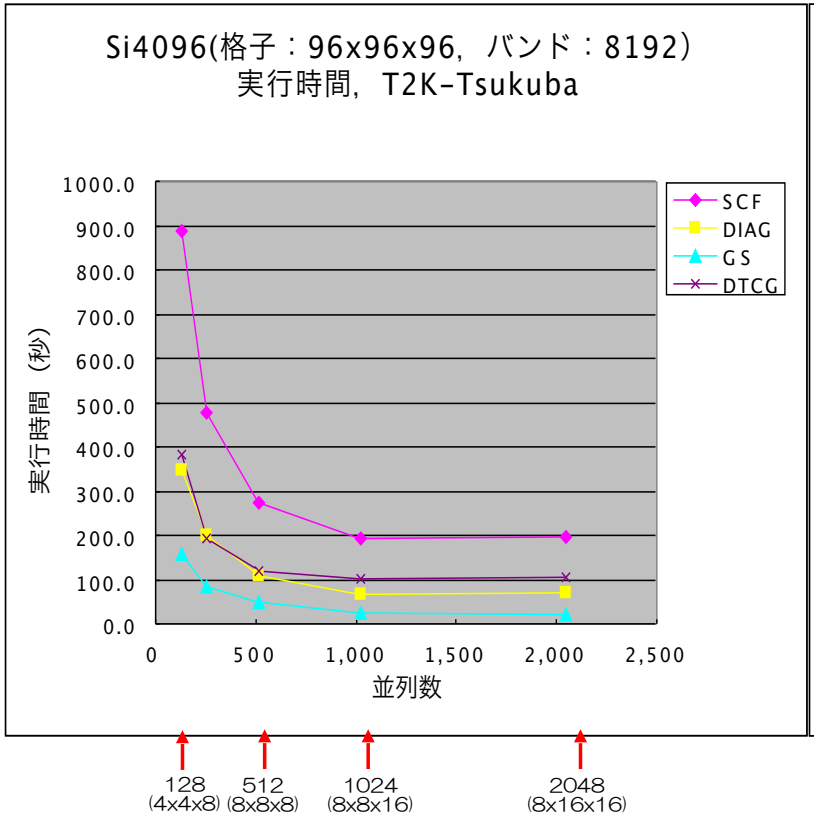
RSDFTの並列特性分析（コスト）

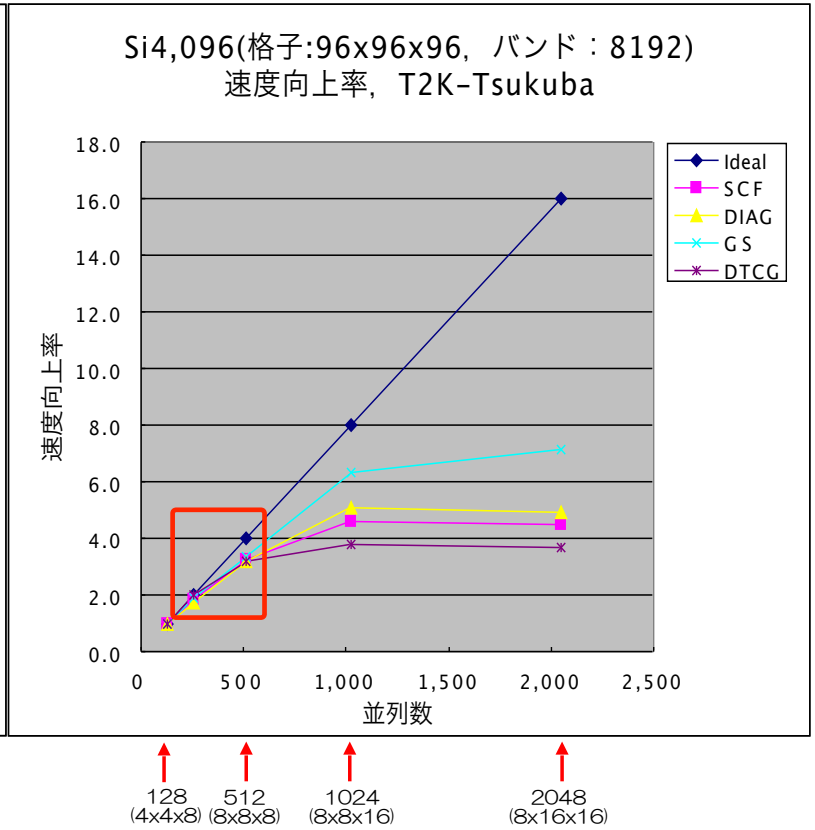
計算機　　：RICC
8,000原子：格子数120x120x120，バンド数16,000
並列数　　：8x8x8（空間方向のみ）
SCFループ1回実行の実測データからSCFループ100回として実行時間を推定

処理内容	コスト	演算	プロセス間通信
初期化 パラメータの読込み 全プロセスへの転送	0.4%		Bcast，Isend/Irecv
SCF部	99.6%	$O(N^3)$	
DO SCFループ(100回と仮定)			
DIAG	30.5%	$O(N^3)$ DGEMM中心	行列生成部: Reduce, Isend/Irecv（HPSI) 固有値ソルバー部：PDSYEVD内(Bcast) ローテーション:部分Bcast，部分Reduce
DTCG	27.4%	$O(N^2)$ 演算＜ロード	スカラー値のallreduce中心 Isend/Irecv(ノンローカル項/HPSI) Isend/Irecv(境界データ交換/BCSET)
GramSchmidt	38.6%	$O(N^3)$ DGEMM中心	Allreduce （内積，規格化変数）
Mixing, 途中結果の出力	3.1%		途中結果出力は毎SCFではないのでコストはもっと少
ENDDO			

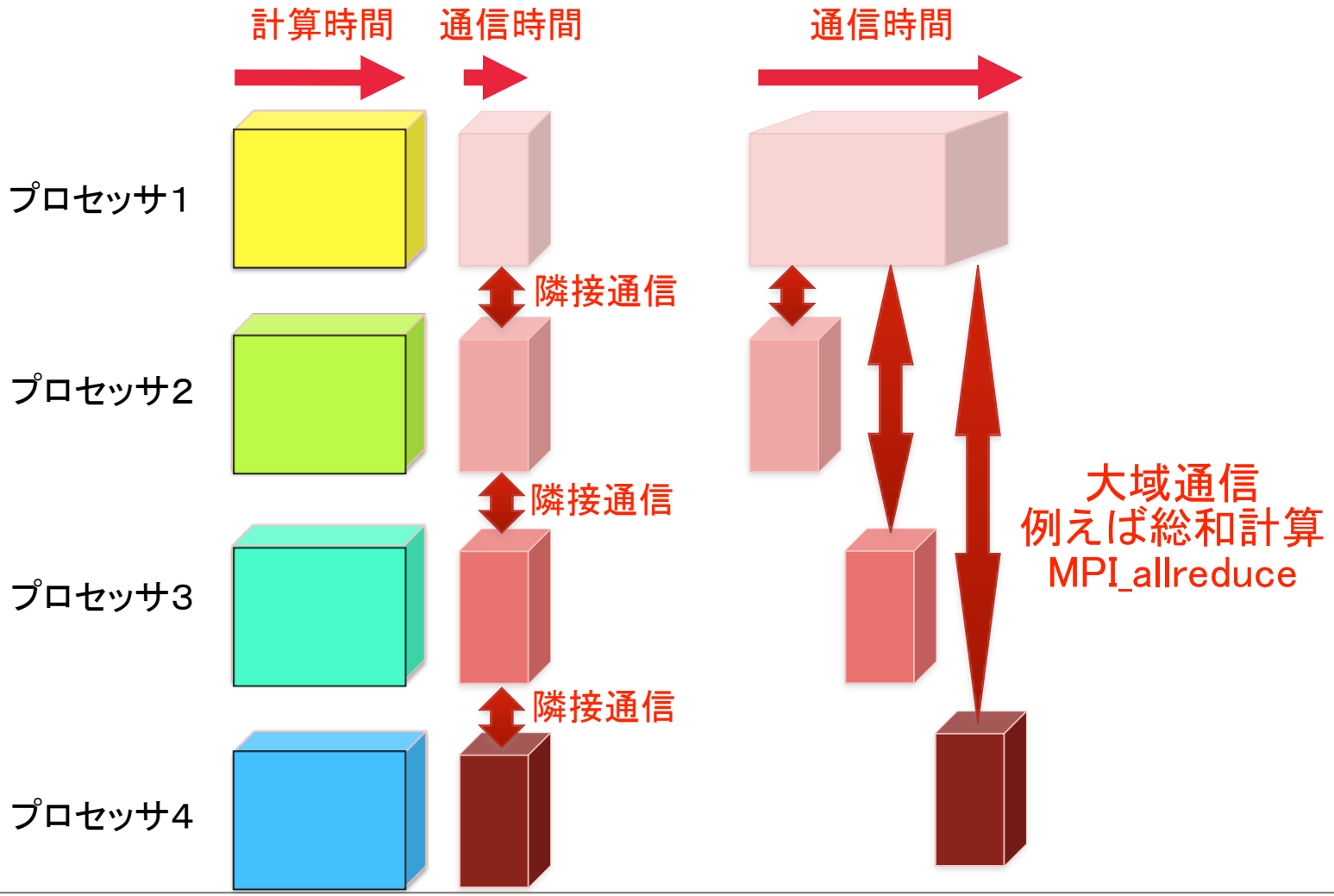
RSDFTの並列特性分析（ブロック毎のスケーラビリティ）

計算機：T2K-Tsukuba
コンパイラ&ライブラリ：PGI + mvapich2-medium
S方向の分割数:128, 256, 512, 1024, 2048





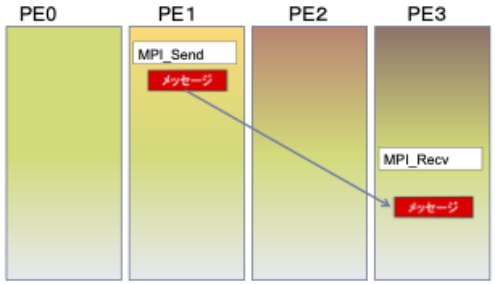
そもそも並列化とは？



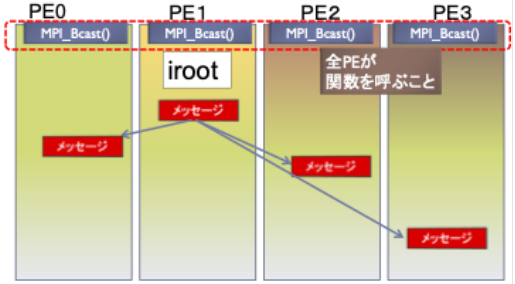
MPIの概要

waveZ

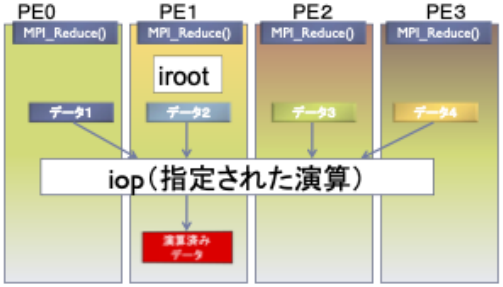
Send－Recvの概念（１対１通信）



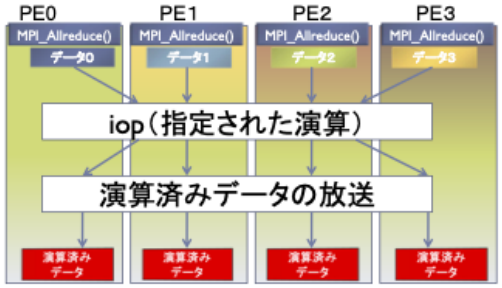
MPI_Bcastの概念（集団通信）



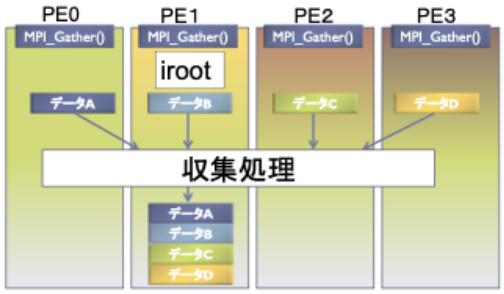
MPI_Reduceの概念（集団通信）



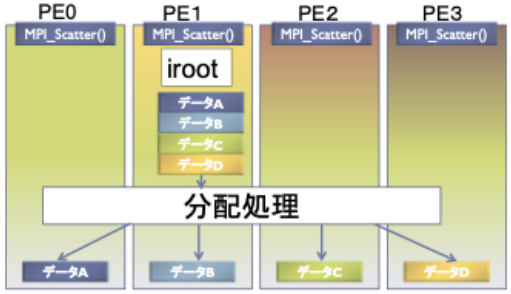
MPI_Allreduceの概念（集団通信）



MPI_Gatherの概念（集団通信）



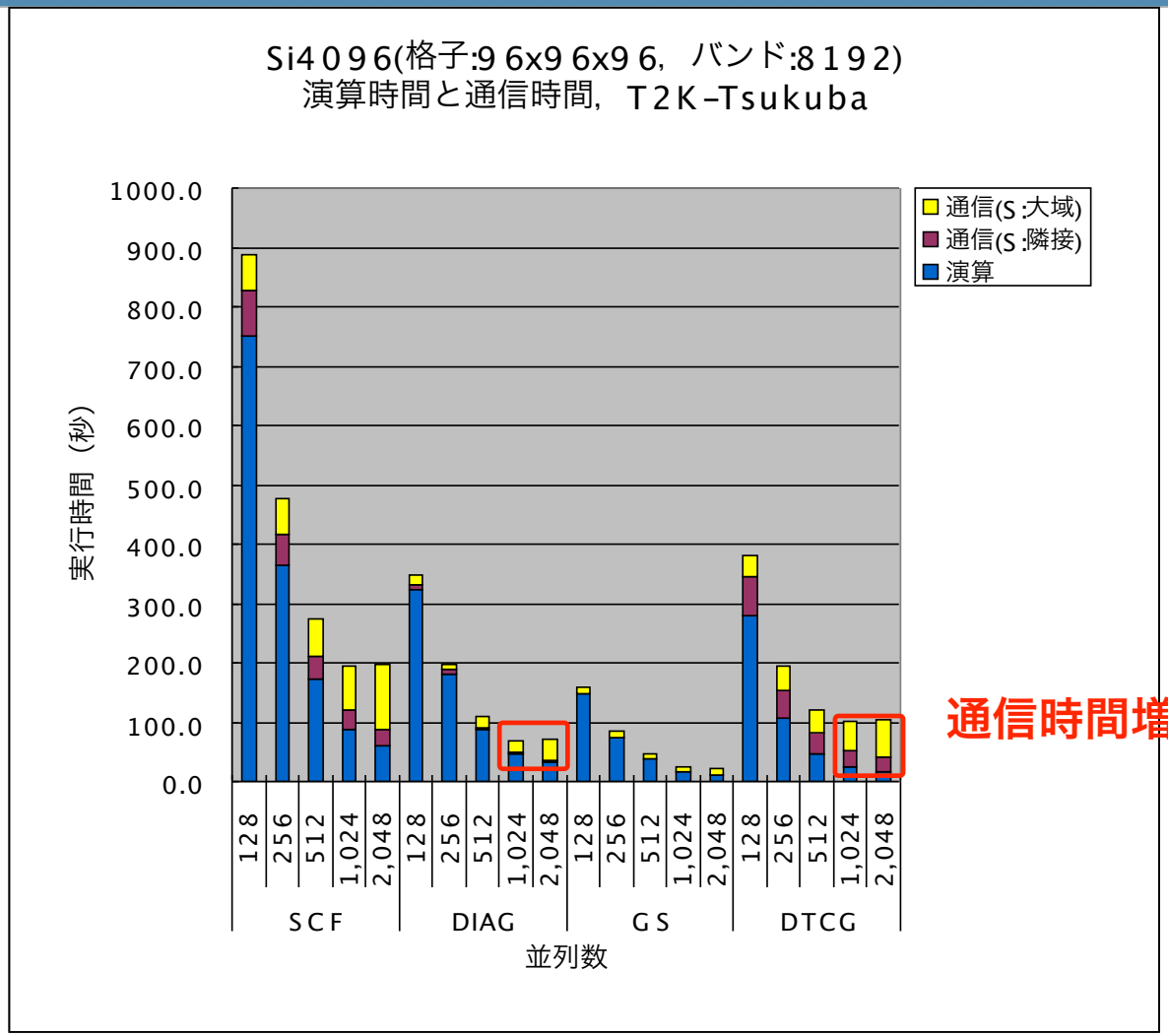
MPI_Scatterの概念（集団通信）



2017年度　計算科学技術特論A

名古屋大学 NAGOYA UNIVERSITY

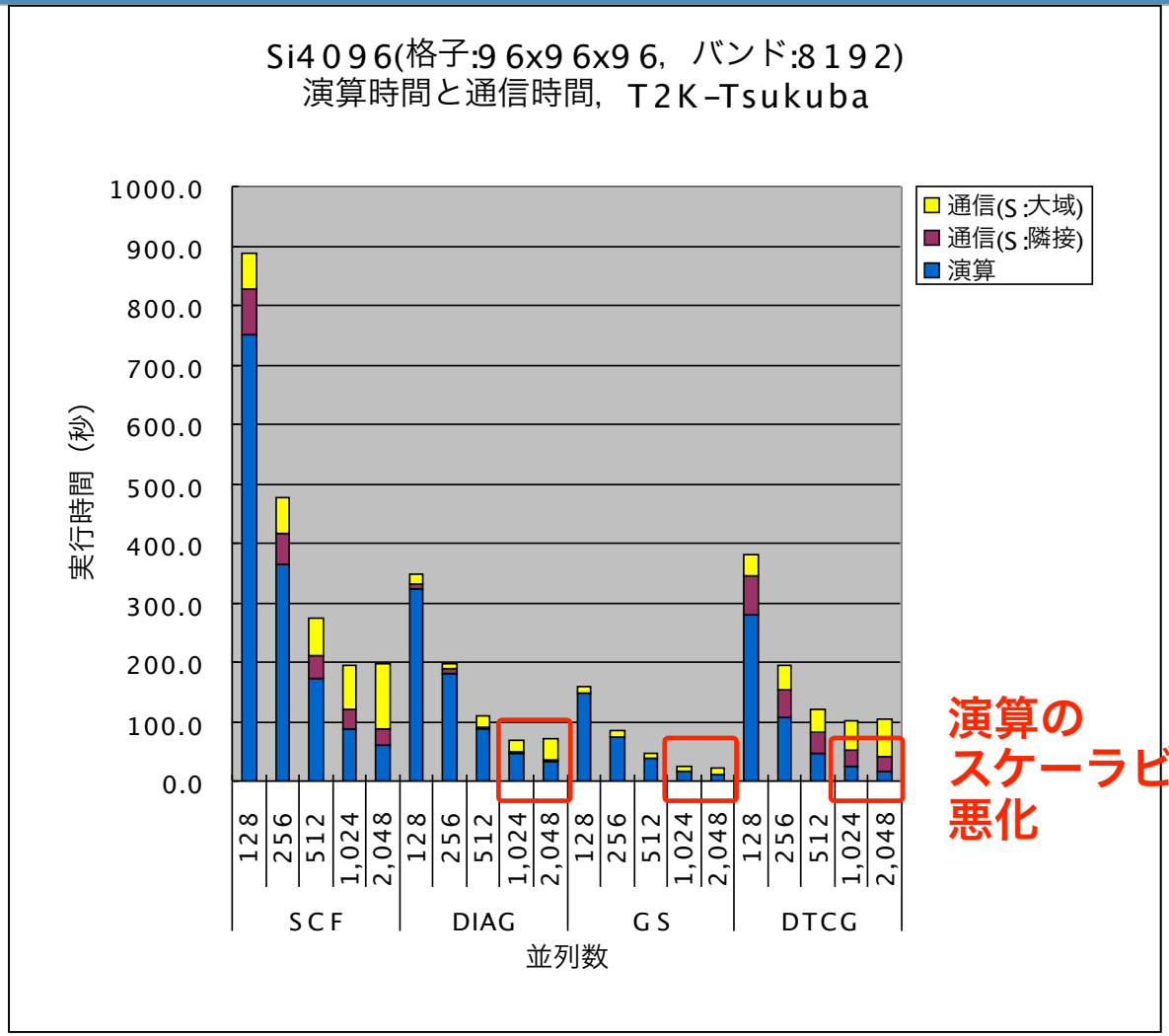
RSDFTの並列特性分析（ブロック毎のスケーラビリティ）



通信時間増大

※PDSYEVDの通信は演算部に含まれている

RSDFTの並列特性分析（ブロック毎のスケーラビリティ）



演算の
スケーラビリティ
悪化

※PDSYEVDの通信は演算部に含まれている

RSDFTの並列特性分析（並列・単体性能）

waveZ

ルーチン	処理内容		演算量	高並列化性能	単体性能
DTCG	ML×ML対称行列の固有値，固有ベクトルを共役勾配法で固有値の小さいものから順にMB本求める.	レイリー商 $\frac{\langle\psi_m\|H_{KS}\|\psi_n\rangle}{\langle\psi_n\|\psi_n\rangle}$ ⟶ minimize	O(ML×ML)→O(N²) **O(N²)**	通信時間増大 演算時間と逆転 並列度の不足	行列ベクトル積 性能は悪い
GramSchmidt	規格直交化	$H_{m,n}=\langle\psi_m\|H_{KS}\|\psi_n\rangle$	O(MLxMB²)→O(N³) **O(N³)**	通信時間減少せず 演算時間と同程度 並列度の不足	行列積化で良好
DIAG	ML次元の部分空間に限ってハミルトニアンの対角化をする.				
	行列要素生成（MatE）	$\psi'_n=\psi_n-\sum_{m=1}^{n-1}\psi_m\langle\psi_m\|\psi_n\rangle$	O(MLxMB²)→O(N³) **O(N³)**	通信時間増大 演算時間と同程度 並列度の不足 Scalapackのスケーラビリティが悪い	行列積化で良好
	固有値求解（pdsyevd）	$\begin{pmatrix} H_{N\times N} \end{pmatrix}\begin{pmatrix}\vec{c}_n\end{pmatrix}=\varepsilon\begin{pmatrix}\vec{c}_n\end{pmatrix}$	O(MB³)→O(N³) **O(N³)**		Scalapackの性能が悪い
	回転（RotV）	$\psi'_n(r)=\sum_{m=1}^{N}c_{n,m}\psi_m(r)$	O(MLxMB²)→O(N³) **O(N³)**		行列積化で良好

ML:格子数，MB:バンド数

並列性能上のボトルネック

waveZ

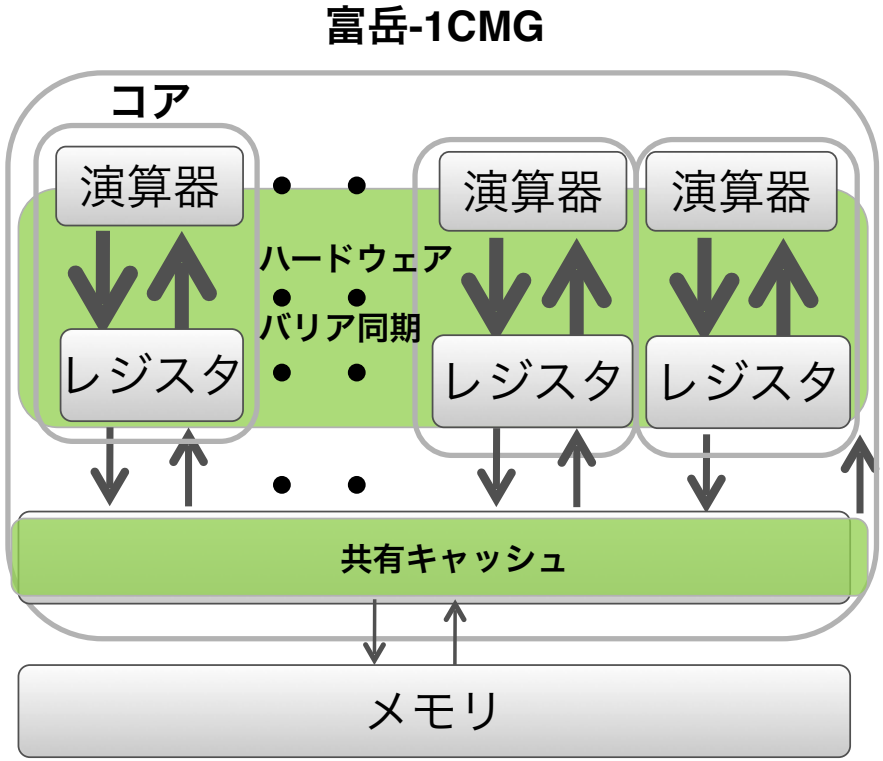
- 今まで示した調査を実施することにより処理ブロック毎に並列性能上の問題がある事が発見される.
- それらを分析するとだいたい以下の6点に分類されると考える.

1	アプリケーションとハードウェアの並列度のミスマッチ（アプリケーションの並列度不足）
2	非並列部の残存
3	大域通信における大きな通信サイズ、通信回数の発生
4	フルノードにおける大域通信の発生
5	隣接通信における大きな通信サイズ、通信回数の発生
6	ロードインバランスの発生

RSDFTのボトルネック

スレッド並列・スレッド性能

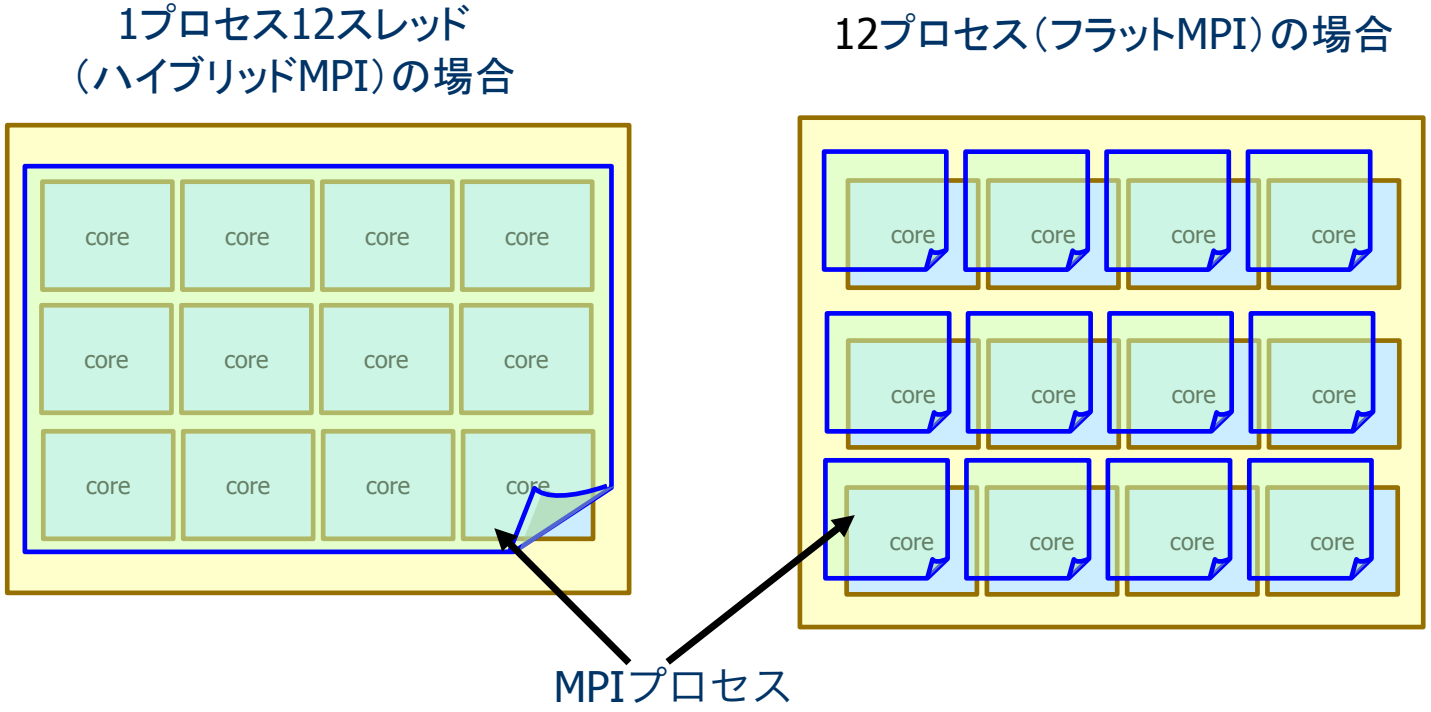
「富岳」のスレッド並列の概要（CPU）



- 富岳は1CMGに12コア搭載.
- 富岳は12コアでL2キャッシュを共有.
- MPI等のプロセス並列に対し富岳はCMG内の12コアを使用するためのスレッド並列化が必要.
- スレッド並列化のためには自動並列化かOpenMPが使用可.
- 富岳のハードウェアバリア同期機能を使用した高速なスレッド処理が可能.
- 1スレッドが1コアに対応.
- 1コアあたりの演算器は以下となっている.
  1M&ADD × 8SIMD × 2pipe

ハイブリッド並列とフラットMPI並列（CPU）

- MPIプロセス並列とスレッド並列の組み合わせをハイブリッド並列という.
- 各コアにMPIプロセスを割り当てる並列化をフラットMPI並列という.
- 京では通信資源の効率的利用，消費メモリ量を押さえる観点でハイブリッド並列を推奨している.

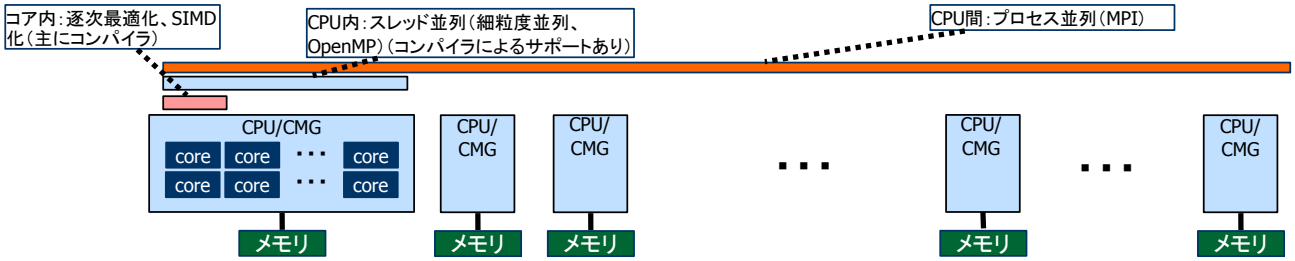


ハイブリッド並列とフラットMPI並列（CPU）

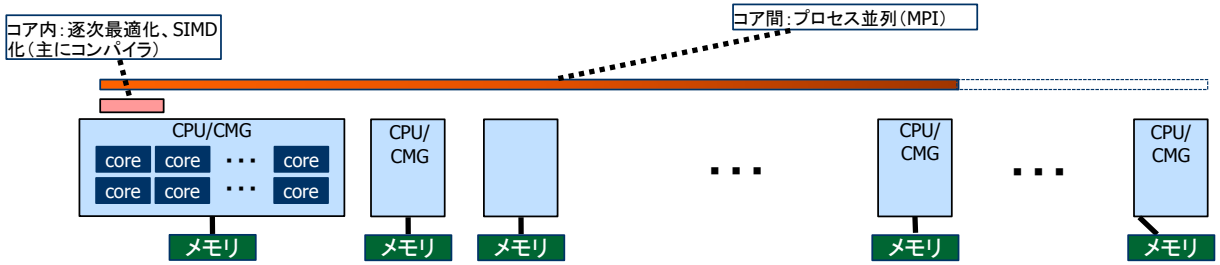
waveZ

- スレッド並列＋プロセス並列のハイブリッド型
  - コア内：コンパイラによる逐次最適化，SIMD化
  - CPU/CMG内：スレッド並列(自動並列化：細粒度並列化†，OpenMP)
  - CPU/CMG間：プロセス並列(MPI)

†細粒度並列化
高速バリア同期を活用し，内側ループをコア間で並列化
ベクトル向け（内側ループが長い）コードの高速化が可能



- プロセス並列型
  - コア内：コンパイラによる逐次最適化，SIMD化
  - コア間：プロセス並列(MPI)



実行効率の観点から、ハイブリッド型を推奨

共有キャッシュを利用したCyclicThread並列（CPU−富岳）

waveZ

3次元目をcyclicでスレッド並列化

```
!$OMP DO SCHEDULE(static,1),PRIVATE(I,J,K)
 do J = 1, NY
      do I = 1, NX
        do K = 1, NZ
          DYV (k,I,J) = (V(k,I,J)  -V(k,I,J-1))*R40 &
                      - (V(k,I,J+1)-V(k,I,J-2))*R41
        end do
      end do
   end do
```

キャッシュに載せる

- 第3軸をcyclic分割 →　1ストリームで3配列がL2に乗る(説明次項)
- 性能が２倍になる

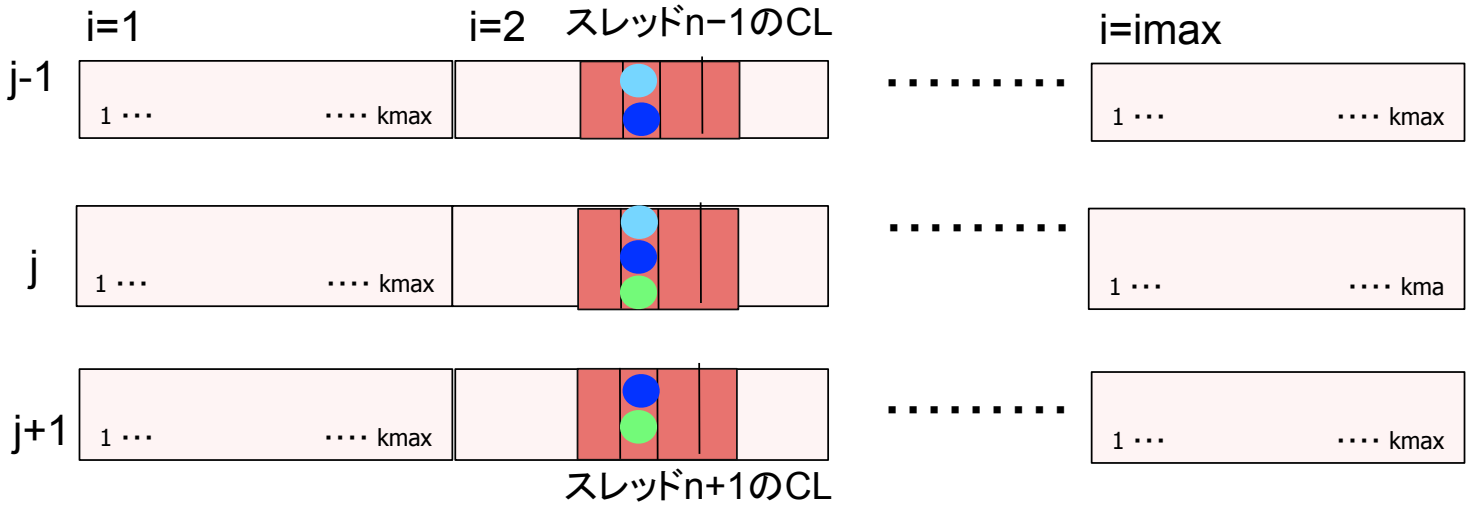
共有キャッシュを利用したCyclicThread並列（CPU−富岳）

プログラム例

```
Do j=1,jmax
  do i=1,imax
    do k=1,kmax
      a(k,i,j)=・・・c0*v(k,i,j-1)+c1* v(k,i,j)+c2* v(k,i,j+1)・・・
    end do
  end do
end do
```

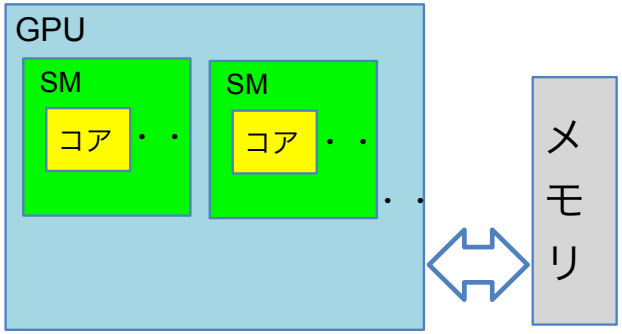
● :スレッドn+1で参照するデータ
● :スレッドnで参照するデータ
● :スレッドn-1で参照するデータ

キャッシュライン(CL)



GPUのスレッド数（GPU）

- 1GPUは複数のSM（Streaming Multiprocessor）を持つ（A100の場合108個）
- 複数のワープ（スレッド）を同時に動作可能な状態にしあるスレッドがメモリ待ちする間に他のスレッドが単純な演算コアを使用することでメモリレイテンシーを隠蔽する仕組みをとっている
- Stream-triadベンチマークでテストした結果メモリレイテンシーを隠すのにSMあたり1500程度のスレッドが必要
- Stream-triadベンチマークでは1GPUあたり15万程度のスレッド数が目安となる



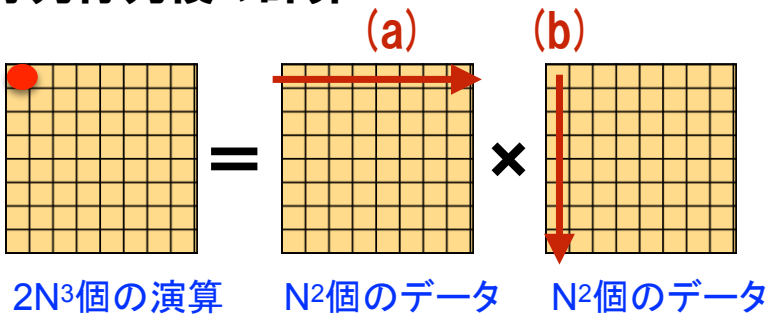
SIMTアーキテクチャ
（Single Instruction Multi Threads）

アプリケーションのタイプ– 性能の観点から –（CPU/GPU）

waveZ

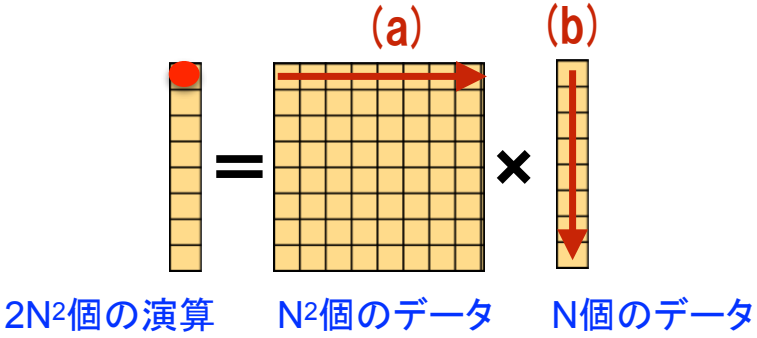
要求B/F値が小さい計算

行列行列積の計算

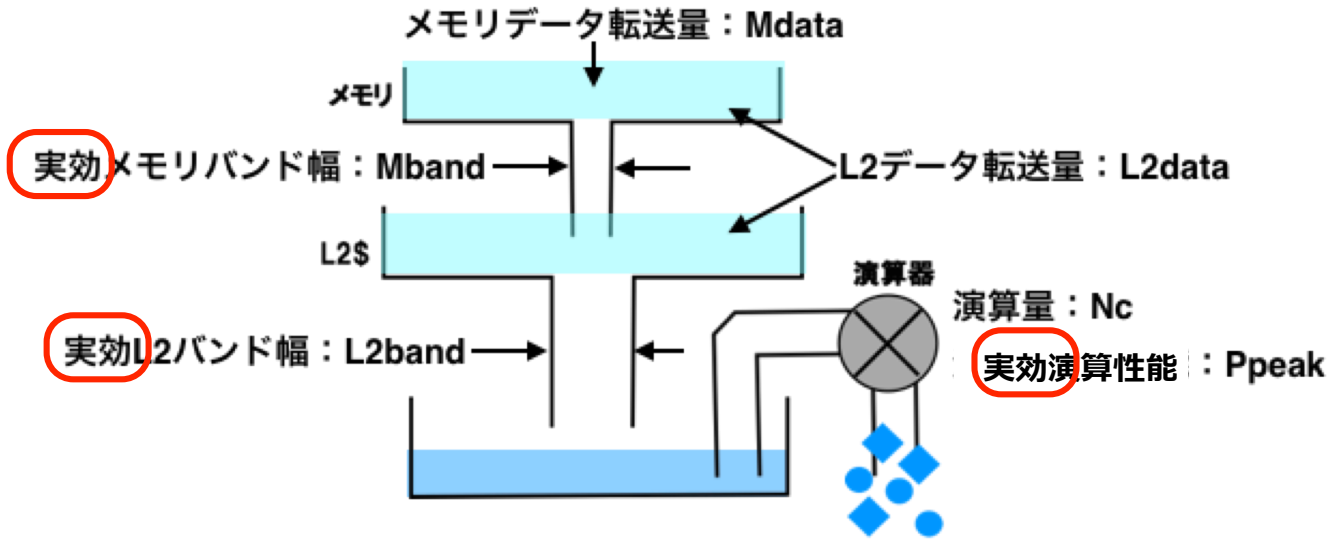


要求B/F値が大きい計算

行列ベクトル積の計算



メモリ・L2 キャッシュ・演算器を考慮した性能モデル



メモリデータ移動時間：t_M=Mdata/Mband

L2データ移動時間：t_{L2}=L2data/L2band

最小演算時間：t_C=Nc/Ppeak

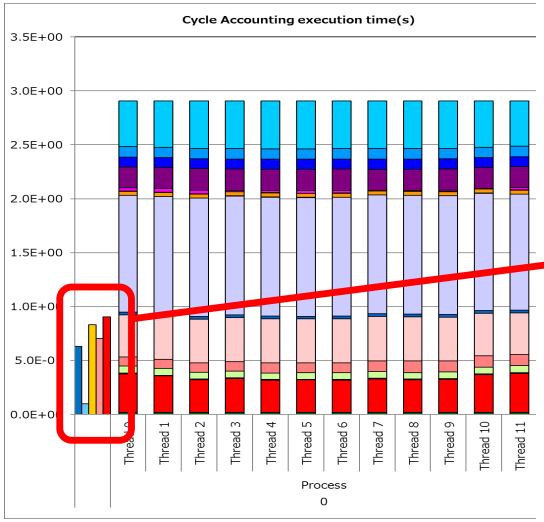
実行時間：$t_E = \max\{t_M, t_{L2}, t_C\}$

ピーク性能比：$C_p = N_c/(\max\{t_M, t_{L2}, t_C\} \times P_{peak})$

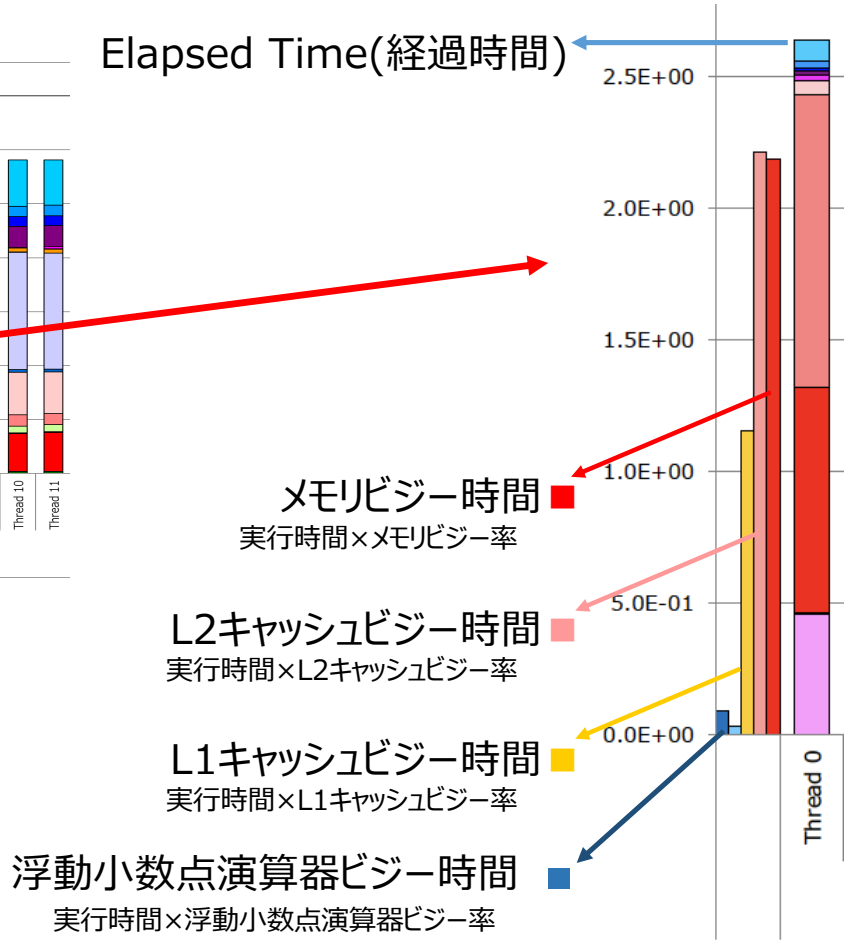
これ以降の話はCPUで検証している．ただしメモリについてはGPUでも成り立つ！

ルーフラインモデルは，
理論値とt_Mとt_Cを用いたモデルである．

CPU性能解析レポート（ビジー時間）



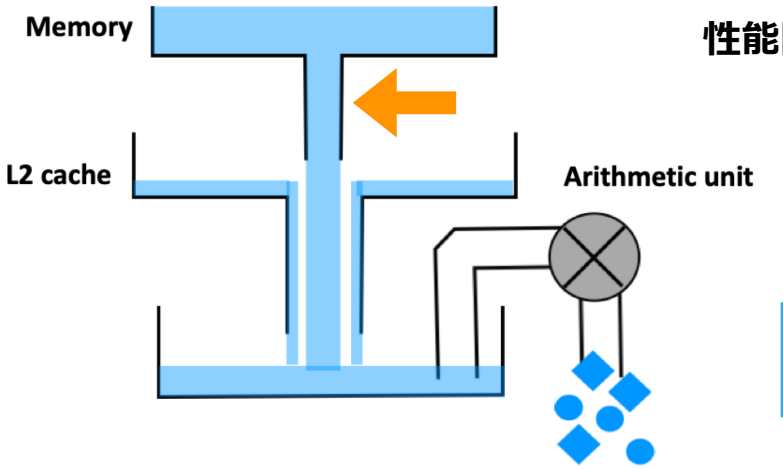
ビジー時間は前頁の
t_E, t_M, t_{L2}, t_c,
に近い 値．詳細は後述



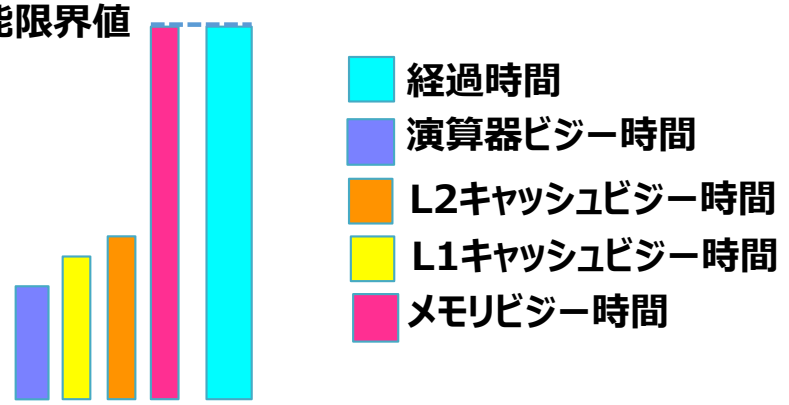
ビジー時間から見たアプリケーションのタイプ

理想的にチューニングされたオンメモリなアプリケーション

CPU/GPU



経過時間＝メモリビジー時間



メモリデータ移動時間：t_M=Mdata/Mband

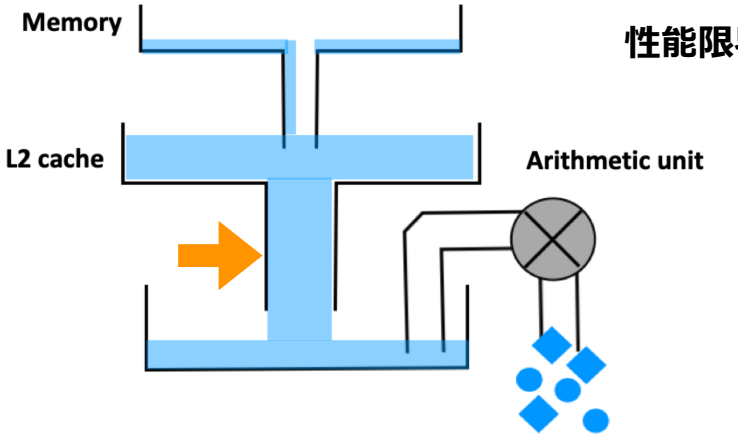
L2データ移動時間：t_{L2}=L2data/L2band

最小演算時間：t_C=Nc/P_{peak}

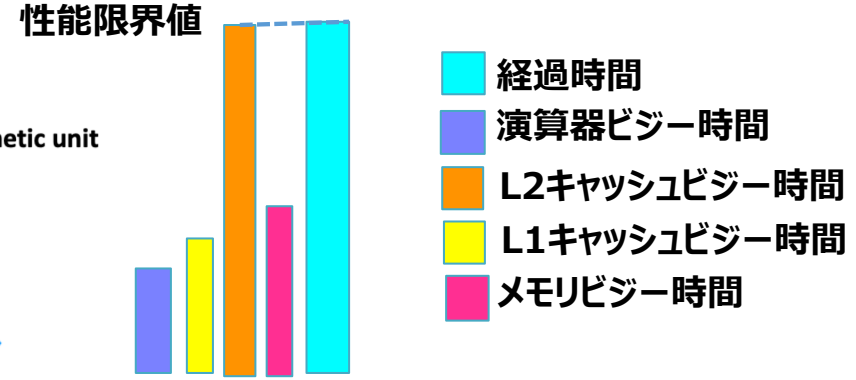
実行時間：$t_E = \max\{t_M, t_{L2}, t_C\}$

ビジー時間から見たアプリケーションのタイプ

理想的にチューニングされたオンキャシュなアプリケーション



経過時間＝$L2ビジー時間



メモリデータ移動時間：t_M=Mdata/Mband

L2データ移動時間：t_{L2}=L2data/L2band

最小演算時間：t_C=Nc/P_{peak}

実行時間：$t_E = \max\{t_M, t_{L2}, t_C\}$

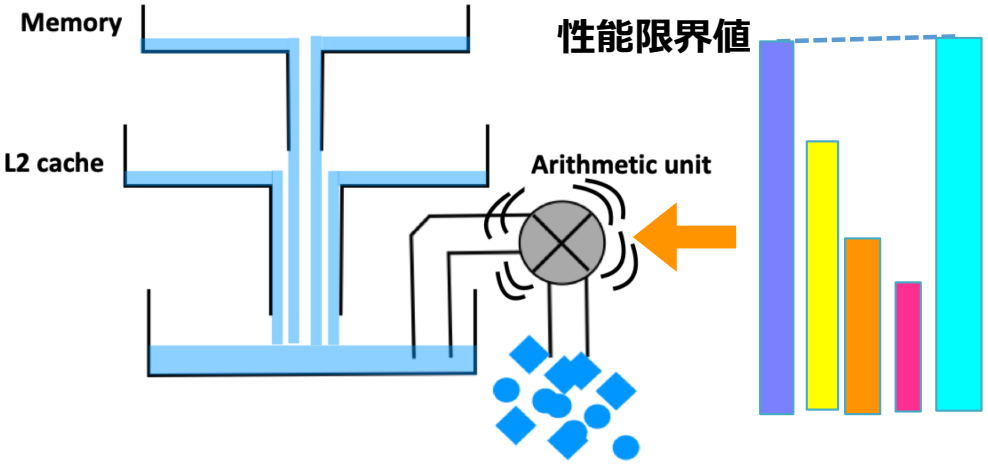
ビジー時間から見たアプリケーションのタイプ

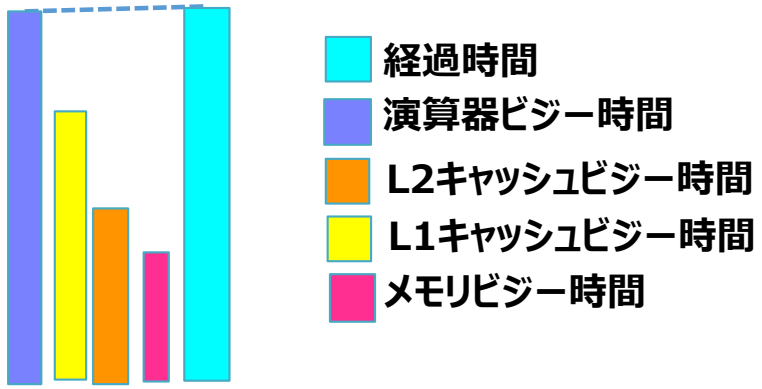
waveZ

理想的にチューニングされたオンキャシュなアプリケーション

経過時間＝演算器ビジー時間

CPU/GPU





メモリデータ移動時間：t_M=Mdata/Mband

L2データ移動時間：t_{L2}=L2data/L2band

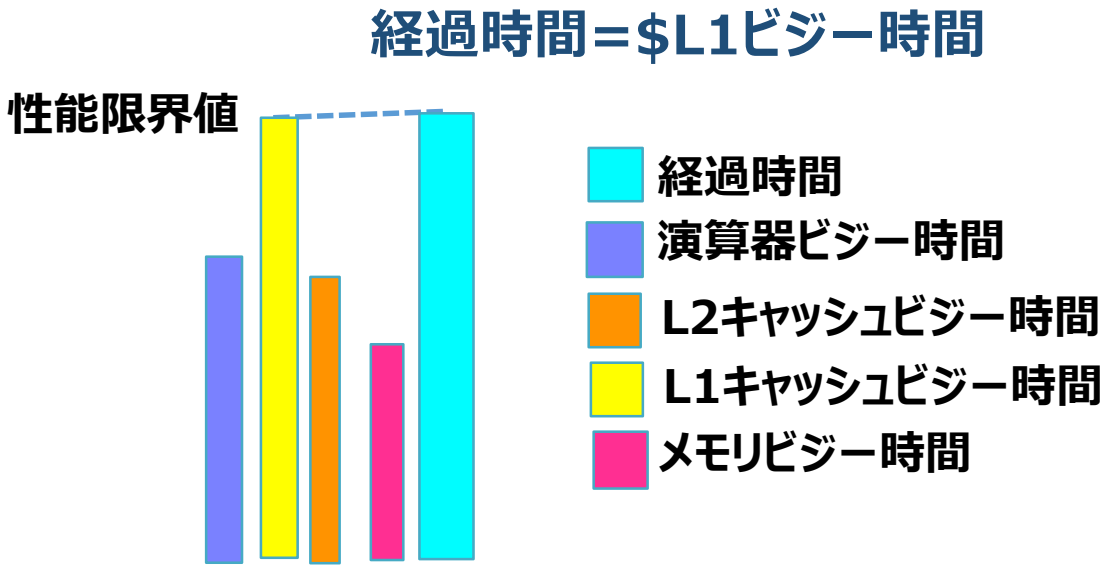
最小演算時間：t_C=Nc/P_{peak}

実行時間：$t_E = \max\{t_M, t_{L2}, t_C\}$

ビジー時間から見たアプリケーションのタイプ

waveZ

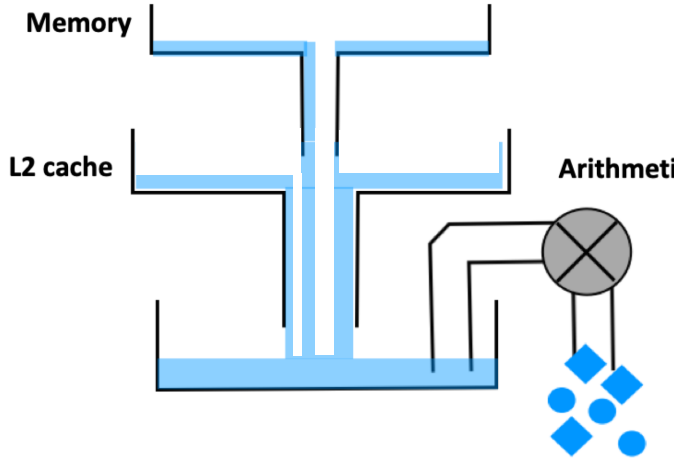
理想的にチューニングされたオンキャシュなアプリケーション

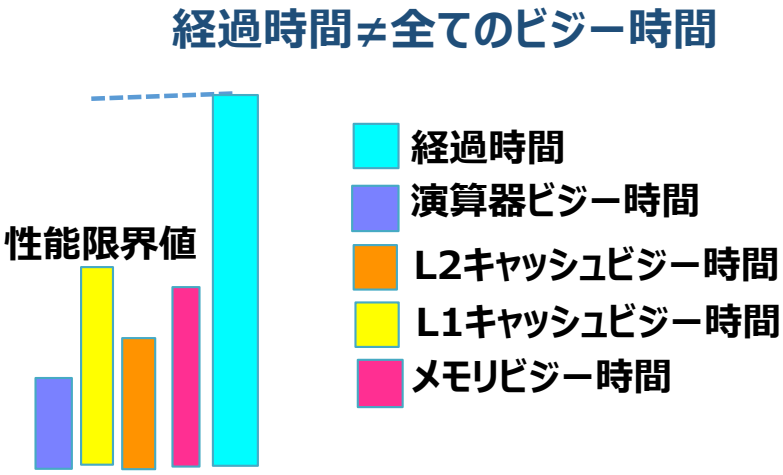


ビジー時間から見たアプリケーションのタイプ

CPU/GPU

十分にチューニングされていないオンメモリ・オンキャッシュなアプリケーション





性能から見たアプリケーションの分類

waveZ

要求B/F値が小さい

番号	単体性能上の分類	アプリケーション例
1	行列行列積に書き換え可能	第一原理 (DFT) 量子計算等
2	要求 B/F 値が小さくループボディがシンプル	高次なステンシル計算等
3	キャッシュブロッキング可能	分子動力学・重力多体問題等
4	要求 B/F 値が小さいがループボディが複雑	プラズマ・気象の物理過程 量子化学計算等
5	要求 B/F 値が大きい	気象の力学過程・流体・ 地震・核融合等
6	要求 B/F 値が大きくリストアクセスを使用	有限要素法を用いた構造・ 流体計算等

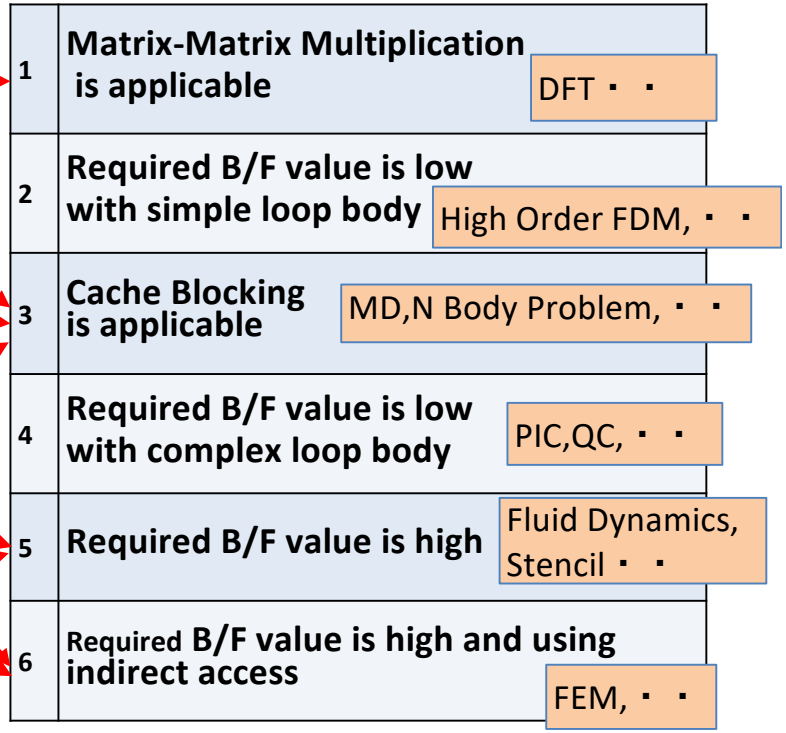
要求B/F値が大きい

ビジー時間から見たアプリケーションタイプと性能から見たアプリケーションの分類の対応

waveZ

性能の高い順に並べると(あくまで目安です)

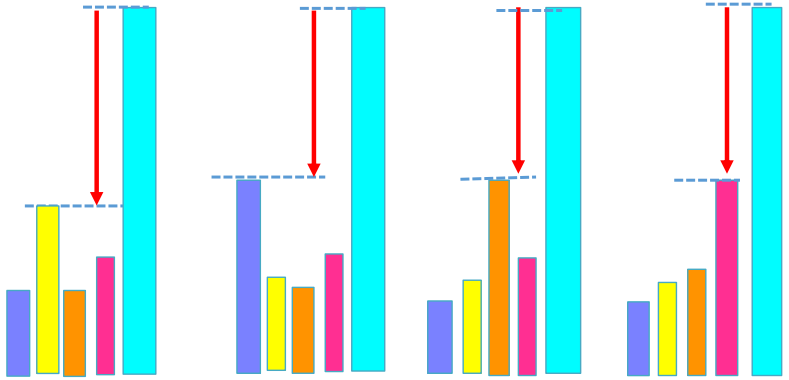
1. 理想的な演算器チューニングが可能なオンキャッシュアプリケーション
2. 理想的な$L1チューニングが可能なオンキャッシュアプリケーション
3. 理想的な$L2チューニングが可能なオンキャッシュアプリケーション
4. 理想的なチューニングが難しいオンキャッシュアプリケーション
5. 理想的なチューニングが可能なオンメモリアプリケーション
6. 理想的なチューニングが難しいオンメモリアプリケーション



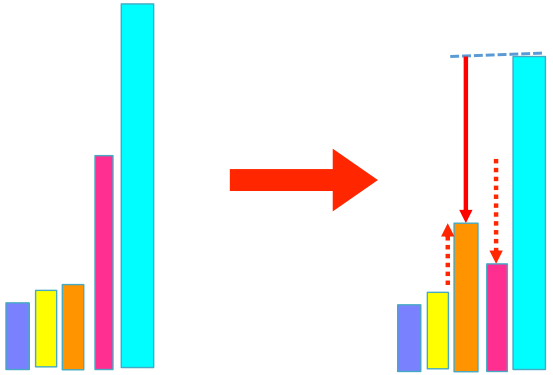
性能チューニングとは？

waveZ

ビジー時間の限界値に
近づける！



オンメモリからオンキャッシュ
チューニング！
その後ビジー時間の限界値
に近づける！



なぜ経過時間>ビジー時間となってしまうか？

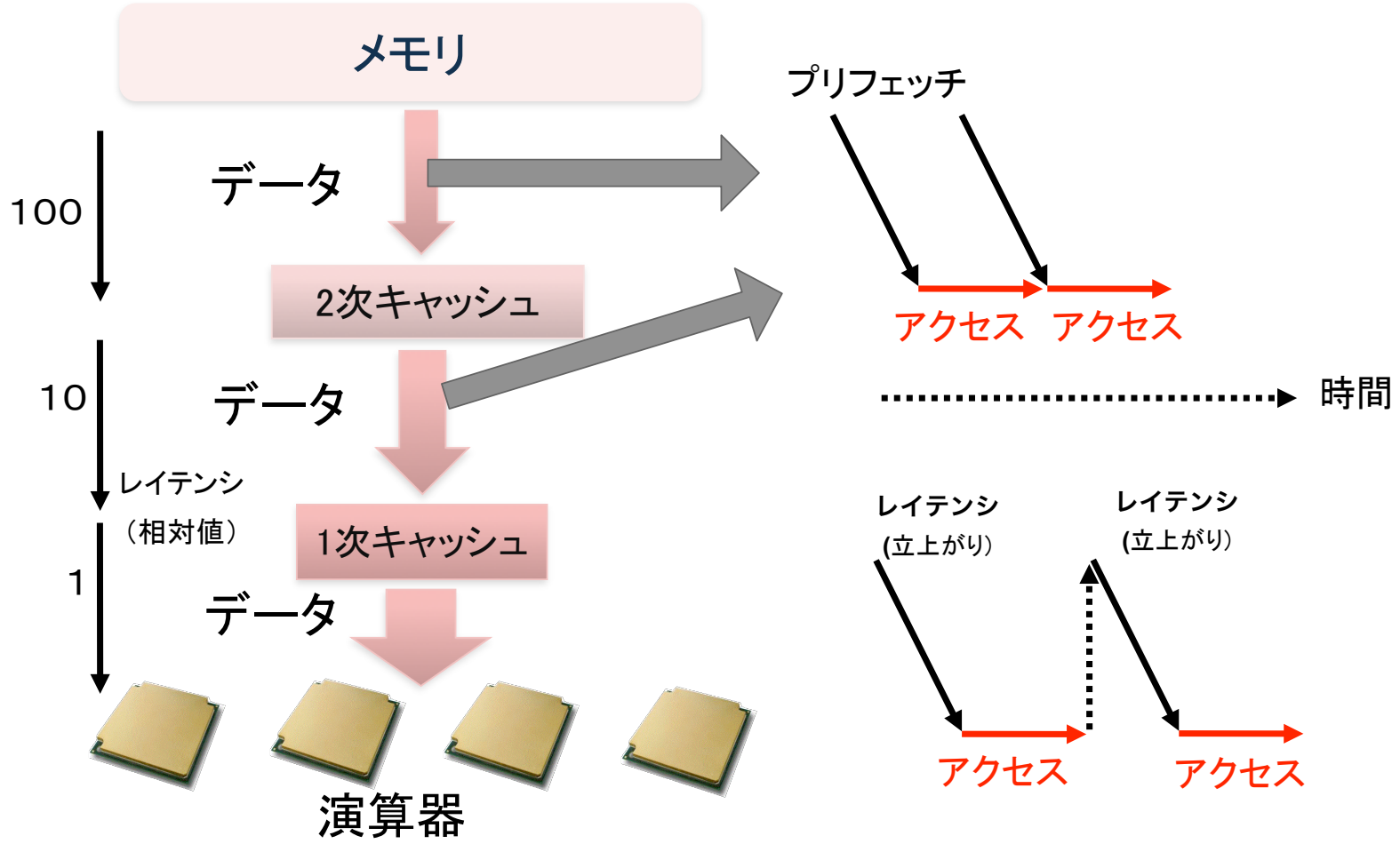
waveZ

代表的な4つの要素

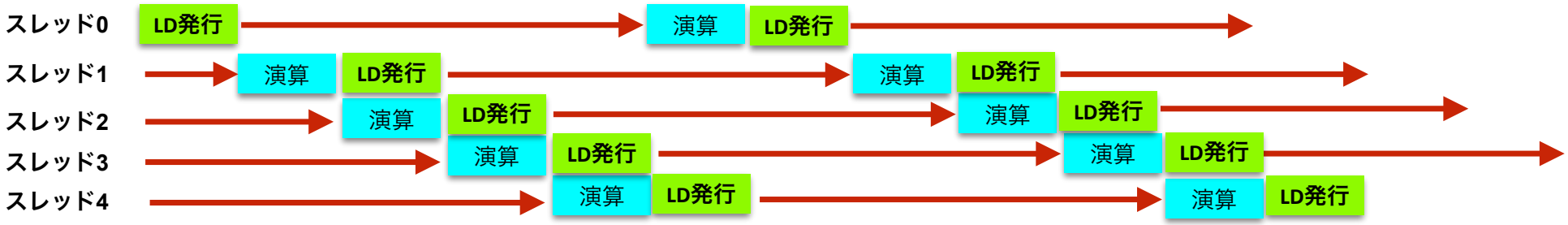
(1) メモリアクセスが非効率
(2) ラインアクセスが非効率
(3) キャッシュの有効利用ができていない
(4) 命令スケジューリングが非効率
・・・・・

色々な問題が考えられる

メモリアクセスが非効率（CPU）



メモリアクセスが非効率（GPU）

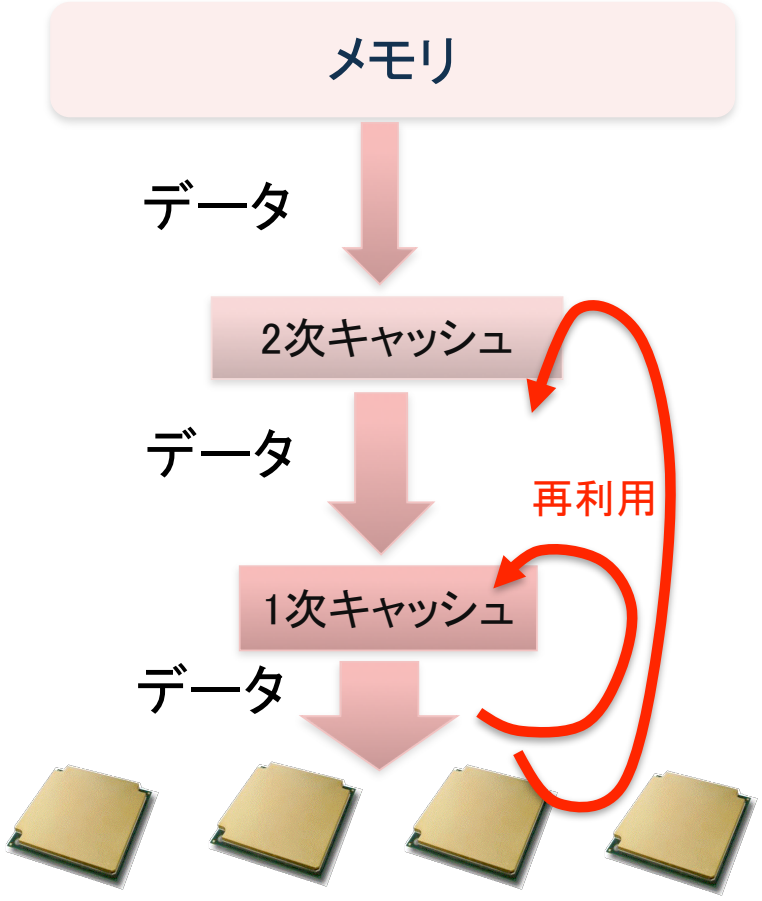


- メモリアクセスは時間がかかる（→はメモリレイテンシ）
- 複数スレッドを同時に動作させる（例は5スレッド）
- スレッド0のメモリアクセス中に1-4スレッドの演算実行
- スレッド0のメモリアクセスは隠蔽できている

- 複数スレッドを同時に動作させる（例は2スレッド）
- スレッド0のメモリアクセス中にスレッド1の演算実行
- スレッド数が少なすぎてスレッド0のメモリアクセスは隠蔽できていない

- この図では各スレッドが別の動作をするように見えているが実際はワープ単位で命令は実行される
- 説明の簡素化のための図である

キャッシュの有効利用ができていない（CPU/GPU）

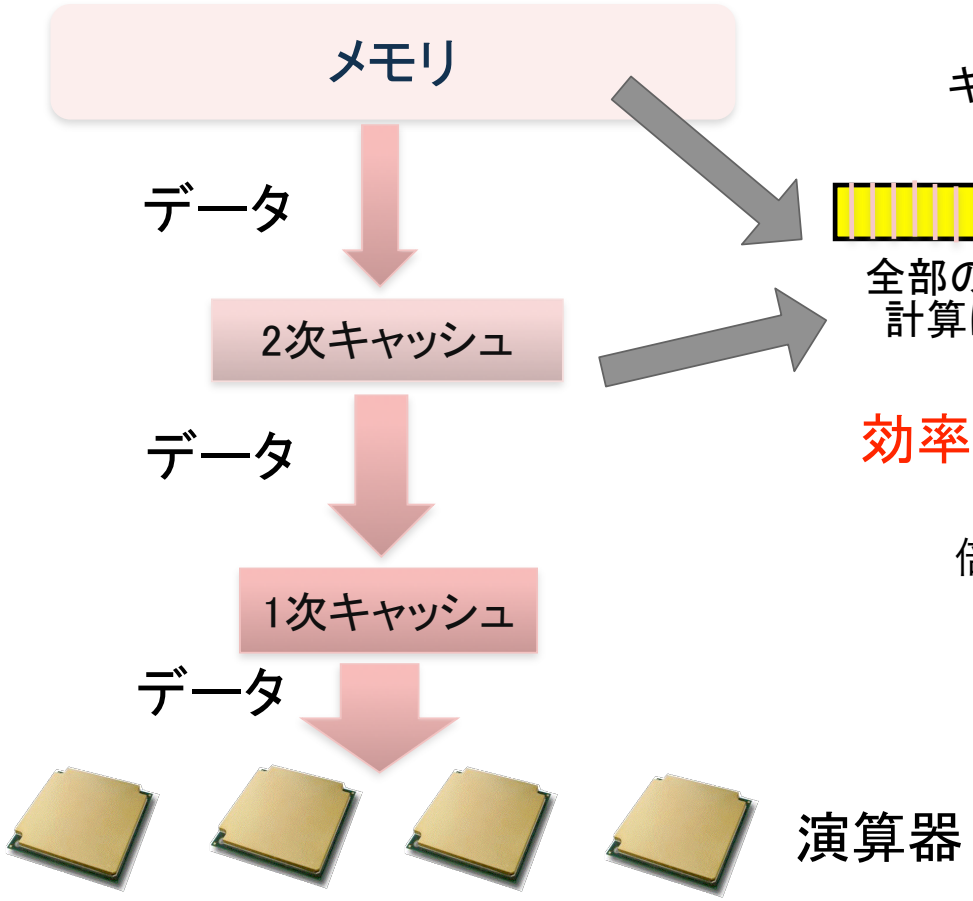


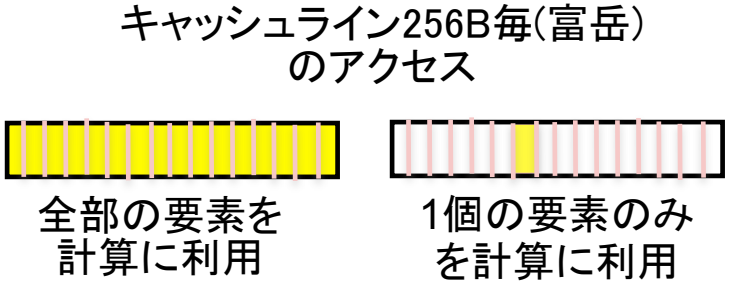
・データ供給能力の高いキャッシュを設ける
・キャッシュに置いたデータを何回も再利用し演算を行なう
・こうする事で演算器の能力を十分使い切る

- **ブロッキング**

演算器

ラインアクセスが非効率（CPU/GPU）





倍精度の場合，最大1/32(富岳)
の効率に落ちる.

- **GPUも1ワープ内のスレッドが不連続なアドレスをアクセスすると非コアレッシングアクセスとなりCPU同様の非効率が生じる場合が多い（跳び幅とキャッシュラインの大きさによる）**

命令スケジューリングが非効率（CPU）

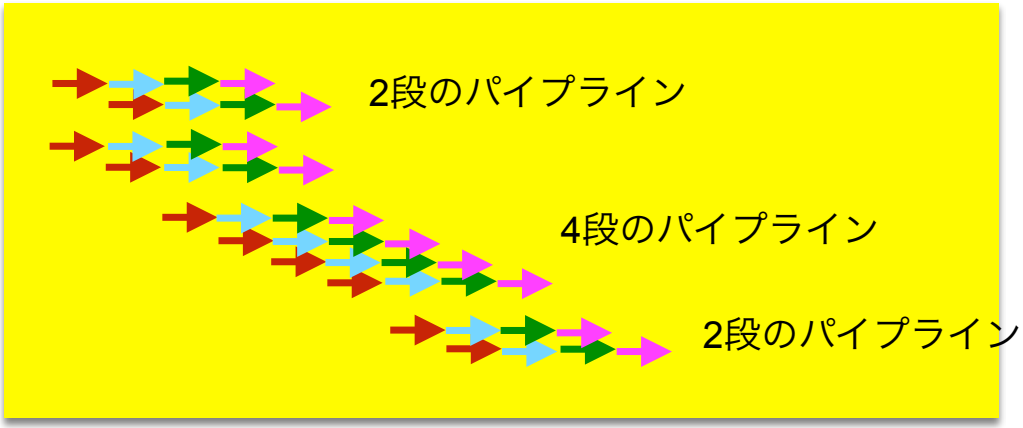
waveZ

CPU-ソフトウェアパイプラインニング

スケジューリング
機能

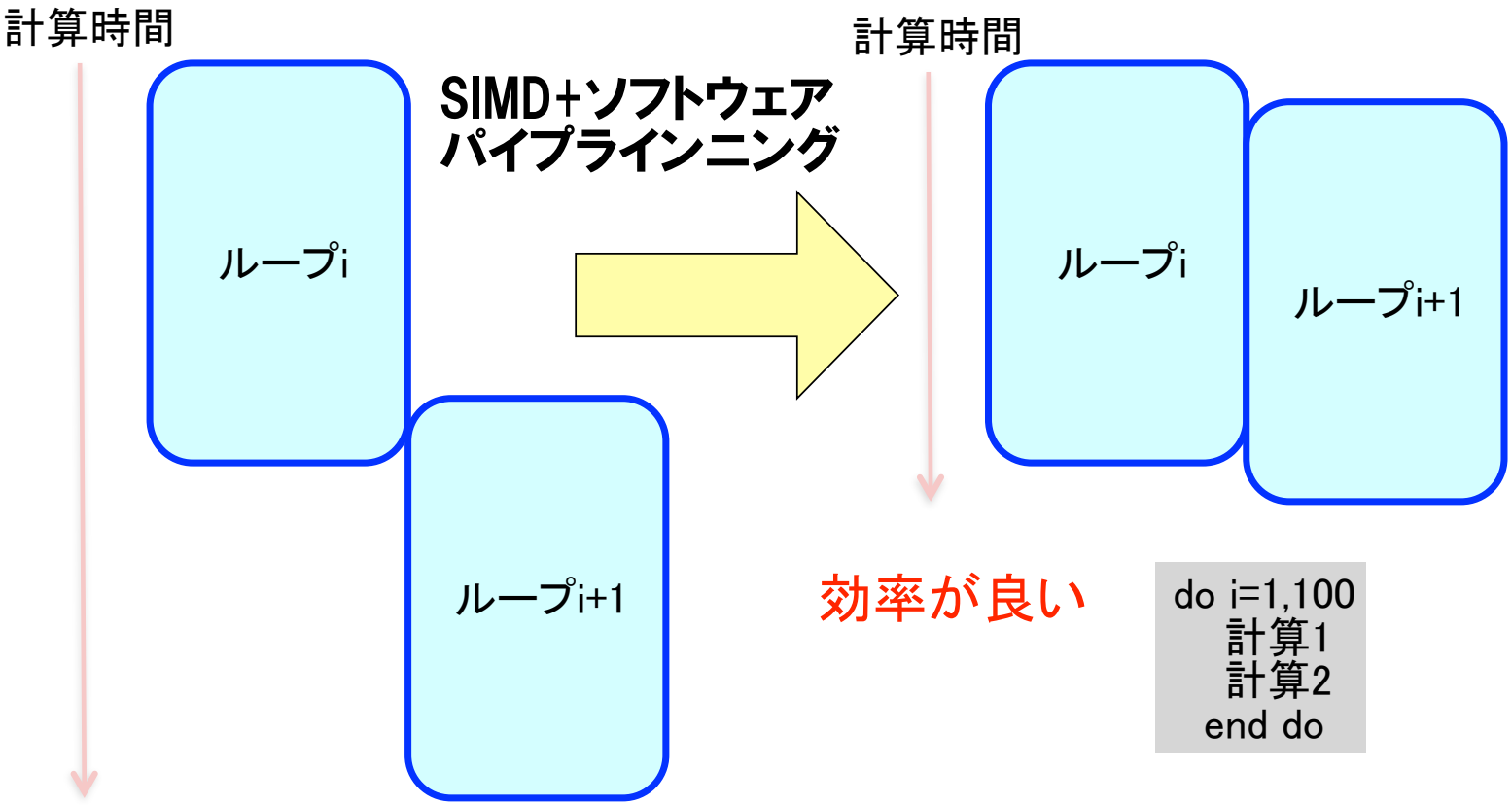
< ソフトウェアパイプラインニング>

```
do i=1,100
      a(i)のロード

      b(i)のロード

      a(i)とb(i)の演算

      i番目の結果のストア

end do
```



< 4要素を11クロックで計算できる>

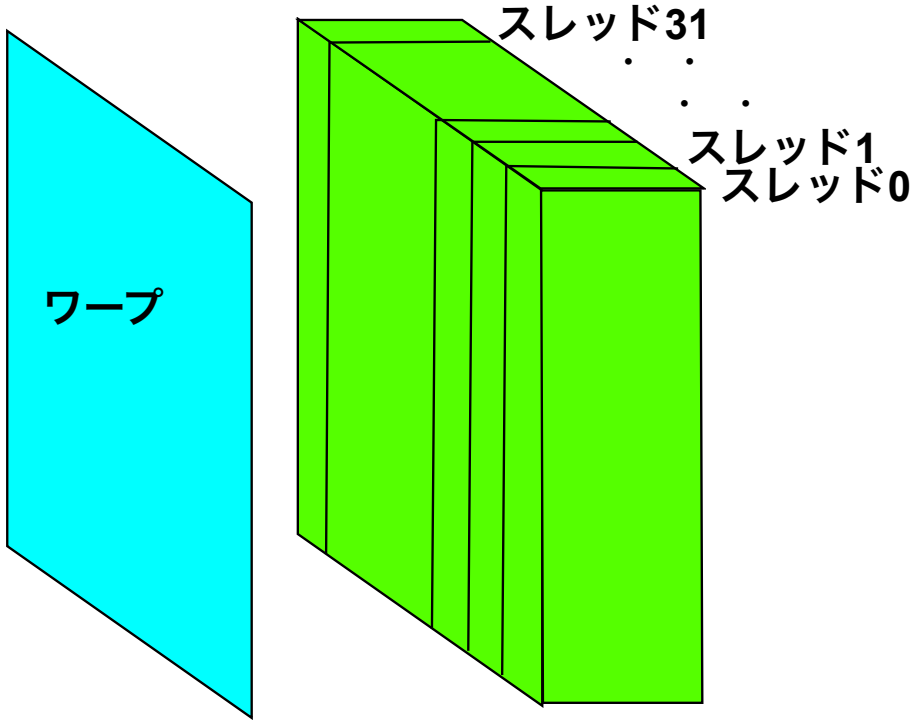
命令スケジューリングが非効率（CPU）



この他に重要なスケジューリング機能として
アウト・オブ・オーダーがある.

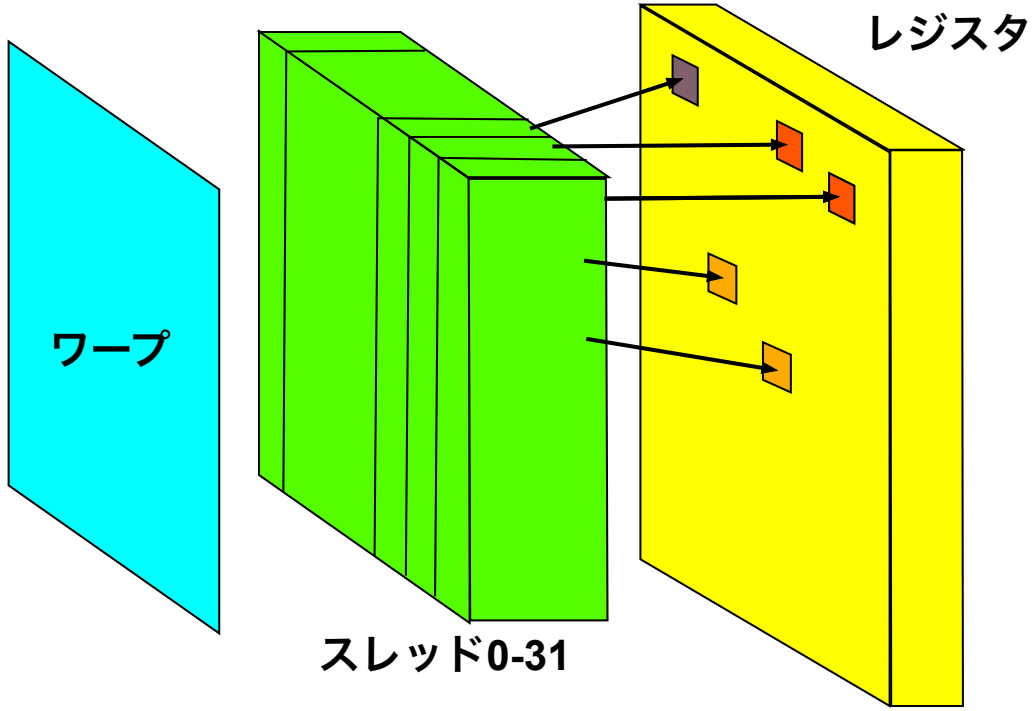
命令スケジューリングが非効率（GPU）

- GPUは基本的に命令はワープ単位で実行されワープ内の32スレッドは同じ動作をする



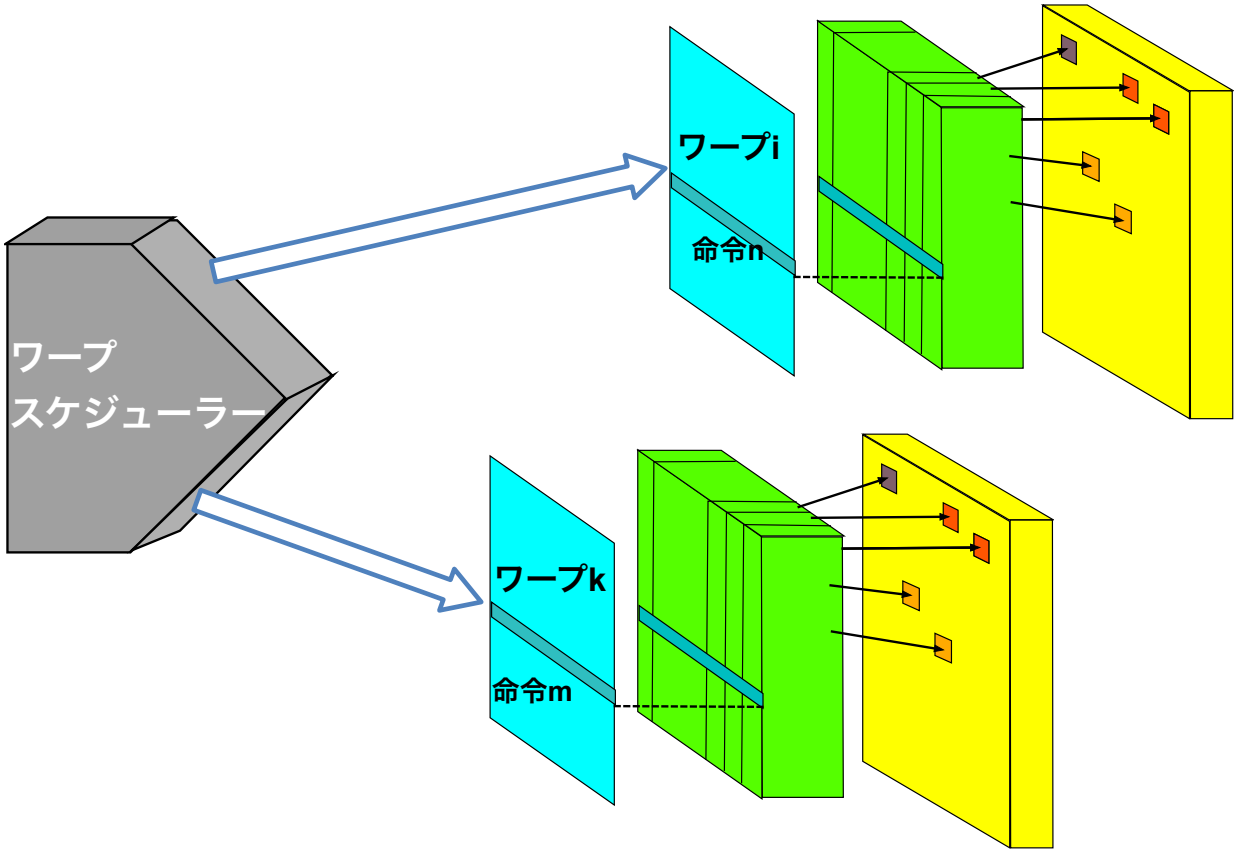
命令スケジューリングが非効率（GPU）

- コンパイル時にスレッドごとに別のレジスタが割り当てられ同じ動作をするがスレッドごとに別のデータを使用しそれぞれの演算を実行する

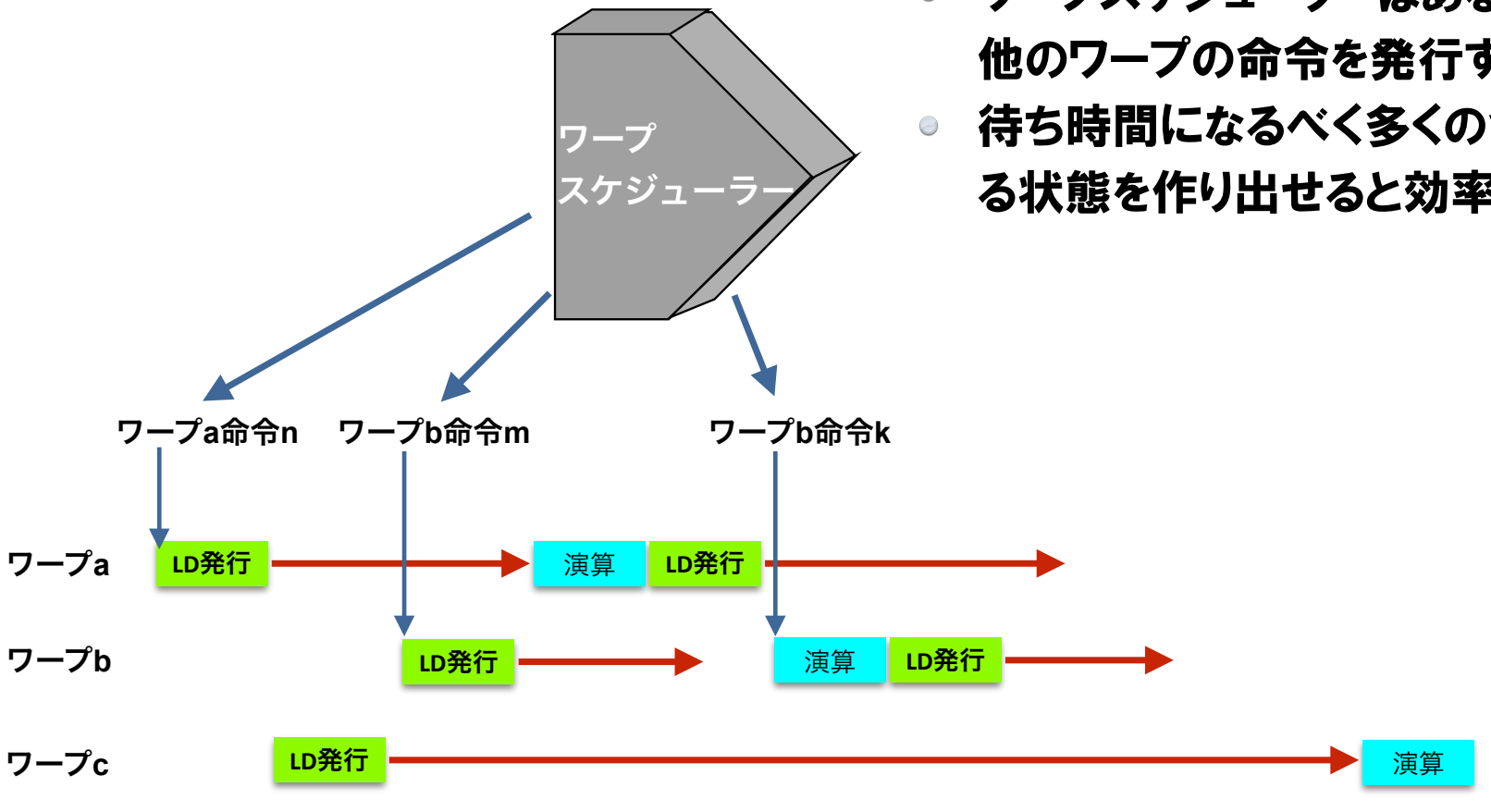


命令スケジューリングが非効率（GPU）

- ワープスケジューラによりワープとワープ内の命令が選択・スケジューリングされ命令が発行される



命令スケジューリングが非効率（GPU）

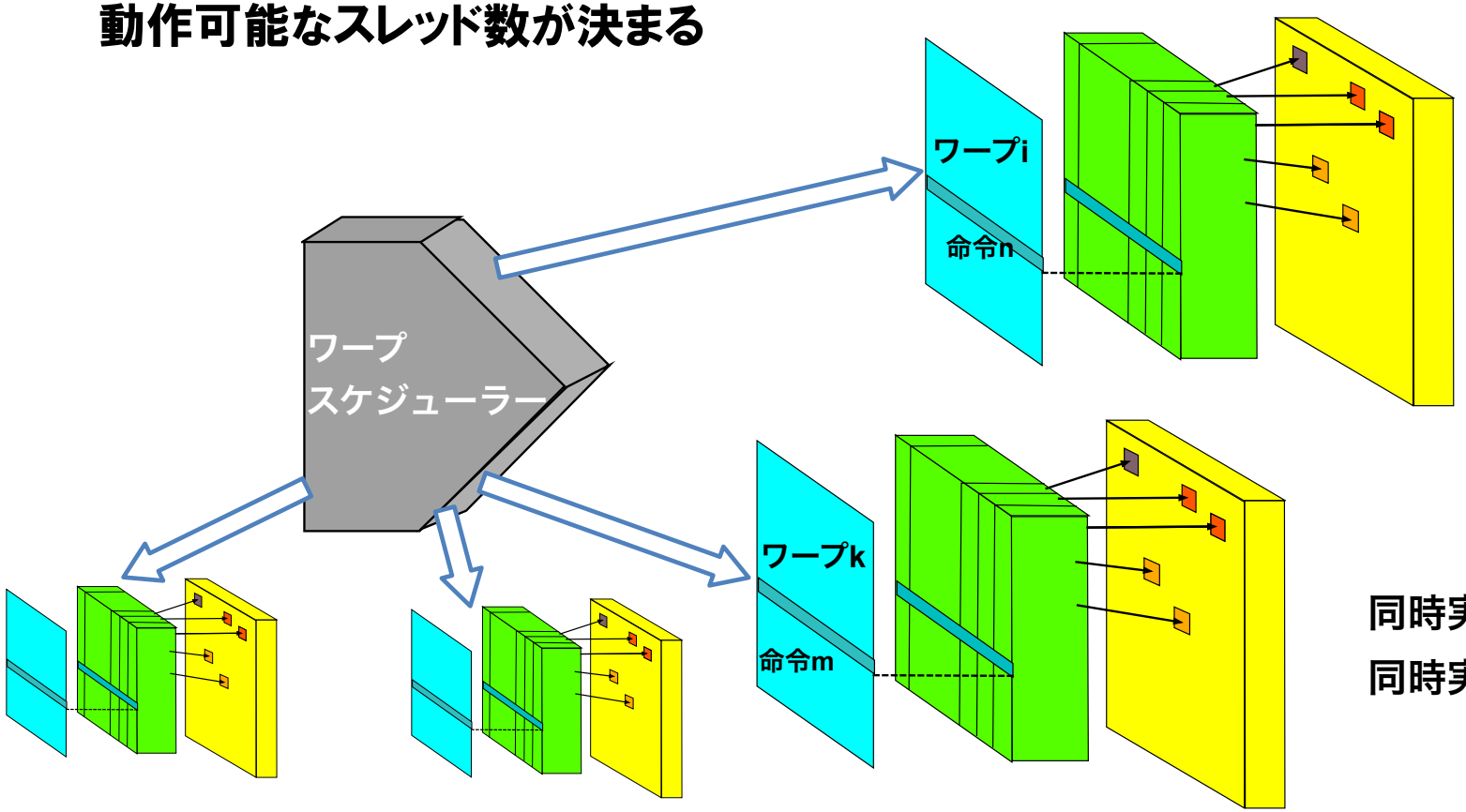


- ワープスケジューラーはあるワープの待ち時間に他のワープの命令を発行する
- 待ち時間になるべく多くの命令が実行されている状態を作り出せると効率がいい

命令スケジューリングが非効率（GPU）

waveZ

- スケジューリングされた同時動作可能なワープ数により同時動作可能なスレッド数が決まる
- GPUのレジスタの物量は多いがそれぞれのスレッドで使用するレジスタ量によっても同時に動作可能なスレッド数が決まる

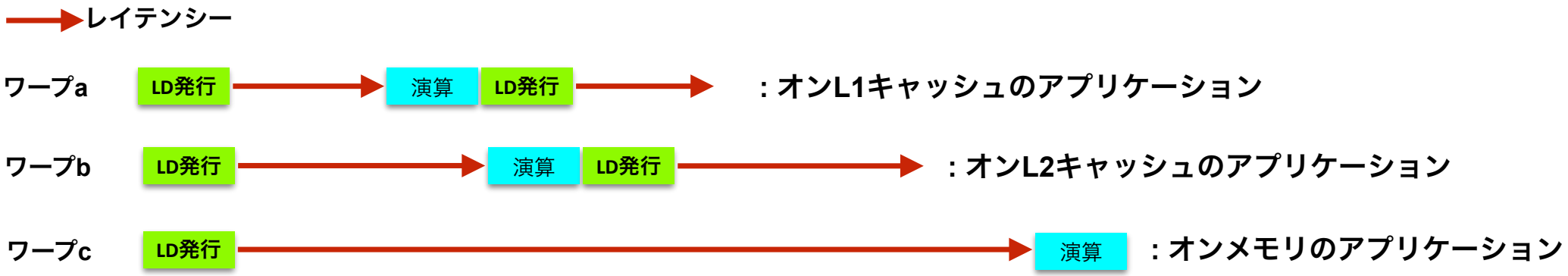


同時実行可能なワープ数：4

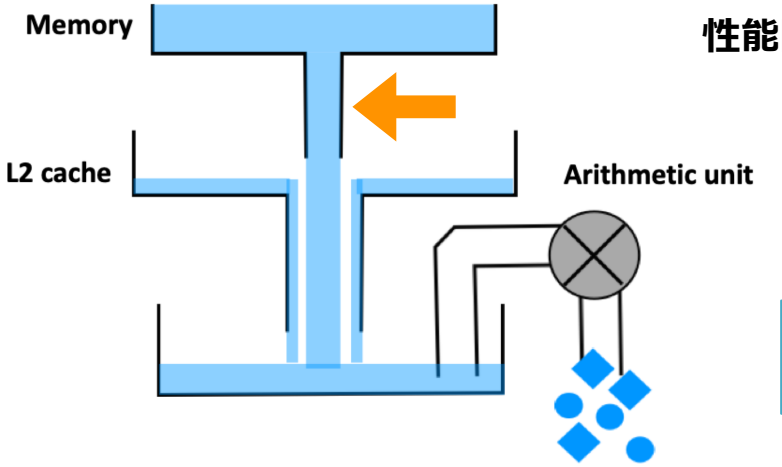
同時実行可能なスレッド数：128

命令スケジューリングが非効率（GPU）

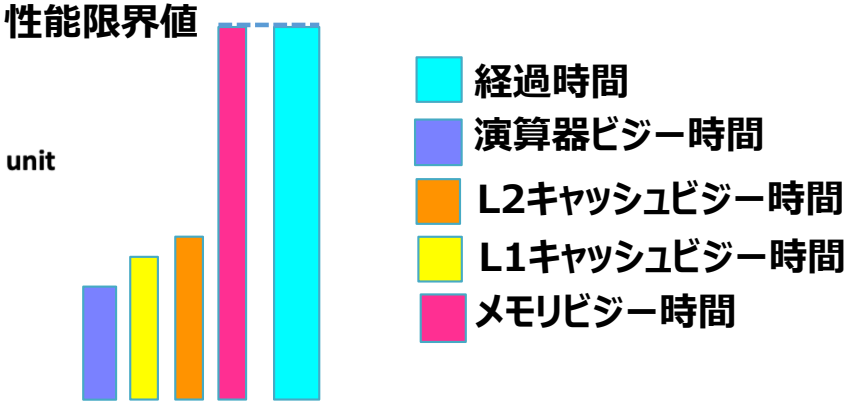
- 同時動作可能なスレッド数が少ないと命令スケジューリングが非効率となる場合がある
- 同時動作可能なスレッド数が十分かどうかはそのスレッドがオンメモリ・オンキャッシュ・オンレジスタかどうかによって決まる
- オンメモリのスレッドはメモリレイテンシーを隠すために同時動作可能なスレッド数が多く必要になる



メモリ律速なアプリケーションの性能見積もり



経過時間＝メモリビジー時間



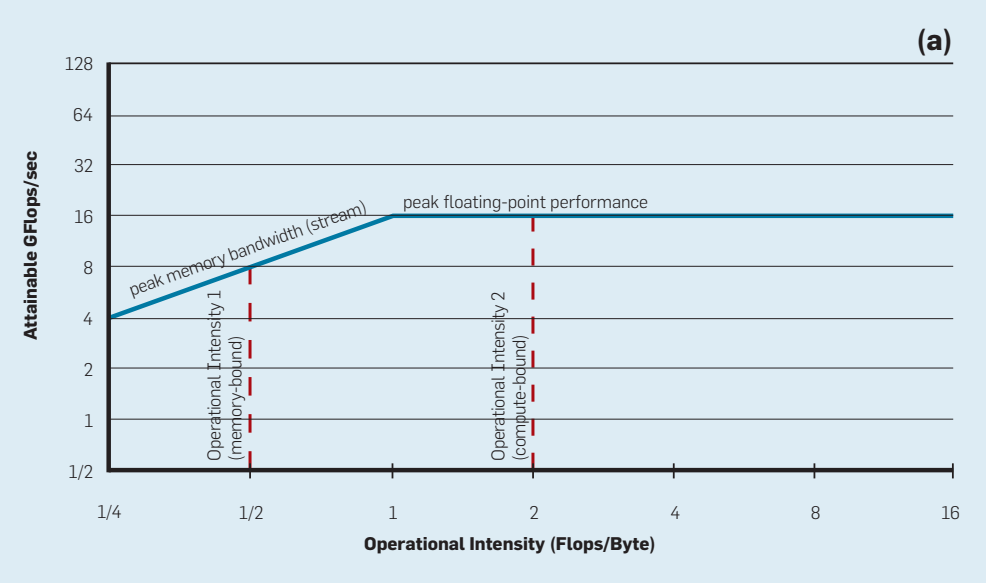
メモリデータ移動時間：t_M=Mdata/Mband

L2データ移動時間：t_{L2}=L2data/L2band

最小演算時間：t_C=Nc/P_{peak}

実行時間：$t_E = \max\{t_M, t_{L2}, t_C\}$

ルーフラインモデル



S. Williams, A. Waterman, and D. Patterson: Roofline: an insightful visual performance model for multicore architectures. Commun. ACM, 52:65–76, 2009.

- ハードウェアの実効的なメモリバンド幅：B
- ハードウェアのピーク性能：F
- アプリケーションの演算強度：X=f/b
  - アプリケーションの要求フロップス値：f
  - アプリケーションの要求バイト値：b

アプリケーションの性能：

$$\min(F,\ B*X)$$

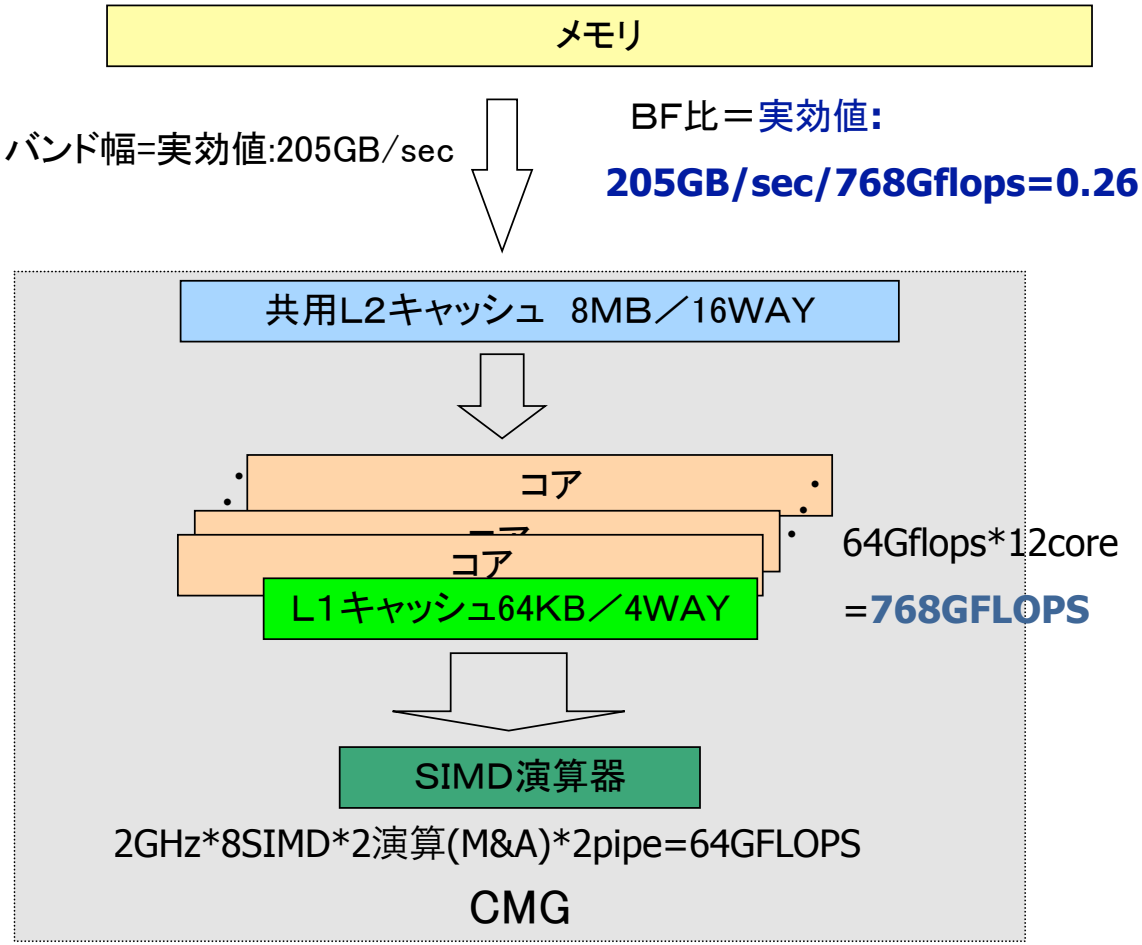
$$B*X=B*(f/b)=B*F*f/(b*F)=(B/F)/(b/f)*F$$

アプリケーションのピーク性能比 ：$\min(1.0,\ (B/F)/(b/f))$

ハードウェアのB/F値をアプリケーションのb/f値で割る.

ルーフラインモデルによる性能見積り

ベースとなる性能値(富岳1CMGの場合)



ルーフラインモデルによる性能見積り

メモリとキャッシュアクセス（1）

配列宣言(3610,-10:70,168)
N1=3610,N2=60,N3=168

```
do k = 1,N3
  do j = 1,N2
    do i = 1,N1
        a(i,j,k) = c(i,j-1,k)+c(i,j,k)*c(i,j+1,k)
    enddo
  enddo
enddo
```

2M 1M 1F 1L2 1F 1L2

要求b/f	24/2=12
性能予測	0.26/12 = 0.021%
実測値	0.022

アプリケーションのピーク性能比の予測値
：ハードウェアのBF値(実効値)/アプリの要求bf値

要求byteの算出:

1store,2loadと考える

8x3 = 24byte

要求flop:

add：1+mult：1 = 2

まとめ

- 並列性能上のボトルネック
- 高並列化手法の手順(簡単な実例・RSDFTの例)
- スレッド並列・スレッド性能
  - CPUのスレッド並列
  - GPUのスレッド並列
  - アプリケーションのタイプ- 性能の観点から -
  - メモリ・L2 キャッシュ・演算器を考慮した性能モデル
  - CPU性能解析レポート(ビジー時間)
  - ビジー時間から見たアプリケーションのタイプ
  - 性能から見たアプリケーションの分類
  - ビジー時間から見たアプリケーションタイプと性能から見たアプリケーションの分類の対応
  - 性能チューニングとは？
  - ルーフラインモデル
  - ルーフラインモデルによる性能見積り