物性第一原理計算プログラムの高速化

東京大学
中村研究室所属
修士一年　大根田　拓

発表の流れ

- プログラム全体の構成
- FFT計算の高速化
  - オリジナルのアルゴリズム
  - キャッシュ最適化
- 行列積の高速化
  - オリジナルのアルゴリズム
  - キャッシュ最適化
- SCIMAでの評価(現状)

第一原理計算のプログラム

- 第一原理計算:シュレディンガー方程式を元に価電子の挙動を計算
  → 直接対象物質(ここでは水素)の物性を調べる
- 計算法にCG(Conjugate Gradient)法、FFTを用いる
- 問題サイズが大きく、iteration回数が多いため、計算時間がかかる

プログラム構造

- 基本構造

```
Do j =1,itcf
  do k = 1,nbnd
        call hlocal
  enddo
  call diagk
enddo
```

hlocal: FFT計算
diagk: 行列積計算
(nbnd=191,itcf=8)

表記していないところは
ベクトルの内積等の
1次元配列演算が中心
→　現時点では対象外

オリジナルでは、nbnd=itcf=1のとき
トータル370秒程度に対し
hlocal55秒、diagk300秒

プログラム高速化基本戦略

- オリジナルはベクトル計算機用に最適化
  - スカラ計算機向けプログラムに修正
- FFT計算部分、行列積部分において計算時間大
  - 主にFFT計算と行列積を対象に高速化

FFT計算部(hlocal)

- オリジナルのアルゴリズム

1. 一方向のバタフライ演算を,直交する面単位でまとめて行う(基数2から5の間で行う)
2. 配列全体の転置を行う

- 1,2を3回繰り返し、演算後再度3回繰り返す

FFT計算部(hlocal)

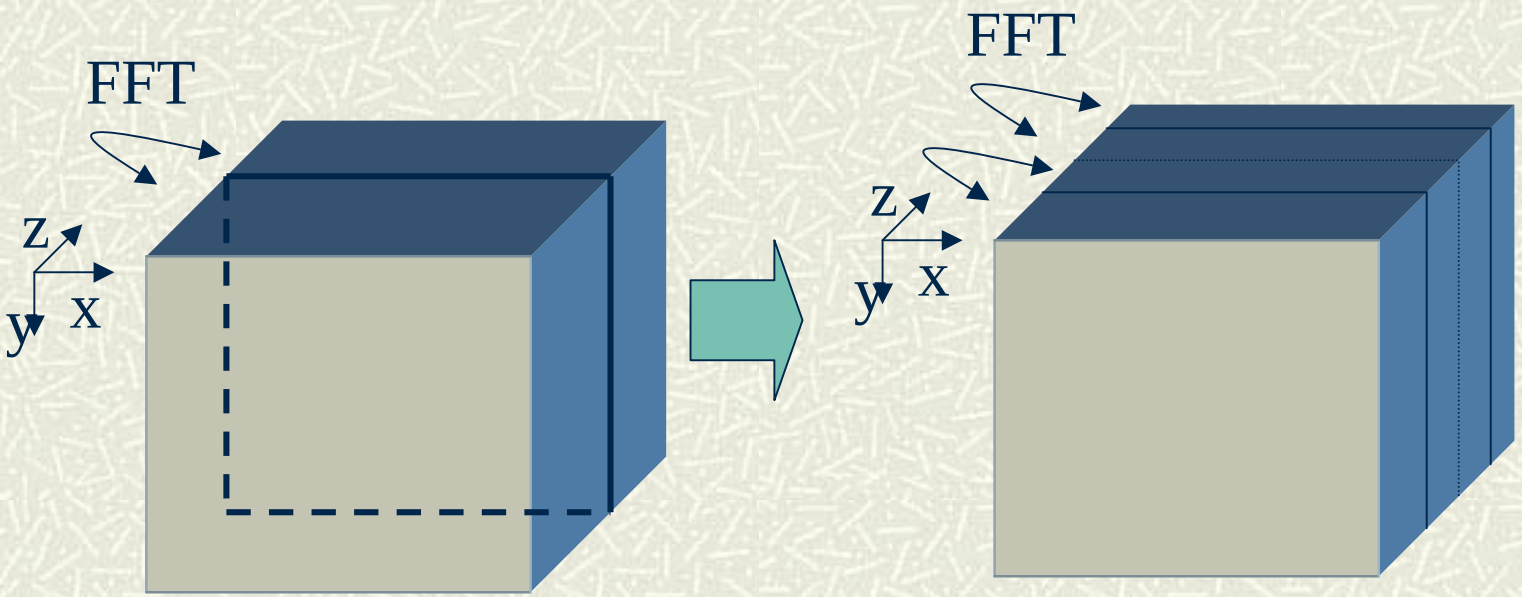
- オリジナルのアルゴリズム

1. 一方向のバタフライ演算を,直交する面単位でまとめて行う(基数2から5の間で行う)
2. 配列全体の転置を行う

- 1,2を3回繰り返し、演算後再度3回繰り返す

オリジナルのFFT計算アルゴリズム(1)

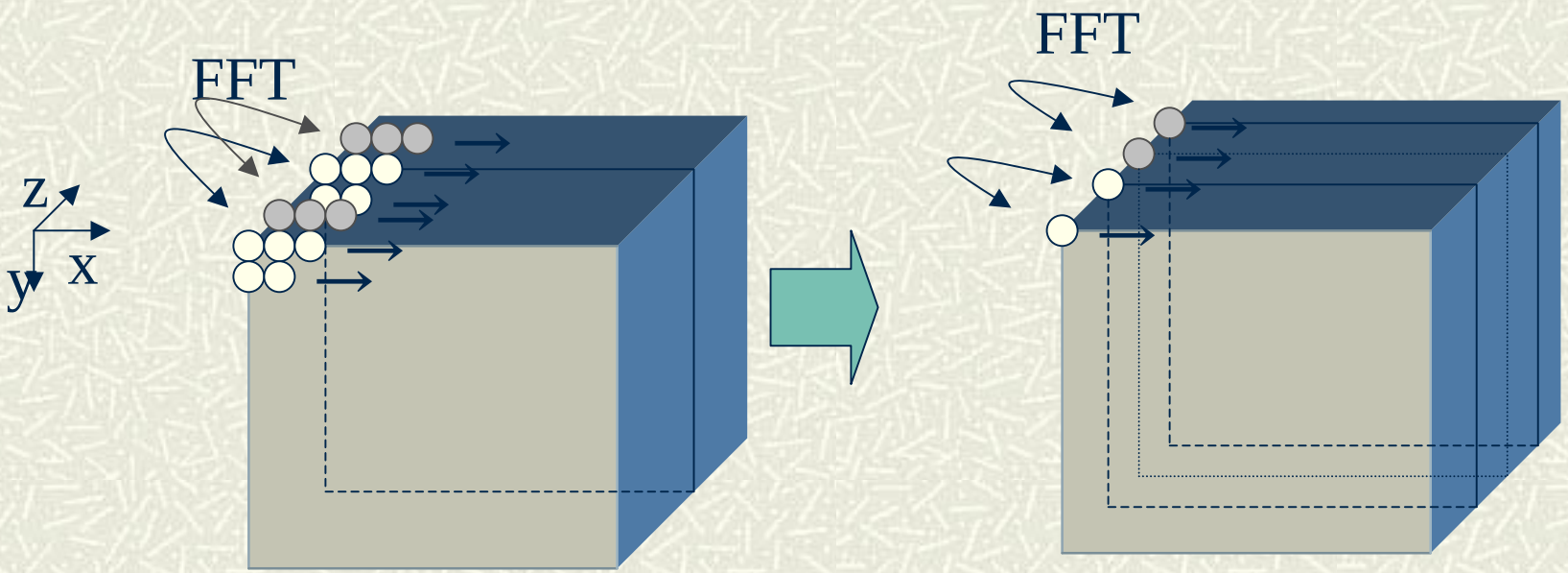
- (例:基数2で2回展開する場合)



面単位で図のz軸方向にバタフライ演算を行う
(基数2から5の間で行う)

オリジナルのFFT計算アルゴリズム(1)

- (例:基数2で2回展開する場合)



面単位で図のz軸方向にバタフライ演算を行う
(基数2から5の間で行う)

FFT計算部(hlocal)

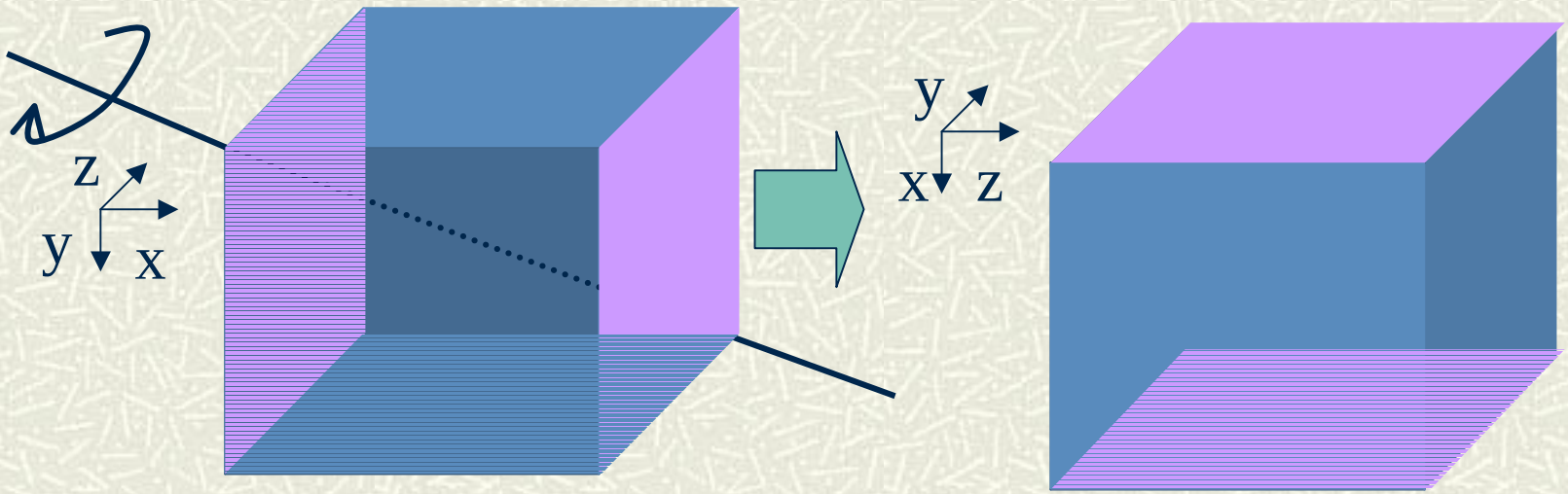
- オリジナルのアルゴリズム

1. 一方向のバタフライ演算を,直交する面単位でまとめて行う(基数2から5の間で行う)
2. 配列全体の転置を行う

- 1,2を3回繰り返し、演算後再度3回繰り返す

オリジナルのFFT計算アルゴリズム(2)

- オリジナルのアルゴリズム



1つの面を計算後、配列全体を転置する

→ 1と2を3回繰り返し、演算をはさんでさらに3回繰り返す

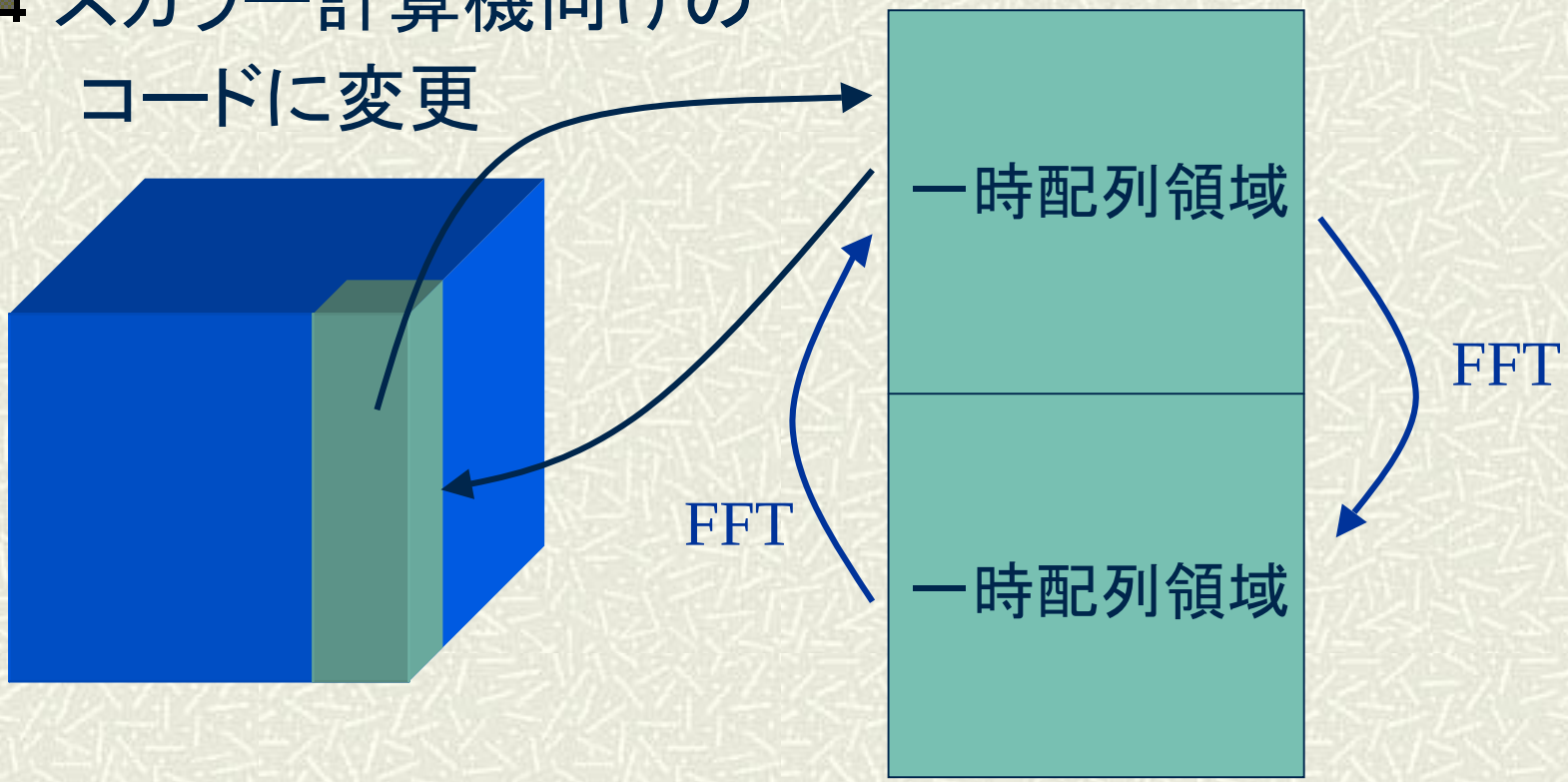
オリジナルプログラム(FFT計算)の問題点

- 配列全体がキャッシュ上に載りきらない場合

ある面についてのバタフライ演算1回、
あるいは転置の度に、配列全要素にアクセスする
　　→　データがキャッシュ上で再利用されない

FFT計算(変更後)

- スカラー計算機向けの
  コードに変更



配列の一部を一時配列にコピーしてFFT計算
全方向について行う

FFT計算部のキャッシュ最適化結果

評価条件:
SGI MIPS R12000 (300MHz)
L1 cache : 32kB
L1 cache ラインサイズ: 64B
L2 cache : 1MB
L2 cache ラインサイズ : 64B

- 110*110*110の3次元配列を対象に評価(performance counter)

	cycle	L1 cache miss
オリジナル	13208×10^6	102×10^6
変更後	1867×10^6	14×10^6

ともにオリジナルに対し15%程度まで減少

行列積部分の計算

複素数の行列積を計算

1. a[x,y] = p[z,x]・hp[z,y]

　　－（ 配列vの生成 ）－

2. p[z,x] = p[z,y]・v[y,x]
3. hp[z,x] = hp[z,y]・v[y,x]

(x = y = 191, z = 13052)

- オリジナル：単純な行列積計算
  → 配列全体がキャッシュに載りきらない場合、
  データが再利用されない
- 1で生成したaを用いて2の前でvを生成し、2と3で利用
  → 1と2,3をあわせて計算することはできない

行列積高速化戦略

1. a[x,y] = p[z,x]・hp[z,y]

   ー（ 配列vの生成 ）ー

2. p[z,x] = p[z,y]・v[y,x]
3. hp[z,x] = hp[z,y]・v[y,x]

(x = y = 191, z = 13052)

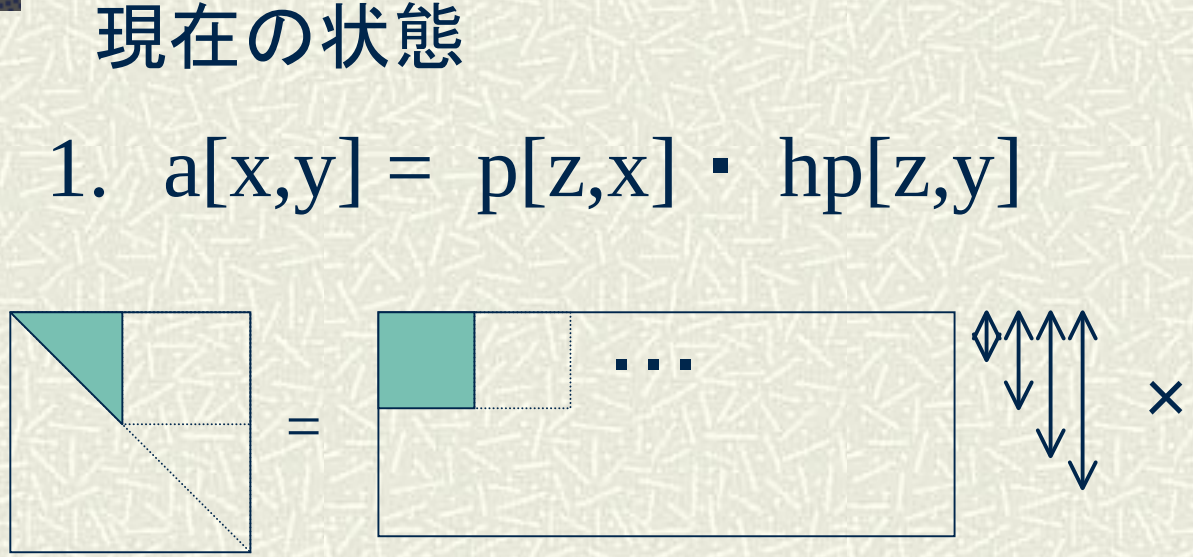
データの再利用性を高める

- キャッシュブロッキングおよびレジスタブロッキング
- 2と3をあわせて計算(vの再利用性向上)

行列積(1)

現在の状態

1. a[x,y] = p[z,x] ・ hp[z,y]

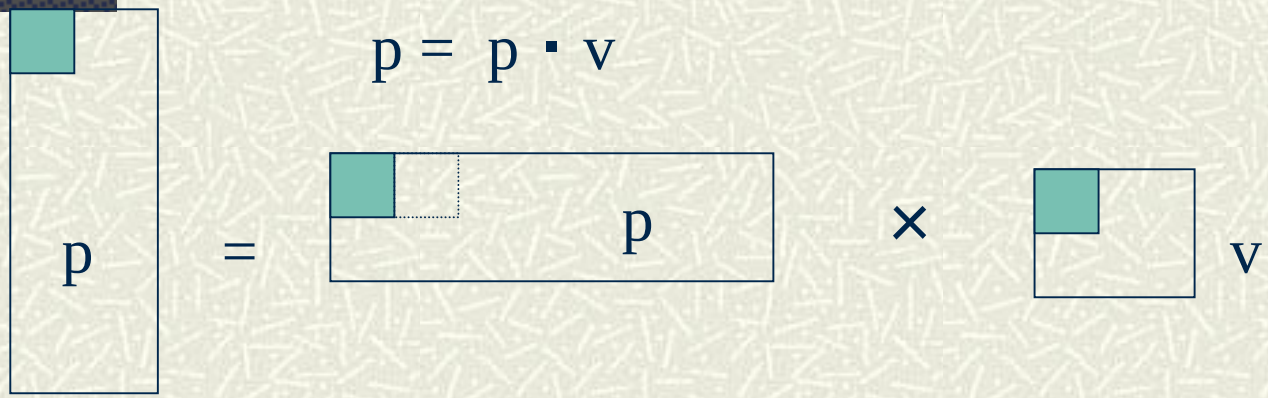


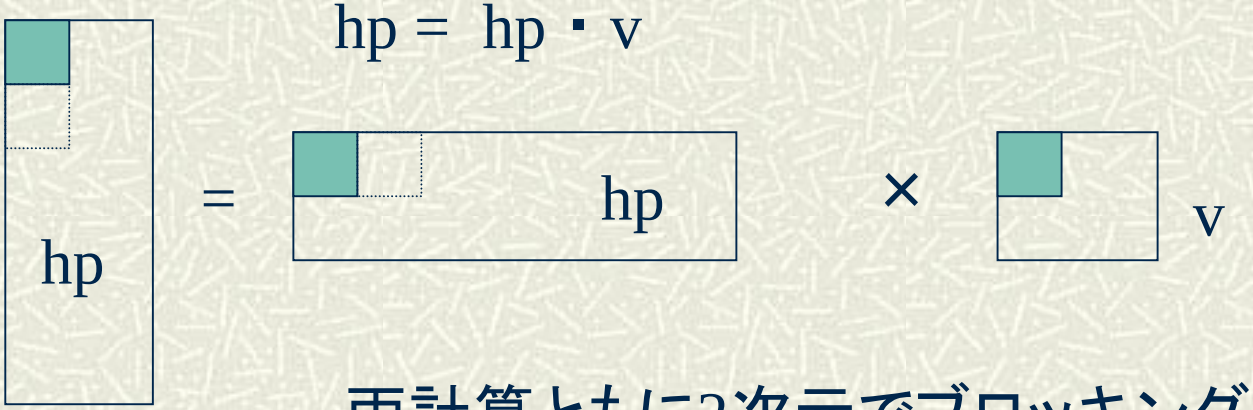
a　　p （横に連続）　　hp （縦に連続）

配列aの上三角部分のみ計算

→ pとhpは再利用性に偏り
現時点では2次元でブロッキング

行列積(2,3)





- 両計算ともに2次元でブロッキング
- vを共通して使うのであわせて計算

行列積

- [1]:データの再利用性に偏り
- [2,3]:2回の行列積の間でvを再利用する

全ての行列積共通:

複素数３＊３でレジスタブロッキング

行列積最適化:評価条件

評価条件:
SGI　MIPS　R12000 (300MHz)
レジスタ: int 32, float 32
L1 cache : 32kB
　　ラインサイズ: 64B
L2 cache : 1MB
　　ラインサイズ : 64B

行列積キャッシュ最適化の評価

＊ Performance counterで評価

	cycle	L2 cache miss
オリジナル	84912×10^6	307×10^6
変更後	9467×10^6	9×10^6

“変更後”のコード:

(1) L2キャッシュを対象にブロッキング

① 行列積1: ブロックサイズをa:100×100、
p & hp:100×276として計算

② 行列積2 & 3: ブロックサイズをv:100×100、
p & hp:100×128とし、あわせて計算

(2) 複素数3*3でレジスタブロッキング

第一原理計算プログラムの キャッシュ最適化結果(現状)

SGI R12000(300MHz)で評価(cycle)

- hlocal: $13208 \times 10^6 \rightarrow 1867 \times 10^6$
- diagk: $84912 \times 10^6 \rightarrow 9467 \times 10^6$ (約)
- 変更後、全体に対し、

hlocal(FFT) : $191 \times 1867 = 356 \times 10^9$

diagk(行列積): $8 \times 9467 = 75 \times 10^9$

→ 7倍前後、対象物質によっては逆転の可能性?

SCIMAでの評価

- SCIMAの場合：

  FFT･行列積ともに、

  - 配列間干渉がなくなる
  - 大粒度転送

  による高速化が見込める

→ シミュレータで評価

現状： FFT計算部分のみ

SCIMAでの評価環境

FFT計算部分(hlocal)を評価

- 評価環境
  1. 40×40×40の3次元配列で評価
  2. (現時点)キャッシュ最適化との比較
  3. L1キャッシュ32k(連想度:4)のモデルと、L1キャッシュ16k(連想度:2)、OCM16kのモデルを対象に比較
  4. ラインサイズ:32B,64B,128B,256Bで評価
- パラメタ

  レジスタ:　int 32, float 32
  同時発行命令:int 2, float(mul) 1, float(div) 1, load/store 1
  オフチップメモリスループット: 8 Byte/cycle
  オフチップメモリスループット: 40 cycle

シミュレーション時の評価環境

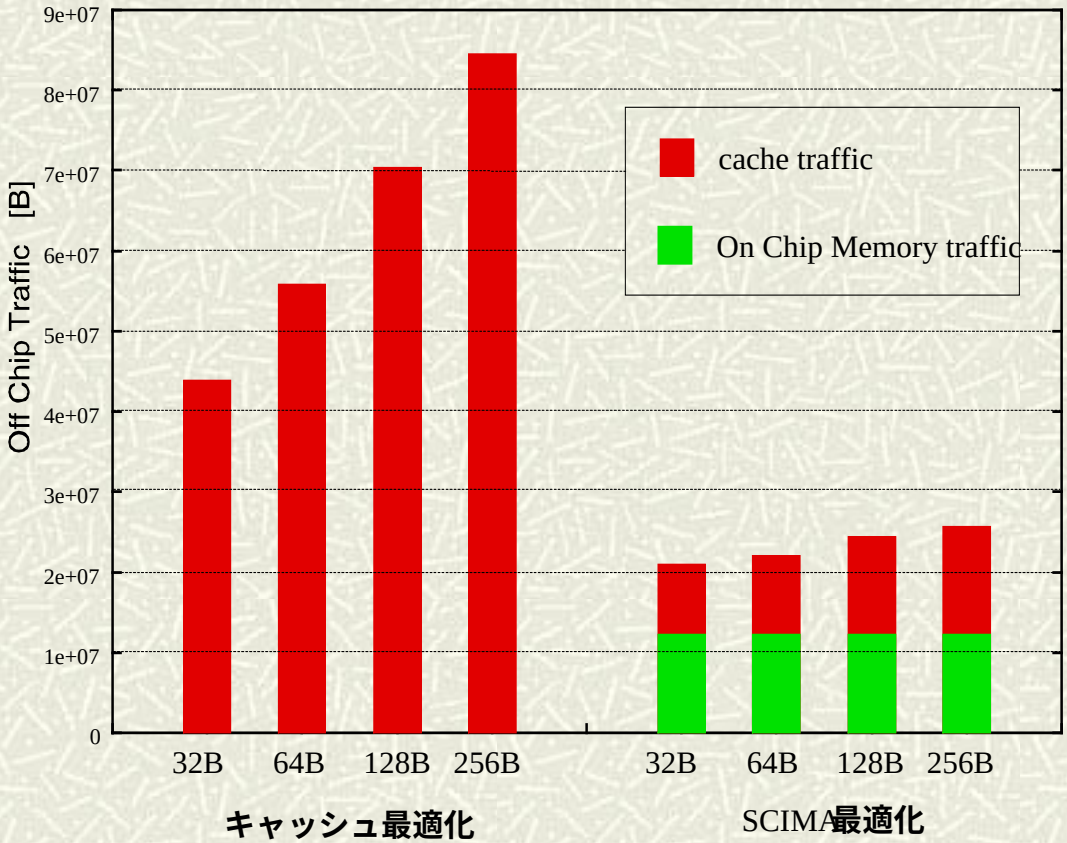
simulation時に、

- アクセスレイテンシ0　スループット∞　のメモリ(perfect memory)
- アクセスレイテンシ40　スループット∞のメモリ
- アクセスレイテンシ40　スループット8のメモリ(normal memory)

で評価

→ CPU動作時間(CPU busy time)、アクセスレイテンシが原因のストール(Latency stall)、スループット不足が原因のストール(Throughput stall)に切り分ける

SCIMAでの評価結果(traffic)

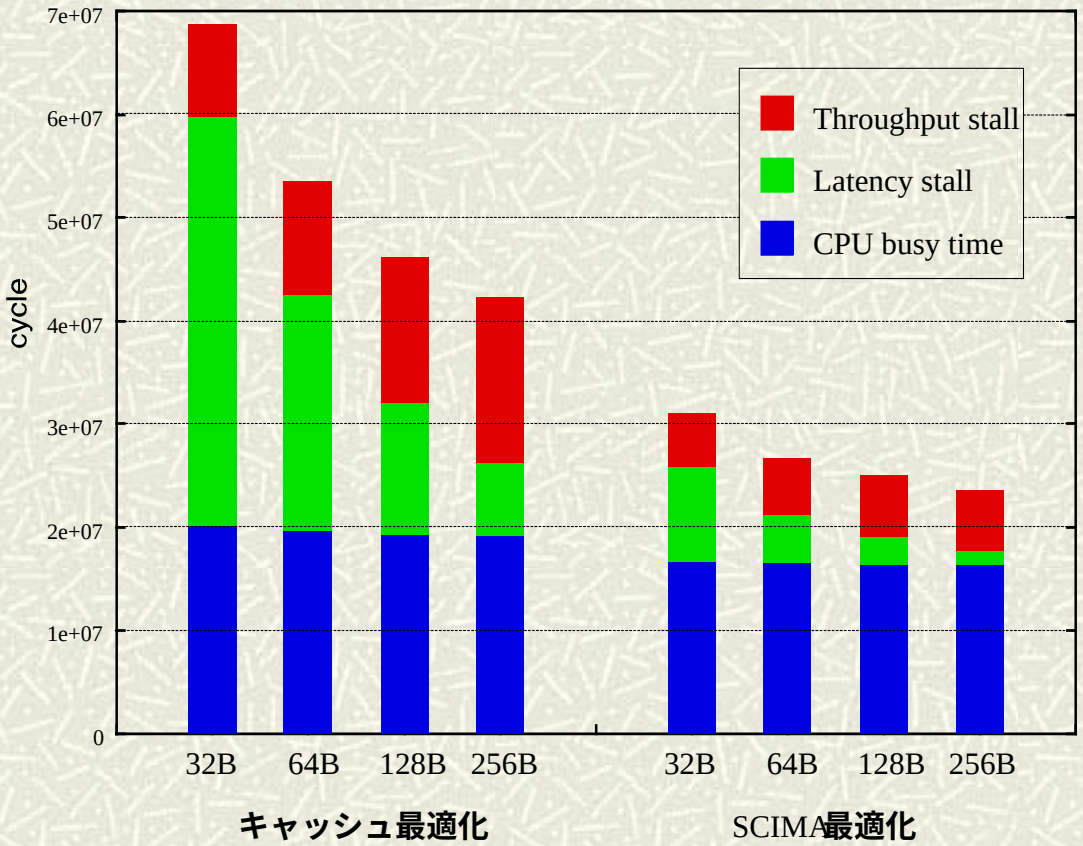


NASPB FTと同様の傾向

キャッシュ最適化
- 配列間干渉が起こる
- ラインサイズ 大 → 無駄なライン転送

SCIMA最適化
- 配列間干渉なし
- OCMでのトラフィックは、ラインサイズによらず一定

SCIMAでの評価結果(cycle)



NASPB FTと同様の傾向

キャッシュ最適化
- ラインサイズ小 → 初期レイテンシの影響 大
- ラインサイズ大 → トラフィック増大

SCIMA最適化
- トラフィック 小
- 大粒度転送により初期レイテンシ 小

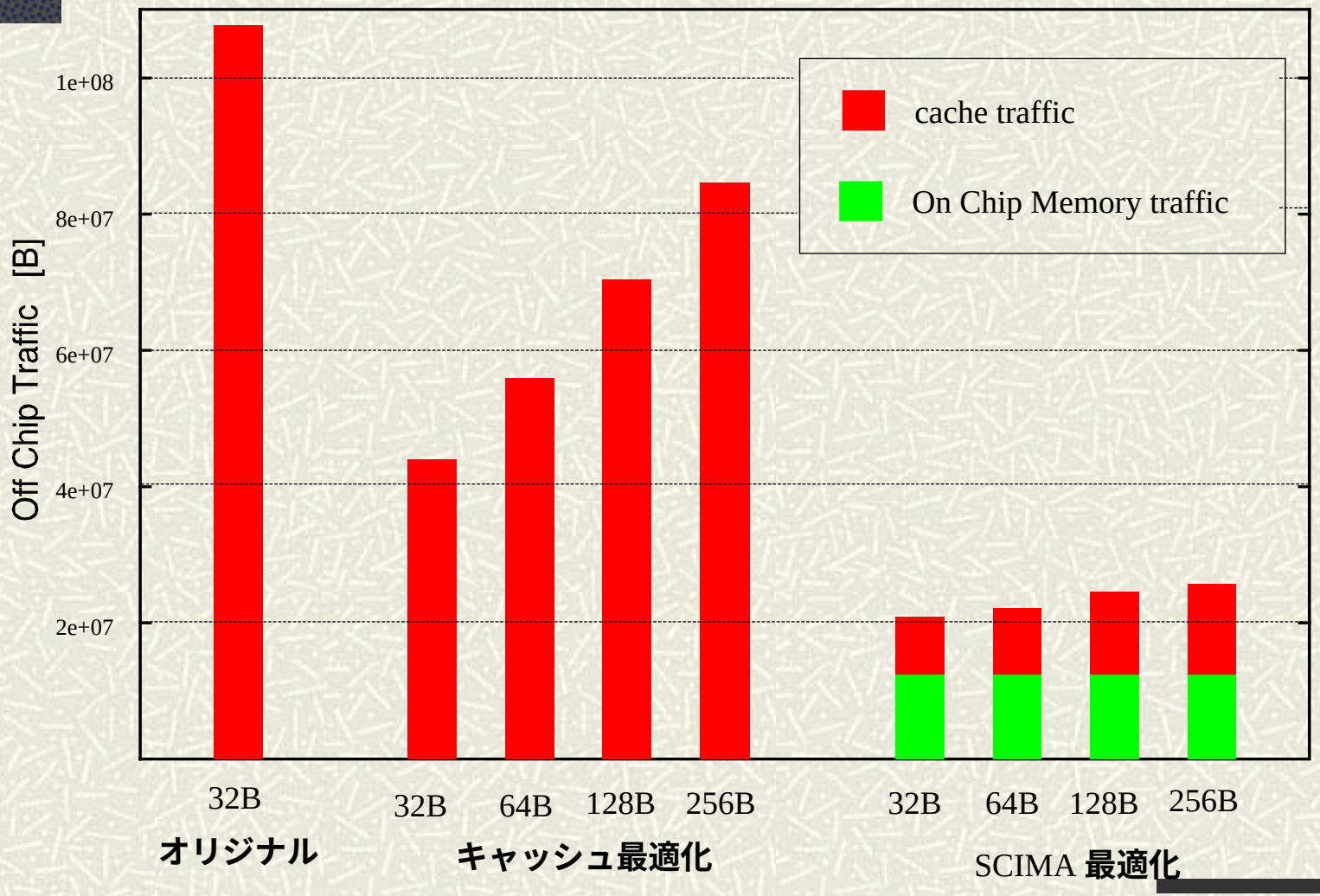
結果の考察(現状)

- 全体:NPB FTと同様の傾向
- SCIMA最適化結果にラインサイズの影響
  → 評価にFFT以外の計算が含まれている
  ことが影響(検討中)
- その他、詳細は検討中

今後の課題

- SCIMAでのFFT計算部評価の詳細
- SCIMAでの行列積部分の評価
- 第一原理計算プログラム全体での評価

付録: SCIMAの評価(対original:traffic)



付録:SCIMAの評価(対original:cycle)