NASPBを用いたSCIMAの評価

中村研究室修士2年

岩本　貢

発表の流れ

- 本研究の目的
- 評価条件
- NASPB　CG
  - 最適化の戦略
  - 結果，考察
- NASPB　FT
  - 最適化の戦略
  - 結果，考察
- NASPB　MG
  - 最適化の戦略
  - 結果，考察
- まとめ

本研究の目的

- HPC向けアーキテクチャSCIMA の有効性の検討
- NAS Parallel Benchmarks, CG,FT,MG を例題とし
  - Cache による最適化
  - SCIMA による最適化

  を比較する.

NASPB の概要

- 数値計算の代表的なベンチマーク.
  - CG : Conjugate Gradient (共役勾配)法を用いた大規模疎行列の固有値計算
  - FT : 3次元FFTおよび逆FFTを用いた偏微分方程式の数値計算
  - MG: Multi Grid 法を用いた3次元ポアソン方程式の数値解の計算
- 評価には Serial 版を使用.

評価条件

- パラメタ
  - 同時発行命令数: Int=2,　FP(積和，除算/平方，各) = 1
  - データキャッシュ: 32kB, 4way, non-blocking L1 cache
    - OCM使用時: 4way cacheのうち 2way をOCM
      2way 16kB cache & 16kB OCM
    評価に用いる class W のデータセットがキャッシュに載らない
  - load, store スループット：8B/cycle (cache,OCMのtotal)
  - オフチップメモリスループット: 4B / cycle
  - オフチップメモリレイテンシ: 40 cycle
  - line サイズ：32B, 64B, 128B, 256B と変化
  - page サイズ : 4kB　<= ラインと比較し粒度が大
- 命令キャッシュ: all hit，分岐予測: perfect
- リザベーションステーションをもつout -of -order実行

評価方法

- MIPS IV に次の命令を追加して評価
  - page-load：メインメモリからOCMへの転送
    page-store：OCMからメインメモリへの転送

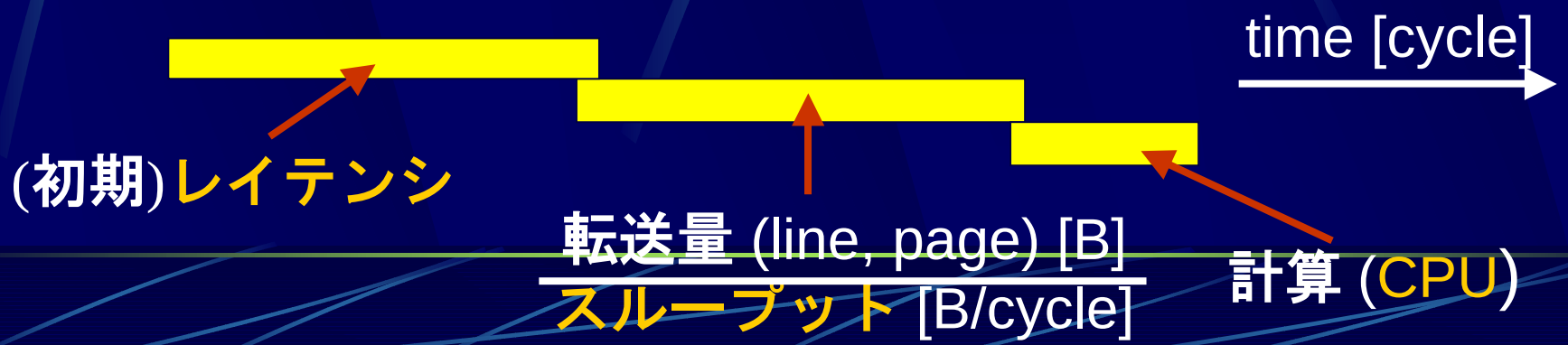
→ 命令は関数としてユーザがソースレベルで挿入

- 評価
  - 評価時間を考慮し，class W のデータセットを選択
  - クロックレベルシミュレータによる評価
    - オフチップトラフィック
    - 所要サイクル数

評価内容

1. オフチップトラフィック
2. 所要サイクル数
   - C_{norm} = 評価条件でのシミュレーション
   - C_{thinf} = 評価条件で Off-Chip throughput = ∞ にした評価
   - C_{ideal} = 評価条件で Off-Chip throughput = ∞, Off-Chip latency = 0 にした評価

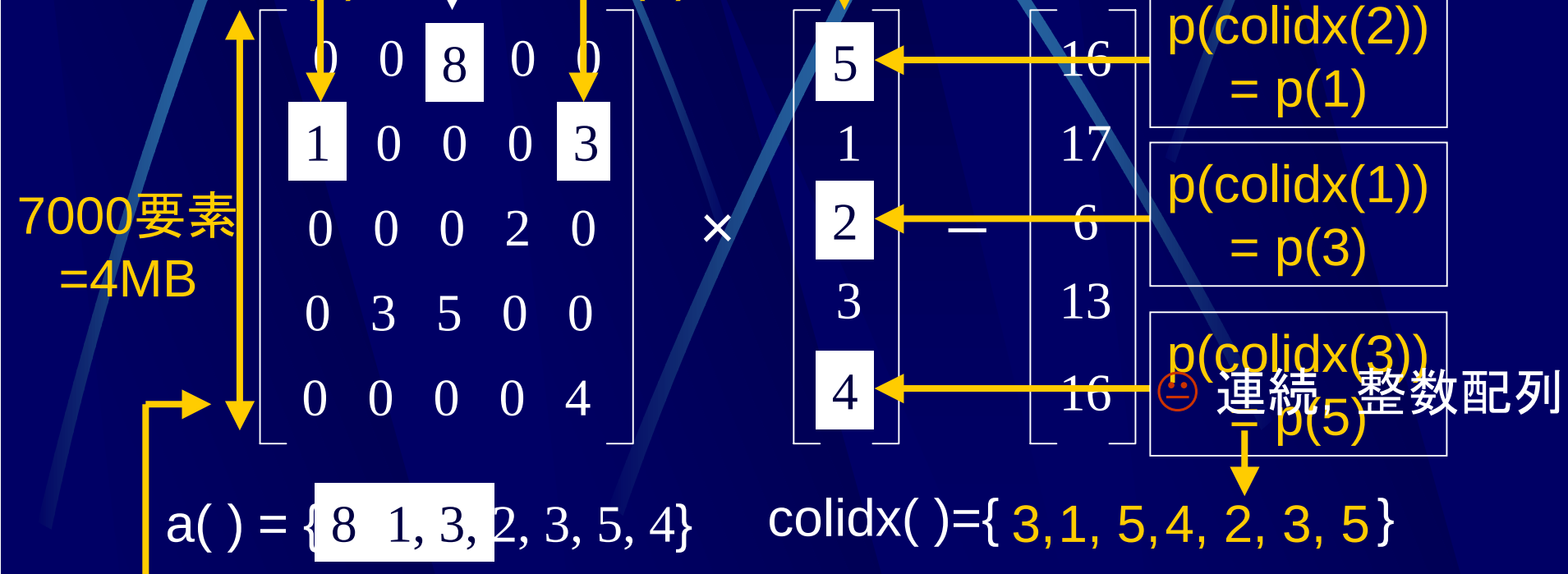
- throughput stall = C_{norm} — C_{thinf}: スループット不足による待ち時間
- latency stall = C_{thinf} — C_{ideal}: 初期レイテンシによる待ち時間
- CPU busy time = C_{ideal}: 計算時間



NASPB CG

- 計算時間の大半: q(k) = a(k) ＊ p(colidx(k))

a( ): 疎行列：非ゼロ率 1%　☹ p( ): 間接ランダム参照，再利用性あり

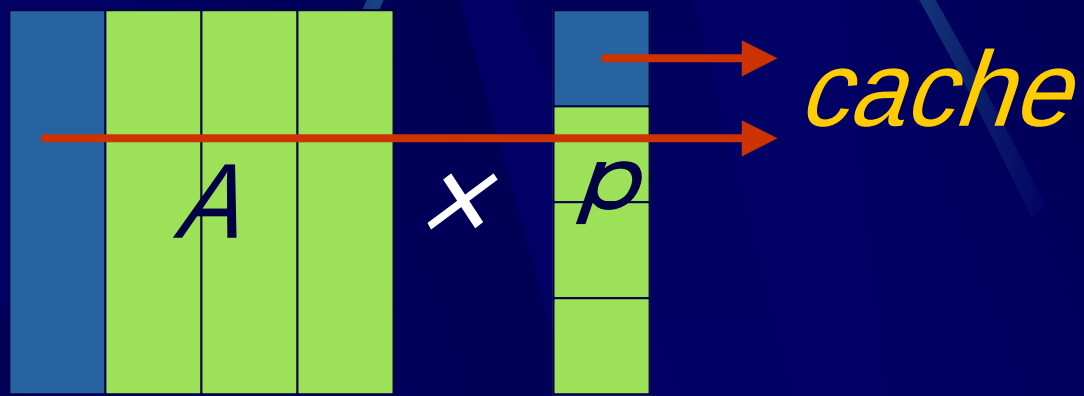


☹ a( ): データサイズ大，連続，再利用性なし

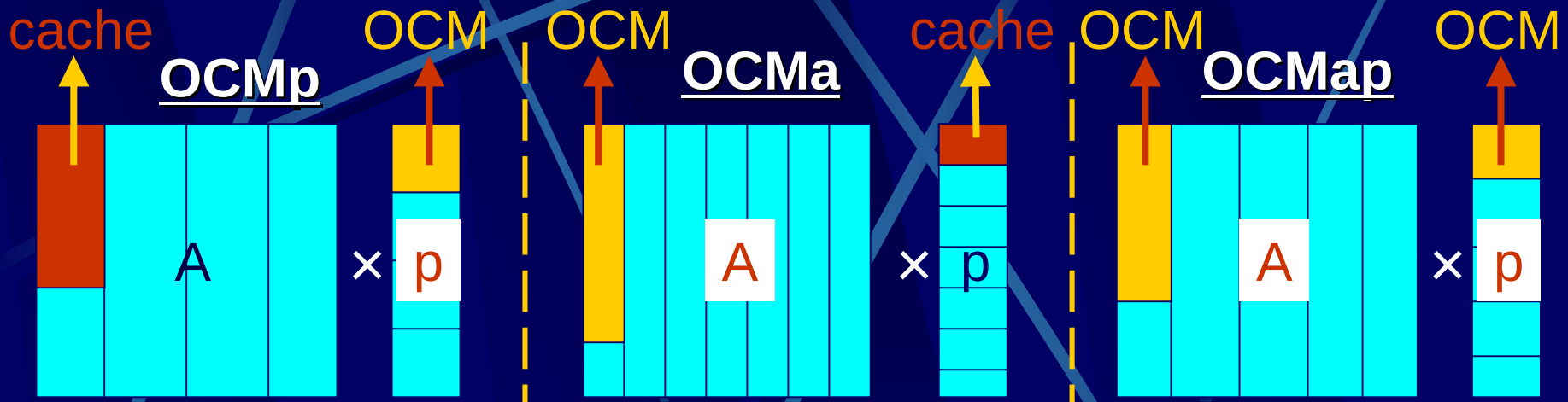
😐 連続，整数配列

キャッシュ最適化

- キャッシュブロッキング
  - 行列a, ベクトルpをブロックに切り, p の再利用性を高める
    → a, pのコンフリクトが生じる可能性あり
  - cache 1way に p を載せるのが最適:4分割



SCIMA 向け最適化



☺再利用性のあるpがOCM上で干渉なく再利用

➡pをできるだけOCMへ => 4分割

☹連続アクセスされる配列aのライン転送

☺aの大粒度転送
レイテンシの影響小

☹pのキャッシュ上での干渉(キャッシュ2way)

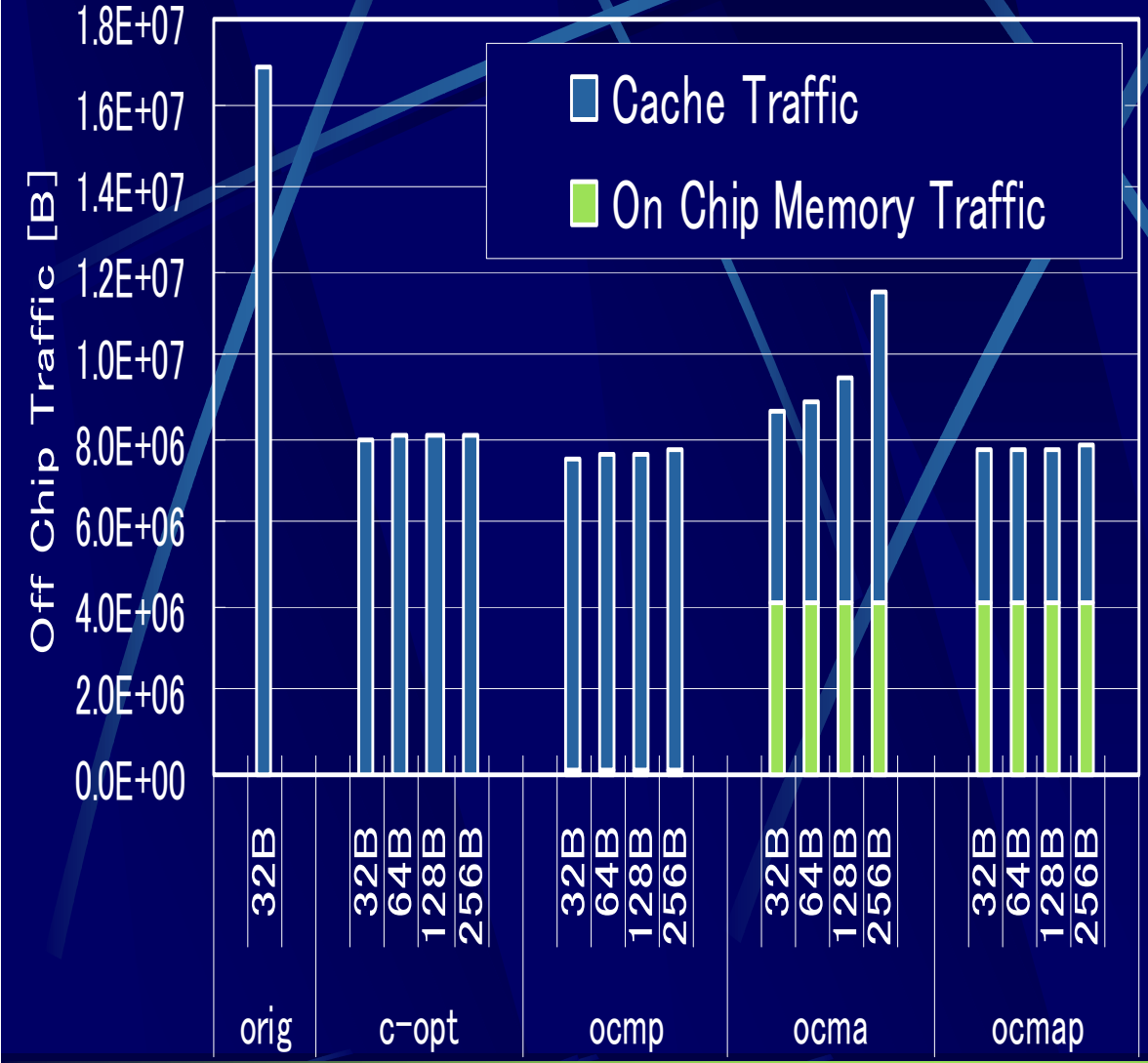
➡pのブロックを小さく => 7分割

☺OCMp, OCMaの利点が共に活かせる

➡ a を1Page, 残りをp => 5分割

☹キャッシュがあまり使われない

CGのオフチップトラフィック



- original と c-opt の比較

  c-opt

  ☺ブロッキングの効果あり

- c-optとSCIMAの比較

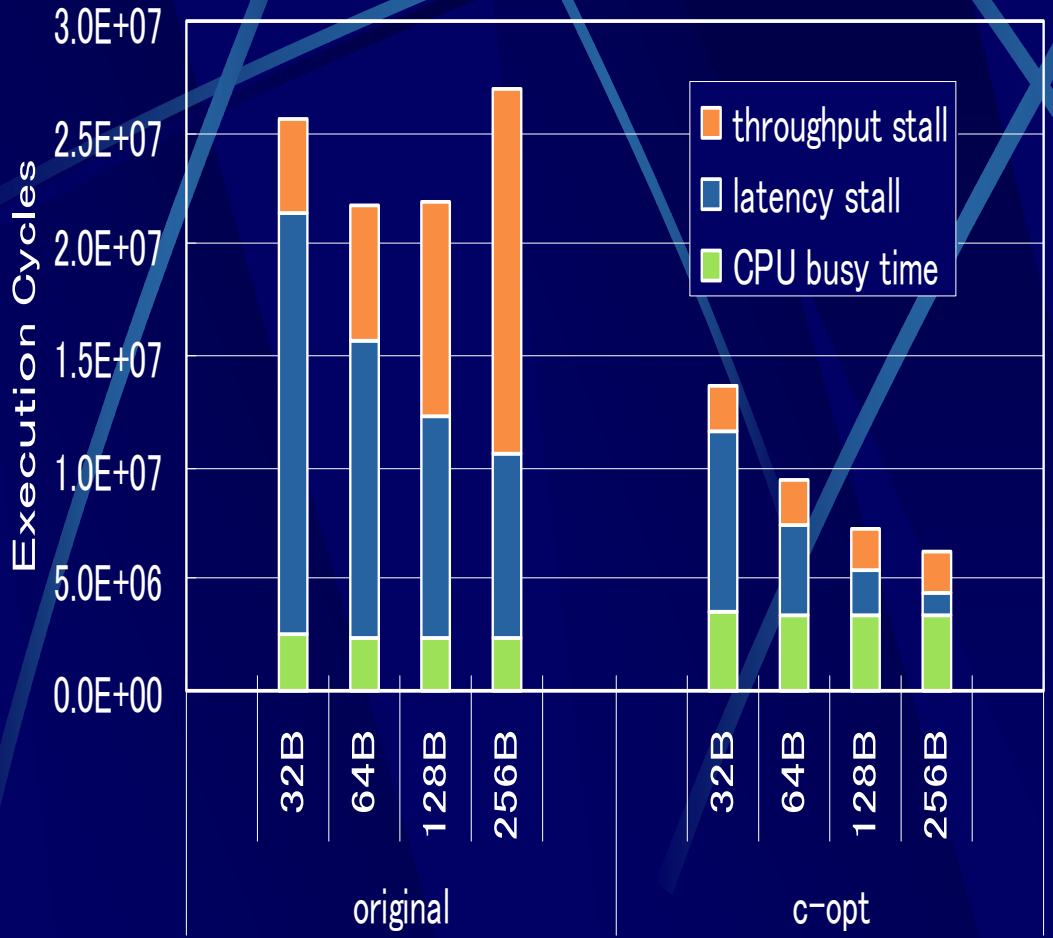
  OCMp/OCMap

  ☺配列間干渉減
  トラフィック小

  OCMa

  ☹キャッシュ上での干渉

CGのサイクル数



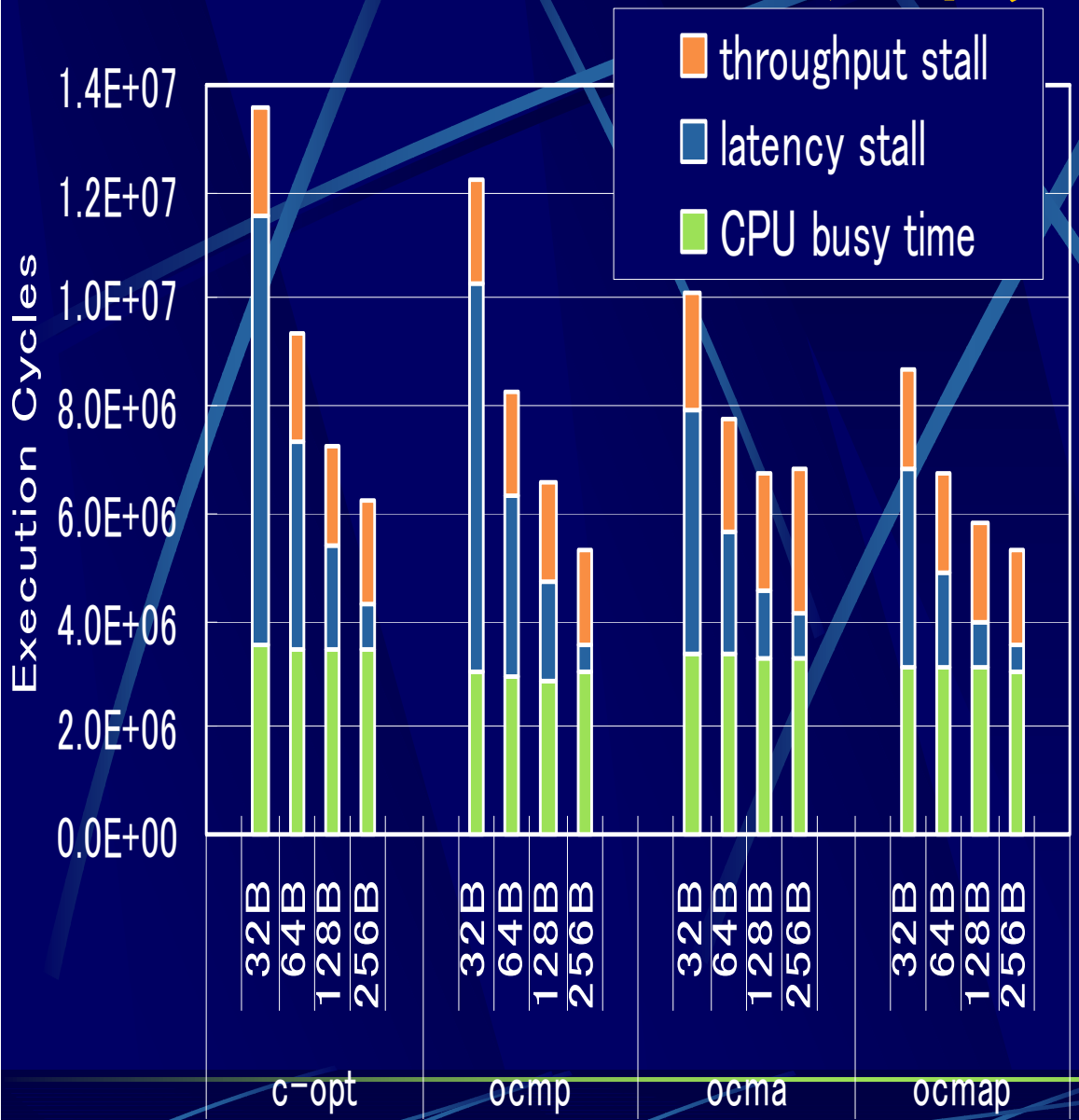
- original と c-optの比較

c-opt

☺ブロッキング効果あり
- throughput stall 減
- latency stall 減

☞SCIMAでさらに
throughput stall
latency stall
が減るか？

CGのサイクル数



- c-optとSCIMAの比較

OCMp

☺throughput stall 減少

☹latency stall は多い

OCMa

☺latency stall減少

☹throughput stallは増加

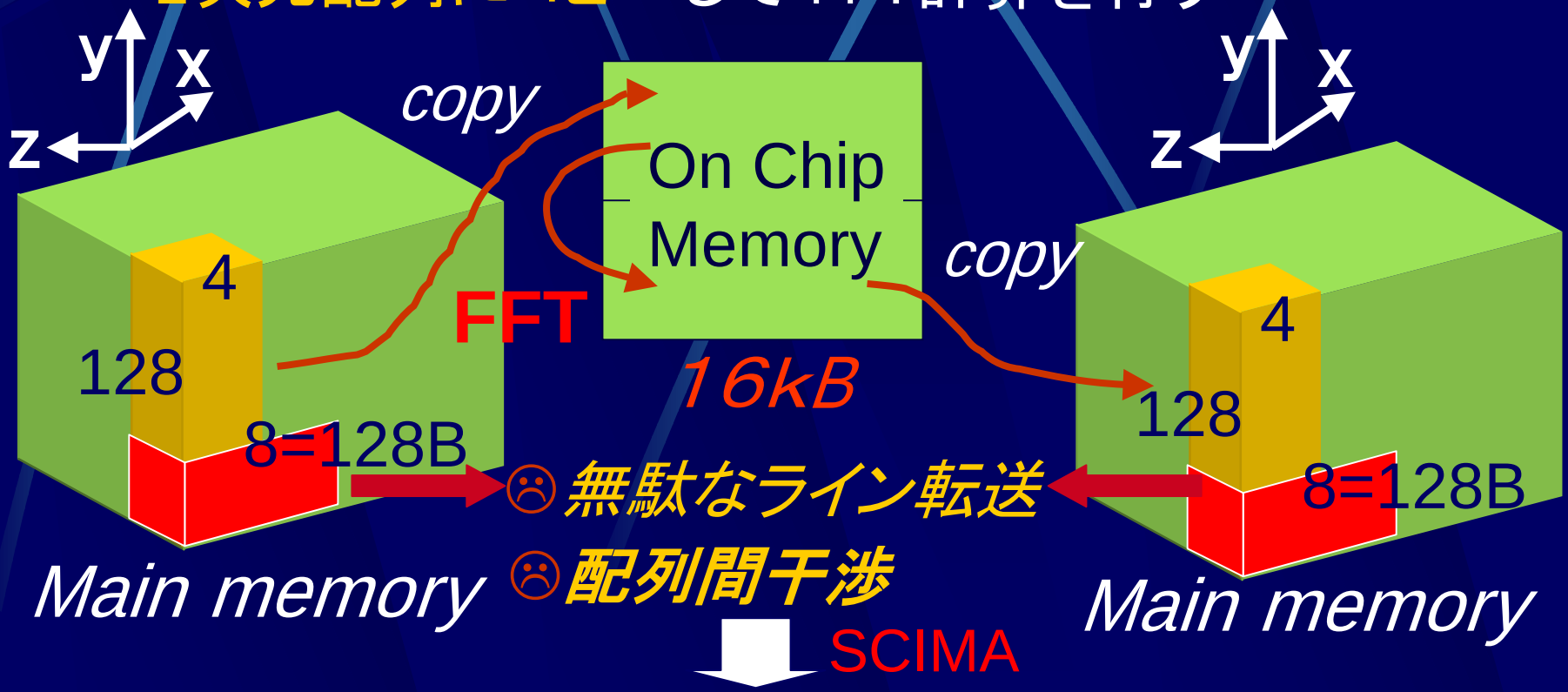
OCMap

☺OCMp, OCMaの利点をもつ
  - throughput stall 小
  - latency stall 小

☺c-opt 比 1.6 倍性能向上 (line = 32B)

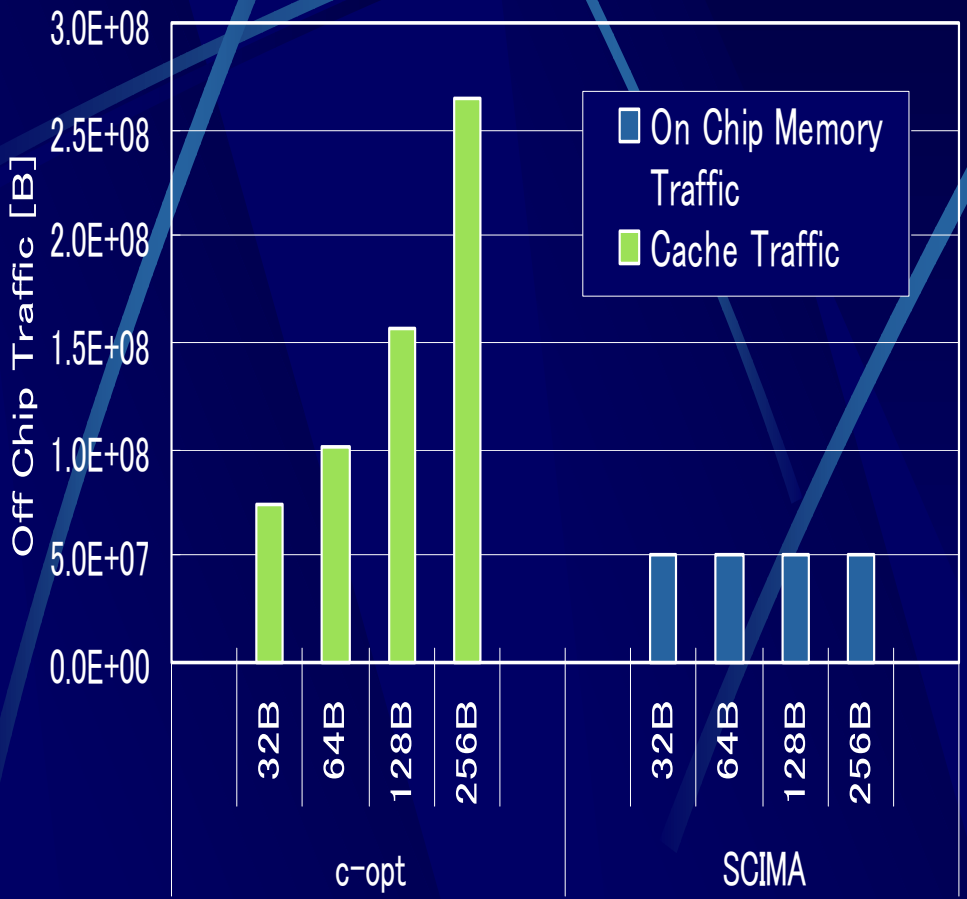
NASPB FT

- 3次元配列(class W，128×128×32 複素数)を1次元配列にコピーして FFT計算を行う



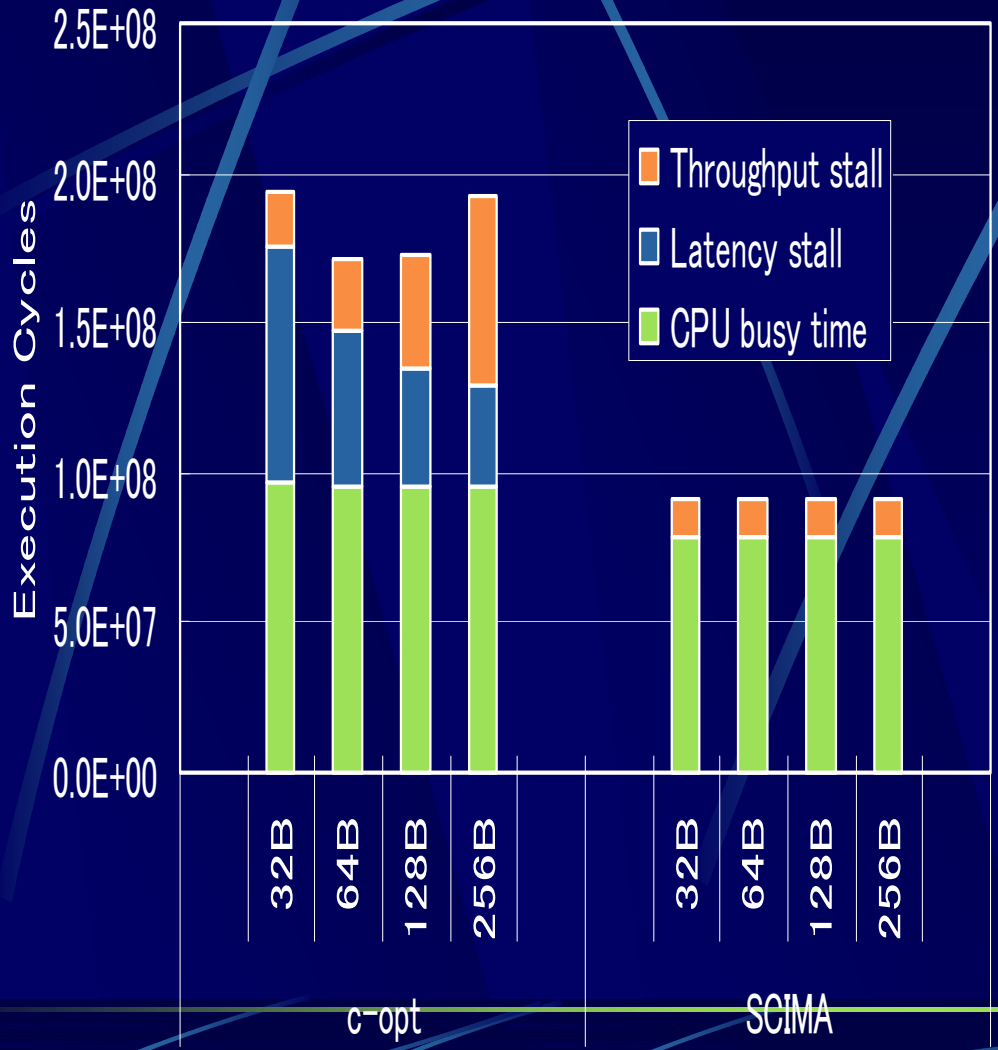
☺ストライド転送，干渉なし，大粒度転送

FTのオフチップトラフィック



- キャッシュ最適化
  1. 配列間干渉
  2. ラインサイズが128B以上でのライン転送によるトラフィック
- SCIMA最適化
  - ☺ 1，2が生じない
  - ☞ キャッシュ最適化よりトラフィックが少ない
  - ☞ ラインサイズによらず一定

FTのサイクル数



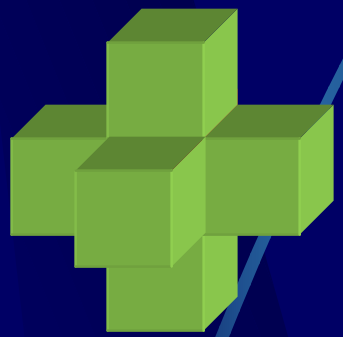
- Throughput stall：
  - トラフィックを反映：SCIMAではラインサイズによらず一定で少ない
- Latency stall：
  - キャッシュ：ライン転送毎にレイテンシがかかる
  - SCIMAでは大粒度転送により少ない

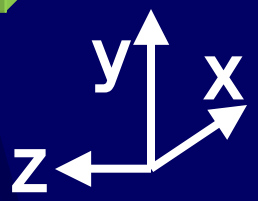
☺32B/line のとき，2.1倍の性能向上

NASPB MG

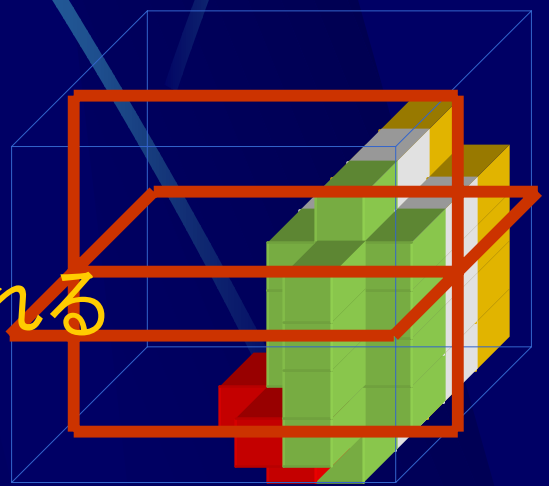
- 3次元 stencil 計算：

$$A(I,J,K) = C \times (B(I-1,J,K) + B(I+1,J,K) + B(I,J-1,K) \ldots$$





- 再利用性のあるデータにアクセスする.
  - ➡ ブロッキングで対応
  - ☞ xy 平面のみ分割
    高々Bのxy平面が3面載れば
    z 方向への再利用性は活かされる



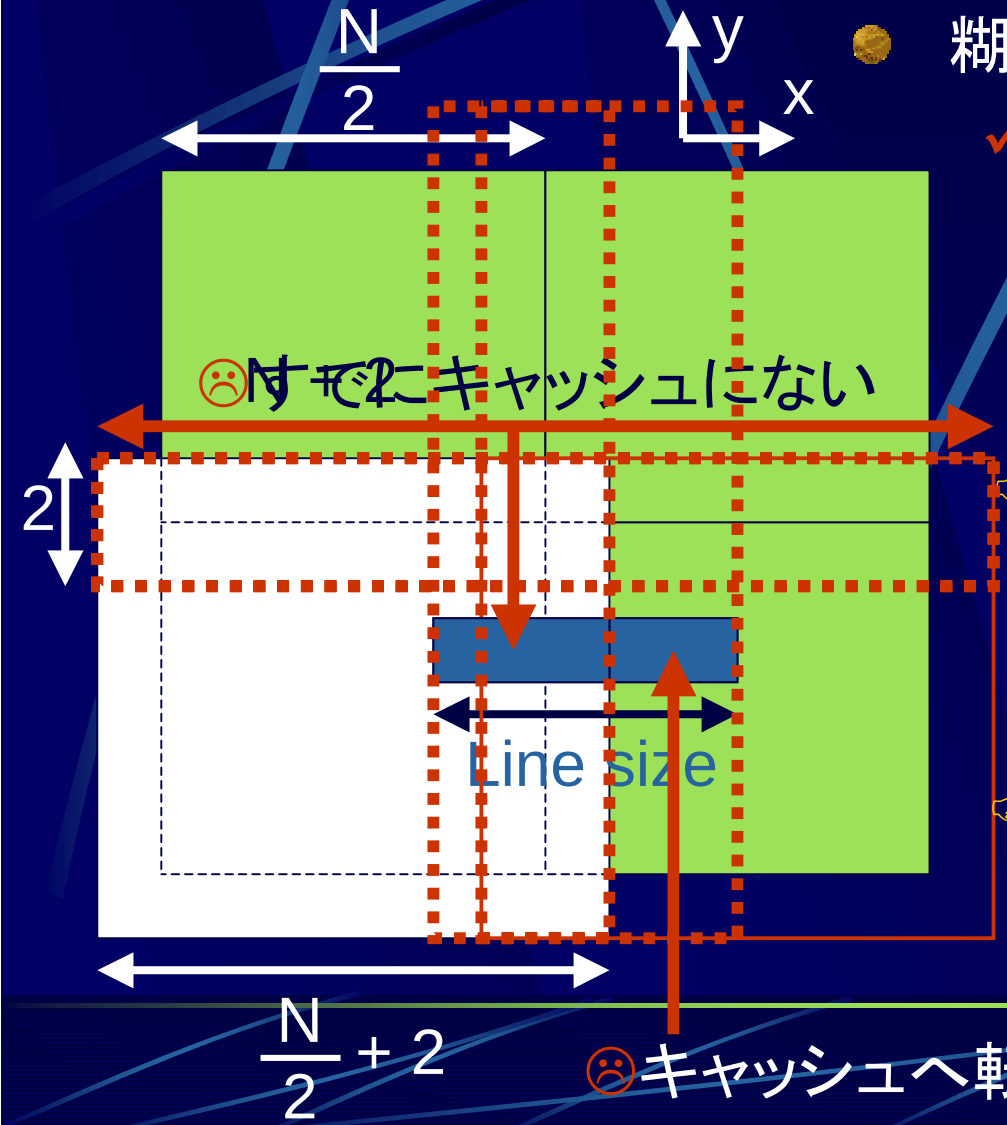
キャッシュから追い出されている

- 格子サイズ: 64 → 32 →・・・→4 →・・・→ 64と変化

ブロッキングは格子サイズ64,32で必要(for 32kB cache)

キャッシュ最適化:コスト関数

- 目的:ブロッキングによるオフチップトラフィックを最小にしたい



- 糊代を考慮 = コスト関数で見積もる
  - ✓Rivera, Tseng (SC `00)
    ラインサイズを考慮に入れない

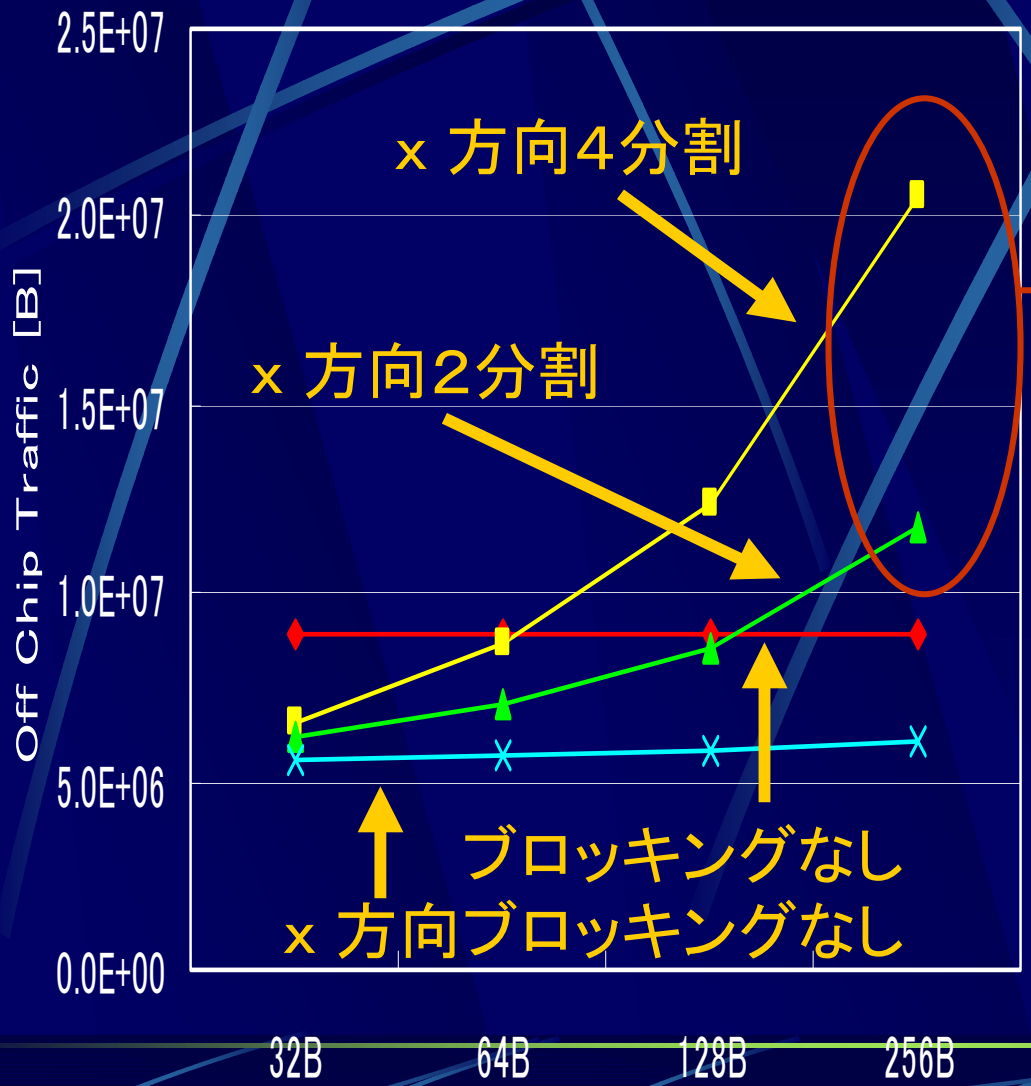
$$2(N+2)+2(N+2)$$

☞実際は x 方向へのブロッキングには L word 分のトラフィック

$$2(N+2)+\underline{L}(N+2)$$

☞x 方向へブロッキングするとトラフィックが多い
➡ ラインが大きいと影響大

ラインサイズの影響



- 格子サイズ　N = 64
- 4way, 32kB キャッシュ

x 方向へブロックを切るとブロッキングしないものよりトラフィックが多い

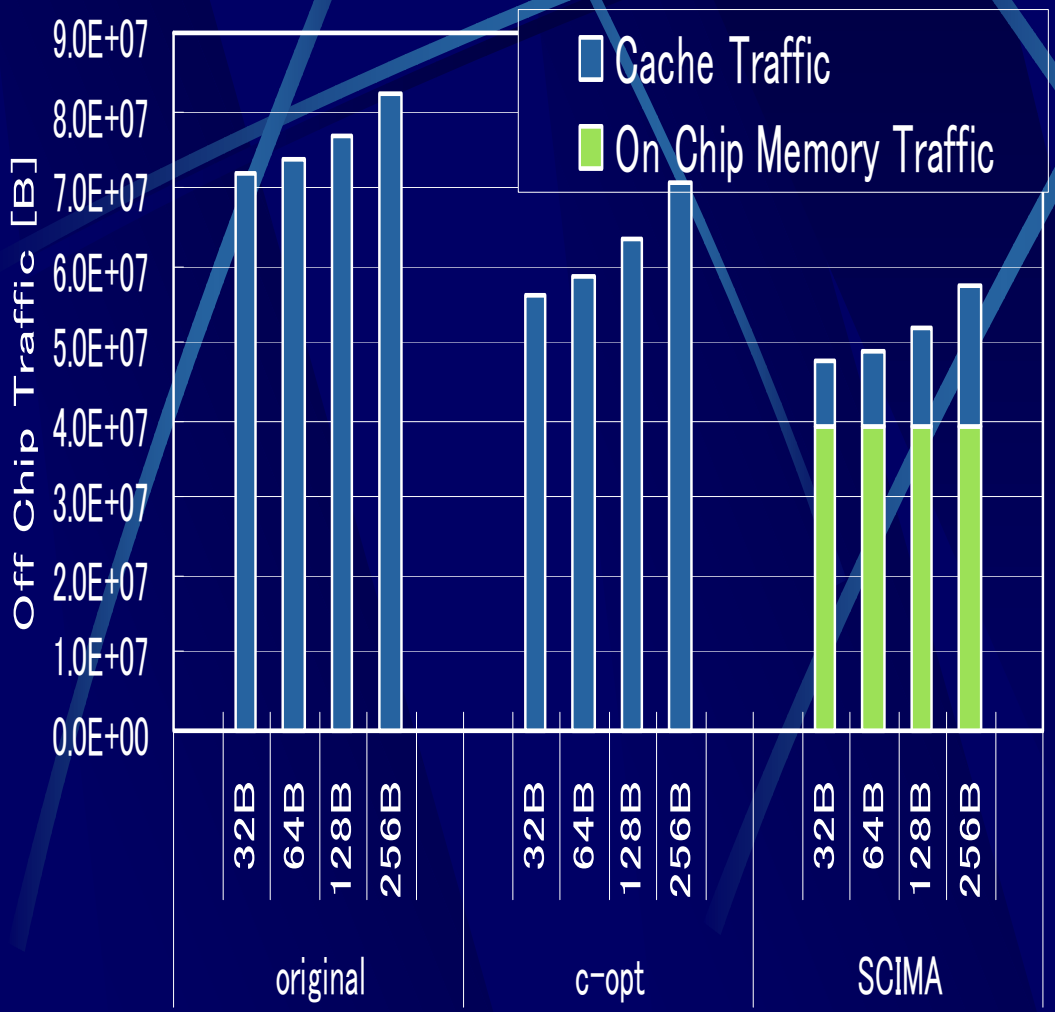
☺提案するコスト関数の方が正しくトラフィックを見積もる

☺ x 方向にブロックを切らない方がトラフィック小

MGの最適化

- キャッシュ最適化
  - 高々 B 3面をキャッシュに載せる
  - ブロッキングが必要な格子サイズ N = 32, 64
  - y 方向のみブロッキング 32 × 8, 64 × 4
  - そのほかはブロッキングなし
  - ☹ 配列間干渉は避けられない
- SCIMA 最適化
  - 全ての配列を オンチップメモリに載せる
  - ☺ 配列干渉なし，大粒度転送
  - ☺ x 方向へもブロッキング可能 => ブロックサイズ 16 ×16 実現できる最小のトラフィック
  - ☹ キャッシュがあまり用いられない

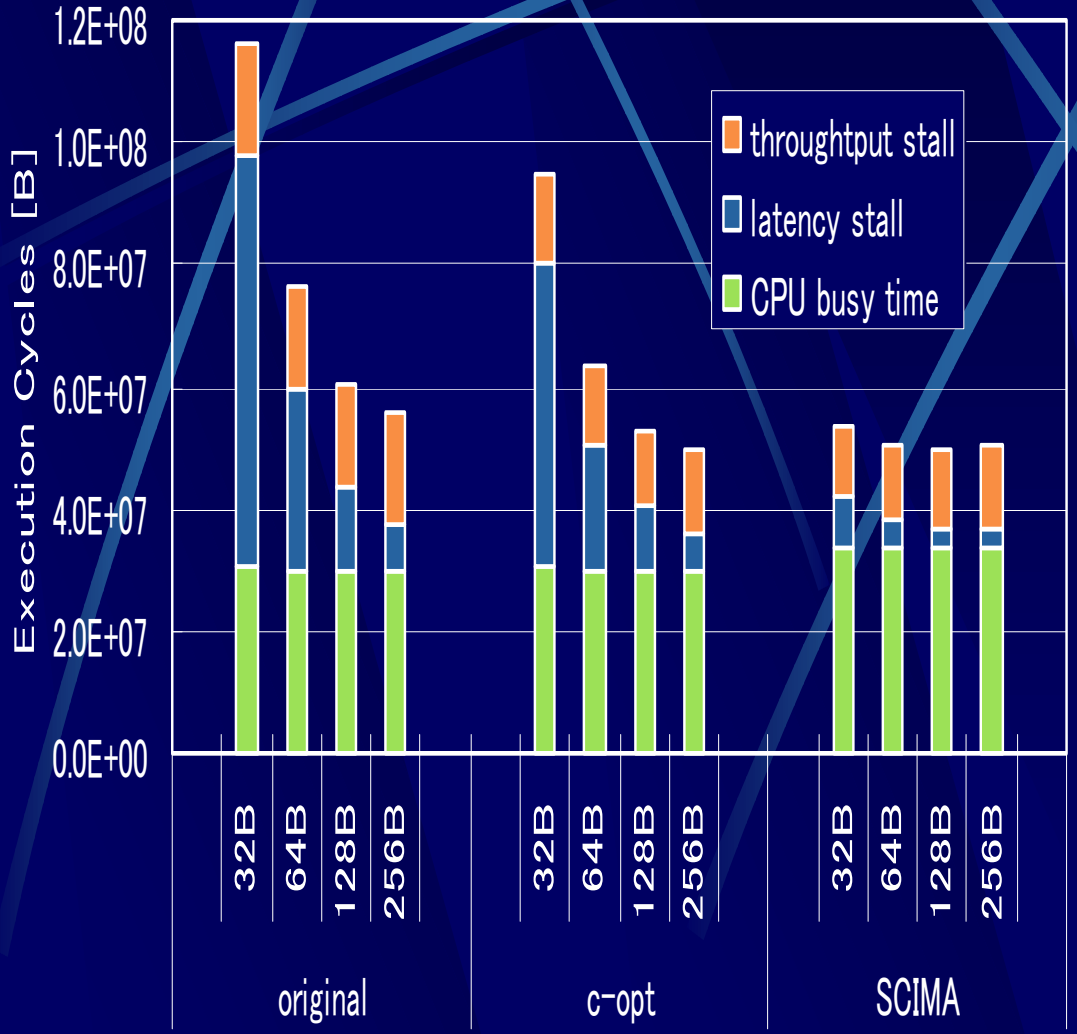
MGのオフチップトラフィック



- ラインによる**無駄な転送がない**
- **配列間干渉なし**

- SCIMAはキャッシュに比べトラフィックが15～19%少ない

MGのサイクル数



- **トラフィック削減**
  Throughput stall **減**
- **大粒度転送**
  Latency stall **減**

- SCIMAはキャッシュに比べ1.7倍のサイクル数減(line 32B)

SCIMAの有効性

- データの配置をユーザが指定できる
  - 干渉を防ぎオフチップトラフィック減少
  - OCMp/ap(CG), FT, MG
- ストライドアクセスによる
  オンチップメモリへの転送
  - HPC等に多い定型的アクセスに効果
  - FT, MG
- Page単位の大粒度転送
  - オフチップアクセス回数の減少
  - OCMa/ap(CG), FT, MG

結論

- NASPBを用いキャッシュアーキテクチャに対するSCIMA アーキテクチャの有効性 を示した
  - SCIMAの利点が性能向上をもたらす

今後の課題

- SCIMA最適化の指針の検討