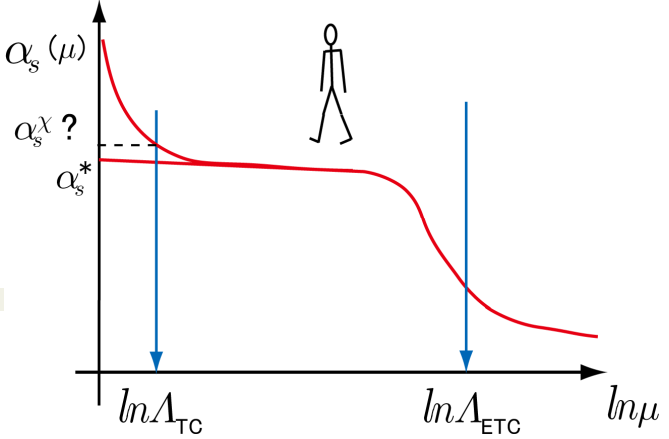
多フレーバー格子ゲージ理論におけるシュレーディンガー汎関数法のGPUを用いた加速について

石川健一（広大理）
宇野隼平（名大理）尾崎祐介（広大理）
武田真滋（金沢大数物）早川雅司（名大理）
山田憲和（高エ研、総研大）

1.はじめに

- LHC稼動： ヒッグス機構、電弱対称性の破れの機構の解明、標準模型を超える模型への足がかり
- 標準模型：
  - 階層性、Fine Tuning 問題 ⇔スカラーヒッグス
  - 超対称性？余剰次元？複合ヒッグス？….
- テクニカラー模型について調べたい

2. Walking テクニカラー模型



- Extended テクニカラー模型で有望な模型

Holdom(1981), Akiba-Yanagida (1986),
Yamawaki-Bando-Matumoto(1986),
Appelquist-Karabali(1986),Wijewardhana(1987)
.....

  - FCNC問題、S-parameter問題、正しい m_q, m_l を出せる湯川結合の問題
  - を解決できる（かも）模型

Damgaard,Heller,Krasnitz,Olsen[PLB400('97)], Iwasaki, Kanaya,Kaya,Saki,Yoshie[PRD69('04)]
Appelquist, Fleming, Nell[PRL('08)],Shamir, Svetitsky, DeGrand[PRD('08)],
Bilgici et al[PRD80('09);Lat09], Bursa,Debbio,Keegam,Pica,Pickup[PRD81('10)].....
本日の9:00-12:45@20aBKの企画講演:
伊藤-Bilgici-Flachi-倉知-Lin-松古-大木-大野木-新谷-山崎:SU(3)
大木-Bilgici-Flachi-伊藤-倉知-Lin-松古-大野木-新谷-山崎:SU(2)

くわしくは3月20日20aBK会場 9:00-
山田憲和氏「格子ゲージ理論で探るLHCの物理」
伊藤,Biligi氏らの研究「large flavor SU(3),SU(2)理論における赤外固定点の探索」
羽場-松崎-山脇氏らの研究「Holographic Techini-dilaton or Confomal Higgs」

3. 多フレーバーゲージ模型の格子計算

- Walking テクニカラー模型
  - Nearly conformal なゲージ理論＝＞ゲージ群、フェルミオンの数、表現

- ここでは、フェルミオンの数がQCDより多い場合を考える。
  - 格子QCDの手法（非摂動的）を使って計算できる。
  - 特に、Coupling の Running の非摂動計算にはシュレーディンガー汎関数法(SF)が使える。

- SFの良い点
  - シュレディンガー汎関数法（繰り込みスキーム）はあまり大きな格子がいらない。16^4ぐらい？

- SFの困難な点
  - フェルミオンの数が多いのでQCDより計算コストがかかる。フレーバー数に比例する。
  - 強結合での結合定数の計算は符号問題があり、なかなか精度良く求まらない。高統計の計算が必要。

3. 多フレーバーゲージ模型の格子計算

- SFの困難な点
  - フェルミオンの数が多いのでQCDより計算コストがかかる。フレーバー数に比例する。
  - 強結合での結合定数の計算は符号問題があり、なかなか精度良く求まらない。高統計の計算が必要。

- これらの困難を乗り越えるためGPUによる加速をおこなったので報告する。

Wilson型Fermion の多フレーバーHMC法のGPUによる加速

4. GPUによる加速方法

- 格子ゲージ理論の計算方法：
  - ハイブリッドモンテカルロ法(HMC)
    - フェルミオン行列式は擬フェルミオンとして扱う
    - 時間方向の（始状態、終状態）境界条件は決まったシュレディンガー汎関数で固定

$$Z = \int DUDP \prod_{j=1}^{\frac{N_f}{2}} D\phi_j^{\dagger} D\phi_j \exp\left[-\frac{TrP^2}{2} - S_G[U] - \sum_{j=1}^{\frac{N_f}{2}} \left| D[U]^{-1} \phi_j \right|^2 \right]$$

$$\phi_j : j = 1 \cdots N_f / 2$$

    - QCD(Nf=2)：擬フェルミオン場は1個
    - SU(3)(Nf=10):擬フェルミオン場は5個　単純には5倍時間がかかる

4. GPUによる加速方法

- シュレディンガー汎関数法なので格子サイズは大きくても16^4 ぐらいまでで良いと考える。（マルチコアの並列化は行う）
- 並列化して動かしてもあまり高並列化は出来ないので並列化しない。
- 統計を稼ぐため複数の独立な計算をたくさん進めたほうが効率は良い。O(10,000)の統計。高統計

- 以上の条件の下で何とか計算が速くなるように工夫する必要がある。
- 1台のPCに複数のGPUを挿して活用できないか？

4. GPUによる加速方法

- 1台のPCに複数のGPUを挿して活用できないか？

$$Z = \int DUDP \prod_{j=1}^{\frac{N_f}{2}} D\phi_j^{\dagger} D\phi_j \exp\left[-\frac{TrP^2}{2} - S_G[U] - \sum_{j=1}^{\frac{N_f}{2}} \left| D[U]^{-1} \phi_j \right|^2 \right]$$

- 1つのWilson型fermionに対する連立方程式の解法の加速

$$D[U]x = b \Rightarrow x = D[U]^{-1} b$$

  - GPU(Nvidia GeForceGTX280)を使うことでCPU単体の計算に比べて 20 倍速くなることを報告した(2008年春の学会：尾崎,石河)

4. GPUによる加速方法

$$Z = \int DUDP \prod_{j=1}^{\frac{N_f}{2}} D\phi_j^{\dagger} D\phi_j \exp\left[-\frac{TrP^2}{2} - S_G[U] - \sum_{j=1}^{\frac{N_f}{2}} \left| D[U]^{-1} \phi_j \right|^2 \right]$$

- これまで: 2-flavor を(Nf/2)個入れる

for $i = 1, N_f/2$

$$D[U]x_i = \phi_i \Rightarrow x_i = D[U]^{-1}\phi_i$$

$$D[U]^{\dagger} y_i = x_i \Rightarrow y_i = D[U]^{\dagger -1} x_i$$

$$Force = Force + F[x_i, y_i]$$

end for

update U <= $Force$

Force の計算を順番に行って足しこんでいた

並列性がある。
まずソルバーをまとめる
（ブロック化）

4. GPUによる加速方法

$$Z = \int DUDP \prod_{j=1}^{\frac{N_f}{2}} D\phi_j^{\dagger} D\phi_j \exp\left[-\frac{TrP^2}{2} - S_G[U] - \left| D[U]^{-1}\Phi \right|^2 \right]$$

- まず(Nf/2)-Flavor 用の擬フェルミオンにまとめる

$$\Phi \equiv (\phi_1, \phi_2, \phi_3 \cdots), X \equiv (x_1, x_2, x_3 \cdots), Y \equiv (y_1, y_2, y_3 \cdots)$$

$$D[U]X = \Phi \Rightarrow X = D[U]^{-1}\Phi$$

$$D[U]^{\dagger} Y = X \Rightarrow Y = D[U]^{\dagger -1} X$$

$$Force = Force + F[X, Y]$$

$$\text{update}\, U <= Force$$

ブロック化されたソルバーでまとめて解いてしまう。
係数行列は共通なのでデータの効率利用が期待される

4. GPUによる加速方法

$$Z = \int DUDP \prod_{j=1}^{\frac{N_f}{2}} D\phi_j^{\dagger} D\phi_j \exp\left[-\frac{TrP^2}{2} - S_G[U] - \left| D[U]^{-1}\Phi \right|^2 \right]$$

- その上でブロックソルバー内でGPUに計算を分担させる

$$D[U]X = \Phi$$

$$D[U]x_1 = \phi_1 \Rightarrow x_1 = D[U]^{-1}\phi_1$$ ← GPU＃1が計算

$$D[U]x_2 = \phi_2 \Rightarrow x_2 = D[U]^{-1}\phi_2$$ ← GPU＃2が計算

$$D[U]x_3 = \phi_3 \Rightarrow x_3 = D[U]^{-1}\phi_3$$ ← GPU＃3が計算

$$\vdots$$

複数のGPUがあれば同時計算が出来る

5. GPUによる加速結果

- 比較するアルゴリズム

1. CPU倍精度ソルバー (Even-Odd site 前処理):
   - 非ブロック化BiCGStab
   - 単純ブロック化BiCGStab
   - Sakurai-Tadano-Kuramashi ブロック化BiCGStab [CPC181(‘10)]

2. GPU加速ソルバー (Even-Odd site 前処理) :
   - ブロック化BiCGStabソルバー(CPU,DP)＋非ブロック化BiCGStabソルバー(GPU,SP)

3. CPU単精度SSE加速 (SSOR前処理)
   - ブロック化BiCGStabソルバー(CPU,DP)＋非ブロック化BiCGStabソルバー(GPU,SP)+Block-SSOR前処理(CPU,SSE,SP)

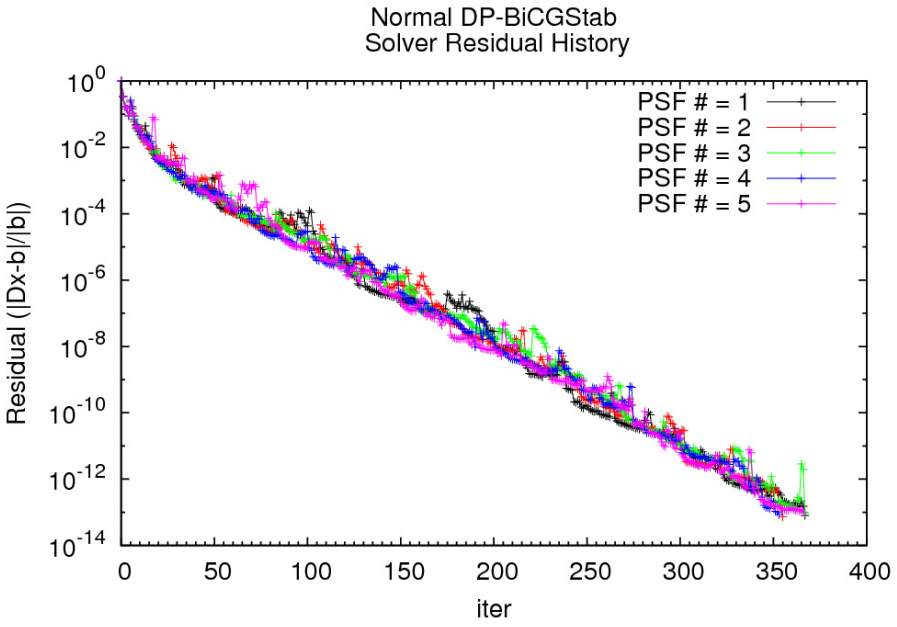
5. GPUによる加速結果

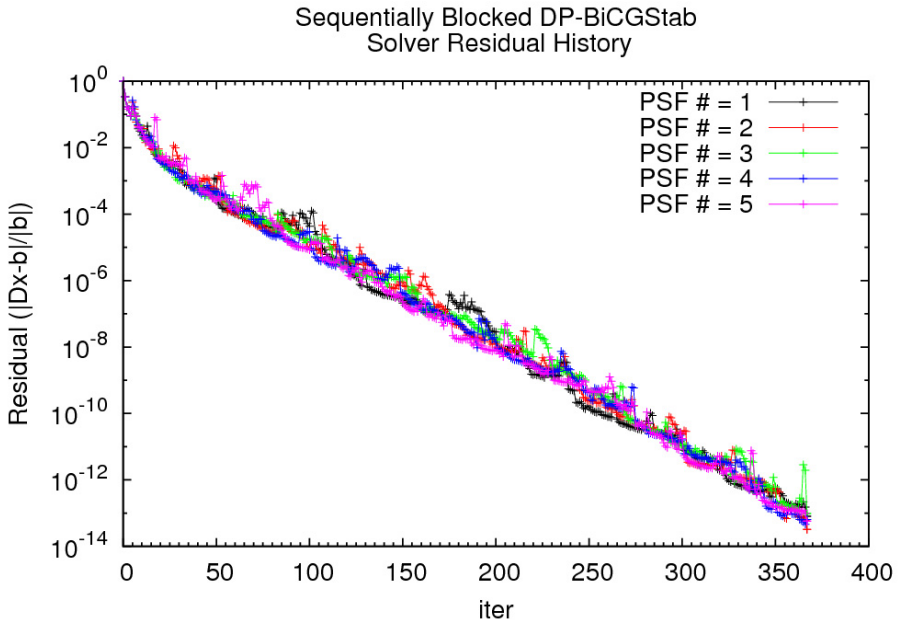
- 計測環境
  - ホスト計算機（CPU）
    - CPU: Core i7 920 (2.67GHz, 4 core)
    - DP: 43 GFlops, SP: 85 GFlops (peak)
  - GPUカード
    - Nvidia GeForce GTX 285 (2枚をホストに x16 PCIeGen2 で接続)
    - 240 core @ 1.48 GHz,　SP: 710 GFlops (peak)
  - OS,コンパイラ
    - CentOS 5.2 (Linux)　ロハ
    - Intel Fortran / C++　買う
    - CUDA 2.3　ロハ

5. GPUによる加速結果

$$16^4,\ \beta = 4.52,\ \kappa = 0.15805,\ N_F = 10$$
$$g_{\mathrm{SF}}^2 \approx 10,\quad aM_{\mathrm{PCAC}} \approx 0.001$$

- CPUのみでのブロック化アルゴリズムの改良の効果





非ブロック化BiCGStab
29.29 sec/solve

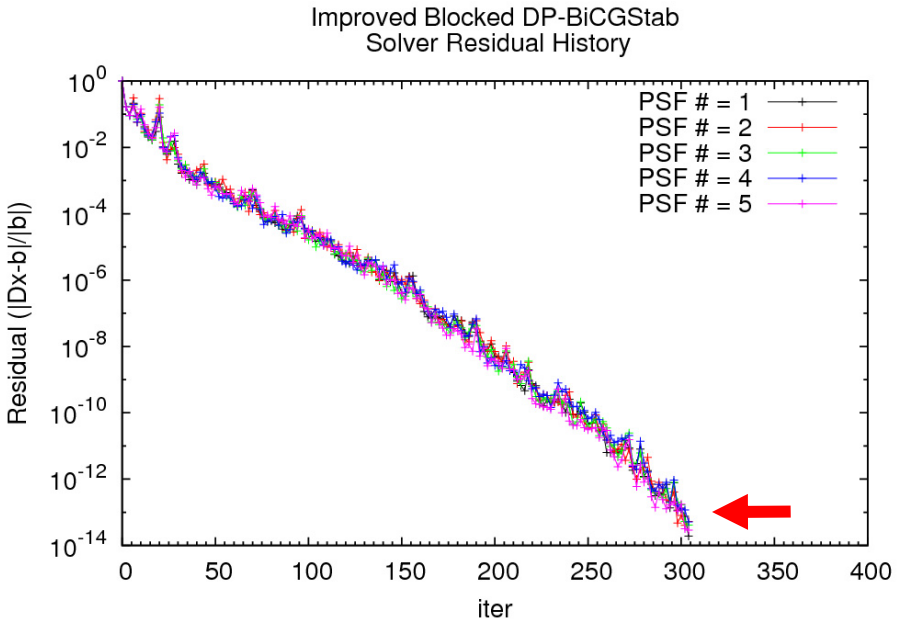
単純ブロック化BiCGStab
24.47 sec/solve

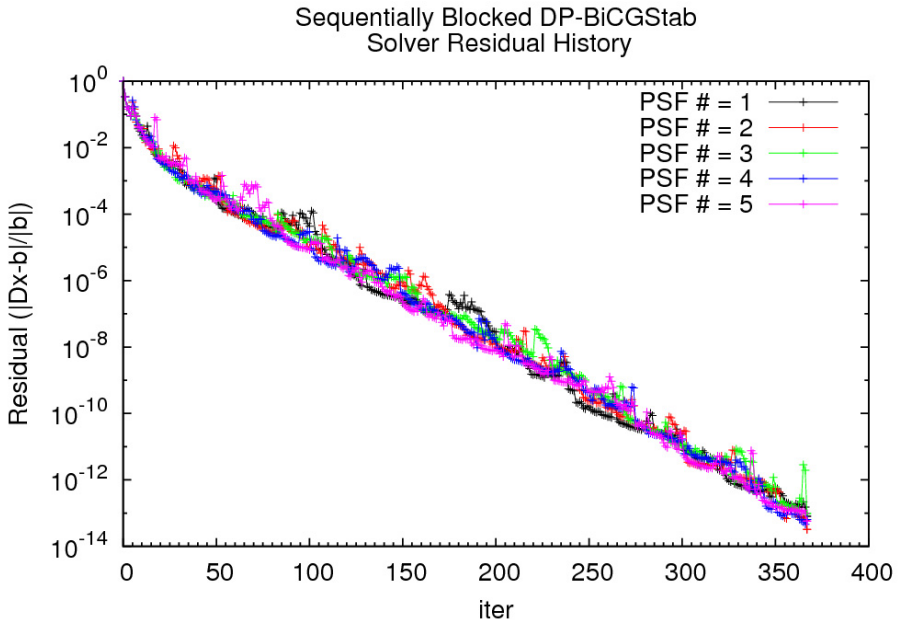
データの局所化？で16%高速化

5. GPUによる加速結果

$$16^4,\ \beta = 4.52,\ \kappa = 0.15805,\ N_F = 10$$
$$g_{\mathrm{SF}}^2 \approx 10,\quad aM_{\mathrm{PCAC}} \approx 0.001$$

- CPUのみでのブロック化アルゴリズムの改良の効果





Sakurai-Tadano-Kuramashi
改良ブロック化BiCGStab
22.68 sec/solve

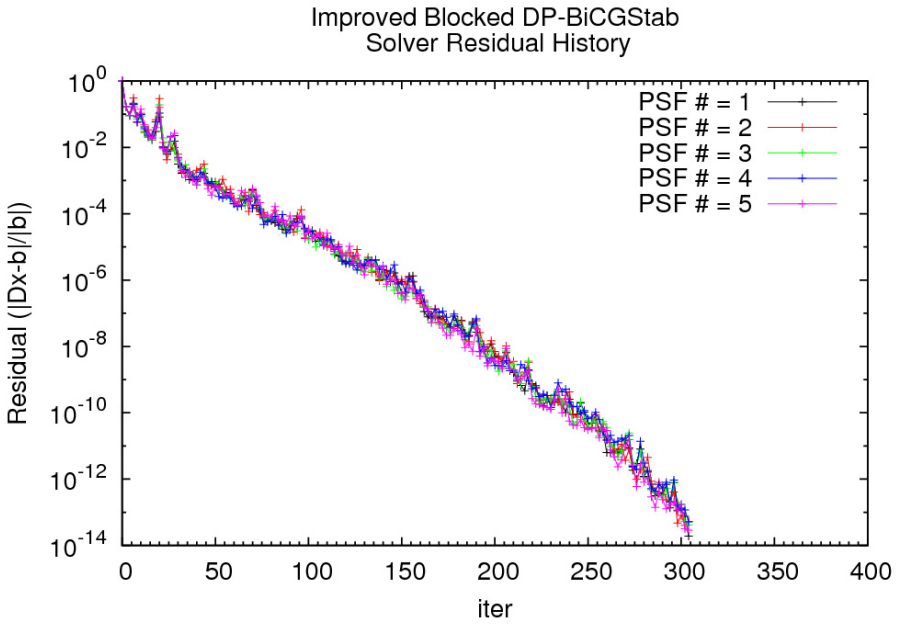
単純ブロック化BiCGStab
24.47 sec/solve

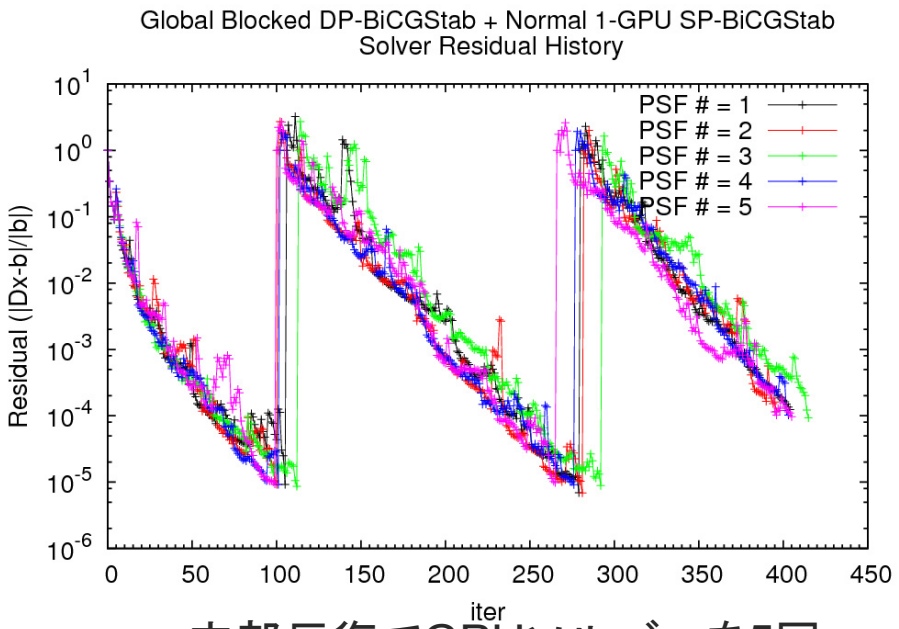
5本の連立方程式を混ぜて解くことで反復回数は18%減,計算時間は8%減

5. GPUによる加速結果

$$16^4,\ \beta = 4.52,\ \kappa = 0.15805,\ N_F = 10$$
$$g_{\mathrm{SF}}^2 \approx 10,\quad aM_{\mathrm{PCAC}} \approx 0.001$$

GPUの効果（1GPU）





Sakurai-Tadano-Kuramashi
改良ブロック化BiCGStab
22.68 sec/solve

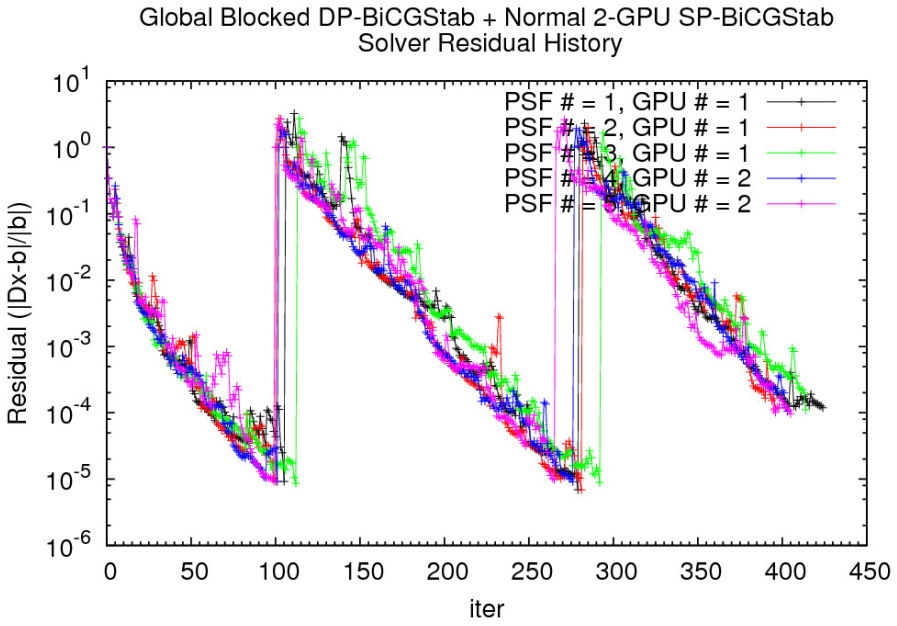
内部反復でGPUソルバーを5回ずつ解いている
4.06 sec/solve

単精度GPUソルバーで反復回数は多いが実行時間は82%減＝1/5.6　（前回の報告はでは1/17でしたが、解いている問題が違うので単純には比較できない）

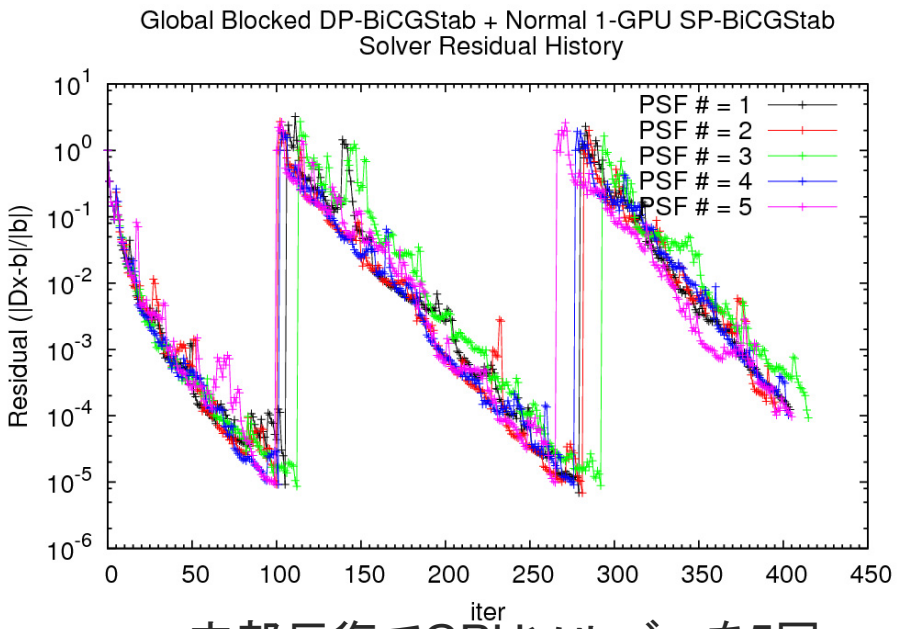
5. GPUによる加速結果

$$16^4,\ \beta = 4.52,\ \kappa = 0.15805,\ N_F = 10$$
$$g_{\mathrm{SF}}^2 \approx 10,\quad aM_{\mathrm{PCAC}} \approx 0.001$$

GPUの効果（1GPU⇒2GPU）



内部反復でGPUソルバーを3回と2回ずつ解いている
2.69 sec/solve



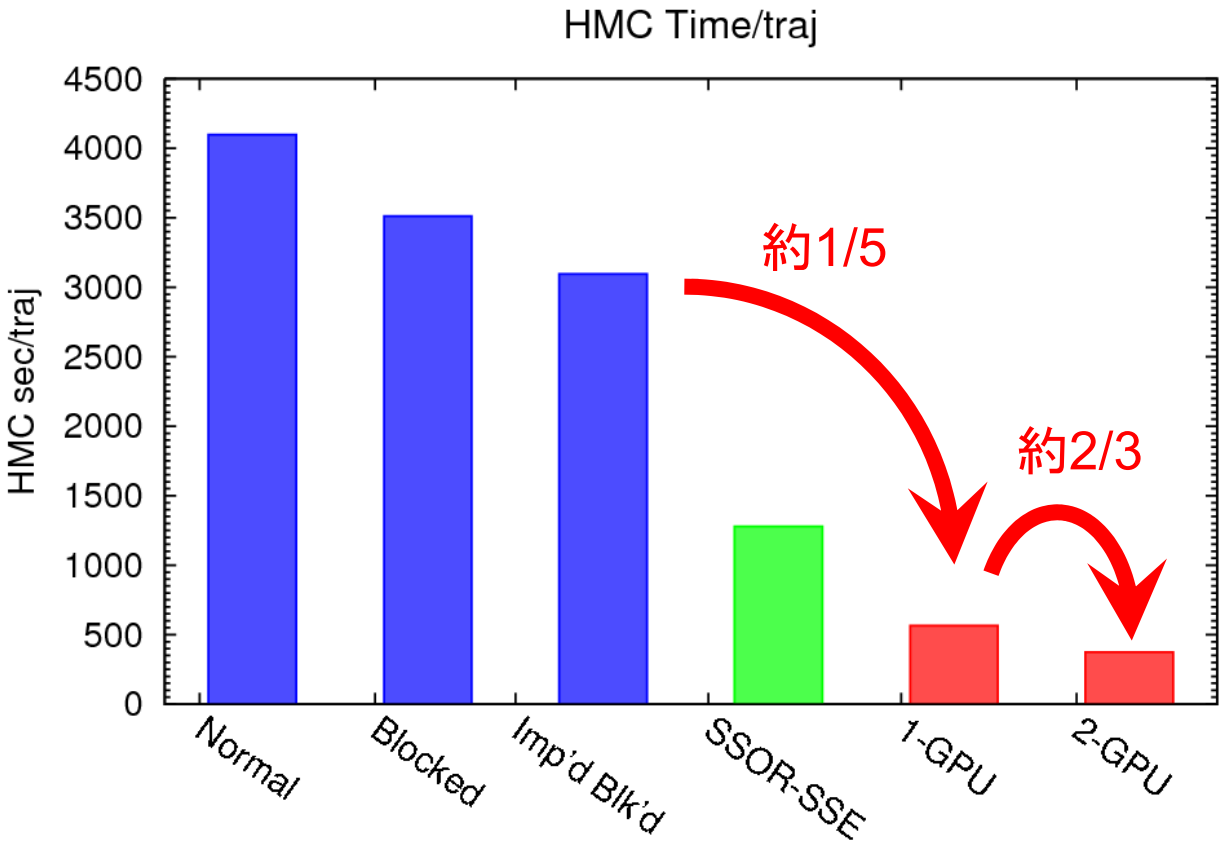
内部反復でGPUソルバーを5回ずつ解いている
4.06 sec/solve

GPUに等価に負荷分散できていないので2倍速くならない
計算時間は34%減:　理想は 3／5 = 0.6 =40%減

5. GPUによる加速結果

$$16^4,\ \beta = 4.52,\ \kappa = 0.15805,\ N_F = 10$$
$$g_{\mathrm{SF}}^2 \approx 10,\quad aM_{\mathrm{PCAC}} \approx 0.001$$

- HMC 1trajにかかった時間の比較



CPUだけでも、単精度SSE+SSOR前処理加速で2倍以上速くなった

GPUをつかうとさらに加速できた。

1GPU＝＞2GPUで66%=2/3 に加速
理想は60%＝3/5

6. まとめ

- 多フレーバーのHMCアルゴリズムを加速したい
  - 擬フェルミオンはそれぞれ独立で並列化可能
  - 複数のGPUで分担して計算するというアイデア

- 1台の計算機で　CPU，GPU比較を行った
  - CPU倍精度計算はとても遅い。
  - CPU単精度＋SSE＋SSORはとても速い。
  - GPU複数使用は多フレーバーには理想的（単純並列化）
  - GPU複数の費用対効果は？

6. まとめ

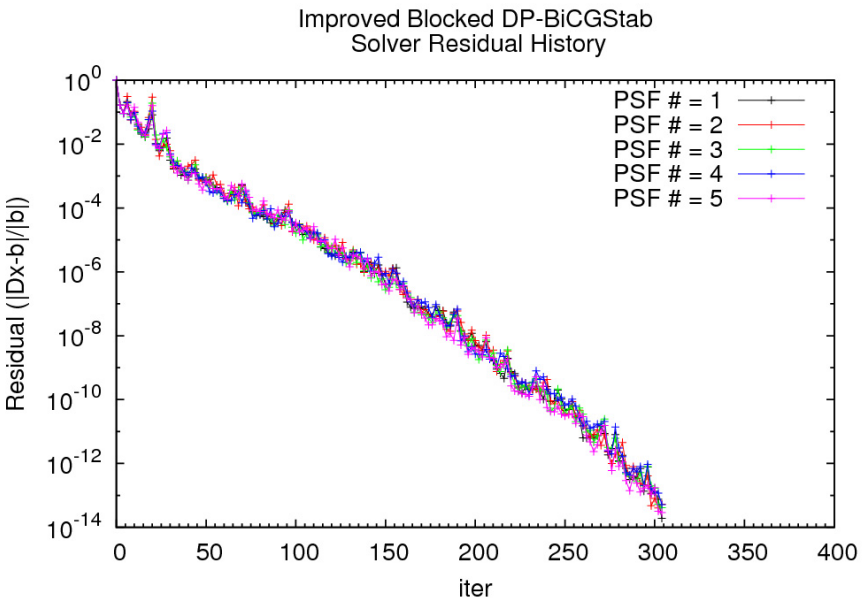
- GPU複数の費用対効果は？
  - Nf＝10　（5 ps-fermion）の時は 5GPUまで
  - CPUを増やすかGPUを増やすか？
  - ホスト計算機の値段　＞　GPU1枚の値段
  - ホスト計算機＋GPU１枚に
  - GPU2枚追加で倍以上速くなるはず
    - （ホスト計算機＋GPU１枚）の値段＜GPU2枚の値段　ホストを増やす
    - （ホスト計算機＋GPU１枚）の値段＞GPU2枚の値段　GPUを増やす
    - 管理の手間を考えるとあまりホストは増やしたくない。。。

おわり

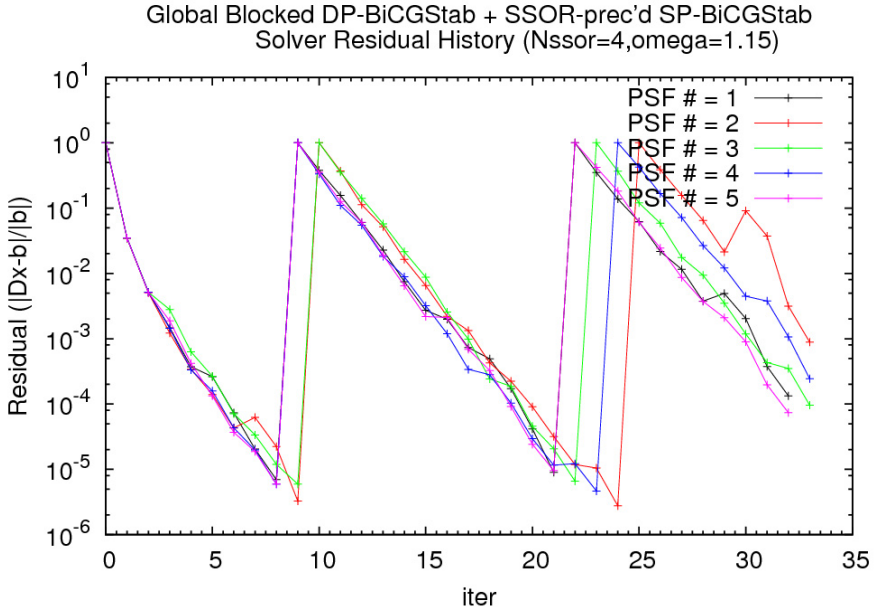
5. GPUによる加速結果

$$16^4,\ \beta = 4.52,\ \kappa = 0.15805,\ N_F = 10$$
$$g_{\mathrm{SF}}^2 \approx 10,\quad aM_{\mathrm{PCAC}} \approx 0.001$$

SSE単精度加速の効果（CPUのみ）



Sakurai-Tadano-Kuramashi
改良ブロック化BiCGStab
22.68 sec/solve



内部反復でSSOR前処理した単精度
ソルバーを5回ずつ解いている
9.23 sec/solve

CPUだけでも、1node計算では強力な前処理（SSOR）が使える。
SSE単精度は割りと強力。GPUがないときはおそらくベスト？
60%減＝2倍以上速くなる。前処理と単精度の組み合わせ