格子QCD共通コード「Bridge++」 2013年度報告

上田　悟（KEK）
新学術領域・HPCI戦略分野5 格子QCD共通コードプロジェクト

プロジェクト

- 新しいことを始めるための Test Bed として
- Lattice QCD コミュニティーの共通インフラ
  - 読み易い、使いやすい
  - 検証などのドキュメントの整備
  - バグなどの合わせ窓口をしっかりする

- プロジェクトサイト: http://bridge.kek.jp/Lattice-code/
- コアメンバー：青木慎也、金谷和至、滑川裕介、根村英克、谷口裕介、浮田尚哉、(筑波大)、青山龍美 (名古屋大)、松古栄夫、元木伸治、上田悟 (KEK)
- サポート：
  - 新学術領域研究「素核宇宙融合による計算科学に基づいた重層的物質構造の解明」（領域代表・青木）H20-24　http://bridge.kek.jp/
  - 計算基礎科学連携拠点　HPCI戦略プログラム分野5 http://www.jicfus.jp/field5/

共通コードの目標

- 可読性: 初心者でも読んで使える
- 可搬性: ノートPCからスパコンまで
- 拡張性: 簡単に拡張できる
- 高性能: 実践でも使用しても問題ない性能

- プログラミング: C++を利用
  - オブジェクト指向
  - デザインパターン
- 様々なコンピュータ・アーキテクチャ
  - MPI, OpneMP/pthread
  - arithmetic accelerators.（CUDA, Xeon Phi）
- 豊富なドキュメント

2013年の主な進捗

これまでの主な活動

- 2012/7: ver1.0を公開。
- 2013/7: ver1.1を公開。
  Ver1.0よりパラメータ周りの整備などの改善
  バグ修正など3回
- 使い方や開発に関する問い合わせ　2件
- 国際会議でコードの紹介 2回

現在

- Wikiによる実装ノートや検証データの充実
- コードの整理・BG/Qなどに対する最適・高速化
- OpenCL (GPGPU), Xeon Phi の利用
- 外部ライブラリー（CUDA）の利用

OpenMP開発

- 京コンピュータなどには OpenMP + MPI 化が不可欠
- OpenMP：directiveを挟むだけで簡単にthread並列化
- OpenMP と C++ が微妙に相性が悪い。
  - オブジェクト指向：機能毎にひとまとめにしよう。
  - OpenMP：性能を出すためには全体をparallelに

1. For文に#pragma parallel forをつけてく
   Thread 生成のオーバーヘッドで遅い
2. Fork & Join を減らすために並列領域を全体に
   戻り値返しの関数Field_z = func(Field_x, Field_y)
   一時変数がthreadごとに生成されて大変なことに
3. 戻り値返しから変数渡しに変更
   z =func(x, y) -> func(z, x, y)
4. 2 と 3 の繰り返し

Bridge++ OpenMP化指針

C++ で OpenMP を使うときに注意すること

1. Parallelのスコープがどこにあるのか。
    1. 自分の中でparallelにするのか？
    2. Parallel領域から呼ばれているか？
    3. Single領域から呼ばれているか？
2. Parallel領域で使用されるクラスはThread safeに

現在の Bridge++ OpenMP 化指針

1. Main の始まりの方でparallelにする。
2. ほぼすべてのクラスがparallel領域で呼ばれることを仮定。
3. #pragma omp for を使わず、thread idで明示的に分割。

OpenCL

- 最近のスパコンにはGPUなどの演算加速器が広く導入
- 色々な device を統一的に扱えるAPI言語
  - CPUs, GPGPUs, Xeon Phi, FPGAs, Cell/B.E. etc…
- Device Manager クラスを用意して、リソースの管理
  - Host / Device 間のメモリー転送
  - OpenCL APIの隠蔽
- 現在、実装のたたき台として製作
  - 理論性能の 1% 程度
  - データレイアウト変更等の最適化はこれから

Xeon Phi

- IntelがGPGPUに対抗して出した multi core device
- 筑波大学に4月から PACS-IX(COMA) として導入予定
- CPU + Phi、Phi単体など並列度に応じた利用が可能
- 基本的な使い方は OpenMP
- 現在、Bridge++の元になった、singleのコードを利用して Xeon Phi を使用する際の注意点などを模索中

外部ライブラリの使用

- CUDA などに最適化されたコードをBridgeとどう組み合わせるか？
  - Bridge++本体や外部のコードにできるだけ手を付けずに共存。
  - 不必要なときは外部のコードが無くてもコンパイルができるように。
- CUDAによるゲージ固定をたたき台として実装。

2014年度に向けて

- 高速化/最適化
  - OpenMP への対応
    - 試行錯誤の繰り返しで OpenMP のノウハウが溜まってきた。
  - OpenCL、Xeon Phi へのβ版
    - これらを使う際の問題点などを探る。
  - 外部ライブラリの利用
    - Bridgeとの整合性をどう取るか
- 検証データなどのドキュメントの整備
  - Wiki の実装ノートなどリアルタイムで更新。
- 利用者の拡大

格子QCD共通コード「Bridge++」を使ってみて下さい