GPU を用いた有限体上の線形方程式の解法の高速化

萩原 尚† 村尾裕一 †

1 はじめに

シミュレーションをはじめとする様々な分野で GPU による並列化が行われているが，整数計算の例はほとんど試みられていない．しかし，多倍長整数の演算は計算量も多く，特にモジュラー算法による表現では，複数の法における独立な計算を行う必要があり，データ並列処理能力の高い GPU を活用することが有効である．

本稿では，GPU を用いた有限体上の線形方程式の解法を提案する．有限体上で正確な解を求める場合，行数に比例して法の数が多くなり，計算量も比例して増えていく．この影響を抑えるために GPU による並列化を試みた．また，線形方程式の解法のアルゴリズムとして，行列ベクトル積を多用するアルゴリズムを採用し，多数のコアを有する GPU の活用を目指した．

結果，Core i7-920 による逐次実行と，GeForce GTX570 を用いた並列実行との比較で最大 47 倍の速度向上が達成できた．

2 並列化の方針

本研究では，線形方程式の解法に Wiedemann のアルゴリズム [1] を用いている．このアルゴリズムは有限体上の疎な線形方程式を解くアルゴリズムであり，処理の内容は大きく分けて行列ベクトル積と最小多項式の計算である．疎な線形方程式の解法用であるアルゴリズムを用いた理由は，最も時間のかかる処理が行列ベクトル積であるため，GPU による並列化の効果が出しやすいと考えたからである．

モジュラー算法とは共通因子をもたない複数の法 p_i を用いて多倍長整数を表現する方法である．一般的な多倍長整数の表現で必要な carry や borrow の概念がなく，各法の計算を単一精度で行うことができ，法ごとに容易に並列化が可能な表現となる．この利点からモジュラー算法は GPU に向いた多倍長整数のアルゴリズムであると考えた．また，モジュラー算法ではすべての計算を法の大きさ未満の精度で行うことができるため，unsigned int を用いる場合に法の大きさとして $0 < p_i < 2^{32}$ が考えられるが，$2^{16} \leq p_i < 2^{32}$ では積が 2^{32} を越えてしまう可能性があるため，64bit の整数を扱う必要が出てくる．しかし，GPU で 64bit の整数演算はハードウェアでサポートされず，他の命令を組み合わせて実行されるため，大きな速度低下を招くと考えられる．よって，本研究では法の大きさに $0 < p_i < 2^{16}$ を採用した．

CUDA を用いた実装では 1 ブロックに 1 つの法についての Weidemann のアルゴリズムの計算を割り当て，1 スレッドでアルゴリズムの内部の並列化を行った．しかし，最小多項式の計算は法の数程度の個数しかなく，計算回数も法の値で異なり，各スレッドの計算量が大きく異なってしまうため，行列ベクトル積の処理のみを GPU で行うことにした．行列ベクトル積を GPU で行う方法は複数あるが，1 スレッドで 1 行の処理を行う方法を採用した．この実装の利点は，1 スレッドあたりの計算量をある程度大きくすることができ，且つ，行列を転置することでグローバルメモリへのアクセスでコアレッシングを行えることである．一方，行列が小さすぎる場合は十分な数のスレッドやブロックを使用しないために GPU の性能を引き出せないことに加え，呼び出しのオーバーヘッドが大きくなり，並列化の効果が得られなくなる．また，行列が大きすぎる場合は行列がグローバルメモリにのりきらないため，メモリアクセス等を検討し複数のカーネルへ処理を分割する必要がある．

3 まとめ

整数演算命令はハードウェアでサポートされていないものがあり，環境として十分であるとは言えないが，本研究では CPU による逐次実行と比較して約 47 倍の速度向上が達成できた．この結果より，整数演算においても浮動小数点演算ほどではないが GPU による並列化が有効であると考えている．今後も，行列の大きさに合わせた実装方法の違いによる計算時間の計測や，消去法など他のアルゴリズムによる実装と比較を行う予定である．

参考文献

[1] Douglas H. Wiedemann, Solving sparse linear equations over finite fields. *IEEE Trans. Information Theory* IT-32, 54-62, 1986.

† 電気通信大学 情報理工学研究科