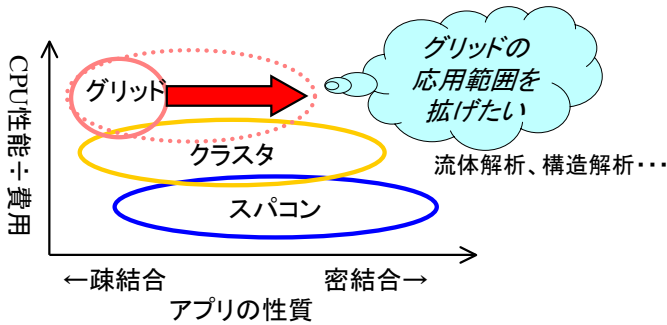


高い耐遅延性を持つガウス消去法

～ 副題: Top500@homeへの道 ～

遠藤 敏夫, 田浦 健次郎 (東京大学)



密結合アプリを広域で動かすための障害は？

- WANのバンド幅不足
 - ノードの故障率
 - セキュリティポリシーの差異
 - WANの高い通信遅延
- 近い将来に解決(?)
- ⇒ 障害であり続ける
- スパコン: $10 \mu s$, グリッド: >10ms

耐遅延アルゴリズムの研究によって
グリッドの応用範囲拡大

密行列連立一次方程式を例題として選択

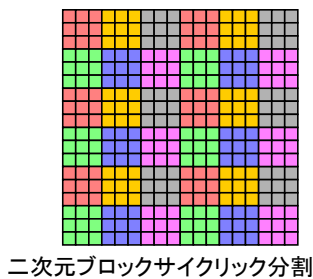
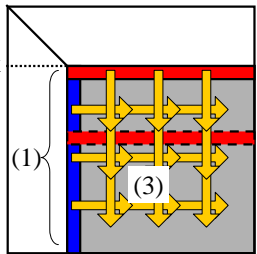
- Top500 supercomputerランキングでも利用
- 遅延の影響大

ガウス消去法とpartial pivoting

連立一次方程式 $Ax=b$ ($A: N \times N$ 行列) を解く古典的手法
良好な計算精度を得るために partial pivoting

for $k=0$ to $N-1$

- (1) k 列で絶対値最大の要素 (pivot) を選ぶ
- (2) pivot を含む行と k 行を交換
- (3) k 行と k 列を用い、 $k+1 \leq i, j < N$ の部分を更新
$$a_{ij} = a_{ij} - a_{ik} a_{kj} \div a_{kk}$$



全CPUコスト: $\frac{2}{3}N^3 + O(N^2)$

全通信コスト: $O(N^2 \sqrt{p})$

同期コスト: $O(N)$ ← 各 k の pivoting (1) における reduction 通信のため

長遅延環境では pivoting の同期がボトルネックに！

Partial pivoting よりも緩和された手法が望ましい

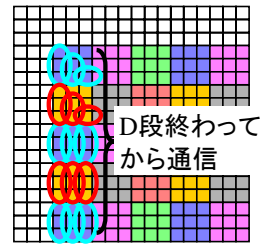
まじめに選ぶと
オーバーラップ処理できない...

提案手法

同期コストを削減しつつ、良好な計算精度を実現する
selective pivoting を提案

- (1) k 列を持つ各プロセスが、局所データのみを用いて $[k, k+D)$ 番目の D 個の pivot を選択
 - (2) 結果を通信により集計
 - (3) 「最良」のプロセスを選び、それが提供した D 個の pivot を決定版として採用
- ⇒ D 段分の行交換、更新

局所的・投機的に
 D 段ガウス消去
してしまう！



特徴

同期コストは $O(N/D)$
高い確率で良好な pivot を選択

必ずしもベストを選ばなくとも良いという方針

他の緩和手法

Threshold pivoting: pivot として選ばれる値が、厳密に最大でなくても良い (最大値の t 倍以上であれば良い)

Pairwise pivoting: 隣り合う2つの行をペアにして、片方を消去。これを下から順に (バブルソート式) に繰り返す

Parallel pivoting: 2行をペアにして消去する処理を、トーナメント式に繰り返す

	Partial	Threshold	Pairwise	Parallel	Selective
計算精度	○	○~△	△	×	大抵○
耐遅延	×	×	○	○	○

並列実装

ノード増減への対応と、行列更新部分の耐遅延性のために以下の特徴を備える (cf. 遠藤 CCGrid04 論文)

- 行列再帰分割 + ランダム入れ替えによるデータマッピング
- 部分行列のノード間移動による動的負荷分散
- ブロックレベルでのメッセージ駆動計算

以下のライブラリを利用

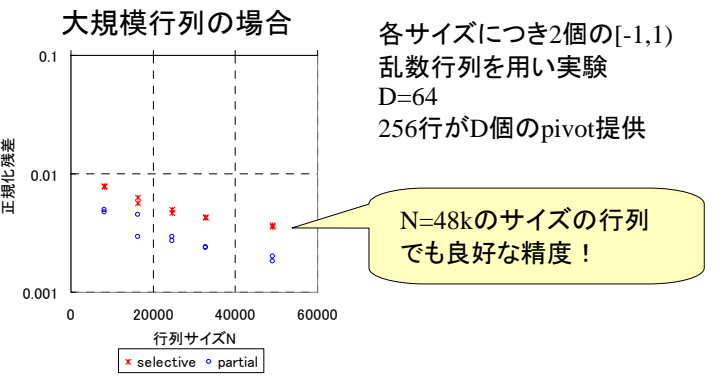
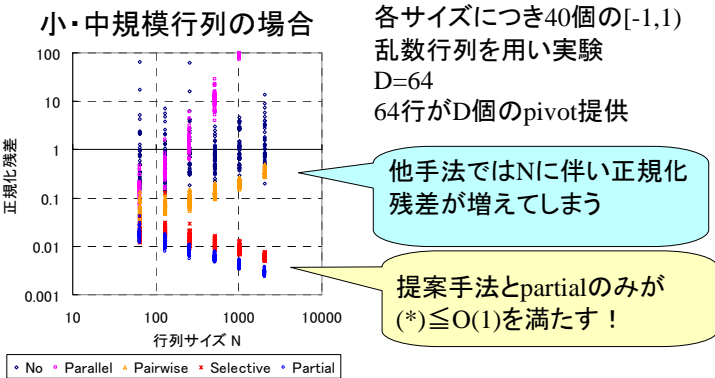
- Phoenix メッセージパッシングライブラリ (金田ら)
- BLAS library by Kazushige Goto

実験結果

計算精度評価

正規化残差 $\frac{\|Ax-b\|_\infty}{\|A\|_\infty \|x\|_\infty N \varepsilon} \dots (*)$ により、計算精度を評価

(*) $\leq O(1)$ であることがTop500での必要条件

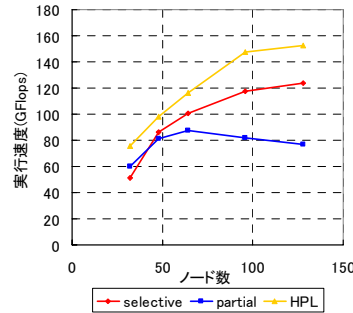


速度性能

東大情報理工Linuxクラスタで実験
CPU: Dual Xeon 2.4/2.8GHz, 1CPUずつ利用
Network: Gigabit ethernet

我々の実装を、HPLと比較
(HPL: Petitetらによるフリーの並列実装。Partial pivoting)

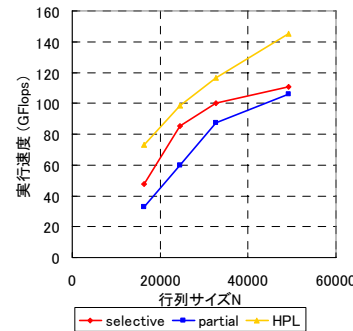
クラスタでのスケーラビリティ



行列サイズN=32768
ブロックサイズSB=256
D=16

HPLに近いスケーラビリティを達成

行列サイズ v.s. 速度



ノード数p=64
ブロックサイズSB=256
D=16

selective, partialとも実装の洗練の余地あり

議論

どんな行列でもうまくいくか?

Selectvie pivotingは各プロセスがそれぞれD個のpivotを見つける方式であるため、各部分行列のランクが全てD未満であると失敗する。たとえば、非零要素が広く散らばったpermutation matrixに対しては失敗する(partial pivotingであれば成功する)。

一方で、上の実験結果は「多くの場合について」selective pivotingが成功することを示している。

では、なぜ「たいてい」うまくいく?

Trefethenらは「平均的に安定」となる条件を論じている:

- (1) Multiplier $(= |a_{ik} / a_{kk}|)$ が1より小さい
 - (2) 各ステップにおける要素変化量の行列のランクが低い
- 上で精度の良くないpairwise, parallel pivotingは(1)を満たすが、(2)を満たしていない。

Selective pivotingは(1)を高い確率で満たそうとする手法である。また、残り行列の全体が1つのpivot行により更新されるため、partial pivotingと同様に(2)を満たす。以上から、平均的安定には(2)を満たす必要があると考えられる。

余計にかかる計算コストは?

Selectvie pivotingでは各プロセスがpivotを選ぶが、最終的に選ばれなかった場合の計算はムダとなる。余分に必要な計算量は、計算全体で $O(DN^2)$ となる。

これは、Dが大きくなければ、本来の計算量 $\frac{2}{3}N^3 + O(N^2)$ に比べて小さい追加であると考えられる。

まとめ

ガウス消去法アルゴリズムを耐遅延にする、selective pivoting手法を提案

- 遅延の影響を大幅に削減 (1/D倍に)
- partial pivotingに匹敵する精度を実現