第 3 回 クラスタシステム上での
並列プログラミングコンテスト

XcalableMP による
並列数値計算プログラミング

電気通信大学大学院 情報理工学研究科 情報・通信工学専攻
2 年次 今村研究室 高橋慶太

課題

Linpack アルゴリズムを XcalableMP を用いて実装
Linpack：連立一次方程式を計算するためのアルゴリズム

予選では問題のサイズが 4096, コア数は 2 のべき乗個 $(1,2,\cdots,128)$ であった.

データの分割

解く対象の行列をどのように分割したか

- 課題におけるコア数が二乗数とは限らない
- 当時の XcalableMP の実装においてまだブロックサイクリックが実装されていなかったこと,
  また XcalableMP の仕様上でも二次元のブロックサイクリック分割 (チェッカーボード分割) を用いた
  部分行列への分割が難しかった

$\Rightarrow$ **列方向のブロックサイクリック分割**

それに合わせてベクトルもブロックサイクリックに分割

データの分割

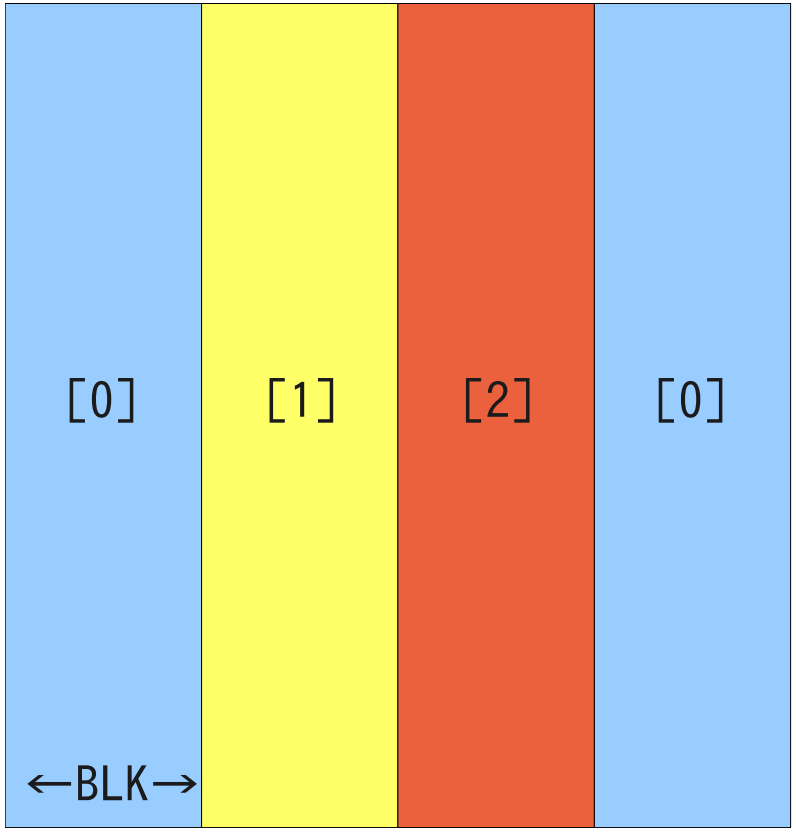

図 1: 行列のブロックサイクリック分割

ブロックサイクリックの実装

XcalableMP でブロックサイクリックが実装されていない

⇓

次のようなプログラムは作成できなかった.

```
double a[NUM][NUM];

#pragma xmp nodes p(*)
#pragma xmp template t(0:4096-1) //NUM-1
#pragma xmp distribute t(cyclic(w)) onto p
#pragma xmp align a[*][i] with t(i)
```

ブロックサイクリックの実装

XcalableMP でブロックサイクリックが実装されていない

⇓

配列の次元を 1 つ増やし, そこでサイクリック分割をすることで擬似的にブロックサイクリック分割をした

```
double  a[DIS][NUM][BLK]; //NUM = BLK*DIS


#pragma xmp nodes p(*)
#pragma xmp template t(0:256-1) //DIS-1
#pragma xmp distribute t(cyclic) onto p
#pragma xmp align a[i][*][*] with t(i)
```

ブロックサイクリックの実装

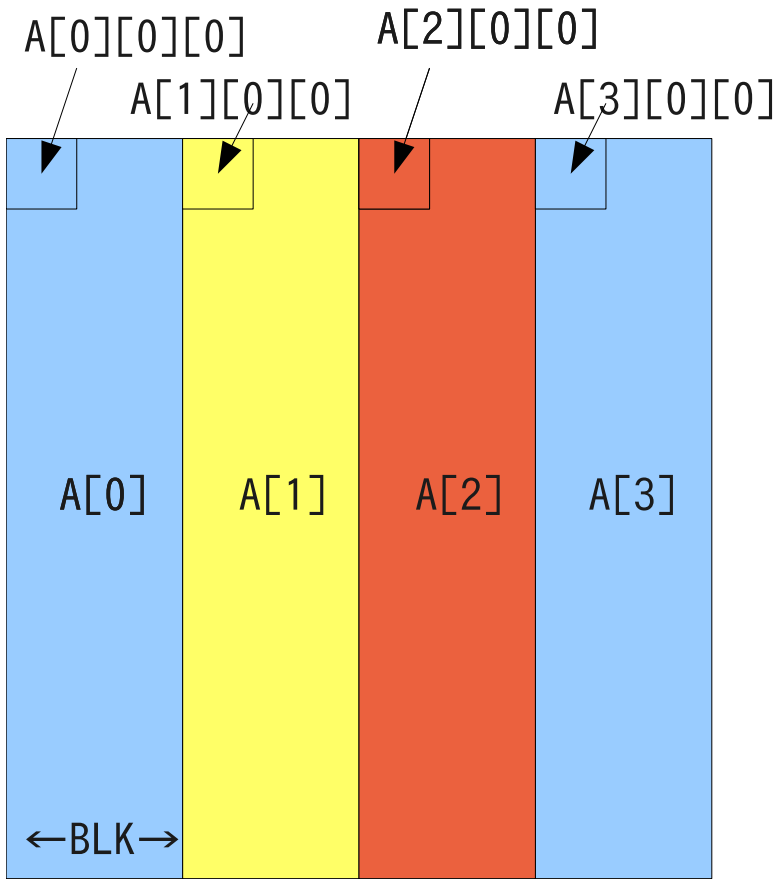

図 2: ブロックサイクリック分割の実装

アルゴリズムの並列化

$n \times n$ 係数行列 A を $n \times m$ の部分行列 $\tilde{A}_i (i = 1, 2, \cdots, n/m)$ に分割し, それぞれの部分行列毎に処理を行う.

1. $\tilde{A}_i$ を持つプロセスがその行列を LU 分解する.
2. LU 分解が終了したらそのプロセスは $\tilde{A}_j (j = i+1, i+2, ..., n/m)$ を持つプロセスに分解した L_i, U_i をブロードキャストで送信する.
3. 送られてきた L_i, U_i を用いて各プロセスは $\tilde{A}_j$ を並列に更新する.
4. $i = i + 1$, $i > n/m$ なら終了. そうでなければ繰り返す.

アルゴリズムの並列化

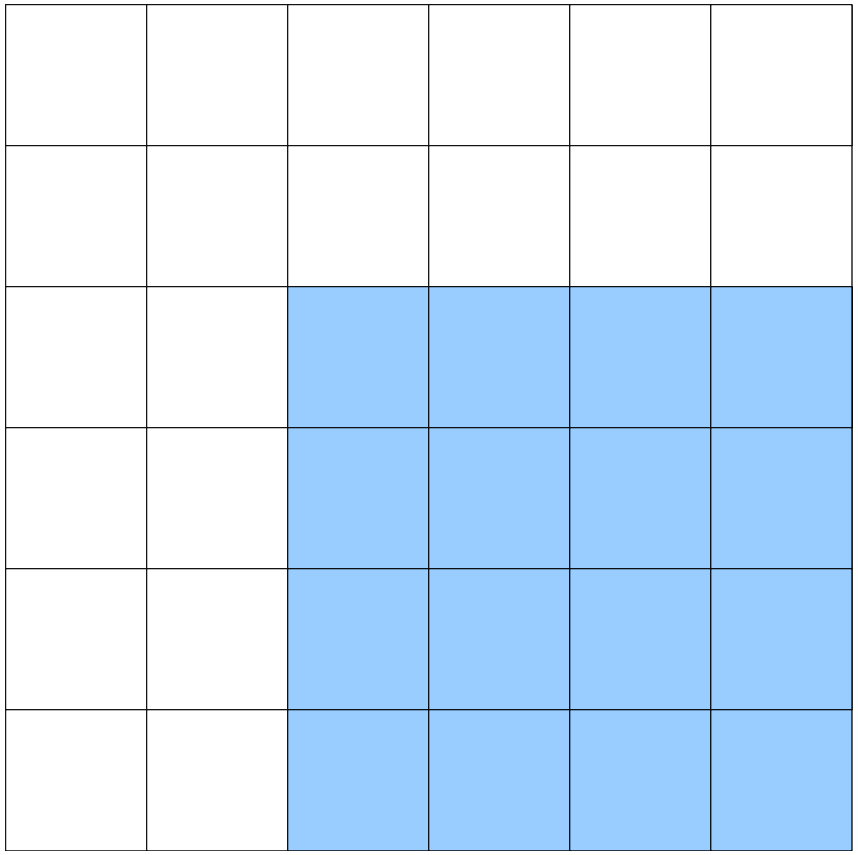

図 3: i ステップ目での動き

アルゴリズムの並列化

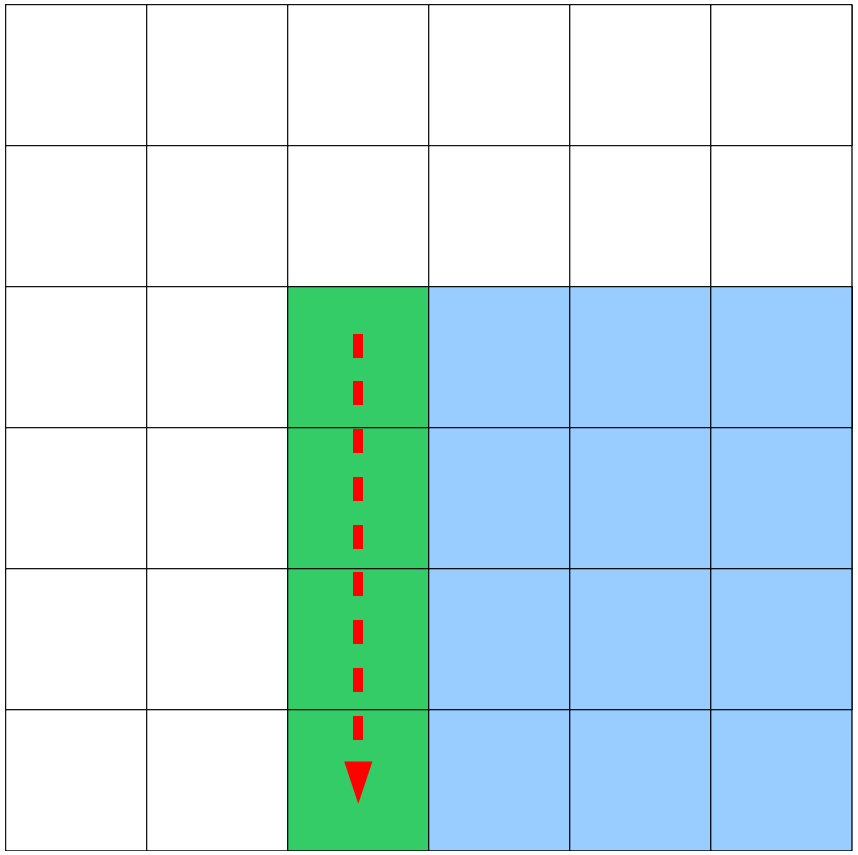

図 4: 1.$\tilde{A}_i$ **の** LU **分解** (1 **コア**)

アルゴリズムの並列化

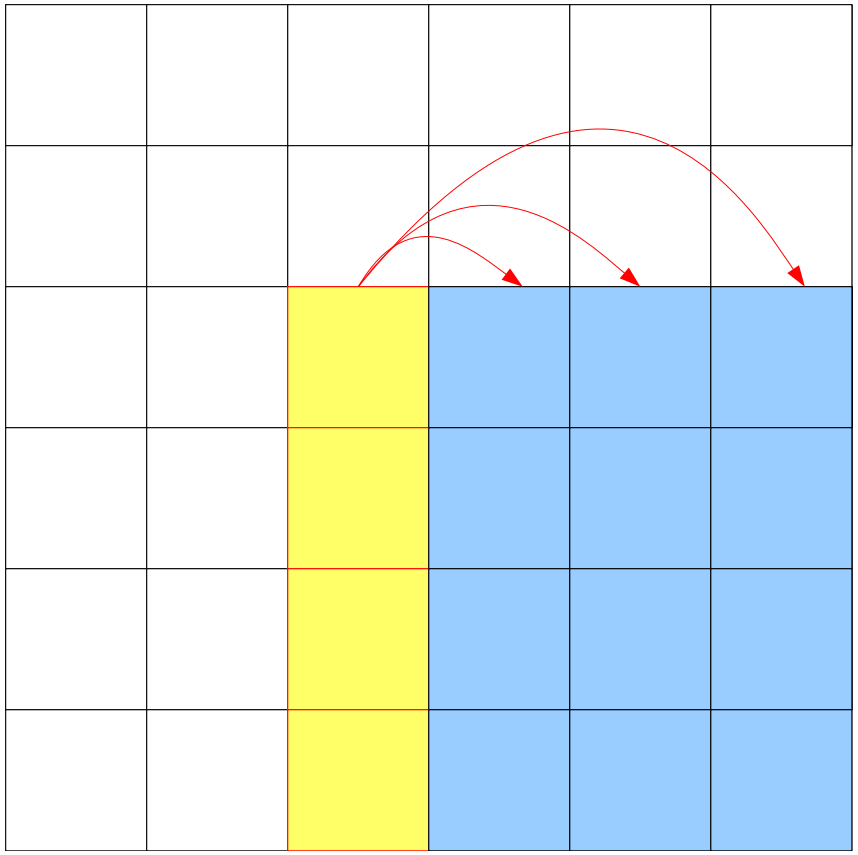

図 5: 2.L_i, U_i **をブロードキャスト**

アルゴリズムの並列化

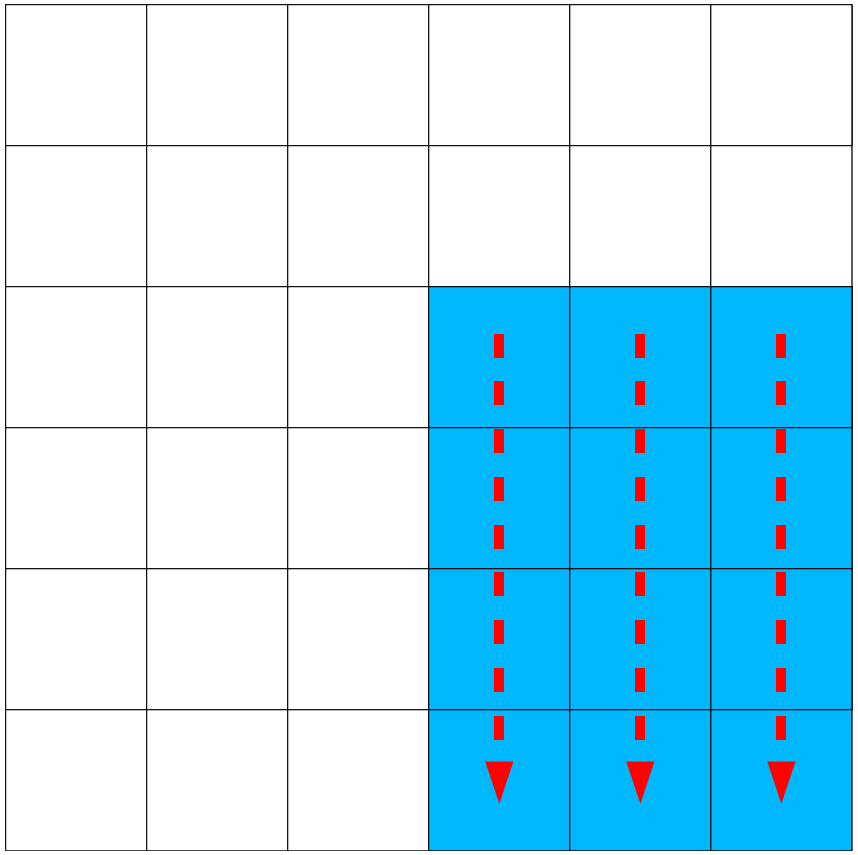

図 6: 3.L_i, U_i **を用いて** $\tilde{A}_j$ **を更新 (並列処理)**

アルゴリズムの並列化

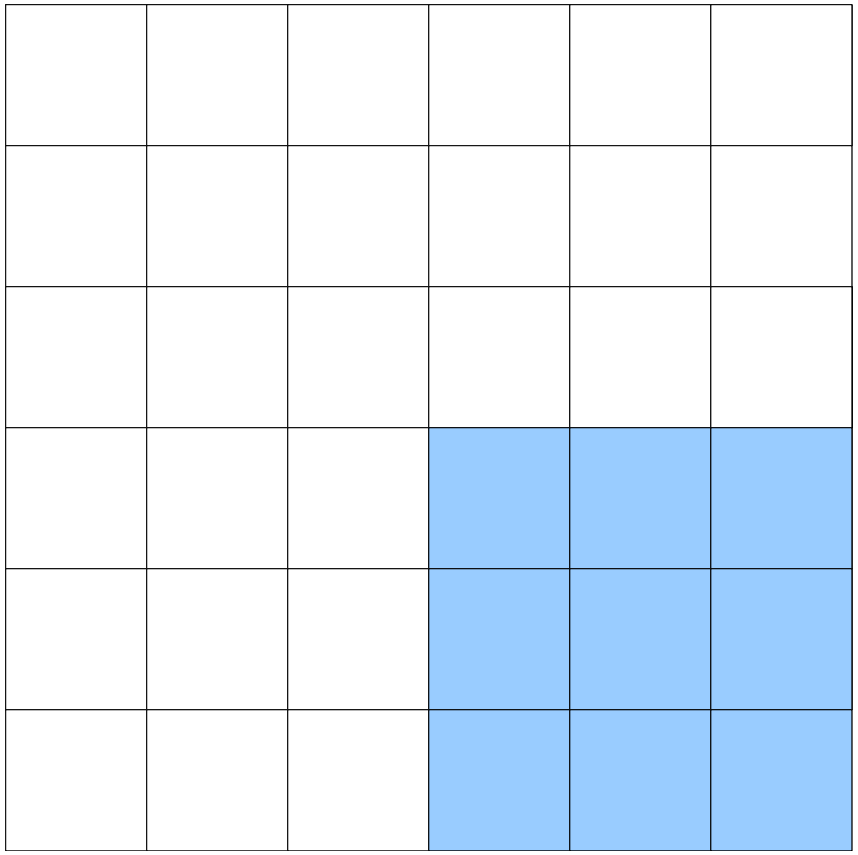

図 7: $4.i$ **ステップ終了**. $i+1$ **ステップへ**

ピボット選択

元のアルゴリズムではピボット選択を使用していた
$\Rightarrow$ 並列プログラムでは $\tilde{A}_i$ の中だけでピボット選択

メリット

- 1 プロセス内だけで処理ができるのでピボット選択や b の要素の入れ替えで通信する必要がない

デメリット

- A 全体からピボットを選択するわけではないため精度が落ちる可能性がある

実行結果

作成したプログラムを実行した結果性能表は表 1 のような結果になった.

表 1: 性能表

コア数	time(s)	**性能値** (GFlops)
1	171.745	0.266945
2	86.8375	0.527958
4	44.4769	1.0308
8	23.9888	1.91116
16	13.5071	3.39426
32	8.53222	5.37334
64	6.30245	7.2744
128	5.75977	7.95978

References

[1] **第 3 回 クラスタシステム上での並列プログラミングコンテスト**
https://www2.cc.u-tokyo.ac.jp/procon2010/

[2] XcalableMP Manual http://www.xcalablemp.org/jp/manual/

[3] **金田　康正　編著**, **並列数値処理 ー高速化と性能向上のためにー** , **コロナ社** (2010)