IS

情報システム学科
*Department of Information Systems*

研究室 並列処理研究室　教員名 水谷　泰治　カテゴリー IT・IoT・AI・ロボティクス

# 教育用並列プログラミング言語を用いた並列プログラミングの学習

## ■ 背景・動機・目的

- CPUのロック周波数の頭打ち（2000年代初頭より）
  - ⇨ CPUの交換だけで劇的に高速化できる時代ではなくなった
- 並列計算可能な装置（マルチコアCPU，クラスタ，GPU等）が増加
  - ➢開発者が明示的に「並列プログラム」を書く必要がある
- 環境毎に使用する言語やライブラリは様々かつ独特
  - ⇨ 初めて並列プログラミングを学ぶ者には学習負荷が高い
- 教育用並列プログラミング言語（T言語）を提案した

T言語による並列プログラミングの学習効果の検証が未調査
⇨ 並列プログラミングの初学者に対して，
①T言語を用いた並列プログラミング演習を実施，
②受講者によるアンケートの実施

## ■ T言語のイメージ

配列総和の逐次プログラム

```
for(i=0; i<N; i++)
  C[i]=A[i]+B[i];
```

Pthreads

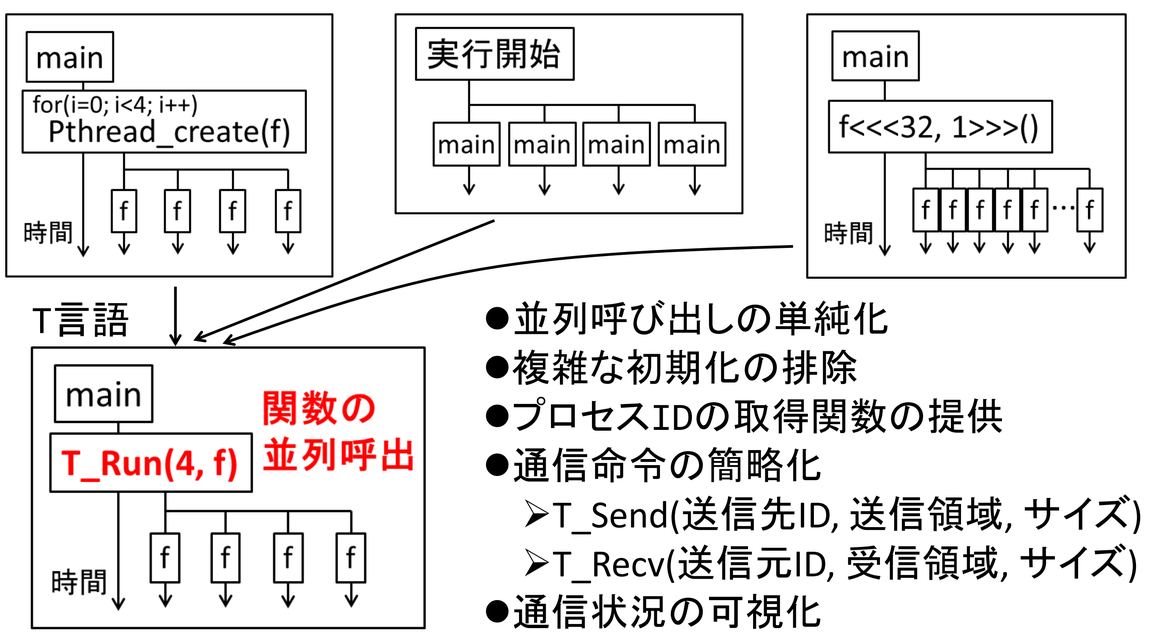
```
#define N 32
int A[N], B[N], C[N];
void *f(void *ptr){
 int i, myrank==(*(int*)(ptr));
 for(i=myrank; i<N; i=i+4){
    C[i]=A[i]+B[i];
 }
}
```

MPI

```
#define N 32
int main(int argc, char *argv[]){
 int A[N/PNUM], B[N/PNUM], C[N], tag=100
 int myrank, size, i;
 MPI_Status s;
 MPI_Init(&argc, &argv);
 MPI_Comm_rank(MPI_COMM_WORLD, &n
 for(i=0; i<N/4; i++){
```

CUDA

```
#define N  32
__global__
void f(int *a, int *b, int *c){
 int i = blockIdx.x;
 c[i] = a[i] + b[i];
}
int main(void) {
 int A[N], B[N], C[N], *deva, *devb, *devc;
 cudaMalloc((void**) &deva, N * sizeof(int));
 cudaMalloc((void**) &devb, N * sizeof(int));
 cudaMalloc((void**) &devc, N * sizeof(int));
 cudaMemcpy(deva, a, N*sizeof(int),
                  cudaMemcpyHostToDevice);
 cudaMemcpy(devb, b, N*sizeof(int),
                  cudaMemcpyHostToDevice);
 f<<<N,1>>>( deva, devb, devc );
 cudaMemcpy(c, devc, N * sizeof(int),
```




- 並列呼び出しの単純化
- 複雑な初期化の排除
- プロセスIDの取得関数の提供
- 通信命令の簡略化
  - ➢T_Send(送信先ID, 送信領域, サイズ)
  - ➢T_Recv(送信元ID, 受信領域, サイズ)
- 通信状況の可視化

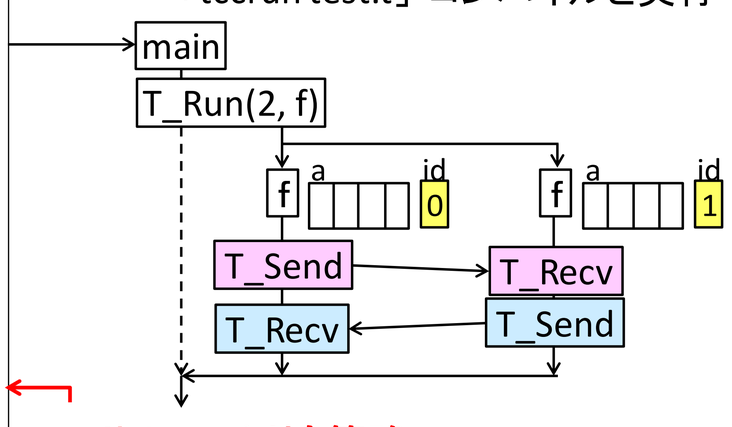
## ■ T言語のプログラム例と実行のイメージ

test.t

```
#include <stdio.h>

int main(void){
  T_Run(2, f);
  return 0;
}

void f(){
  int a[N] = {...};
  int id=T_GetMyNum();
  if (id == 0){
    T_Send(1, a, N*sizeof(int));
    T_Recv(0, a, N*sizeof(int));
  }else if (id == 1){
    T_Recv(1, a, N*sizeof(int));
    T_Send(0, a, N*sizeof(int));
  }
}
```

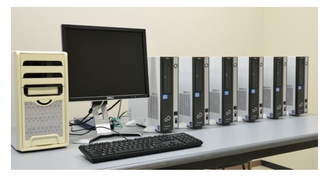



**MPIやCUDAよりも簡略**

```
MPI_Send(a, N, MPI_INT, 1, tag, MPI_COMM_WORLD);
```

```
cudaMalloc((void**)&da, N, sizeof(int));
cudaMemcpy(da, a, N*sizeof(int),
                  cudaMemcpyHostToDevice);
```

## ■ 並列プログラミング演習



- 初めて並列プログラミングを学習する学生を対象
  - ➢並列計算を専門とする研究室に配属された大学4年生8人
  - ➢大学1・2年生レベルのプログラミングは学習済（C, Java）
- 演習内容
  - ➢PCクラスタ上でT言語プログラムを作成
  - ➢性能測定を行い，発表会で結果を発表
- 演習終了後にアンケート（記名式）

| 課題 | 期間 | 概要 | | 言語 |
|---|---|---|---|---|
| 1 | 1週 | 簡単な通信プログラム | | T言語 |
| 2 | 2週 | 行列積（データ分散なし） | | T言語 |
| 3 | 2週 | 行列積（データ分散あり） | | T言語 |
| 4A | 3週 | 選択 | フラクタル画像の生成 | T言語 |
| 4B | | | ガウスの消去法 | T言語 |
| 5M | 2週 | 課題1-4をMPIで再実装 | | C+MPI |
| 5G | | 簡単なGPUプログラム | | CUDA |

課題5のみHPC分野で使われている言語を使用

## ■ アンケート結果(1)

1. T言語における並列プログラミングにおいて，T_Run関数を用いた関数の実体が複数呼び出されることで並列処理を行うことの難易度はどうでしたか？

簡単:5
どちらとも:2
難しい:1

2. T_Run関数で呼び出された各関数はそれぞれが独立に動作しており，それぞれ個別の識別番号が割り振られているという概念の難易度はどうでしたか．

簡単:7
どちらとも:0
難しい:1

▲【自由記述】「初めての並列プログラミングであり，並列に動作するイメージがつかみにくかった」，「初めは難しかった」

3. T_Send関数とT_Recv関数の難易度はどうでしたか？ 簡単:7, どちらとも:0, 難しい:1

4. tccrunコマンドの使い方の難易度はどうでしたか？ 簡単:8, どちらとも:0, 難しい:0

5. T言語プログラムの可視化の機能は使用しましたか？

よく(4回程度以上)使用:0　少し(1-3回程度)使用:2　全く使用せず:6

▲「確認のために一度使用したがとてもわかりづらいと思います」，「特に必要な場面がなかったため」，「見方がよくわからなかった」

7. 課題5でMPIとCUDAを使ったプログラムを作成する際，課題4までで用いたT言語で役に立ったことや役に立たなかったことがあれば教えてください．

- **課題1～4で考え方自体は身に付いた**という回答が多かった

▲「基本的な考え方をある程度理解できていたため，MPIやCUDAの特有の処理も理解しやすかった」，「引数が全然が違っていたのですがやり方自体はT言語と同じなので考え方が役に立ちました」

9. 総合的に考えて，課題1から4までをT言語で学び，最後にMPIやCUDAを学ぶというスタイルは良かったと思いますか？

良かった:6
どちらとも:2
悪かった:0

▲良かった理由:「MPIを扱った最後の課題が簡単だったと思えたのは，その扱い方がT言語の扱い方を元に簡単に類推できるからであり，MPIも簡単に理解できたから」「ローカル言語から世界で使われているものを触るのは良いと思いました．ローカル言語だけだと少し不安でした．

▲どちらとも言えない理由:「もう少しMPIやCUDAの演習を増やした方がいいと思う」，「MPI等の機能をもう少し扱ってみたかった」

## ■ まとめ

- 教育用並列プログラミング言語を用いた並列プログラミング演習
- アンケートの実施
  - ➢T言語のシンプルさに対する肯定的な回答が多かった
  - ➢学習負荷の軽減に一定の効果はあったと考えれる

(1) 水谷泰治, 西口敏司, 橋本渉. "並列プログラミングの学習における教育用並列プログラミング言語の適用". 情報処理学会第80回全国大会, 2G-06, (2018-03).