Tugas Pemrograman Parallel

Oleh:

D121171519 - Glenn Claudio Ivan Petrus

Departemen Teknik Informatika

Fakultas Teknik

Universitas Hasanuddin

2020

Estimasi Nilai Pi dengan Metode Integrasi – Parallel Sections

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>

long long num_steps = 1000000000;
double step;

int main () {
      int i, j, k, l;
      double exec_t, x, pi, sum = 0.0;
      long start, finish;

      start = clock();
      step = 1.0/(double) num_steps;
      #pragma omp parallel sections num_threads(4)
      {
            #pragma omp section
            for(i=0;i<250000000;i++)
            {
                  x = (i+0.5)*step;
                  #pragma omp critical
                  sum += 4.0/(1.0+x*x); // terjadi race condition
            }

            #pragma omp section
            for(j=250000000;j<500000000;j++)
            {
                  x = (j+0.5)*step;
                  #pragma omp critical
                  sum += 4.0/(1.0+x*x); // terjadi race condition
            }

            #pragma omp section
            for(k=500000000;k<750000000;k++)
            {
                  x = (k+0.5)*step;
                  #pragma omp critical
                  sum += 4.0/(1.0+x*x); // terjadi race condition
            }
```

```
            #pragma omp section
            for(l=750000000;l<1000000000;l++)
            {
                  x = (l+0.5)*step;
                  #pragma omp critical
                  sum += 4.0/(1.0+x*x); // terjadi race condition
            }
      }
      pi = step *sum;
      finish = clock();
      exec_t = (double) (finish - start)/CLOCKS_PER_SEC;

      printf("nilai pi %1f\n", pi);
      printf("exec_t %1f\n", exec_t);

      return 0;
}
```

Output Estimasi Nilai Pi dengan Metode Integrasi – Parallel Sections

```
nilai pi 3.141591
exec_t 28.772000
```

Estimasi Nilai Pi dengan Metode Integrasi – Parallel For

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>

long long num_steps = 1000000000;
double step;

int main () {
	int i;
	double exec_t, x, pi, sum = 0.0;
	long start, finish;

	start = clock();
	step = 1.0/(double) num_steps;
	#pragma omp parallel num_threads(4)
	{
		#pragma omp for  schedule(static)
		for(i=0;i<num_steps;i++)
		{
			x = (i+0.5)*step;
			#pragma omp critical
			sum += 4.0/(1.0+x*x); // terjadi race condition
		}
	}
	pi = step *sum;
	finish = clock();
	exec_t = (double) (finish - start)/CLOCKS_PER_SEC;

	printf("nilai pi %1f\n", pi);
	printf("exec_t %1f\n", exec_t);

	return 0;
}
```

Output Estimasi Nilai Pi dengan Metode Integrasi – Parallel For

```
nilai pi 3.141591
exec_t 29.115000
```