IF3230 Sistem Paralel dan Terdistribusi
Praktikum 1 Pengenalan dan Penggunaan OpenMP - Quicksort

A. Persiapan Praktikum

Pada praktikum kali ini anda ditugaskan untuk mengeksplorasi OpenMP, suatu API yang dapat digunakan untuk pemrograman berbasis *shared memory multiprocessing*.

Anda akan menggunakan sebuah mesin dengan IP address 167.205.32.40. Mesin tersebut dapat diakses dari jaringan dalam ITB atau menggunakan fasilitas VPN. Gunakan username berupa NIM dengan password "guest" untuk login. Praktikum menggunakan OpenMP dapat dilakukan pada mesin tersebut. Berikut dilampirkan sebuah program *omp_hello.c* serta cara untuk menjalankan program tersebut.

Petunjuk Kompilasi

Lakukan kompilasi dengan menambahkan argumen -fopenmp. Contoh perintah kompilasi untuk file omp_hello.c

```
gcc -g -Wall -o omp_hello omp_hello.c -fopenmp
```

File *omp_hello.c*

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

void Hello(void); /* Thread function */

int main(int argc, char *argv[]) {
    int thread_count = strtol(argv[1], NULL, 10);

    #pragma omp parallel num_threads(thread_count)
    Hello();

    return 0;
}

void Hello(void) {
    int my_rank = omp_get_thread_num();
    int thread_count = omp_get_num_threads();

    printf("Hello from thread %d of %d\n",
        my_rank, thread_count);
}
```

Petunjuk Menjalankan Program

Untuk menjalankan program yang telah dibuat, perintah yang dapat digunakan adalah sebagai berikut. Angka 4 menunjukkan banyaknya *threads* yang dapat digunakan pada program.

```
./omp_hello 4
```

B. Pengenalan OpenMP

OpenMP adalah API yang dapat digunakan untuk mengatur eksekusi program secara paralel dengan multi-threading dan memory sharing. OpenMP terdiri dari tiga komponen API yaitu *compiler directives*, *runtime library routines*, dan *environment variables*. Compiler directives dipakai untuk mengatur eksekusi kode secara paralel. Directives pada umumnya memengaruhi satu blok setelahnya. Runtime library routines dipakai untuk mendapatkan informasi atau mengatur alur kerja program saat eksekusi program OpenMP. Environment variables dipakai untuk mengubah tingkah laku eksekusi program OpenMP.

Struktur directives untuk C/C++

```
#pragma omp <directive-name> [clause, ...] newline
```

clause bersifat opsional, namun jika ada dapat diberikan tanpa memperhatikan urutan.

Tidak diperbolehkan untuk melakukan *goto* ke dalam atau keluar bagian kode paralel.

Contoh compiler directives
#pragma omp paralel

Contoh runtime library routines
int omp_get_num_threads(void)

Contoh environment variables
export OMP_NUM_THREADS=8

Pada *repository* project telah diunggah beberapa program kecil yang meringkas fungsi-fungsi apa saja yang dapat diterapkan dari OpenMP. Silahkan pelajari program kecil itu sebagai acuan. Adapun untuk mendapatkan penjelasan yang lebih lengkap dapat dilihat pada link berikut.
http://computing.llnl.gov/tutorials/openMP/
http://www.openmp.org/mp-documents/OpenMP3.1.pdf
http://bisqwit.iki.fi/story/howto/openmp/

C. Persoalan, Deliverables, dan Pengumpulan

Anda diminta untuk mengimplementasikan permasalahan *quicksort* paralel menggunakan OpenMP. Karena untuk tugas ini digunakan OpenMP, pertukaran data dilakukan secara *shared memory*. Tugas ini dikerjakan dalam kelompok 2 orang.

Quicksort adalah algoritma sorting yang berbasis *divide and conquer*, dimana sekuens data yang akan di-sort secara rekursif dibagi menjadi 2 bagian, dimana salah satu bagian memiliki elemen yang lebih kecil dibandingkan elemen yang berada pada bagian lain.

Algoritma *quicksort* serial dapat diliat sebagai berikut:

```
void quicksort_seq(int *a, long lo, long hi) {
      long i, div;
      if (lo < hi) {
         int x = a[lo];
         div = lo;
         for (i = lo+1; i < hi; i++) {
              if (a[i] <= x) {
                  div++;
                  swap(a, div, i);
              }
         }
         swap(a, lo, div);
         quicksort_seq(a, lo, div);
         quicksort_seq(a, div+1, hi);
      }
}
```

Paralelisasi dilakukan saat pemanggilan rekursif, masing-masing dijalankan secara paralel. Akan ada nilai tambahan jika proses partisi juga dilakukan secara paralel.

Deliverables yang harus dikumpulkan pada program ini adalah *source code* program anda serta laporan berisi:

1. Deskripsi solusi paralel
2. Analisis solusi
3. Jumlah *thread* yg digunakan
4. Pengukuran performansi untuk tiap kasus uji dibandingkan dgn *quicksort* serial
5. Analisis perbandingan performansi serial & paralel (mengapa?)

Pengukuran performansi dilakukan pada bagian distribusi awal data ke tiap-tiap proses dan saat proses sorting dimulai hingga selesai. Adapun untuk kasus-kasus pengurutannya adalah sebagai berikut.

1. N elemen pada array
    a. Nilai N: 50.000, 100.000, 200.000, 400.000

Mekanisme pengumpulan untuk tugas ini sebagai berikut:

1. Lakukan fork pada repo di gitlab
2. Lakukan git clone (boleh di server)
3. Lakukan commit setelah selesai mengerjakan satu program
4. Lakukan merge request dari repo hasil fork anda di akhir praktikum untuk penilaian dengan judul Praktikum1_<NIM1>_<NIM2>.
5. Batas waktu pengumpulan adalah **Jumat, 17 Februari 2017, 20:00 WIB**
6. Segala bentuk kecurangan akan mendapatkan konsekuensi