PEMBAHASAN UAS KTP 2019

Pilihan Ganda

1. Pemrograman komputasi paralel lebih mudah dilakukan daripada komputasi berurutan (serial) karena pada komputasi paralel, eksekusi program dilakukan oleh beberapa mesin sekaligus. **FALSE**

2. Kecepatan program yang tidak bisa diparalelkan akan membatasi kecepatan program secara keseluruhan. **TRUE? FALSE?**

3. Kecepatan eksekusi program berbanding lurus dengan jumlah prosesor yang digunakan. **TRUE**

4. Thread, jika digunakan pada program untuk eksekusi komputasi secara konkuren, dapat memunculkan masalah race condition

5. Taksonomi Flynn membagi program komputer berdasarkan set instruksi yang dimilikinya saja. **FALSE**

6. Memori dapat didistribusikan secara fisik maupun logik. **TRUE**

7. Arsitektur UMA membuat memori utama dapat diakses dengan latensi yang lebih tinggi dibandingkan bandwidth-nya. **FALSE**

8. Istilah komputasi multicore mengacu pada penggunaan banyak chip prosesor dalam sebuah komputer **FALSE**

9. Massively parallel processor memiliki karakteristik identik dengan cluster, terutama pada jaringan interkoneksi khususnya. **FALSE**

10. Pemrograman komputasi parallel bisa menggunakan mekanisme komunikasi perpesanan message passing interface (MPI) **TRUE**

11. Pemrograman komputasi message-passing mengkodekan paralelisme secara eksplisit. **MUNGKIN TRUE/false**

12. Memprogram dengan paradigma message passing berarti adalah memprogram secara synchronous. **TRUE?**

13. Operasi dasar dalam paradigma komputasi message passing adalah mengirim dan menerima **TRUE?ya**

14. Urutan penerimaan pesan pada penerima tidak bisa dijamin pada blocking message passing operation **FALSE**

15. Solusi untuk masalah idling dan deadlocking yang diuraikan di atas adalah mengandalkan kemampuan API pemrograman **FALSE**

16. GPU dapat menjalankan ratusan atau bahkan ribuan thread sekaligus **TRUE**

17. Bahasa yang digunakan dalam CUDA merupakan perluasan dari bahasa pemrograman C++ **TRUE ?**

18. Arsitektur CUDA membagi sumber daya komputasi menjadi vertex dan pixel shaders **FALSE (ini yang true SHADERS)**

19. Pengembangan program menggunakan CUDA bisa menggunakan compiler C umum seperti GCC **FALSE**

20. Komputer yang memiliki GPU CUDA dan driver perangkat NVIDIA dapat menjalankan kode CUDA C. **TRUE**

21. Untuk mendapatkan speedup melalui implementasi sekuensial, program paralel harus memiliki beberapa proses aktif secara bersamaan, bekerja pada tugas yang berbeda. Mekanisme dimana penugasan task untuk eksekusi proses disebut distributing

22. Dekomposisi ke banyak proses yang memiliki tugas kecil adalah coarse-grained. **FALSE**

23. Algoritma quicksort dapat menggunakan metode dynamic task generation untuk proses task generation **TRUE**

24. Exploratory decomposition biasanya digunakan untuk solusi kasus optimasi **TRUE?**

(Exploratory decomposition is used to decompose problems whose underlying computations correspond to a search of a space for solutions.)

25. Maximum degree of concurrency adalah jumlah rata-rata tugas yang dapat diproses secara paralel selama pelaksanaan program **FALSE**

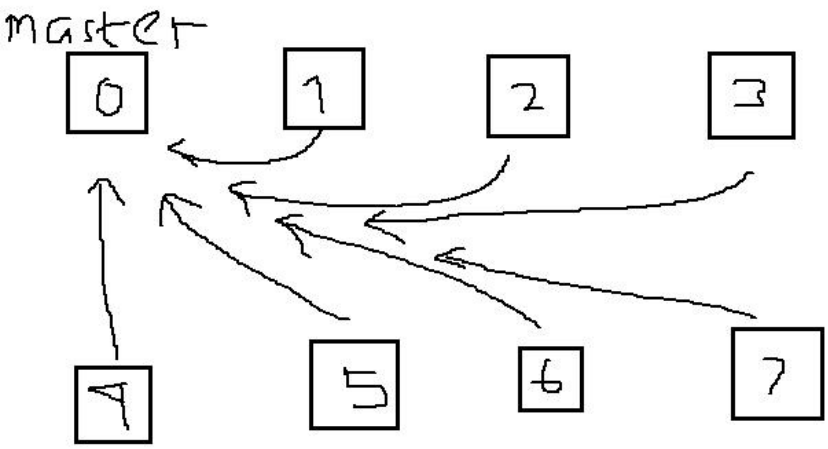
(Maximum Degree of concurrency = jumlah maksimum pekerjaan yang bisa dieksekusi secara bersamaan dalam program paralel pada suatu waktu tertentu.)

Essay

1. Sebutkan dan jelaskan tiga decomposition techniques yang anda ketahui, dan berikan contoh problem yang dapat diselesaikan dengan techniques tersebut!

Jawaban :
1. Data Decomposition : Identify the data on which computations are performed.
• Partition data into sub-units
• Data can be input, output or intermediate for different computations
2. **Rekursive :** Generally suited to problems that are solved using the divide and-conquer strategy.
3. **Exploratory :** In many cases, the decomposition of the problem goes hand in-hand with its execution.
These problems typically involve the exploration (search) of a state space of solutions

2. Perhatikan potogan code sumber MPI berikut. Jika program ini dijalankan menggunakan 8 processor (0,1,2,..7), Jelaskan menggunakan gambar, proses apa yang terjadi antar prosesor?



3. Ubahlah code tersebut menjadi CUDA code, dan tuliskan komentar pada code anda, posisi serial code dan parallel code.

Jawaban :
```
#include <stdio.h>

//Code Serial
void functionSerial(int n, float a, float* x, float* __restrict y)
{
    for (int i = 0; i < n; ++i)
        y[i] = a * x[i] + y[i];
}

//Code Parallel
__global__
```

```
void functionParallel(int n, float a, float* x, float* y)
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < n) y[i] = a * x[i] + y[i];
}

int main(void)
{
    int N = 100;
    float* x, * y, * d_x, * d_y , *d_xx, *d_yy, *yy;
    x = (float*)malloc(N * sizeof(float));
    y = (float*)malloc(N * sizeof(float));

    cudaMalloc(&d_x, N * sizeof(float));
    cudaMalloc(&d_y, N * sizeof(float));
    cudaMalloc(&d_xx, N * sizeof(float));
    cudaMalloc(&d_yy, N * sizeof(float));

    for (int i = 0; i < N; i++) {
        x[i] = 2.0f;
        y[i] = 1.0f;
    }

    cudaMemcpy(d_x, x, N * sizeof(float), cudaMemcpyHostToDevice);
    cudaMemcpy(d_y, y, N * sizeof(float), cudaMemcpyHostToDevice);

    functionParallel << <(N + 255) / 256, 256 >> > (N, 3.0f, d_x, d_y);

    cudaMemcpy(y, d_y, N * sizeof(float), cudaMemcpyDeviceToHost);

    cudaMemcpy(d_xx, x, N * sizeof(float), cudaMemcpyHostToDevice);
    cudaMemcpy(d_yy, y, N * sizeof(float), cudaMemcpyHostToDevice);

    functionSerial(N, 3.0f, d_xx, d_yy);

    cudaMemcpy(yy, d_yy, N * sizeof(float), cudaMemcpyDeviceToHost);

    cudaFree(d_x);
    cudaFree(d_y);
    cudaFree(d_xx);
    cudaFree(d_yy);
    free(yy);
    free(x);
    free(y);
}
```