Problem Session 1 HT23 Parallel Programming Models

Problem 1

Given a sequential vector addition

```
float a[N], b[N], c[N];
for(i : 0..N-1)
  c[i] = a[i] + b[i]
```

N= 1,000,000
System with P processors with shared address space and a shared cache. The size of the cache line is 64 bytes
Each float is a 32-bit number (4 bytes)
The goal is to parallelize this loop. Assume that the overhead of statically scheduling a task is 1ms, and 2ms for the case of dynamic scheduling.

(a) What type of array distribution achieves best locality in this scenario? Why?

- Blockwise
- Cyclic
- Block-cyclic

(b) Now assume P=8. What task sizes will provide good performance? Does the selection of array distribution impact this decision. If yes, suggest a data structure (using C notation) for the parallelized version of the loop.

(c) Assume that the scheduling of each task involves 1ms of scheduling and mapping time. Furthermore the computation of each individual element c[x] = a[x] + b[x] requires 1 microsecond. What will be the execution time for the coarsest granularity.

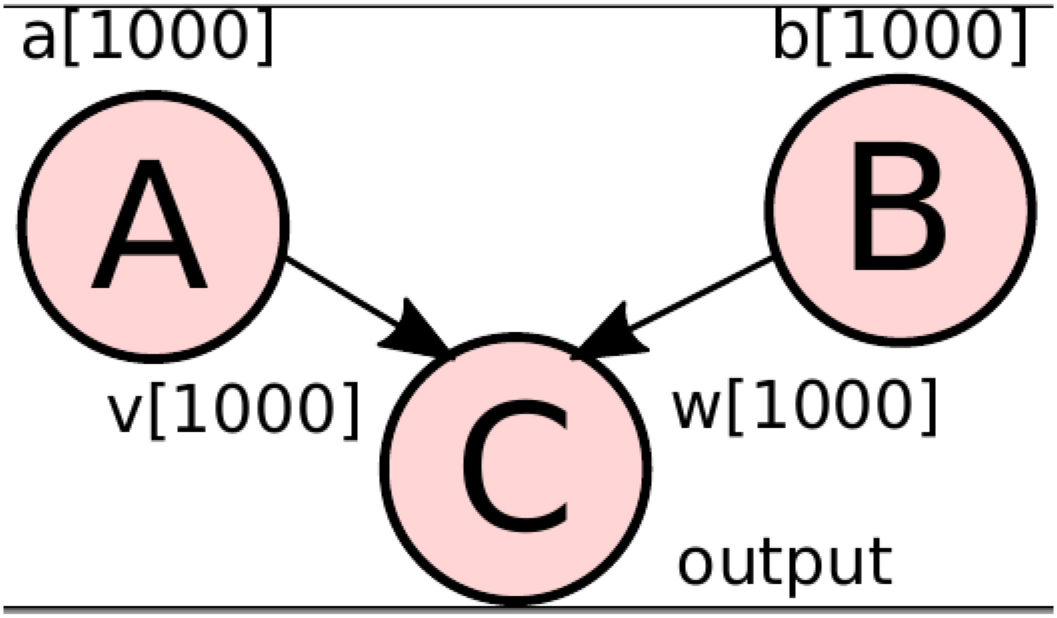
(d) Assume now that P= 10000. Repeat the calculation of the execution time.

(e) What is the optimal granularity that guarantees that less than 10% of time is spent in scheduling? How many processors can we make use of?

(f) Take the granularity resulting from section (d). What is the execution time if we execute the code on P=100 processors?

Problem 2

Given the following simple task graph



Tasks A and B read two vectors a[1000] and b[1000] and produce two vectors v[1000] and w[1000] which are consumed by task C. Task C processes the two vectors and produces one output element of size 8 bytes.
Each element is a floating point element of 8 bytes (type: double).
Two computer systems are considered. A shared address space computer with 2 processors (P0 and P1), and a distributed address space with 2 nodes (D0 and D1), each with one processor each (P0 and P1).
The two vectors a[] and b[] are initially located in the memory attached to processors 0. The problem is completed with the output is written into the memory of P0

Execution times for the tasks are as follows

- Task A, B: 10ms
- Task C:

    - 10 ms (if both arrays are in the local cache)
    - 15 ms (if one of the arrays is not in the local cache)
- The network linking D0 and D1 has a bandwidth of 2MB/s (consider that O/S and NIC overheads are negligible)

The goal of this exercise is to find the best schedule for each computer system (shared memory and distributed memory) that minimizes the execution time. What is the execution time?

Problem 3

Which of the following loops is vectorizable? Explain why.
Assume that none of the vectors a[ ], b[ ], or c[ ] are overlapping

```
void vadd1(double c[], double b[], double a[])
{
for(i=0; i<N; i++)
  c[i] = a[i] + b[i]
}
```

```
void vadd2(double c[], double b[], double a[])
{
for(i=0; i<N-1; i++)
  c[i] = a[i] + a[i+1]
}
```

```
void vadd3(double c[], double b[], double a[])
{
for(i=1; i<N; i++)
  c[i] = a[i] + c[i-1]
}
```

Problem 4

Consider a computation consisting of 1000 seconds (sequential time).
The system will automatically context switch every 100 ms. The cost of context switch is 2us, and the cost of thread creation is 10us.
The selected thread granularity is 1s. All threads are statically created at the beginning.
(a) What is the execution time if the system has 10 or 1000 processors?
(b) Repeat the exercise considering a thread granularity of 1ms

Problem 5

Consider a matrix of dimension 100 x 100. Draw the following data distributions for p processors:
(a). Column-cyclic distribution.
(b). Row-cyclic distribution.
(c). Blockwise column-cyclic distribution with blocksize 5.
(d). Blockwise row cyclic distribution with blocksize 5.

Problem 6

Apply loop coalescing transformation on the following nested loop. Transform it in the singly nested loop with loop variable i. Consider parallel execution of resulted singly nested loop on 8 processors. What array elements would be executed by processor 1?

```
for(j=0; j<5; j++)
  for(k=0; k<3; k++)
    for(l=0; l<5; l++)
      a[j][k][l] = a[j+1][k+1][l] + 1;
```

Problem 7

Given the following nested loop. Draw the tiled iteration space for the loop. Apply loop tiling and interchange operation on the loop with tile_size =2 (=block size).

```
for(i=1; i<=10; i++)
  for(j=1; j<=10; j++)
    a[i][j] = a[i][j-1] + b[i][j];
```