PAR – In-Term Exam – Course 2022/23-Q2

April 26^{th}, 2023

Problem 1 (5.0 points) Given the following code instrumented with *Tareador*:

```
#define N 6  // always a multiple of 2
int m[N][N];
...
// initialization
for (int i=0; i<N-1; i+=2) {   // loop step is 2
   sprintf(label, "init-%d", i);
   tareador_start_task(label);
   for (int k=0; k<N; k++) {
      m[i][k]   = foo(m[i][k], i);
      m[i+1][k] = foo(m[i][k], i);
   }
   tareador_end_task(label);
}

// calculation
for (int i=1; i<N; i++) {  // loop step is 1
   sprintf(label, "calc-%d", i);
   tareador_start_task(label);
   for (int k=0; k<N; k++)
      m[i][k] = goo(m[i-1][k], m[i][k]);
   tareador_end_task(label);
}
```

Assume that the execution time for each `init-i` and `calc-i` task is 20 and 4 time units, respectively, functions `foo()` and `goo()` do not perform any memory access and that the execution cost in time for the rest of the code is negligible. **We ask you to**:

1. (1.0 points) Draw the *Task Dependence Graph* (*TDG*) based on the above *Tareador* task definitions and for $N = 6$. Include for each task its name and its cost in time units. Notice that the outer loop in the initialization phase has a step value of 2.

2. (1.0 points) Compute the values for T_1, T_∞, the amount of *Parallelism* and P_{min}. Draw the temporal diagram for the execution of the *TDG* on P_{min} processors.

   Solution:

3. (1.0 points) Indicate the best assignment for the tasks in the previous *TDG* to 2 processors that minimizes the computation time. Draw the temporal diagram showing the execution of the *TDG* with the proposed mapping. Compute the values for T_2 and S_2.

4. (2.0 points) Consider a distributed memory architecture with P processors and N any value multiple of 2. Let's assume that matrix m is stored in the memory of processor 0 and that the assignment of task to the P processors is as follows:

   - *init-i* tasks are assigned to `P` different processors (i.e. the number of processors `P` is equal to the number of tasks *init-i*).
   - All *calc-i* tasks are assigned to processor 0

We ask you to: Write the expression that determines the execution time, T_p as a function of N (NOT as a function of P, since P is directly related with the value of N), clearly identifying the contribution of the computation time and the data sharing overheads, assuming the data sharing model explained in class in which the overhead to perform a remote memory access is $t_s + t_w \times m$, being t_s the start-up time, t_w the time to transfer one element and m the number of elements to be transferred; at a given time, a processor can only perform one remote access to another processor and serve one remote access from another processor.

Problem 2 (5.0 points) Consider a multiprocessor system with a hybrid NUMA/UMA architecture composed of 2 identical NUMAnodes. Each NUMAnode has 16 Gbytes of main memory and 4 processors, each processor with its own private cache of 16 Mbytes. Memory cache lines are 16 bytes wide and data coherence is guaranteed using a *Write-Invalidate MSI protocol* within each NUMAnode and using a *Write-Invalidate MSU Directory-based* cache coherency protocol between NUMAnodes. Processors 0 to 3 belong to *NUMAnode0* and processors 4 to 7 to *NUMAnode1*. **We ask you to** answer the following two questions:

1. (1.0 point) Compute the total number of bits that are necessary **in each cache memory** to maintain the coherence, indicating the function of those bits.

2. (1.0 point) Compute the total number of bits that are necessary **in each NUMAnode's directory** to maintain the coherence, indicating the function of those bits.

Given the following OpenMP code to be executed on the multiprocessor system described above:

```
#define NUM_THREADS 8
#define M 1
int hist[NUM_THREADS][M];
#pragma omp parallel num_threads(NUM_THREADS)
{
  int id=omp_get_thread_num();
  hist[id][0]=foo();
}
```

and assuming that: 1) no accesses to `hist` are performed before the execution of the parallel region; 2) the initial memory address of `hist` is aligned to the beginning of a memory/cache line and other variables are stored in registers; 3) the size of an `int` data type is 4 bytes; and 4) the Operating System applies the *"First touch"* policy for data allocation in memory. **We ask you to answer the following question:**

3. (0.25 points) How many directory entries will be required to store the coherence information of `hist` and which NUMAnodes will hold them after the execution of the previous parallel region?

Considering the following execution order for the multiple instances of `hist[id][0]=foo()`: *id=0,2,4,6,1,3,5,7* and the fact that function `foo()` does not perform any memory access to any memory line, **we ask you to:**

4. (1.5 points) Complete the table in the provided answer sheet with the required information in its columns: affected $line_m$, hit or miss in cache, CPU command by processor k ($PrRd_k/PrWr_k$), bus transaction(s) by the Snoopy in processor k ($BusRd_k$ / $BusRdX_k$ / $BusUpgr_k$ / $Flush_k$ / $Nothing$), new line state ($I/S/M$) for the copies of $line_m$ in the affected processors, and directory entry information for the affected NUMAnode: number of node, state ($U/S/M$) and presence bits (0/1, where the lowest ordered bit, the rightmost one, corresponds to NUMAnode0).

After the execution of the previous parallel region processor 0 executes the following sequential code:

```
int sum = 0; // sum is stored in a register
for (int i=0; i<8; i++) // i is stored in a register
  sum += hist[i][0];
```

5. **We ask you to answer the following questions:**
   - (0.25 points) During the execution of the sequential code by processor 0, do you expect any coherence protocol transaction/s between the two NUMANodes?
   - (0.25 points) Once the sequential code has been executed, which will be the home NUMAnode for the memory lines containing `hist` accessed by processor 0?
   - (0.25 points) Which will be the value of the state and presence bits in the directory for those memory lines?

And finally,

6. (0.5 points) Do you observe any potential efficiency problem in the parallel region related to the cache coherence protocol? Briefly justify your answer and indicate a modification of the code that avoids this problem.

Student name: ....

Answer for question 2.4

Time Unit		Affected $line_m$	hit or miss	Processor command	Bus transaction/s	Processor : Cache line state	Directory entry		
							NUMAnode #	State	Presence bits
1	hist[0][0]=foo()								
2	hist[2][0]=foo()								
3	hist[4][0]=foo()								
4	hist[6][0]=foo()								
5	hist[1][0]=foo()								
6	hist[3][0]=foo()								
7	hist[5][0]=foo()								
8	hist[7][0]=foo()								