Començat el	diumenge, 14 de novembre 2021, 21:22
Estat	Acabat
Completat el	diumenge, 14 de novembre 2021, 22:04
Temps emprat	42 minuts 13 segons
Qualificació	**9,00** sobre 9,00 (**100**%)

Pregunta **1**

Correcte

Puntuació 1,00 sobre 1,00

For the following parallel computation expressed using tasks

```
#define N 1024
#define MIN 16

void doComputation (int * vector, int n) {
   int size = n / 4;
   for (int i = 0; i < n; i += 4)
      tareador_start_task("compute");
      compute(&vector[i], size);
      tareador_end_task("compute");
}

void partition (int * vector, int n) {
   if (n > MIN) { // MIN is multiple of 4
      int size = n / 4;
      for(int i=0; i<4; i++)
         partition(&vector[i*size], size);
      }
   else
      doComputation(vector, n);
   return;
}
void main() {
   ...
   partition (vector, N); // N is multiple of 4
   ...
}
```

How many tasks are generated during the execution of the program?

Resposta: 256 ✔

256 tasks will be generated, where each task will work on 4 elements of the vector.

Pregunta **2**

Correcte

Puntuació 1,00 sobre 1,00

If the granularity of a task is defined as the number of invocations to function `compute` (e.g. granularity 4 means that tasks perform 4 invocations of that function). Which is the granularity for the tasks in this task decomposition?

Resposta: 1 ✔

Each task performs 1 invocation to function `compute`.

Pregunta **3**

Correcte

Puntuació 1,00 sobre 1,00

Assuming that the execution of each `compute` invocation takes 16 time units, the creation of each individual task takes 2 time units and that we neglect the execution time of the rest of the code, calculate the parallel execution time for an infinite number of processors in the system (T_∞).

Resposta: 528 ✔

Tasks are created sequentially in this code. And the creation of each task takes 2 time units. Therefore, we need 512 time units to create all the tasks.

Having infinite processors we can start executing tasks as soon as they have been created (we assume no dependencies). Therefore, most tasks will do their work while we are creating tasks. Only the lat task created will not fully overlap its execution with creation of other tasks. Thus, we need to add 16 additional time units to finish the computation.

Consequently, 512+16=528

Pregunta **4**

Correcte

Puntuació 1,00 sobre 1,00

If the task decomposition is changed to

```
#define N 1024
#define MIN 16

void doComputation (int * vector, int n) {
   int size = n / 4;
   for (int i = 0; i < n; i += 4)
      compute(&vector[i], size);
}

void partition (int * vector, int n) {
   if (n > MIN) { // MIN is multiple of 4
      int size = n / 4;
      for(int i=0; i<4; i++)
         partition(&vector[i*size], size);
      }
   else
      tareador_start_task("doComputation");
      doComputation(vector, n);
      tareador_end_task("doComputation");
   return;
}
void main() {
   ...
   partition (vector, N); // N is multiple of 4
   ...
}
```

How many explicit tasks are generated during the execution of the program?

Resposta: 64 ✔

Pregunta **5**

Correcte

Puntuació 1,00 sobre 1,00

If the granularity of a task is defined as the number of invocations to function `compute` (e.g. granularity 4 means that tasks perform 4 invocations of that function). Which is the granularity for the tasks in this task decomposition?

Resposta: 4 ✔

Pregunta **6**

Correcte

Puntuació 1,00 sobre 1,00

Assuming that the execution of each compute invocation takes 16 time units, the creation of each individual task takes 2 time units and that we neglect the execution time of the rest of the code, calculate the parallel execution time for an infinite number of processors in the system (T_∞).

Resposta: 192 ✔

Well done!

Pregunta **7**

Correcte

Puntuació 1,00 sobre 1,00

Finally, if the task decomposition is changed to

```
#define N 1024
#define MIN 16

void doComputation (int * vector, int n) {
   int size = n / 4;
   for (int i = 0; i < n; i += 4)
      compute(&vector[i], size);
}

void partition (int * vector, int n) {
   if (n > MIN) { // MIN is multiple of 4
      int size = n / 4;
      for(int i=0; i<4; i++)
         tareador_start_task("partition");
         partition(&vector[i*size], size);
         tareador_end_task("partition");
      }
   else
      doComputation(vector, n);
   return;
}
void main() {
   ...
   partition (vector, N); // N is multiple of 4
   ...
}
```

How many explicit tasks are generated during the execution of the program?

Resposta: 84 ✔

Pregunta **8**

Correcte

Puntuació 1,00 sobre 1,00

If the granularity of a task is defined as the number of invocations to function `compute` (e.g. granularity 4 means that tasks perform 4 invocations of that function). Which is the granularity for those tasks in this task decomposition that end up executing instances of function `compute`?

Resposta: 4 ✔

Pregunta **9**

Correcte

Puntuació 1,00 sobre 1,00

Assuming that the execution of each compute invocation takes 16 time units, the creation of each individual task takes 2 time units and that we neglect the execution time of the rest of the code, calculate the parallel execution time for an infinite number of processors in the system (T_∞).

Resposta: 88 ✔

Well done!

◄ Questions after video lesson 6

Salta a...

Additional material ►