2º curso / 2º cuatr.

Grado en Ing. Informática

Doble Grado en Ing. Informática y Matemáticas

Arquitectura de Computadores: Exámenes y Controles

Examen de Prácticas AC 05/07/2013 resuelto

Material elaborado por los profesores responsables de la asignatura:
Mancia Anguita, Julio Ortega

1 Enunciado Examen de Prácticas del 05/07/2013

Cuestión 1.(0.5 puntos) **(a)** (0.05) ¿Cuántos nodos de cómputo (servidores) tiene atcgrid y cuántos chips de procesamiento (encapsulados, procesador) tiene una placa madre de atcgrid? **(b)** (0.05) ¿Cuántos cores de procesamiento tiene un chip de procesamiento (encapsulado) de atcgrid? **(c)** (0.05) ¿Qué gestor de colas ha utilizado en prácticas para enviar trabajos a atcgrid? **(d)** (0.25) Suponga que debe ejecutar en atcgrid el fichero ejecutable `hello` que tiene en su home del PC del aula de prácticas (o en su PC portátil si está trabajando en el aula de prácticas con su propio portátil), ¿qué haría para ejecutarlo en atcgrid? ¿Cómo sabría que ya ha terminado la ejecución? ¿dónde podría consultar los resultados de la ejecución? **(e)** (0.1) Suponga que debe ejecutar en atcgrid el *script* `scripthello.sh` que tiene en su home del PC del aula de prácticas (o en su PC portátil si está trabajando en el aula de prácticas con su propio portátil), ¿qué haría para ejecutarlo?

Cuestión 2. (1.2 puntos) Conteste a las siguientes cuestiones sobre el código `examen1.c` de la siguiente figura (considere que la variable de control `dyn_var` está a `false`):

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
int main(int argc, char **argv)
{  int i, n=20, tid, chunck;
   int a[n],suma=0,sumalocal;
   if(argc < 3)    {
      fprintf(stderr,"[ERROR]- Faltan datos de entrada \n");    exit(-1);
    }
   n = atoi(argv[1]); if (n>20) n=20; chunck = atoi(argv[2]);
   for (i=0; i<n; i++)    a[i] = i;
   #pragma omp parallel num_threads(4) default(none) private(sumalocal,tid) shared(a,suma,n,chunck)
  {  sumalocal=0;    tid=omp_get_thread_num();
     #pragma omp for private(i) schedule(static, chunck)
     for (i=0; i<n; i++)
     {   sumalocal += a[i];
        printf(" Thread %d suma de a[%d]=%d sumalocal=%d \n",tid,i,a[i],sumalocal);
     }
     #pragma omp atomic
        suma += sumalocal;
  }
  printf("Suma=%d\n", suma);
}
```

(a) (0.15) Indique qué orden usaría para compilar el código de `examen1.c` desde una ventana de comandos (*Shell* o intérprete de comandos) si el ejecutable se quiere llamar `examen1` (utilice en la compilación alguna de las opciones de optimización que ha utilizado en la práctica de optimización de código).

(b) (0.25) En el código hay tres puntos en los que se puede imprimir en pantalla: un `fprintf` y dos `printf`. Razone qué `fprintf y printf` de los que hay en el código se ejecutan y cuántas veces se ejecuta cada uno de ellos.

(c) (0.15) ¿Cuántos parámetros de entrada necesita el programa y para qué se utiliza cada uno de ellos en el código?

(d) (0.4) Escriba **todo** lo que imprime el programa (hay que poner **todo el texto que imprime un thread cada vez que imprime algo por pantalla**) si el usuario ejecuta (razone sus respuestas):

```
(1). examen1 4
(2). examen1 8  1
(3). examen1 8  2
```

(e) (0.25) Elimine del código todos los `default`, `private` y `shared` y realice en el código las modificaciones que sean estrictamente necesarias (sin añadir `default`, `private`, `firstprivate`, `lastprivate` y `shared`) para que imprima exactamente lo mismo que imprimía en la versión original en el último `printf` del código.

Cuestión 3.(0.3 puntos) Indique cuál de los siguientes códigos ofrece mejores prestaciones y explique los motivos:

```
for (i=0;i<100;i++)
    if ((i%2) = = 0)
          a[i]=x;
    else
         a[i]=y;
```

```
for (i=0;i<100;i+=2) {
      a[i]=x;
      a[i+1]=y; }
```