

CLASE PREXAMEN MPI

La forma del examen igual que la anterior: 30min TEST + 1h 5min PROG
Recordar la modificación del secuencial en un archivo a parte.

Sobre entrega de

PRACTICAS: Antes de examen ordinario

Paralelismo básico
ya es en 7 de rotas
mínimo de programación
adecuada a lo mejor
que se piden

NOTA: Mejor dejar
PRINT desactivado
en la entrega.

Secuencial

```

6 #include <stdio.h>
7 #include <stdlib.h>
8 #include <time.h>
9 #include <sys/time.h>
10 #include "getmem.h"
11 #include "argshand.h"
12 #include "utils.h"
13
14 #define DEBUG 1
15 #define PRINT 0
16
17 /*-----*/
18 void ParametersError()
19 {
20     puts("Options are:");
21     puts("-t -h To show this help");
22     puts("-t -r <n rows>      ");
23     exit(0);
24 }
25
26 /*-----*/
27 //Init values of a vector of size Rows.
28 void InitVectorInt(int *pVector, const int Rows)
29 {
30     for (int i=0;i<Rows;i++)
31         pVector[i]=lrand48()%10;
32 }
33
34 /*-----*/
35 //Print a vector of size Rows
36 void PrintVectorInt(int *pVector, const int Rows, char * String)
37 {
38     printf("%s ----- \n", String);
39     for (int i=0;i<Rows;i++)
40         printf("%d", pVector[i]);
41     puts("");
42 }
43
44 /*-----*/
45 //Adds Vector1 and Vector2 in Vector3
46 void Add2VectorInt(int *pVector1, int *pVector2, int *pVector3,
47                     const int Rows)
48 {
49     for (int i=0;i<Rows;i++)
50         pVector3[i]=pVector1[i]+pVector2[i];
51 }
52
53 /*-----*/
54 int main(int argc, char **argv)
55 {
56     int Rows;
57     int * pVector1, * pVector2, * pVector3;
58
59     if (ExistArg("-h",argc,argv))
60         ParametersError();
61
62     if (!ExistArg("-r",argc,argv))
63     {
64         fputs("Parameter -r is necessary.\n",stderr);
65         ParametersError();
66     }
67     else
68     {
69         Rows = atoi(Arg("-r",argc,argv));
70         if (Rows <1)
71         {
72             puts("Rows<1");
73             exit(1);
74         }
75     }
76 }
77
78 printf("Rows=%d.\n", Rows);
79
80 //Init seed of random number generator
81 srand48(0);
82
83 //Calloc of Getmem put data to zeroes
84 pVector1 = (int *) GetMem(Rows,sizeof(int),"Main:Vector1");
85 pVector2 = (int *) GetMem(Rows,sizeof(int),"Main:Vector2");
86 pVector3 = (int *) GetMem(Rows,sizeof(int),"Main:Vector3");
87
88 //Init values of Vector1 and 2
89 #if (DEBUG==1)
90     struct timeval tv1,tv2,tv3,tv4; struct timezone tz;
91     gettimeofday(&tv1,&tz);
92 #endif
93     InitVectorInt (pVector1,Rows);
94     InitVectorInt (pVector2,Rows);
95 #if (DEBUG==1)
96     gettimeofday(&tv2, &tz);
97 #endif
98
99 #if (PRINT==1)
100     PrintVectorInt(pVector1,Rows,"Vector1");
101     PrintVectorInt(pVector2,Rows,"Vector2");
102 #endif
103
104 #if (DEBUG==1)
105     gettimeofday(&tv3, &tz);
106 #endif
107     Add2VectorInt(pVector1,pVector2,pVector3,Rows);
108 #if (DEBUG==1)
109     gettimeofday(&tv4, &tz);
110     double TInitV = (tv2.tv_sec- tv1.tv_sec) + (tv2.tv_usec-tv1.tv_usec)*1e-6;
111     printf("Tiempo de inicializar los vectores = %12.4g sec\n", TInitV);
112     double TSUMV = (tv4.tv_sec- tv3.tv_sec) + (tv4.tv_usec-tv3.tv_usec)*1e-6;
113     printf("Tiempo de sumar los vectores = %12.4g sec\n", TSUMV);
114     printf("Tiempo de inicializar y sumar los vectores = %12.4g sec\n",
115           TInitV+TSUMV);
116 #endif
117
118 #if (PRINT==1)
119     PrintVectorInt(pVector3,Rows,"Vector1+Vector2");
120 #endif
121
122 free((void *)pVector3);
123 free((void *)pVector2);
124 free((void *)pVector1);
125
126 return 0;
127 }
```

Librerías

Parametros del
programa

Inicialización
de vector

Mostrar
vector

Operaciones
de vectores

Obtener memoria

Empieza a contar tiempo

Inicializaciones

Termina contador para inicialización

Empieza contador para operación

Operación

Termina contador de operación

Imprime tiempos

Liberar memoria

NPI **Básico** **NOTA** **OBTENIBLE** **~7**

```

6 #include <stdio.h>
7 #include <stdlib.h>
8 #include <time.h>
9 #include "mpi.h"
10 #include "getmem.h"
11 #include "argshand.h"
12 #include "utils.h"
13
14 #define PRINT 1
15
16 /*-----*/
17 void ParametersError()
18 {
19     puts("Options are:");
20     puts("-t -h To show this help");
21     puts("-t -r <n rows>      ");
22     exit(0);
23 }
24
25
26 /*-----*/
27 //Init values of a vector of size Rows.
28 void InitVectorInt(int *pVector, const int Rows)
29 {
30     for (int i=0;i<Rows;i++)
31         pVector[i]=lrand48()%10;
32 }
33
34
35 //Print a vector of size Rows
36 void PrintVectorInt(int *pVector, const int Rows, char * String)
37 {
38     printf("%s ----- \n", String);
39     for (int i=0;i<Rows;i++)
40         printf("%d", pVector[i]);
41     puts("");
42 }
43
44
45 //Adds Vector1 and Vector2 in Vector3
46 void Add2VectorInt(int *pVector1, int *pVector2, int *pVector3,
47                     const int Rows)
48 {
49     for (int i=0;i<Rows;i++)
50         pVector3[i]=pVector1[i]+pVector2[i];
51 }
52
53
54 int main(int argc, char **argv)
55 {
56     int Rows;
57     int * pVector1, * pVector2, * pVector3;
58
59     if (ExistArg("-h",argc,argv))
60         ParametersError();
61
62     if (!ExistArg("-r",argc,argv))
63     {
64         fputs("Parameter -r is necessary.\n",stderr);
65         ParametersError();
66     }
67     else
68     {
69         Rows = atoi(Arg("-r",argc,argv));
70         if (Rows <1)
71         {
72             puts("Rows<1");
73             exit(1);
74         }
75     }
76 }
77
78 printf("Rows=%d.\n", Rows);
79
80 //Init seed of random number generator
81 srand48(0);
82
83 //Calloc of Getmem put data to zeroes
84 pVector1 = (int *) GetMem(Rows,sizeof(int),"Main:Vector1");
85 pVector2 = (int *) GetMem(Rows,sizeof(int),"Main:Vector2");
86 pVector3 = (int *) GetMem(Rows,sizeof(int),"Main:Vector3");
87
88 //Init values of Vector1 and 2
89 #if (PRINT==1)
90     struct timeval tv1,tv2,tv3,tv4; struct timezone tz;
91     gettimeofday(&tv1,&tz);
92 #endif
93     InitVectorInt (pVector1,Rows);
94     InitVectorInt (pVector2,Rows);
95 #if (DEBUG==1)
96     gettimeofday(&tv2, &tz);
97 #endif
98
99 #if (PRINT==1)
100     PrintVectorInt(pVector1,Rows,"Vector1");
101     PrintVectorInt(pVector2,Rows,"Vector2");
102 #endif
103
104 #if (DEBUG==1)
105     gettimeofday(&tv3, &tz);
106 #endif
107     Add2VectorInt(pVector1,pVector2,pVector3,Rows);
108 #if (DEBUG==1)
109     gettimeofday(&tv4, &tz);
110     double TInitV = (tv2.tv_sec- tv1.tv_sec) + (tv2.tv_usec-tv1.tv_usec)*1e-6;
111     printf("Tiempo de inicializar los vectores = %12.4g sec\n", TInitV);
112     double TSUMV = (tv4.tv_sec- tv3.tv_sec) + (tv4.tv_usec-tv3.tv_usec)*1e-6;
113     printf("Tiempo de sumar los vectores = %12.4g sec\n", TSUMV);
114     printf("Tiempo de inicializar y sumar los vectores = %12.4g sec\n",
115           TInitV+TSUMV);
116 #endif
117
118 #if (PRINT==1)
119     PrintVectorInt(pVector3,Rows,"Vector1+Vector2");
120 #endif
121
122 free((void *)pVector3);
123 free((void *)pVector2);
124 free((void *)pVector1);
125
126 return 0;
127 }
```

→ Siempre no olvidar librería MPI

operación se
mantiene igual
(como norma no se modifica
a menos que haga falta)

Iniciar parte paralela

Parámetros
solo por
hilo maestro
ya que es
muy simple

→ Inicialización (sin paralelizado)

Enviamos a todas las tareas
el número de elementos de
los vectores ya que
solo lo leyo la tarea 0

Esta parte
es muy similar
a HowToShareAVector
de PThreads

asignación de memoria
El tamaño es el de
su Rank

Hacemos dos Scatter
que lo manda 0 (maestro)

En vez de llamar al método
lo hace aquí. Se podría quitar
indirecto el método.

→ Se hace un Gather para
recuperar la información

NO OLVIDAR
Liberación de memoria siempre :)
(valigrid lo agradece)

→ Fin de parte paralela

