ACSO II Memoria de la Práctica 4. Juan Domingo Jiménez Jerez

Contenido: Introducción
Análisis (tablas)
Comparación de resultados (gráficas)

Introducción

La práctica se ha realizado con un ordenador portátil con procesador Intel Core i5-320M CPU @ 2.60GHz, el cual dispone de 2 núcleos y 4 procesadores lógicos. Por lo tanto he comprobado el funcionamiento de las distintas soluciones simulando en la máquina virtual 1,2,3 y 4 procesadores aunque solo disponga de 2 procesadores reales.

Los parámetros de todas las versiones se leerán de un fichero llamado *mandel.params.* El profesor me dijo que lo situase en *$HOME/mandel.params* , pero no podía leerlos de ahí porque necesitaba permisos de ROOT. Así que decidí situarlo en *$HOME/juando/pvm3/mandel.params.* Siendo */juando/* mi user.
Por lo tanto, si se quiere simular en otro CPU habrá que modificar el código y cambiar la ruta.

La carpeta pvm3 contiene varias soluciones:

Secuencial_ppm: Primera versión secuencial que dibuja una región seleccionable de la figura de Mandelbrot en un archivo .ppm.

Paralelo1: Versión original del algoritmo PVM modificada para que genere un archivo .ppm con la región deseada de la figura de Mandelbrot .

Paralelo2: Segunda versión del algoritmo PVM. Esta versión reparte las líneas de otra forma a los trabajadores.

Análisis

En este apartado se analizarán los tiempos obtenidos para cada versión, simulando 1,2,3 y 4 procesadores, para la resolución máxima (4096x4096) y para una resolución medio- baja (1024x768), el número de trabajadores a analizar aumentará exponencialmente.
Antes de mostrar los resultados es conveniente explicar mi modificación de la versión paralela.

Paralelo2 es una versión propia del algoritmo PVM. Esta versión reparte las líneas de otra forma a los trabajadores. Para su realización he partido de la base de que no todas las líneas requieren la misma carga de trabajo para ser calculadas puesto que no todas necesitan el mismo número de iteraciones. Tal y como estaba planteada la versión original es más fácil que tengan más trabajo unos trabajadores que otros ya que cada trabajador trabaja con un conjunto de líneas en orden. Paralelo2 reparte las líneas por igual en la medida de lo posible para cada trabajador y por lo tanto es más justo a la hora de repartir las líneas, cada línea se repartirá a un trabajador.
Ejemplo con 3 trabajadores:

Trabajador 1: 1,4,7,…….SY-2
Trabajador2: 2,5,8,…….SY-1
Trabajador3: 3,6,9,…….SY

Para implementarlo se realizó una pequeña modificación en el código del archivo trabajador.c:

Versión Original (Paralelo1)

```
   a= SY / cuantos;
   b= SY % cuantos; // modulo.

   mias=a;           // seguo que me tocan
   inicial= a*un_id; // Que línea es la pr

   if(b){
     // no es un divisor exacto;
     if(b>un_id){
       mias=mias+1;
       inicial= inicial + un_id; // las qu
     }else{
       //Se encargan los anteriores
       inicial= inicial + b; // las que ha
     }// se desplza la inicial de cada uno
   }
 }

 linea=inicial;
 while(linea<(mias+inicial)){
   unsigned short *a = datos;
  for(col=0;col<SX;col++)
     *(a++)= Mandel(col,linea,n_col);

   pvm_initsend(PvmDataRaw);  // INforma a
   pvm_pkuint(&linea,1,1);
   pvm_pkushort(datos,SX,1);

   if(pvm_send(masterTid,RESULTADOS_MSG)){
     pvm_perror("[MASTER] al mandadr un nu
     pvm_exit();
     exit(1);
   }
   linea++;
 }
```

Versión modificada (Paralelo2)

```
   a= SY / cuantos;
   b= SY % cuantos; // modulo.

   mias=a;           // seguo que me tocan
   inicial=un_id; // Que línea es la primer

   if(b){
     // no es un divisor exacto;
     if(b>un_id){
       mias=mias+1;
     }// se desplza la inicial de cada uno
   }
 }

 linea=inicial;

 while(linea<=(SY-cuantos+inicial)){
   unsigned short *a = datos;
  for(col=0;col<SX;col++)
     *(a++)= Mandel(col,linea,n_col);

   pvm_initsend(PvmDataRaw);  // INforma a
   pvm_pkuint(&linea,1,1);
   pvm_pkushort(datos,SX,1);

   if(pvm_send(masterTid,RESULTADOS_MSG)){
     pvm_perror("[MASTER] al mandadr un num
     pvm_exit();
     exit(1);
   }
   linea+=cuantos;
 }
```

Después de comprobar que la modificación del programa funcionaba correctamente solo queda comprobar si se consigue una mejora de tiempo. Como veremos en las tablas de las mediciones, con paralelo2 obtenemos exactamente los mismo tiempos que con paralelo1 aunque el reparto de las líneas debería de favorecer claramente a paralelo2, esto me hace pensar que el cuello de botella se encuentra en la cola de recepción del maestro ya que los trabajadores envían las líneas al maestro en cuanto las tienen calculadas y la cola de recepción del maestro puede crecer más deprisa de lo que este proceso puede consumir.

En las tablas siguientes se muestras los resultados del análisis. Algunas mediciones de Paralelo2 están sin completar porque no merece la pena perder el tiempo midiendo sabiendo que se va a obtener el mismo resultado que con Paralelo1.

Secuencial 1024x768

1Proc; Sec 1024x768	
N_col=100	0.27
N_col=1024	2.05
N_col=10k	19.1
2Proc; Sec 1024x768	
N_col=100	0.269
N_col=1024	2.03
N_col=10k	18.9

3Proc; Sec 1024x768	
N_col=100	0.262
N_col=1024	2.02
N_col=10k	18.7
4Proc; Sec 1024x768	
N_col=100	0.263
N_col=1024	1.99
N_col=10k	18.78

Paralelo1 1024x768

1Proc;P1:1024x768	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100	0.265	0.271	0.274	0.282	0.292	0.311	0.358	0.441	0.605
N_col=1024	2.005	2.012	2.028	2.02	2.036	2.064	2.11	2.17	2.39
N_col=10k	18.72	18.71	18.75	18.81	18.91	18.95	19	19.1	19.23
2Proc;P1:1024x768	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100	0.259	0.223	0.168	0.16	0.165	0.172	0.209	0.29	0.4
N_col=1024	2.005	1.8	1.366	1.14	1.11	1.13	1.14	1.2	1.342
N_col=10k	18.79	16.99	12.96	10.24	10.1	10.09	10.13	10.16	10.26
3Proc;P1:1024x768	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100	0.256	0.206	0.121	0.121	0.121	0.1227	0.153	0.201	0.285
N_col=1024	1.98	1.8	1.23	0.879	0.77	0.778	0.792	0.844	1.006
N_col=10k	18.7	16.95	12.05	7.96	7.09	7.08	7.08	7.123	7.195
4Proc;P1:1024x768	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100	0.249	0.21	0.14	0.096	0.101	0.104	0.132	0.182	0.256
N_col=1024	1.97	1.78	1.24	0.765	0.6	0.61	0.62	0.65	0.744
N_col=10k	18.74	16.89	11.85	6.85	5.57	5.55	5.50	5.54	5.62

Paralelo2 1024x768

1Proc;P2:1024x768	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100	0.27	0.271	0.276	0.280	0.293	0.316	0.357	0.441	0.61
N_col=1024	1.98	1.98	1.99	2.02	2.03	2.04	2.1	2.2	2.38
N_col=10k	18.62	18.65	18.7	18.75	18.86	18.88	18.9	19.1	19.22
2Proc;P2:1024x768	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100	0.254	0.219	0.16	0.151	0.158	0.170	0.197	0.26	0.384
N_col=1024	1.99	1.79	1.352	1.11	1.10	1.116	1.14	1.19	1.322
N_col=10k	18.67	16.96	12.71	10.17	10.13	10.08	10.12	10.15	10.25
3Proc;P2:1024x768	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100									
N_col=1024									
N_col=10k									
4Proc;P2:1024x768	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100	0.261								
N_col=1024									
N_col=10k	18.7			6.8					5.61

Secuencial 4096x4096

1Proc; Sec 4096x4096	
N_col=100	5.96
N_col=1024	45.45
N_col=10k	415.3
2Proc; Sec 4096x4096	
N_col=100	5.68
N_col=1024	43.08
N_col=10k	401.7

3Proc; Sec 4096x4096	
N_col=100	5.8
N_col=1024	43.04
N_col=10k	401.5
4Proc; Sec 4096x4096	
N_col=100	5.8
N_col=1024	44.0
N_col=10k	402.03

Paralelo1 4096x4096

1Proc;P1:4096x4096	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100	5.1	5.15	5.22	5.24	5.28	5.3	5.33	5.4	5.57
N_col=1024	42.38	42.3	42.38	42.5	42.7	42.8	43	43.2	43.2
N_col=10k	400	400	400	400	400	400	400	400	401
2Proc;P1:4096x4096	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100	5.25	4.34	3.3	2.9	2.94	2.91	3	3.13	3.16
N_col=1024	42.22	37.81	28.4	23.16	22.8	22.82	22.84	22.94	23.05
N_col=10k	398.7	360.6	272.7	217.6	216.4	214.5	215.6	216.5	217.6
3Proc;P1:4096x4096	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100	5.08	4.20	2.77	2.15	2.11	2.048	2.17	2.21	2.29
N_col=1024	42.09	37.69	26.1	17.53	16.09	16.06	16.07	16.08	16.11
N_col=10k	398	360.9	255.1	169	150.5	149.3	149.6	150.1	151
4Proc;P1:4096x4096	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100	5.20	4.24	2.71	1.85	1.66	1.63	1.85	1.91	1.95
N_col=1024	42.46	37.79	26.18	15.18	12.79	12.47	12.48	12.49	12.57
N_col=10k	398.8	361.6	253.2	150.9	118.9	116.5	116.4	116.8	116.8

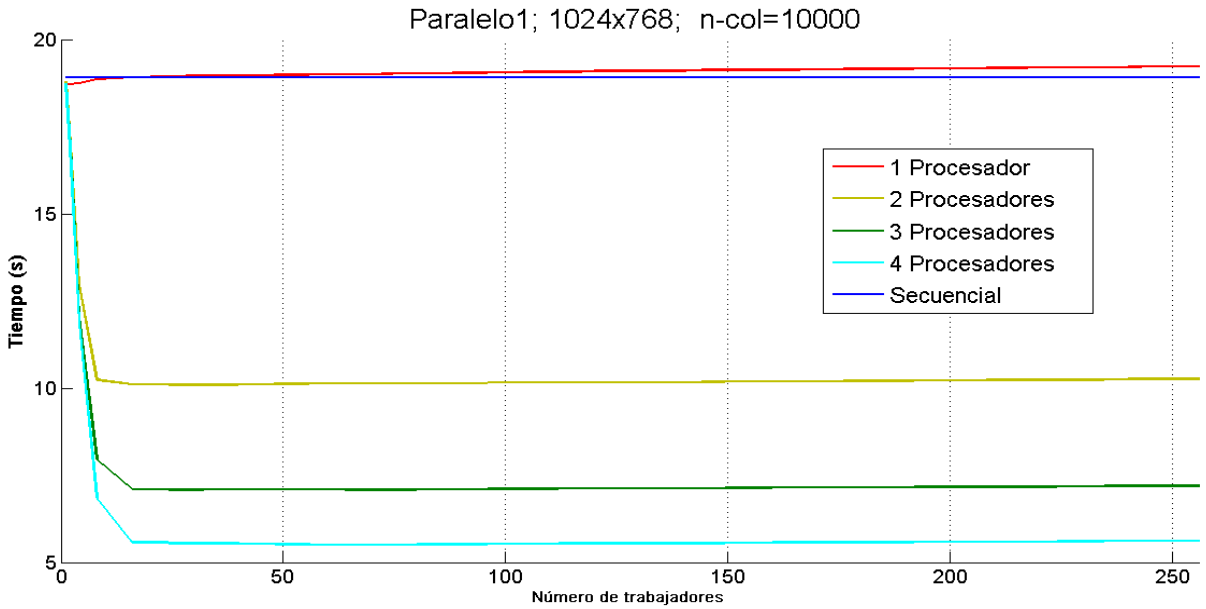
Paralelo2 4096x4096

1Proc;P2 4096x4096	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100	5.19	5.14	3.2	5.26	5.27	5.28	5.31	5.41	5.6
N_col=1024	42.35	4.32	3.16	42.37	42.49	42.68	42.8	43.11	43.2
N_col=10k	399	399	400	400	400	400	400	400	401
2Proc;P2 4096x4096	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100	5.15	4.2	3.2	2.89	2.87	2.88	2.91	3.12	3.14
N_col=1024	42.3	37.8	28.3	23.1	22.75	22.75	22.82	23	23.1
N_col=10k	398	360	272.3	217.1	216.1	213.2	215.1	216.1	217.4
3Proc;P2 4096x4096	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100									
N_col=1024									
N_col=10k									
4Proc;P2:4096x4096	1T	2T	4T	8T	16T	32T	64T	128T	256T
N_col=100									
N_col=1024		16							
N_col=10k	398.9	361.3	252.8		118.9	116.4	116.5	116.6	116.7

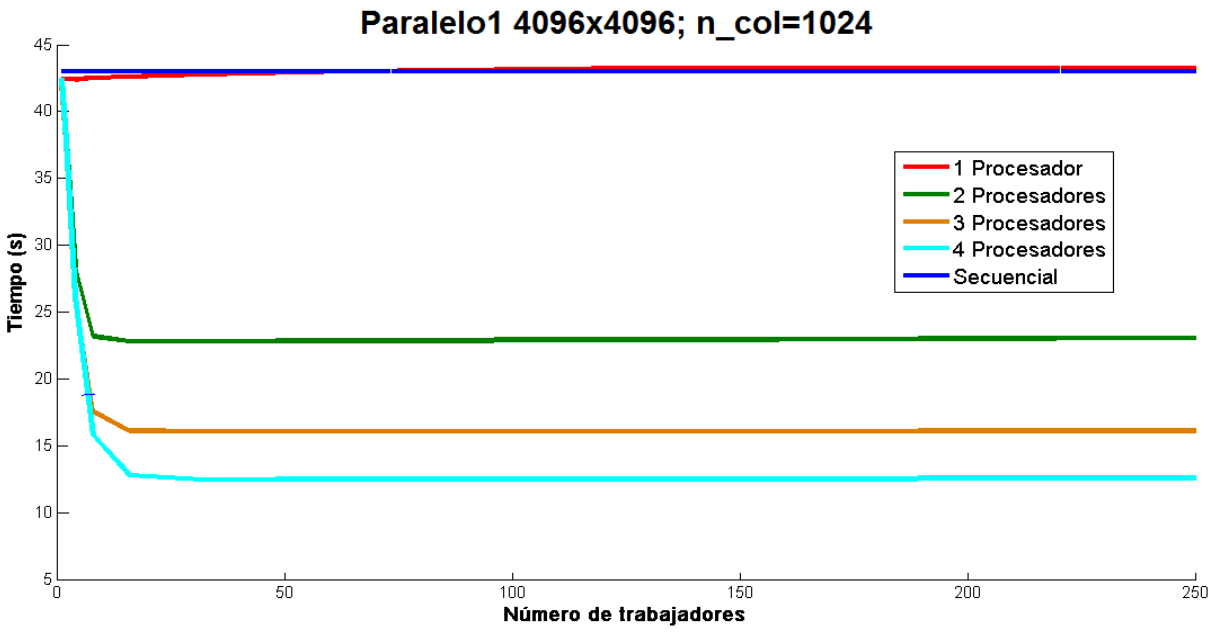
Comparación de resultados

En este apartado vamos a comparar los resultados obtenidos en función del número de trabajadores y del número de procesadores entre la versión secuencial y Paralelo1 (no incluimos Paralelo2 porque se obtienen los mismo resultados que con Paralelo1).

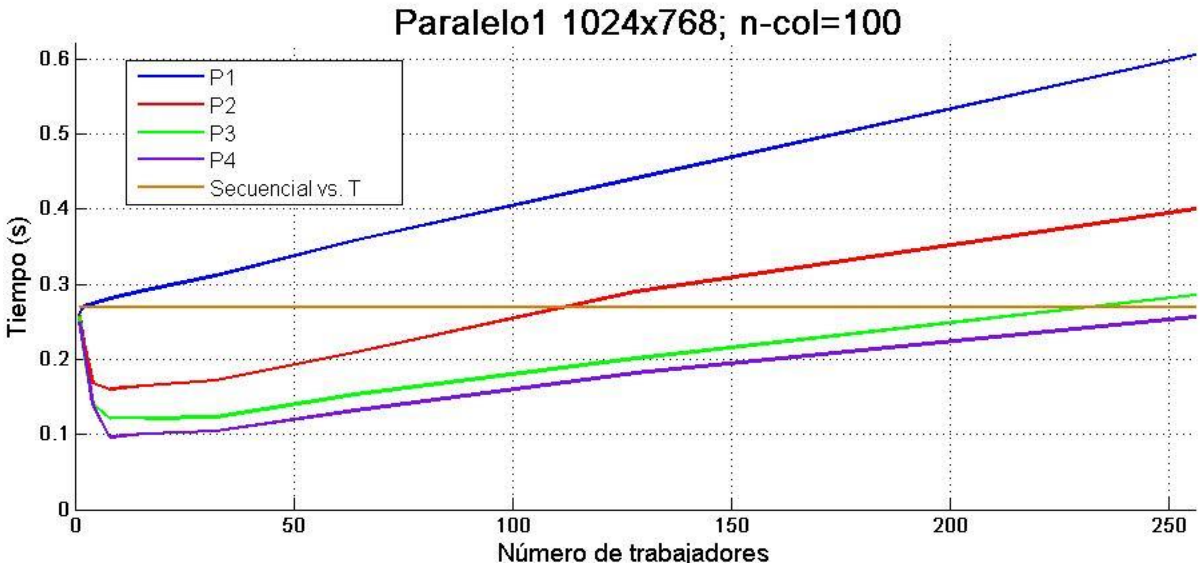
En la primera gráfica utilizaremos una resolución media-baja (1024x768) y un número alto de iteraciones (n_col=10000):



En la segunda gráfica utilizaremos la resolución máxima (4096x4096) y un número moderado de iteraciones (n_col=10000):



Podemos observar cómo llega un punto en el que al aumentar el número de trabajadores los tiempos dejan de mejorar o incluso empeora. Esto es debido a la comunicación entre los trabajadores y el maestro, llega un momento en el que la carga por procesado de cálculos comienza a perder fuerza frente a la carga de la gestión de las comunicaciones. Esto lo podemos ver más claramente en la siguiente gráfica en la que el número de iteraciones es muy bajo.



La carga de trabajo es bastante baja (nunca hay que hacer demasiadas iteraciones) y sin embargo hay que gestionar mucha comunicación entre los distintos trabajadores invocados, luego llega un momento en el que la carga por procesado de cálculos comienza a perder fuerza frente a la carga de la gestión de las comunicaciones. A partir de ese momento empieza a ser inútil o incluso contraproducente aumentar el número de trabajadores invocados.