ALEJANDRO GÜEREÑA MORÁN

DISEÑO E IMPLEMENTACIÓN DE SISTEMAS OPERATIVOS TAREA 1

I OBJETIVOS

Medir y comparar la ejecución de procesos que requieren uso intensivo del CPU y procesos que requieren E/S.

II BIBLIOGRAFÍA

Silberschatz & Galving, "SISTEMAS OPERATIVOS", McGraw Hill.

III RECURSOS

Una PC con Linux

- Las herramientas de desarrollo bajo Linux

Se recomienda no usar máquinas virtuales, para este caso sería preferible usar Cygwin si no se dispone de una computadora con Linux

IV ACTIVIDADES

1 Ejecución de procesos concurrentes que requieren uso intensivo del CPU y E/S

Compilación y ejecución del programa de prueba

Copia el programa que se muestra en la Figura 1 a un archivo de texto llamado `cpuyes.c`. Este programa de prueba realiza un ciclo que ejecuta instrucciones del CPU y después permanece bloqueado durante un tiempo tal que la duración final del proceso es de 10 segundos. Este proceso recibe como argumento el porcentaje de tiempo que hará uso intensivo del CPU. Por ejemplo:

```
$ ./cpuyes 100
```

100 % de su tiempo hace uso del CPU

```
$ ./cpuyes 50
```

50 % de su tiempo hace uso del CPU y el 50 % restante entrada y salida

```
$ ./cpuyes 0
```

100 % de su tiempo hace uso de E/S

Para que la duración del programa `cpuyes` sea de 10 segundos es necesario ajustar el valor de la constante `CICLOS`. Realiza la modificación y prueba ejecutando un proceso que demande 100 % de su tiempo el CPU de manera que la ejecución del programa se haga en 10 segundos.

Compila el programa utilizando el siguiente comando:

```
$ gcc –o cpuyes cpuyes.c
```

El archivo `cpuyes` es el programa ejecutable. Para ejecutarlo solo teclea:

```
$ ./cpuyes 100
```

```
#include <stdio.h>
#include <sys/time.h>
#define CICLOS 30000000 // Modifique este valor para que la duración sea 10 segundos
int main(int argc,char *argv[])
{
        long long start_ts;
        long long stop_ts;
        long long elapsed_time;
        long lElapsedTime;
        struct timeval ts;

        int i,j;
        int porc;

        if(argc<2)
                porc=100;
        else
                porc=atoi(argv[1]);

        printf("CPU=%d, E/S=%d\n",porc,100-porc);

        gettimeofday(&ts, NULL);
        start_ts = ts.tv_sec * 1000000 + ts.tv_usec; // Tiempo inicial

        for (i=0; i<porc ; i++) // Ejecutar instrucciones del CPU
                for(j=0;j<CICLOS;j++); // Uso intensivo del CPU

        usleep((100-porc)*100000);

        gettimeofday(&ts, NULL);
        stop_ts = ts.tv_sec * 1000000 + ts.tv_usec; // Tiempo final

        elapsed_time = stop_ts - start_ts;
        printf("proceso %d, %d microsegundos\n",getpid(),elapsed_time);
}
```

Figura 1. Programa de prueba que ejecuta instrucciones haciendo uso del CPU y E/S

1.2 Arrancador de procesos

El programa `arranca` que se muestra en la Figura 2 permite indicar cuantos procesos `cpuyes` se ejecutarán concurrentemente y que porcentaje de su tiempo demandarán CPU. Al final de la ejecución mostrará la tasa de trabajos[1].

[1] Tasa de trabajos son los trabajos terminados en una unidad de tiempo que es un segundo.

Para compilar solo teclee:

```
$ gcc –o arranca arranca.c
```

```
#include <stdio.h>
#include <sys/time.h>
int main()
{
        long long start_ts;
```

```
        long long stop_ts;
        long long elapsed_time;
        long lElapsedTime;
        struct timeval ts;
        int pid;
        int i,procs,pcpu;
        char spcpu[5];
        int status;
        float tasa;

        printf("Procesos ............ :");
        scanf("%d",&procs);
        printf("Porcentaje uso de CPU :");
        scanf("%d",&pcpu);

        sprintf(spcpu,"%d",pcpu);

        gettimeofday(&ts, NULL);
        start_ts = ts.tv_sec; // Tiempo inicial

        for(i=0;i<procs;i++)
        {
                pid=fork();
                if(pid==0)
                        execlp("nice","nice","--adjustment=0","./cpuyes",spcpu,0);
        }

        for(i=0;i<procs;i++)
                wait(&status);

        gettimeofday(&ts, NULL);
        stop_ts = ts.tv_sec; // Tiempo final

        elapsed_time = stop_ts - start_ts;
        printf("------------------------------\n");
        printf("TIEMPO TOTAL, %d segundos\n",elapsed_time);
        tasa=(float) procs / (float) elapsed_time;
        printf("Tasa de trabajos = %1.4f\n",tasa);
}
```

Figura 2.- Programa `arranca.c`

2 Ejecución de procesos que demandan uso intensivo del CPU

Ejecuta el programa `arranca` solicitando 1 proceso que demande 100 % de su tiempo CPU. Toma nota de la tasa de trabajos.

Repite el experimento con 2,3,4, hasta llegar a 10 procesos que demanden 100% CPU

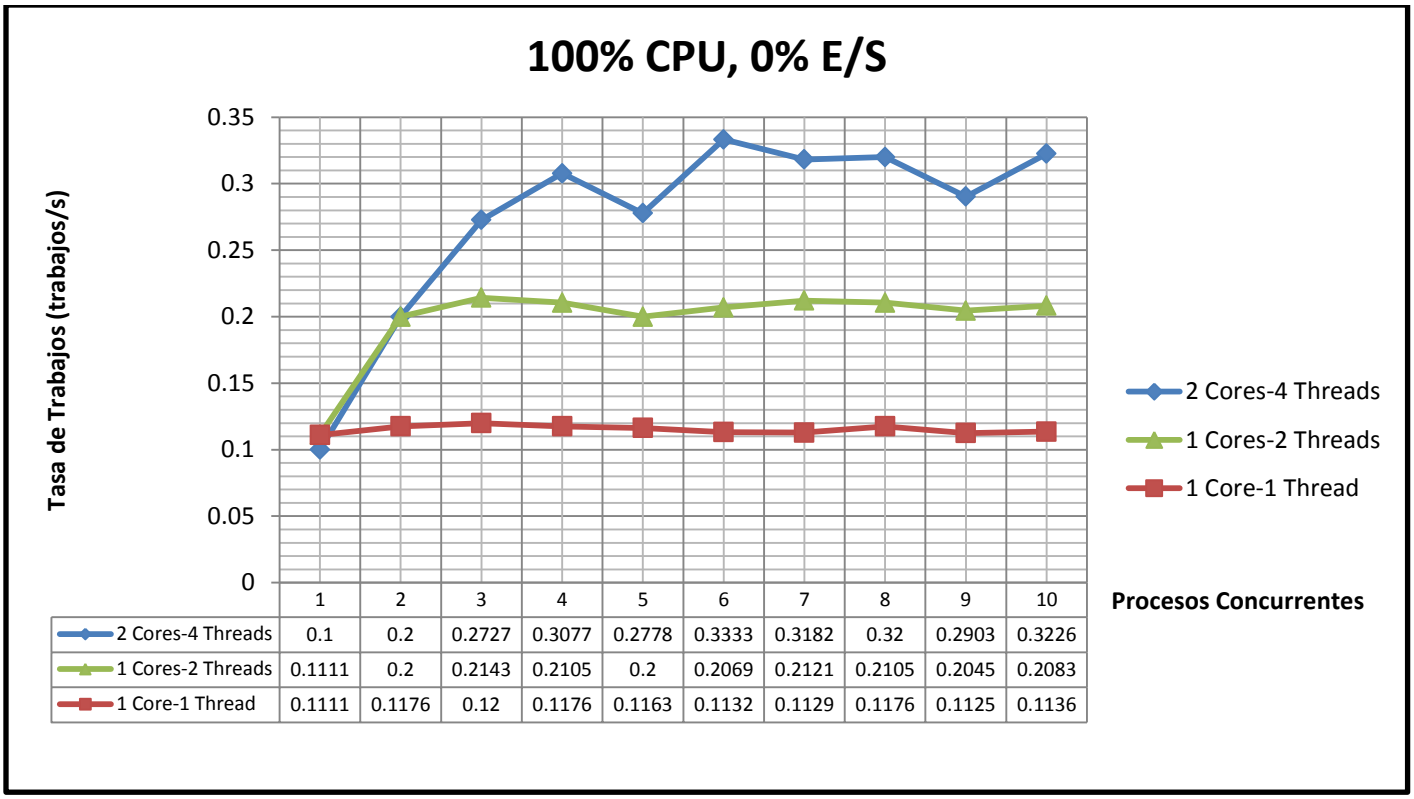
Si el procesador tiene tecnología HT o es Multicore, realiza los experimentos deshabilitando procesadores de manera que solo uno lo ejecute. Esto se puede hacer utilizando el programa `taskset`, ejemplo:

```
$ taskset 01 ./arranca
```

2.1 Resultados

Como se muestra en la Figura 3, realice una gráfica. Indique cuantos procesadores ejecutaron el proceso.

La computadora usada tiene un procesador i5-2540M, de # 2 Cores y # 4 Threads.



	1	2	3	4	5	6	7	8	9	10
2 Cores-4 Threads	0.1	0.2	0.2727	0.3077	0.2778	0.3333	0.3182	0.32	0.2903	0.3226
1 Cores-2 Threads	0.1111	0.2	0.2143	0.2105	0.2	0.2069	0.2121	0.2105	0.2045	0.2083
1 Core-1 Thread	0.1111	0.1176	0.12	0.1176	0.1163	0.1132	0.1129	0.1176	0.1125	0.1136

Figura 3. Gráfica de tiempo con respecto a los procesos en ejecución (100% CPU, 0% E/S).

2.2 Preguntas

1.- Según la gráfica desarrollada, ¿mejora la tasa de trabajos agregando procesos?

Para la gráfica de 1 Core – 1 Thread no mejora.

Para la gráfica de 1 Core – 2 Threads mejora entre 1 proceso a 2 procesos. Luego, al agregar más procesos, el incremento o decremento de la tasa de trabajos es mínimo.

Para la gráfica de 2 Core – 4 Threads mejora entre 1 proceso a 5 procesos. Luego, al agregar más procesos, el incremento o decremento de la tasa de trabajos varía, pero en menor magnitud que al aumentar de 1 a 5 procesos.

2.- Explique por qué

Primero, de la gráfica se observa que la tasa de trabajos es mayor sí existe un mayor número de cores y core-threads que puedan ejecutar concurrentemente mayor número de procesos, ya que el sistema operativo tiene más unidades funcionales en donde enviar los procesos para su ejecución.

También, al agregar más procesos, la variación de la tasa de trabajos tiene a hacerse mínima. Esto se debe a que el sistema operativo reparte la mayor cantidad de procesos a los distintos Cores disponibles, pero al ocuparse todos los Cores disponibles, los demás procesos tiene que esperar su turno hasta que el sistema operativo pueda asignarlos a un Core que se desocupe. Por lo tanto, al incrementar los procesos, estos

tienen que esperar hasta que los demás vayan terminando, incrementando el tiempo en que se ejecutan todos los procesos.

3.- En la grafica desarrollada ¿Dónde se da una asíntota?

Se da una asíntota cuando los procesos llegan a 3 o 5 procesos para las gráficas con 1 Core físico con 1 y 2 Threads respectivamente. La gráfica de 2 Cores- 4 Threads no muestra una asíntota significativa.

4.- Explique el por qué se da una asíntota en ese punto.

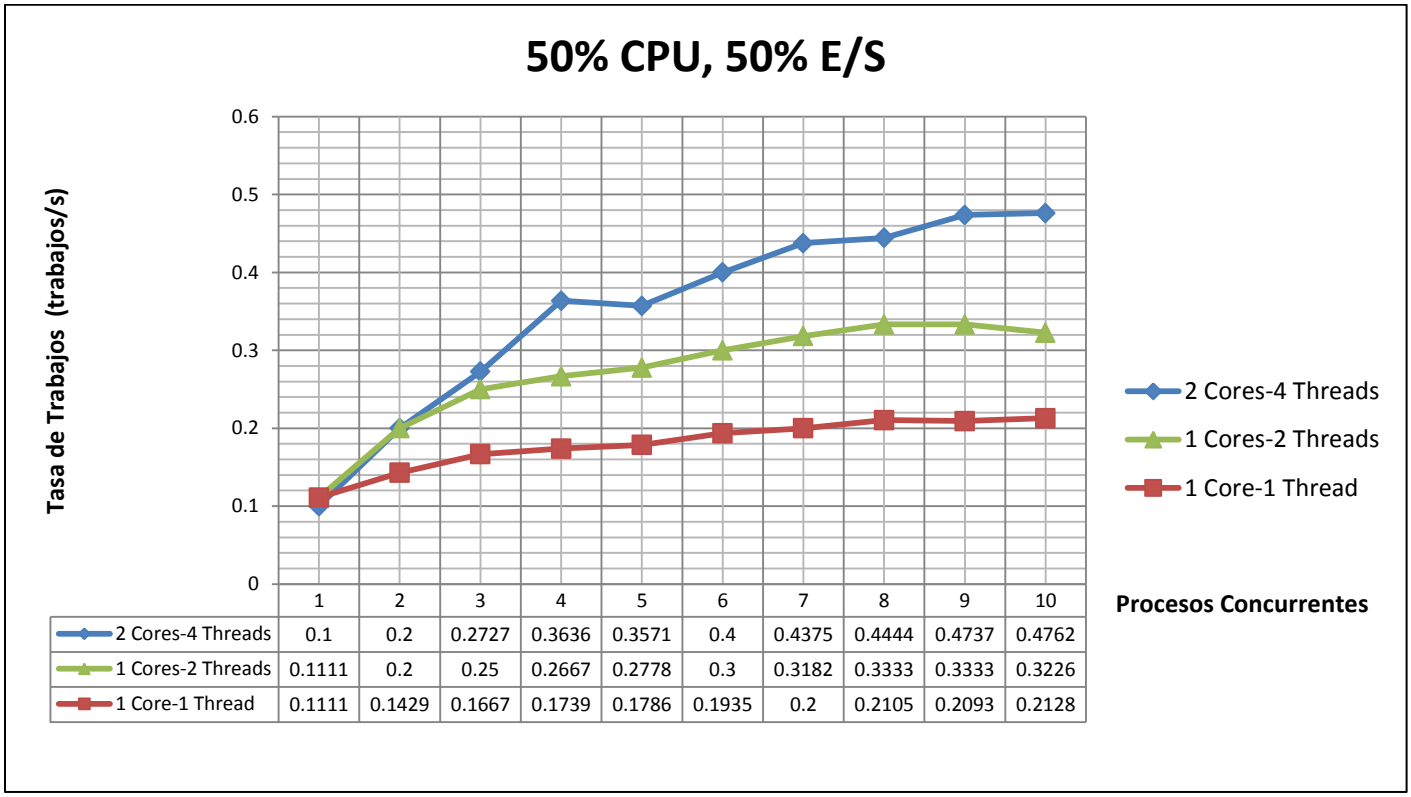
Se dan estas asíntotas en estos puntos porque es el número de procesos que puede asignar el sistema operativo para su ejecución concurrente, que es proporcional a las unidades funcionales disponibles. A partir de esos puntos, los procesos tienen que esperar por una unidad funcional disponible.

3 Ejecución de procesos concurrentes que permanecen bloqueados (puede ser en espera de E/S)

Ejecuta el programa `arranca` solicitando 1 proceso que demande 0 % de su tiempo CPU, es decir 100% de su tiempo demanda entrada y salida. Toma nota de la tasa de trabajos.

Repite el experimento con 2,3,4, hasta llegar a 10 procesos.

La computadora usada tiene un procesador i5-2540M, de # 2 Cores y # 4 Threads.



	1	2	3	4	5	6	7	8	9	10
2 Cores-4 Threads	0.1	0.2	0.2727	0.3636	0.3571	0.4	0.4375	0.4444	0.4737	0.4762
1 Cores-2 Threads	0.1111	0.2	0.25	0.2667	0.2778	0.3	0.3182	0.3333	0.3333	0.3226
1 Core-1 Thread	0.1111	0.1429	0.1667	0.1739	0.1786	0.1935	0.2	0.2105	0.2093	0.2128

Figura 4. Gráfica de tiempo con respecto a los procesos en ejecución (50% CPU, 50% E/S).

3.1 Preguntas

1.- Según la gráfica desarrollada, ¿mejora la tasa de trabajos agregando procesos?

Si mejora las tasas de trabajos, aunque se nota que las gráficas con mayor número de unidades funcionales, tienen un incremento mayor que las que tienen menos. También la variación de la tasa de trabajos es mayor respecto a cuando se tiene procesos de 100% CPU.

2.- Explique por qué

Como se tiene un menor uso de CPU, de 50%, respecto al ejercicio anterior, que es de 100%, esto provoca que los procesos se ejecuten con mayor rapidez y los Cores estén ocupados menos tiempo, provocando que los procesos que estén esperando, tengan menos tiempo de espera.

3.- En la grafica desarrollada ¿Dónde se da una asíntota?

Se da una asíntota cuando los procesos llegan a 9 o 10 procesos para las gráficas con 1 Core físico con 1 y 2 Threads respectivamente. La gráfica de 2 Cores- 4 Threads no muestra una asíntota significativa.

4.- Explique el por qué se da una asíntota en ese punto.

La asíntota se da con un mayor de procesos respecto al ejercicio anterior, indicando que se pueden ejecutar más trabajos por unidad de tiempo, pero aun así, existe el límite de las unidades funcionales de procesamiento, que determinará una tasa de trabajos fija al tener más procesos.

4 Ejecución de procesos concurrentes que el 50 % de su tiempo utilizan CPU y el 50 % de su tiempo permanecen bloqueados (puede ser en espera de E/S)

Ejecuta el programa `arranca` solicitando 1 proceso que demande 50 % de su tiempo CPU, es decir el 50% de su tiempo restante demanda entrada y salida. Toma nota de la tasa de trabajos.

Repite el experimento con 2,3,4, hasta llegar a 10 procesos.

Como se muestra en la Figura 3, realice una gráfica. Indique las características del CPU (si es Core Solo, HT, Core Duo, Quad Core).

La computadora usada tiene un procesador i5-2540M, de # 2 Cores y # 4 Threads.

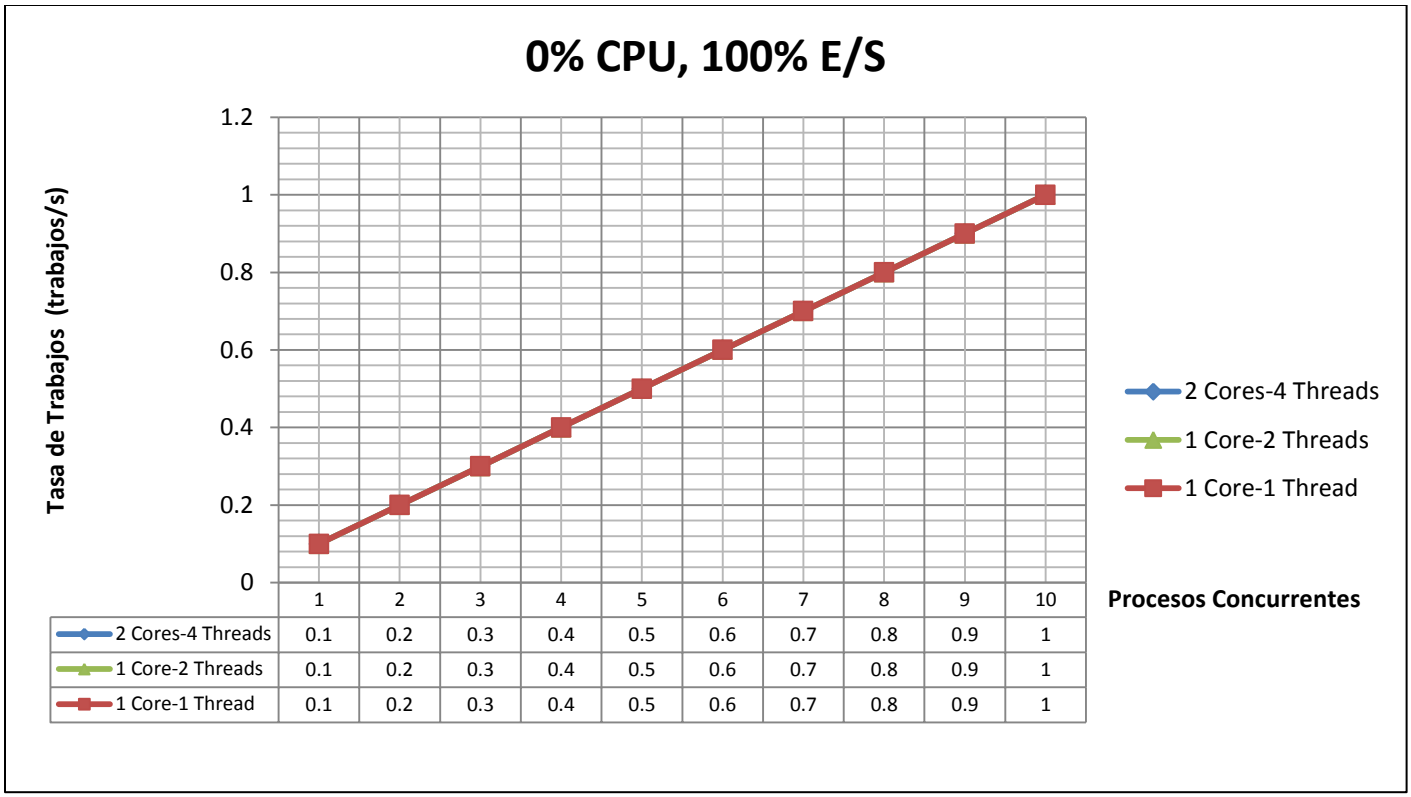


Figura 5. Gráfica de tiempo con respecto a los procesos en ejecución (0% CPU, 100% E/S).

4.1 Preguntas

1.- Según la gráfica desarrollada, ¿mejora la tasa de trabajos agregando procesos?

Sí, todas tienen el mismo incremento en la tasa de trabajos.

2.- Explique por qué

Como estos procesos no dependen de los CPUs, los procesos se quedan esperando por dispositivos E/S, por lo tanto, todos se ejecutan concurrentemente y el sistema operativo no tiene que asignarlos a los Cores disponibles.

3.- En la grafica desarrollada ¿Dónde se da una asíntota?

No existe asíntota para estas gráficas.

4.- Explique el por qué se da una asíntota en ese punto.

No existe asíntota porque no dependen de la cantidad de unidades de procesamiento, sino de las unidades de E/S. Si tuviéramos un límite de unidades de E/S para esta práctica, las gráficas de tasa de trabajo tendrían una asíntota dependiente a ese número de unidades.

5.- Considerando que se tiene el dato del porcentaje de tiempo que los trabajos usan CPU, y este porcentaje puede ser cualquier valor. Determine una fórmula en función del porcentaje de tiempo de CPU donde pueda calcularse el número máximo del total de trabajos por segundo que puede obtenerse si agregamos un número infinito de procesos.

$$\frac{limf(x)}{x \to \infty} = \frac{x * \%CPU}{x * (\%CPU + 1)} + (x * \%ES)$$