Trabajo práctico concurrentes (TP3)

Lucas Salse, Cristian Velazquez

3 de marzo de 2020

1. Introduccion

En el siguiente trabajo práctico se buscará implementar un simulador de un procesador con dos núcleos. A partir de la red de Petri dada por el enunciado, la cual representa a un procesador mono núcleo, se deberá extender la misma a una red que modele un procesador con dos núcleos. Ademas implementaremos una política que resuelva los conflictos que se generan con las transiciones que alimentan los buffers de los núcleos (CPU_Buffer).

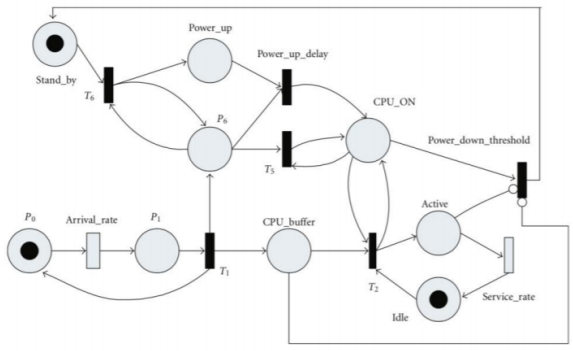


Figura 1: modelo a emplear

2. Consideraciones

- Se implementaron dos CPU, cada una con su propio buffer de entrada, como así también con sus transiciones de encendido y apagado.
- Se tuvo en cuenta una sola entrada (una entrada común) para cada CPU
- La transición de entrada posee un tiempo de arribo, el cual simula el tiempo que demora cada proceso en ser ejecutado por cada CPU.
- Los CPU también tienen diferentes tiempos los cuales representan el periodo en el que se ejecuta una actividad.
- Un CPU solo se podrá apagar en el caso que no esté realizando ninguna actividad y que no exista un proceso en espera de ser ejecutado, esto se modela mediante los inhibidores correspondientes
- Cuando un proceso ingresa y no está el CPU activo, se deberá encenderlo solo en ese momento.

3. RDP utilizada y su correspondiente análisis:

La red de petri que se utilizo fue la que se muestra en la figura 2.

Secuencia de disparos: La secuencia es infinita porque el token de la transición de entrada del sistema vuelve con el disparo de las transiciones T1 y T8, entonces siempre ocurren disparos cada un cierto tiempo. Además,

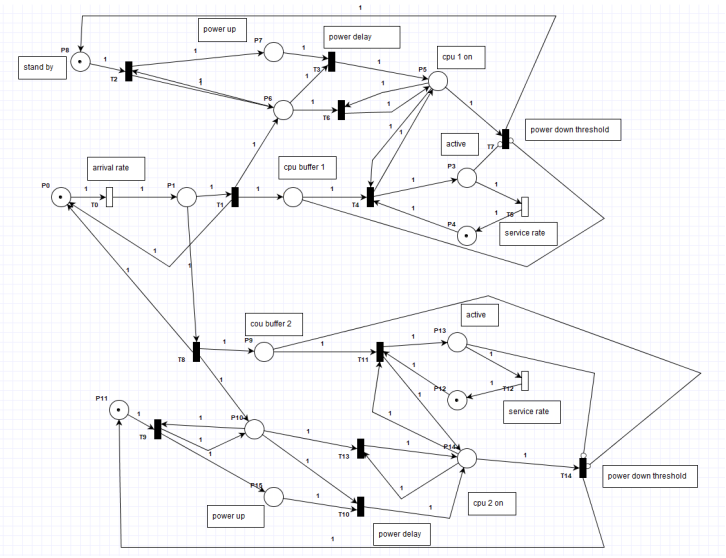


Figura 2: RdP usada

Petri net state space analysis results

Bounded	false
Safe	false
Deadlock	false

Figura 3: RDP analisis

esto se puede ver en la figura 3, donde se observa que no hay puntos muertos por lo que demuestra que siempre habrá disparos infinitos.

Rdp limitada: El número de marcas alcanzables no es conocido, ósea no está limitado. Esto se puede ver en el resultado del análisis de la figura 3.

Rdp segura: No es segura porque hay plazas que contiene más de un token cuando el sistema evoluciona. Si observamos la figura 3, se comprueba

Forwards incidence matrix I^+

	T0	T1	T10	T11	T12	T13	T14	T2	T3	T4	T5	T6	T7	T8	T9
P0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
P1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
P11	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
P12	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
P13	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
P14	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0
P15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
P2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
P5	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0
P6	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
P7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
P8	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
P9	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figura 4: Matriz forwards i+

que no es segura.

Rdp seudo viva: No es seudo viva porque todas las transiciones pueden ser disparadas a medida que el marcado cambia de estado.

Rdp quasi viva: No es quasi viva porque todas las transiciones pueden ser disparadas, no hay un estado determinado que provoque que una transición quede deshabilitada para siempre.

Rdp liveness: Como no existe una marca donde haya un deadlock, está libre de interbloqueo por lo que ningún marcado es un punto muerto. Figura 3.

Conflictos: Tiene conflicto estructural, esto se puede observar fácilmente en las figuras: figura 5, figura 4 y figura 6 donde por ejemplo la plaza p1 tiene

Backwards incidence matrix I^-

	T0	T1	T10	T11	T12	T13	T14	T2	T3	T4	T5	T6	T7	T8	T9
P0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
P10	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1
P11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
P12	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
P13	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
P14	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0
P15	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P5	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
P6	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
P7	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
P8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
P9	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Figura 5: Matriz forwards i-

un conflicto con la transición t1 y t8.

Múltiples disparos: Hay múltiples disparos porque existe una concurrencia, esto se puede ver en la figura 7, en este marcado hay 3 transiciones que pueden ser disparadas.

Invariantes: En la figura 8 se observa en la T-invariantes que existen 4 caminos posibles y 5 loops, osea donde se devulven los tokens.

Además, se puede observar en la ecuación de las P-invariantes siempre habrá una cantidad de tokens en las ecuaciones de la figura 8, independientemente del marcado.

Combined incidence matrix I

	T0	T1	T10	T11	T12	T13	T14	T2	T3	T4	T5	T6	T7	T8	T9
P0	-1	1	0	0	0	0	0	0	0	0	0	0	0	1	0
P1	1	-1	0	0	0	0	0	0	0	0	0	0	0	-1	0
P10	0	0	-1	0	0	-1	0	0	0	0	0	0	0	1	0
P11	0	0	0	0	0	0	1	0	0	0	0	0	0	0	-1
P12	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0
P13	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0
P14	0	0	1	0	0	0	-1	0	0	0	0	0	0	0	0
P15	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	1
P2	0	1	0	0	0	0	0	0	0	-1	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0
P5	0	0	0	0	0	0	0	0	1	0	0	0	-1	0	0
P6	0	1	0	0	0	0	0	0	-1	0	0	-1	0	0	0
P7	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0
P8	0	0	0	0	0	0	0	-1	0	0	0	0	1	0	0
P9	0	0	0	-1	0	0	0	0	0	0	0	0	0	1	0

Figura 6: Matriz incidencia I

Marking

	P0	P1	P10	P11	P12	P13	P14	P15	P2	P3	P4	P5	P6	P7	P8	P9
Initial	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	0
Current	1	0	0	0	0	1	1	0	0	1	0	1	0	0	0	0

Enabled transitions

T0	T1	T10	T11	T12	T13	T14	T2	T3	T4	T5	T6	T7	T8	T9
yes	no	no	no	yes	no	no	no	no	no	yes	no	no	no	no

Figura 7: Matriz marcado

4. Implementación de la política:

Para solucionar el conflicto de la plaza entrada, en donde se debe decidir a qué CPU debe llegar el proceso (actividad), lo resolvimos preguntando por

T-Invariants

T0	T1	T10	T11	T12	T13	T14	T2	T3	T4	T5	T6	T7	T8	T9
1	0	0	1	1	1	0	0	0	0	0	0	0	1	0
1	1	0	0	0	0	0	0	0	1	1	1	0	0	0
1	1	0	0	0	0	0	1	1	1	1	0	1	0	0
1	0	1	1	1	0	1	0	0	0	0	0	0	1	1

The net is covered by positive T-Invariants, therefore it might be bounded and live.

P-Invariants

P0	P1	P10	P11	P12	P13	P14	P15	P2	P3	P4	P5	P6	P7	P8	P9
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0

The net is not covered by positive P-Invariants, therefore we do not know if it is bounded.

P-Invariant equations

M(P0) + M(P1) = 1
M(P12) + M(P13) = 1
M(P11) + M(P14) + M(P15) = 1
M(P3) + M(P4) = 1
M(P5) + M(P7) + M(P8) = 1

Figura 8: Analisis de invariantes

la posición del marcado el cual representa la plaza buffer1 y buffer2, por lo tanto el proceso de entrada irá al CPU que tiene acumulado la menor cantidad de tokens en dichas plazas, y de ese modo no se sobrecargará un CPU. En el caso de que ambos CPU posean la misma cantidad de procesos en su buffer, se lo eligirá mediante un “random”.

En el momento que un hilo no pueda ejecutar su correspondiente transición, se encolará. Cuando otro hilo pueda producir un dispara con su transición el marcado cambiará y se deberá corroborar mediante el uso de una función lógica (and) que cantidad de hilos tenemos encolados en las transiciones y si dichas transiciones se encuentran ahora sensibilizadas. Por lo tanto se deberá desencolar el hilo que pueda ahora si disparar su transición.

4.1. Justificación de la cantidad de hilos usados:

Para determinar la cantidad de hilos que se deberán emplear para dicho trabajo se hizo uso del análisis de los invariantes con la herramienta PIPE. Mediante el cual se pudo observar en las P-invariantes (plazas invariantes figura7), que existen 5 ecuaciones que nos garantizan que siempre habrá un token, entonces mediante eso pudimos determinar la secuencia de transiciones que debe implementar cada hilo, y finalmente con la secuencia obtenemos la cantidad total de hilos que son 7 hilos.

5. Análisis de tiempo:

5.1. Tiempo de las dos CPU iguales

El tiempo está en milisegundos. Tiempos implementados (mismo tiempo):

Transicion T0: = 10

Transición T5: alfa = 20; beta = 2000

TransiciónT12: alfa = 20; beta = 2000

Procesos realizados por el CPU1 = 497

Procesos realizados por el CPU2 = 503

Tiempo empelado = 11647 [ms]

Esto se ve en la figura 9.

5.2. Un CPU con el doble de tiempo

Transición T0: 10

Transición T5: alfa = 40; beta = 2000

```
Soy el hilo Thread-6 que DISPARO la transicion: 4
Ejecuci□n n□: 5511. Test ran: 1, Failed: 0.
Ejecuci□n n□: 5511. Test ran: 1, Failed: 0.

Soy el hilo: Thread-6  transicion disparada T12:  proceso realizado CPU2
Procesos realizados por el CPU2 : 503
************************************************************
Soy el hilo Thread-4 que DISPARO la transicion: 6
Ejecuci□n n□: 5512. Test ran: 1, Failed: 0.
Ejecuci□n n□: 5512. Test ran: 1, Failed: 0.

Soy el hilo Thread-4transicion disparada T14: se apago la CPU2
************************************************************
Soy el hilo Thread-2 que DISPARO la transicion: 10
Ejecuci□n n□: 5513. Test ran: 1, Failed: 0.
Ejecuci□n n□: 5513. Test ran: 1, Failed: 0.

Soy el hilo: Thread-2  transicion disparada T5:  proceso realizado CPU1
Procesos realizados por el CPU1 : 496
************************************************************
Soy el hilo Thread-2 que DISPARO la transicion: 9
Ejecuci□n n□: 5514. Test ran: 1, Failed: 0.
Ejecuci□n n□: 5514. Test ran: 1, Failed: 0.

************************************************************
Soy el hilo Thread-2 que DISPARO la transicion: 10
Ejecuci□n n□: 5515. Test ran: 1, Failed: 0.
Ejecuci□n n□: 5515. Test ran: 1, Failed: 0.

Soy el hilo: Thread-2  transicion disparada T5:  proceso realizado CPU1
Procesos realizados por el CPU1 : 497
Soy el hilo Thread-1 que DISPARO la transicion: 12
el tiempo final es 11647
************************************************************
Ejecuci□n n□: 5516. Test ran: 1, Failed: 0.
Ejecuci□n n□: 5516. Test ran: 1, Failed: 0.

Soy el hilo Thread-1transicion disparada T7: se apago la CPU1
************************************************************
```

Figura 9: Impresion por pantalla (tiempos iguales)

TransiciónT12: alfa = 20; beta = 2000

```
Soy el hilo Thread-2 que DISPARO la transicion: 10
Ejecuci□n n□: 4931. Test ran: 1, Failed: 0.
Ejecuci□n n□: 4931. Test ran: 1, Failed: 0.

Soy el hilo: Thread-2  transicion disparada T5:  proceso realizado CPU1
Procesos realizados por el CPU1 : 630
************************************************************

Soy el hilo Thread-6 que DISPARO la transicion: 4
Ejecuci□n n□: 5025. Test ran: 1, Failed: 0.
Ejecuci□n n□: 5025. Test ran: 1, Failed: 0.

Soy el hilo: Thread-6  transicion disparada T12:  proceso realizado CPU2
Procesos realizados por el CPU2 : 370
Soy el hilo Thread-4 que DISPARO la transicion: 6
el tiempo final es 15325
************************************************************
```

Figura 10: Impresion por consola (CPU1 dolbe de tiempo)

Procesos realizados por el CPU1 = 630

Procesos realizados por el CPU2 = 370

Tiempo empelado = 15325 [ms]

Esto se ve en la figura 10.

5.3. Un CPU con el triple de tiempo

Transición T0: 10

Transición T5: alfa = 60; beta = 2000

TransiciónT12: alfa = 20; beta = 2000

```
Soy el hilo: Thread-2  transicion disparada T5:  proceso realizado CPU1
Procesos realizados por el CPU1 : 679
************************************************************
Soy el hilo Thread-1 que DISPARO la transicion: 12
Ejecución nº: 4832. Test ran: 1, Failed: 0.
Ejecución nº: 4832. Test ran: 1, Failed: 0.

Soy el hilo Thread-1transicion disparada T7: se apago la CPU1
************************************************************
Soy el hilo: Thread-6  transicion disparada T12:  proceso realizado CPU2
Procesos realizados por el CPU2 : 321
Soy el hilo Thread-4 que DISPARO la transicion: 6
el tiempo final es 19758
************************************************************
```

Figura 11: Impresion por consola (CPU1 triple de tiempo)

Procesos realizados por el CPU1 = 679

Procesos realizados por el CPU2 = 321

Tiempo empelado = 19758 [ms]

Esto se ve en la figura 11.

6. Análisis de P-Invariantes

Para poder efectuar el análisis de los P-invarientes se procedió a testear que se cumplan los mismos mediante una clase “Tester”, que utiliza un Junit test.

Procedimiento:

Una vez producido algún disparo de manera satisfactoria de una transición, se comprobara (se llevará a cabo el test) de los P-Invariantes, el cual consiste en lo siguiente. Debido a que cambio el marcado de la red por el disparo de una transición, tomamos la matriz P-Invariantes y realizamos una operación de producto punto con la matriz Marcado (previamente traspuesta). Esto nos debería dar como resultado un vector que contiene los valores de P-Resultados obtenidos mediante el análisis de la red. Finalmente comparamos este resultado con el vector(o matriz) P-resultados buscando que sean iguales, dando como resultado un test satisfactorio.

7. Análisis de los T-invariantes

Para llevar a cabo el análisis de los T-invariantes se tiene como objetivo poder determinar los “caminos” establecidos por el software en base a la red de petri e ir corroborando que los mismos se cumplan.

Para esto se tiene un archivo el cual contiene todas las transiciones disparadas de la red durante su ejecución, en base a esto se buscará cada uno de esos caminos los cuales están formados por ciertas transiciones que lo componen.

Para poder llevar a cabo todo este análisis se procedió a buscar patrones

lo cual tiene como objetivo ahorrar tiempo para determinar lo que buscamos.

Es importante determinar que existen dos conceptos diferentes los cuales son:

Reconocimiento de patrones (Pattern recognition): Extrae información de algo para obtener un patrón. El resultado es un patrón.

Búsqueda de patornes (pattern matching): Busca algo que coincida con un patrón. El resultado son todas las coincidencias con el patrón.

Para esto se emplearon expresiones regulares las cuales son simplemente una secuencia de caracteres, esta expresión será el patrón que servirá para buscar los caminos.

Utilizamos las expresiones regulares para determinar cada uno de los caminos y luego ir verificando que se cumplan, una vez que se reconocieron todas y en nuestro caso como es un modelo de un procesador de dos núcleos el cual trabaja en base a una entrada de procesos, al finalizar su procesamiento se apagan cada uno de los CPU; por lo tanto vemos reflejado en el análisis que todas las transiciones corresponden a un camino.

8. Análisis de tiempo detallado

8.1. Análisis del comportamiento con la Transición T0(Arribos): = 10

1: Cpu 1 y cpu 2 con tiempos iguales

CPU1: alfa = 20; beta = 2000

CPU2: alfa = 20; beta = 2000

Procesos realizados por el CPU1 =497

Procesos realizados por el CPU2 = 503

Tiempo empelado = 13 [s]

2: Un CPU con el doble de tiempo

CPU1: alfa = 20; beta = 2000

CPU2: alfa = 40; beta = 2000

Procesos realizados por el CPU1 =369

Procesos realizados por el CPU2 =631

Tiempo empelado = 16 [s]

3: Un CPU con el triple de tiempo

CPU1: alfa = 20; beta = 2000

CPU2: alfa = 60; beta = 2000

Procesos realizados por el CPU1 =326

Procesos realizados por el CPU2 =674

Tiempo empelado = 20,5 [s]

Se puede observar para el caso 1 que ambas Cpu por lo general realizan la misma cantidad de procesos, para los casos 2 y 3 el tiempo total final va aumentando a medida que una Cpu es más lenta y también aumenta la diferencia en tareas que realiza cada una. Esto se produce porque se disminuye la velocidad de una Cpu por lo que la segunda Cpu realiza más tareas, entonces como el tiempo de arribos es pequeño se produce una acumulación de arribos donde se nota que en este análisis una Cpu más lenta que la otra provoca que se tarde más tiempo la ejecución del programa y que haya diferencias en la cantidad de procesos que realiza cada una. En otras palabras, como siempre hay arribos, una Cpu que termina su tarea ya puede estar atendiendo otro

proceso entonces influye mucho que una Cpu sea más rápido que el otra.

8.2. Análisis del comportamiento con la Transición T0 (Arribos): = 30

1: Cpu 1 y cpu 2 con tiempos iguales

CP1U: alfa = 20; beta = 2000

CPU2: alfa = 20; beta = 2000

Procesos realizados por el CPU1 = 491

Procesos realizados por el CPU2 = 509

Tiempo empelado = 31,7 [s]

2: Un CPU con el doble de tiempo

CPU1: alfa = 20; beta = 2000

CPU2: alfa = 40; beta = 2000

Procesos realizados por el CPU1 = 479

Procesos realizados por el CPU2 = 521

Tiempo empelado = 32,5 [s]

3: Un CPU con el triple de tiempo

CPU1: alfa = 20; beta = 2000

CPU2: alfa = 60; beta = 2000

Procesos realizados por el CPU1 = 409

Procesos realizados por el CPU2 = 591

Tiempo empelado = 31,7 [s]

Encontramos que para el caso 1 ambas Cpus producen la misma cantidad de procesos por lo general ya que sus tiempos son iguales. Para este análisis los casos 2 y 3 tienen poca discrepancia en los tiempos empleados en la

ejecución del programa, pero si influye que una Cpu sea más lenta porque una Cpu produce menos tareas que la otra, pero estas diferencias no son tan grandes como el primer analisis, ya que el tiempo de arribo es un poco más grande por lo tanto se puede ver que existen ocasiones donde una Cpu cuando termina su proceso debe esperar que un arribo llegue. Entonces la Cpu más lenta en algunos casos cuando se cumple el tiempo para que llegue otro arribo a pesar de tardar mas en realizar su tarea, en ese transcurso de espera ya la realizo y está disponible para atender otra, por lo que para esta situación ambas Cpus pueden atender un proceso.

8.3. Análisis del comportamiento con la Transición T0 (Arribos): = 60

1: Cpu 1 y cpu 2 con tiempos iguales

CPU1: alfa = 20; beta = 2000

CPU2: alfa = 20; beta = 2000

Procesos realizados por el CPU1 = 523

Procesos realizados por el CPU2 = 477

Tiempo empelado = 61,7 [s]

2: Un CPU con el doble de tiempo CPU1: alfa = 20; beta = 2000

CPU2: alfa = 40; beta = 2000

Procesos realizados por el CPU1 =509

Procesos realizados por el CPU2 =491

Tiempo empelado = 61,3 [s]

3: Un CPU con el triple de tiempo

CPU1: alfa = 20; beta = 2000

CPU2: alfa = 60; beta = 2000

Procesos realizados por el CPU1 = 481

Procesos realizados por el CPU2 = 519

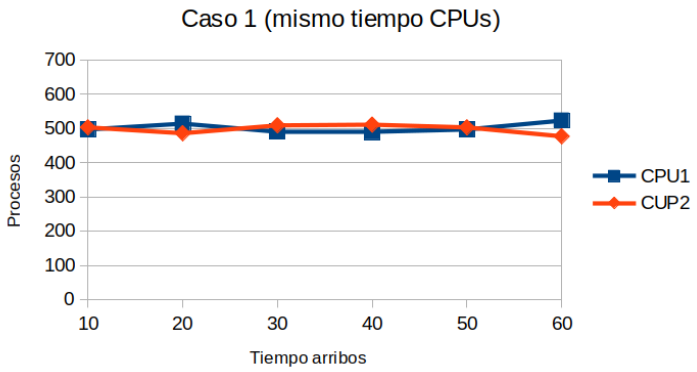
Tiempo empelado = 61,7 [s]

se nota que al haber un mayor aumento para el tiempo de los arribos provoca que tarde mucho más la ejecución, ya que las CPUS deben esperar que los procesos lleguen a ellos. O sea que los arribos se producen con un mayor retardo de tal forma que genera que no influyan las diferencias de tiempos que tardan en realizar los procesos las Cpus. Ambas Cpus tienen el tiempo suficiente para realizar su tarea por lo que un Cpu con la misma velocidad o el doble o triple que la otra, no genera diferencias en la cantidad de procesos que atiende cada una. Entonces en la mayoría de las ejecuciones del programa se puede ver que para los casos 1,2 y 3 los tiempos empleados entre estos son casi similares y la cantidad de procesos que realiza cada Cpu también.

8.4. Comparación de los procesos realizados por las cpus

Cpus iguales variando el tiempo de la transición T0 (Arribos):

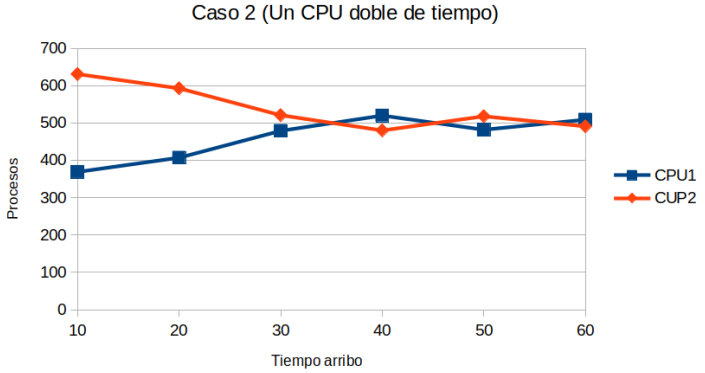
arribos	alfa	beta	CPU1	CUP2	TIEMPO
10	20	20	497	503	13
20	20	20	514	486	22,7
30	20	20	491	509	31,7
40	20	20	489	511	41,3
50	20	20	497	503	51,6
60	20	20	523	477	61,7



El aumento o disminución de los arribos no afecta en la cantidad de procesos que realizada cada Cpu, ya que estos son iguales. Existe una varianza muy aleatoria, pero es producida no es por los arribos.

Una Cpu con el doble de tiempo variando el tiempo de la transición T0 (Arribos):

arribos	alfa	beta	CPU1	CUP2	TIEMPO
10	20	40	369	631	16
20	20	40	407	593	22,7
30	20	40	479	521	32,5
40	20	40	520	480	42,1
50	20	40	482	518	52
60	20	40	509	491	61,3

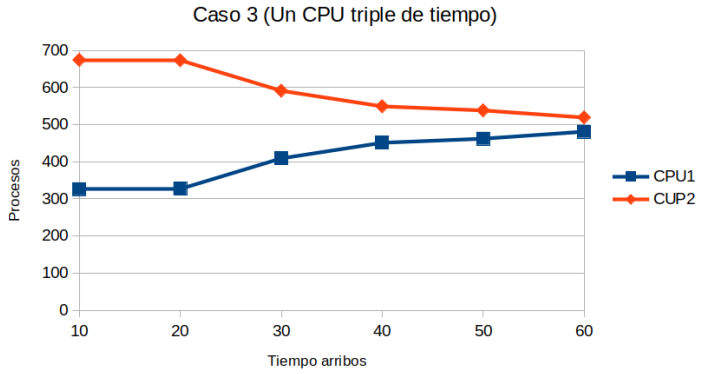


Se puede observar que con un Cpu el doble de rápido que otra, a medida que los arribos llegan con un mayor retraso las diferencias entre los procesos que realiza cada Cpu es menor. Esto sucede porque cuando un cpu termino

su tarea en algunas ocasiones debe esperar que llegue otro proceso y durante ese período no hace ninguna operación.

Una Cpu con el triple de tiempo variando el tiempo de la transición T0 (Arribos):

arribos	alfa	beta	CPU1	CUP2	TIEMPO
10	20	60	326	674	20,5
20	20	60	327	673	22,6
30	20	60	409	591	31,7
40	20	60	451	549	41,3
50	20	60	462	538	51,3
60	20	60	481	519	61,7



En este caso es como el anterior, a medida que aumenta el tiempo en T0 (Arribos) las diferencias disminuyen, pero como existe un Cpu con el triple de velocidad que la otra, esta realiza más tareas que la otra. Entonces se puede llegar a la conclusión que, si los tiempos de llegada entre procesos es muy notoria, tener un procesador muy rápido, lento o igual no hace que existan grandes diferencias entre la cantidad de procesos que realiza cada uno, incluso si el tiempo de Arribos fuera muy alto provocaría que ambos realicen una cantidad de procesos similar porque estos deben esperar que un proceso les llegue para poder atenderlo.