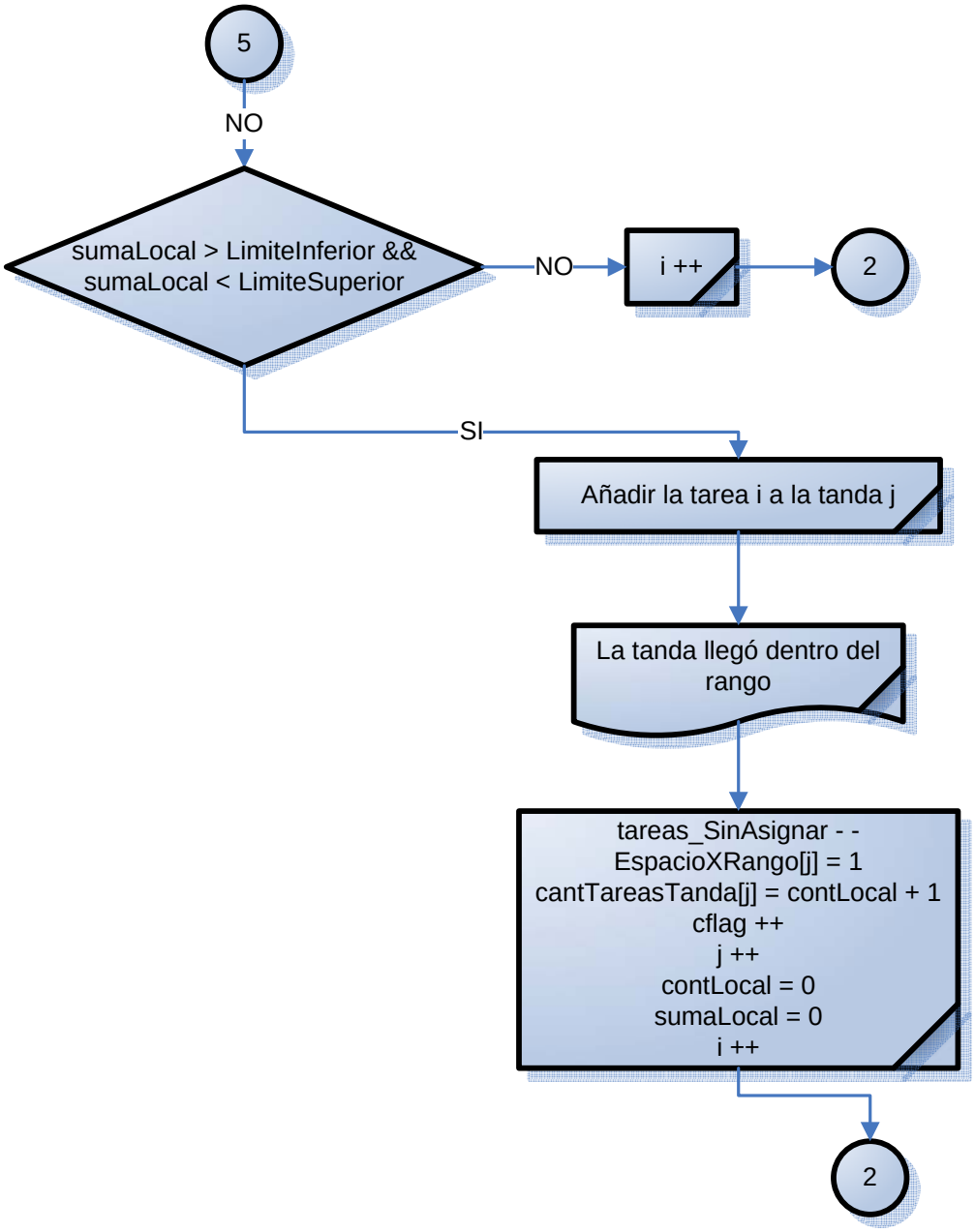
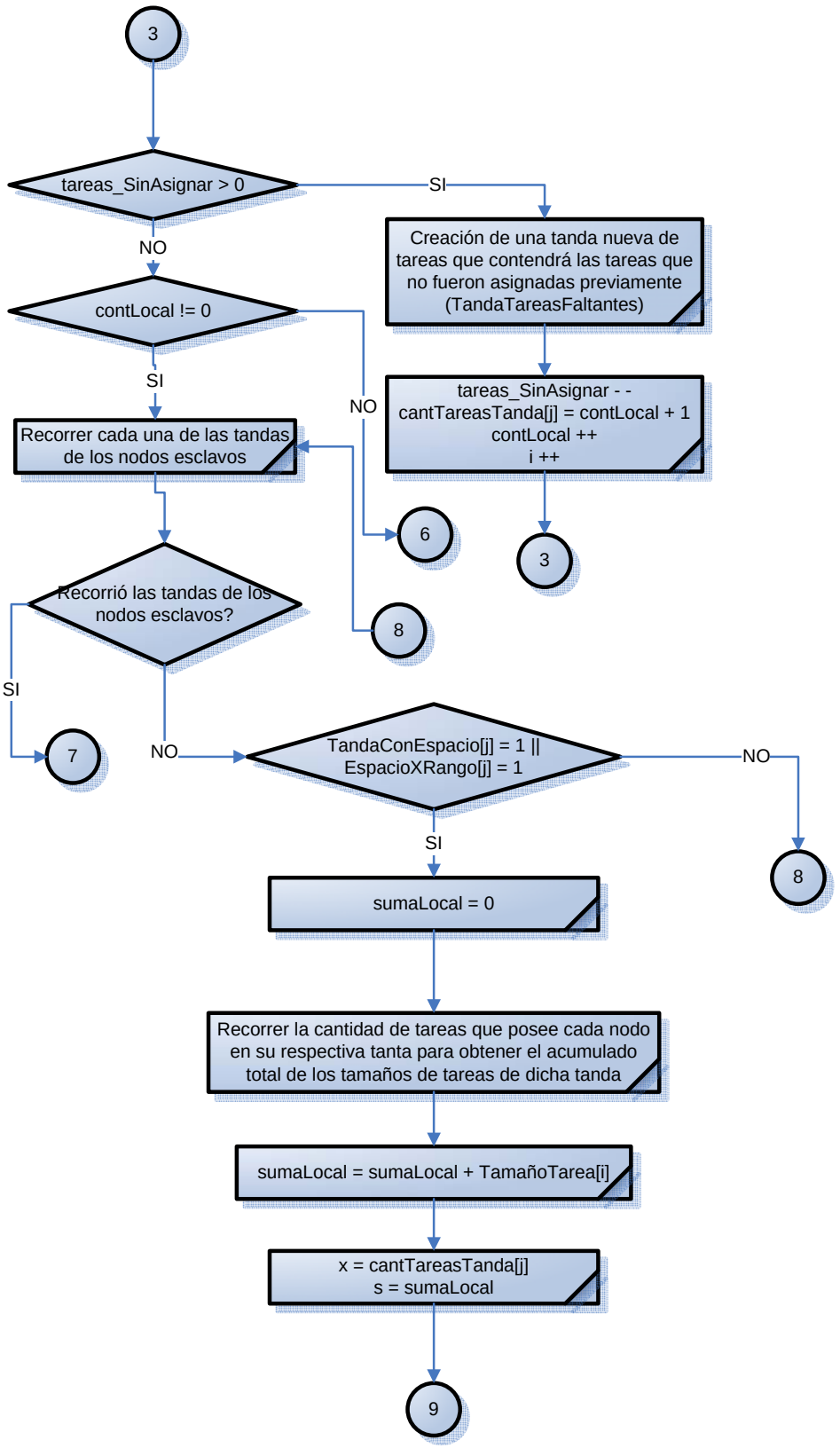
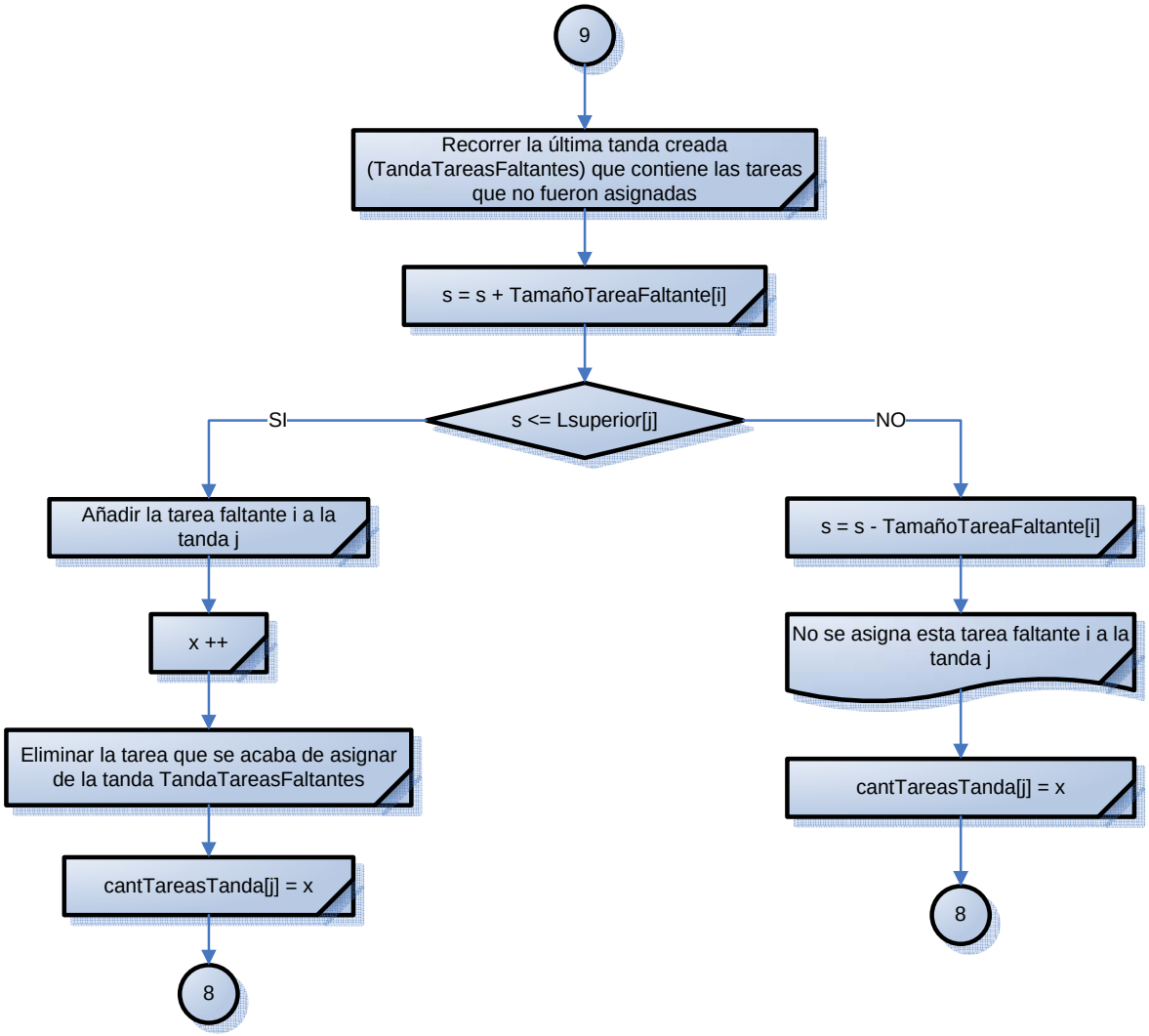
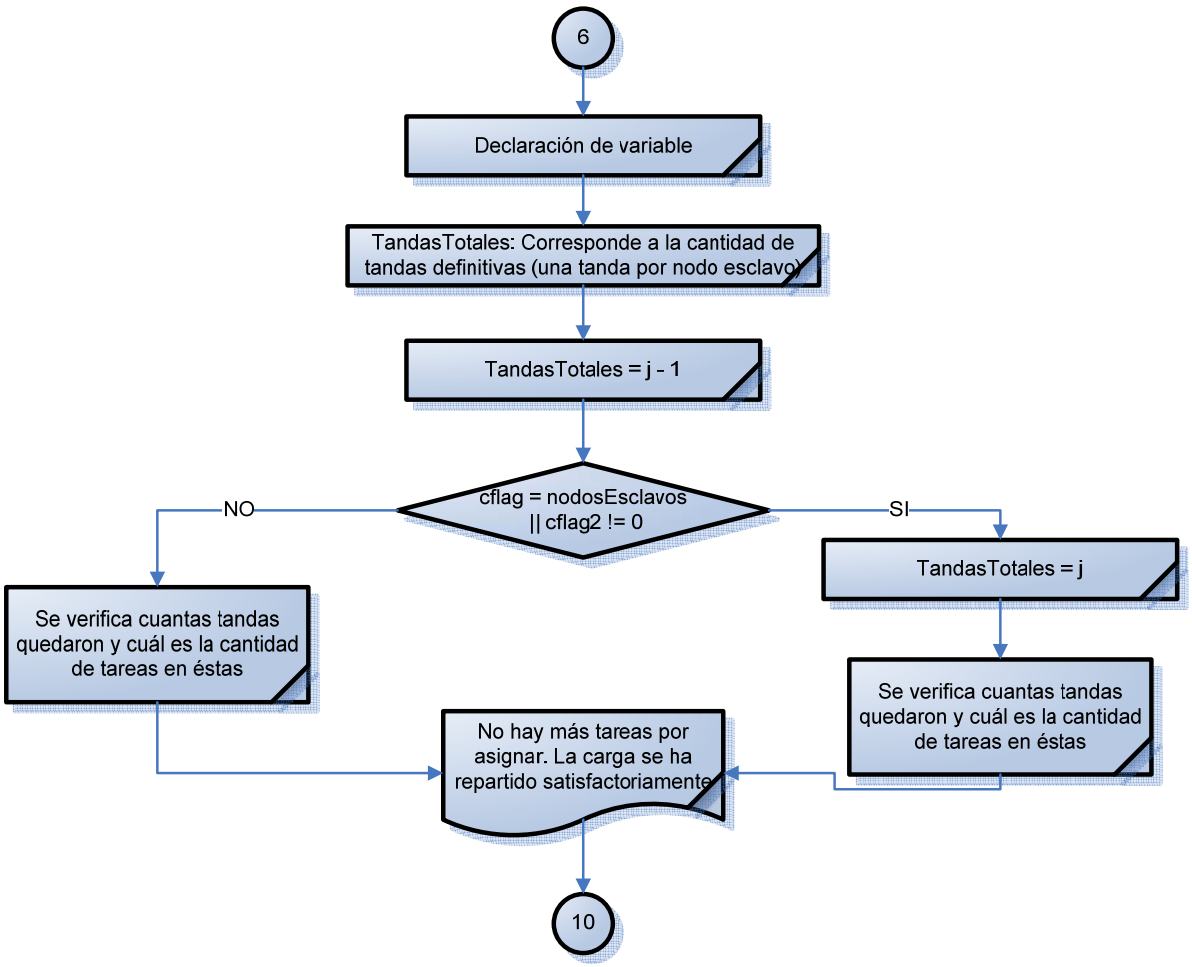
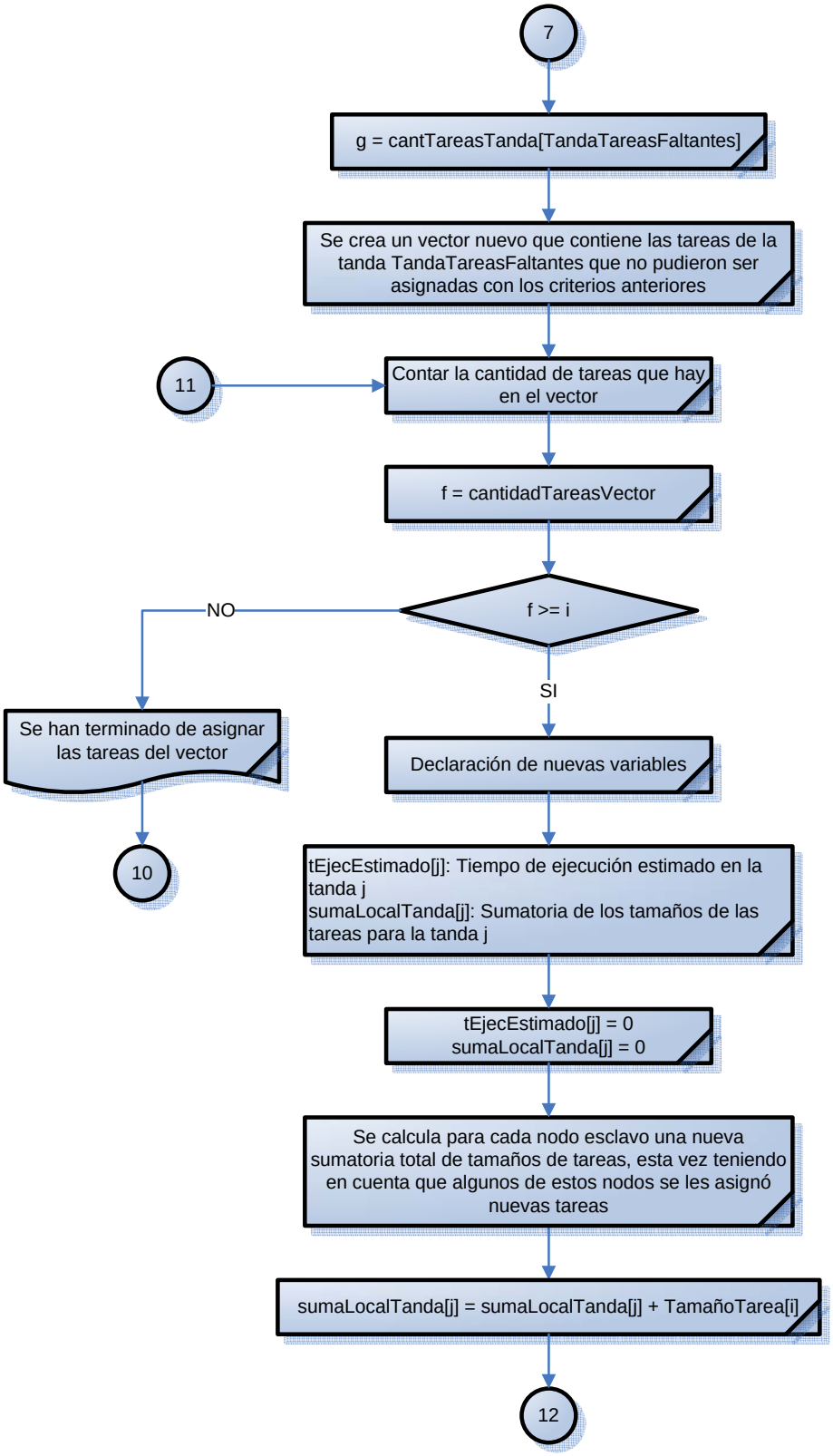

5
NO
sumaLocal > LimiteInferior &&
sumaLocal < LimiteSuperior
NO
i ++
2
SI
Añadir la tarea i a la tanda j
La tanda llegó dentro del rango
tareas_SinAsignar - -
EspacioXRango[j] = 1
cantTareasTanda[j] = contLocal + 1
cflag ++
j ++
contLocal = 0
sumaLocal = 0
i ++
2


3
tareas_SinAsignar > 0
SI
NO
Creación de una tanda nueva de tareas que contendrá las tareas que no fueron asignadas previamente (TandaTareasFaltantes)
contLocal != 0
SI
NO
tareas_SinAsignar - -
cantTareasTanda[j] = contLocal + 1
contLocal ++
i ++
Recorrer cada una de las tandas de los nodos esclavos
6
3
Recorrió las tandas de los nodos esclavos?
8
SI
7
NO
TandaConEspacio[j] = 1 ||
EspacioXRango[j] = 1
NO
SI
8
sumaLocal = 0
Recorrer la cantidad de tareas que posee cada nodo en su respectiva tanta para obtener el acumulado total de los tamaños de tareas de dicha tanda
sumaLocal = sumaLocal + TamañoTarea[i]
x = cantTareasTanda[j]
s = sumaLocal
9


9
Recorrer la última tanda creada (TandaTareasFaltantes) que contiene las tareas que no fueron asignadas
s = s + TamañoTareaFaltante[i]
s <= Lsuperior[j]
SI
NO
Añadir la tarea faltante i a la tanda j
x ++
Eliminar la tarea que se acaba de asignar de la tanda TandaTareasFaltantes
cantTareasTanda[j] = x
8
s = s - TamañoTareaFaltante[i]
No se asigna esta tarea faltante i a la tanda j
cantTareasTanda[j] = x
8


6
Declaración de variable
TandasTotales: Corresponde a la cantidad de tandas definitivas (una tanda por nodo esclavo)
TandasTotales = j - 1
cflag = nodosEsclavos || cflag2 != 0
NO
SI
Se verifica cuantas tandas quedaron y cuál es la cantidad de tareas en éstas
TandasTotales = j
Se verifica cuantas tandas quedaron y cuál es la cantidad de tareas en éstas
No hay más tareas por asignar. La carga se ha repartido satisfactoriamente
10


7
g = cantTareasTanda[TandaTareasFaltantes]
Se crea un vector nuevo que contiene las tareas de la tanda TandaTareasFaltantes que no pudieron ser asignadas con los criterios anteriores
11
Contar la cantidad de tareas que hay en el vector
f = cantidadTareasVector
f >= i
NO
SI
Se han terminado de asignar las tareas del vector
10
Declaración de nuevas variables
tEjecEstimado[j]: Tiempo de ejecución estimado en la tanda j
sumaLocalTanda[j]: Sumatoria de los tamaños de las tareas para la tanda j
tEjecEstimado[j] = 0
sumaLocalTanda[j] = 0
Se calcula para cada nodo esclavo una nueva sumatoria total de tamaños de tareas, esta vez teniendo en cuenta que algunos de estos nodos se les asignó nuevas tareas
sumaLocalTanda[j] = sumaLocalTanda[j] + TamañoTarea[i]
12

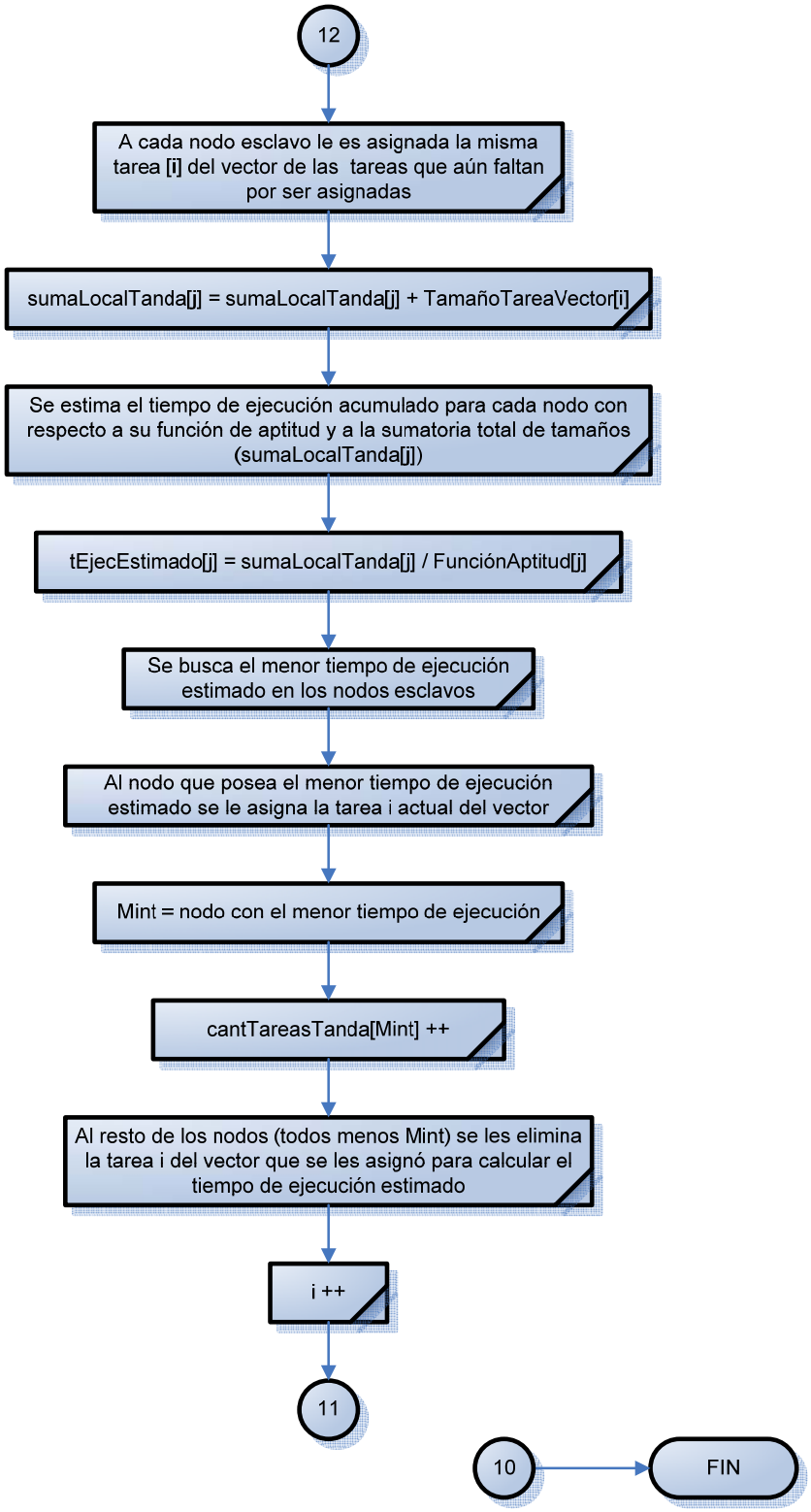


Figura 9. Diagrama de flujo – Método Propuesto

Para la realización de las pruebas se tomó un cluster heterogéneo a nivel de hardware modificando la cantidad de nodos esclavos y la cantidad de tareas iniciales. Los resultados obtenidos fueron de tomar 3, 5, 10, 20 y 50 nodos esclavos y ejecutando para cada uno de estos casos 5, 10, 20, 50, 100, 200, 500 y 1000 tareas iniciales. Hay que hacer constar que estas tareas sólo hacen parte de las condiciones iniciales y no de la cantidad total de tareas que ejecuta el cluster como tal, ya que el nodo maestro cuenta con un temporizador capaz de generar más tareas.

Los valores que se trabajaron en la simulación para los pesos asociados a la función de aptitud para cada uno de los nodos esclavos fueron los siguientes:
a = 0.5
b = 0.35
c = 0.15

Estos valores fueron escogidos de esta forma debido a que el tipo de cluster utilizado es de alto rendimiento, y como tal se requiere mayor velocidad de procesamiento, seguido de la capacidad de memoria y el tiempo de latencia.
Si se quisiera un cluster de otro tipo (disponibilidad o confiabilidad), sería necesario cambiar los pesos acorde al tipo a la preferencia del usuario; esto es, si fuese cluster de alta disponibilidad sería necesario utilizar mayor peso en la capacidad de memoria, seguido del tiempo de latencia y un peso menor para la velocidad de procesamiento. Pero si se quisiera un cluster de alta confiabilidad los valores cambiarían también.
Cabe aclarar que el algoritmo funciona para cualquier modificación en los pesos, sólo alterando el orden en el que se encuentren los nodos (mejor a peor aptitud) y no a la toma de decisión acorde al método como tal.

Para poder correr el algoritmo en el simulador se debe utilizar la siguiente línea de código variando las condiciones iniciales dependiendo de las preferencias y las pruebas que se requieran efectuar, tal y como se explicó en la figura 6 cambiando los parámetros:

ns tesis.tcl *nNodos Topologia Tarea Tamaño Asignación*

donde:
tesis.tcl es el nombre que se le dio al algoritmo que controla la ejecución de la simulación
nNodos: corresponde a la cantidad total de nodos en el cluster (en este trabajo se consideró trabajar con 3, 5, 10, 20 y 50 nodos esclavos, por lo que el parámetro que acá se varía sería 4, 6, 11, 21 y 51 nodos en el cluster considerando que sólo hay un nodo maestro)
Topologia: corresponde al tipo de topología con la que se desea trabajar (en nuestro caso es la topología en estrella dada por el valor 2)
Tarea: corresponde a la cantidad inicial de tareas (para esta investigación 5, 10, 20, 50, 100, 200, 500 y 1000)
Tamaño: corresponde al tamaño de todas las tareas (iniciales y generales). Este tamaño puede variar en ser aleatorio – 0, ó fijo –x (las pruebas realizadas se realizaron con tamaños aleatorios)

Asignación: corresponde al método con el que se desea probar (aleatorio – 0, uniforme – 1, propuesto -2)

6.2 RESULTADOS

Las siguientes tablas y gráficas presentan los tiempos de ejecución totales del algoritmo de acuerdo a la cantidad de tareas iniciales y a la cantidad de nodos esclavos con cada una de las condiciones iniciales mencionadas previamente para cada uno de los métodos (aleatorio, uniforme y propuesto).
Cabe aclarar que la cantidad total de tareas ejecutadas dentro del cluster es la suma entre las tareas iniciales dadas como parámetro inicial y las tareas generadas por el nodo maestro.

Tabla 2. Resultados de la ejecución de carga de trabajo con 3 nodos esclavos

Cantidad de tareas totales ejecutadas	**Tiempo de ejecución para cada método [seg]**		
	Aleatorio	**Uniforme**	**Propuesto**
99	6,361607	6,449856	5,71228
141	8,585431	8,912146	7,595445
149	8,767952	9,413576	8,042756
250	14,42972	15,71695	13,50117
396	23,14886	24,54513	21,01382
479	27,21146	29,05732	25,03099
553	31,74227	34,24748	29,50111
1179	66,19169	71,70027	61,83991

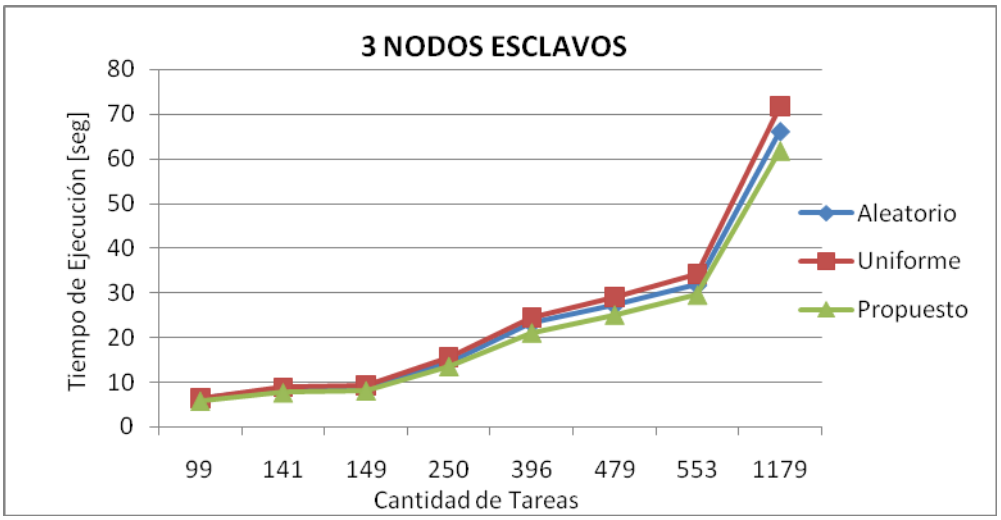


Figura 10. Resultados de la ejecución de carga trabajo con 3 nodos esclavos

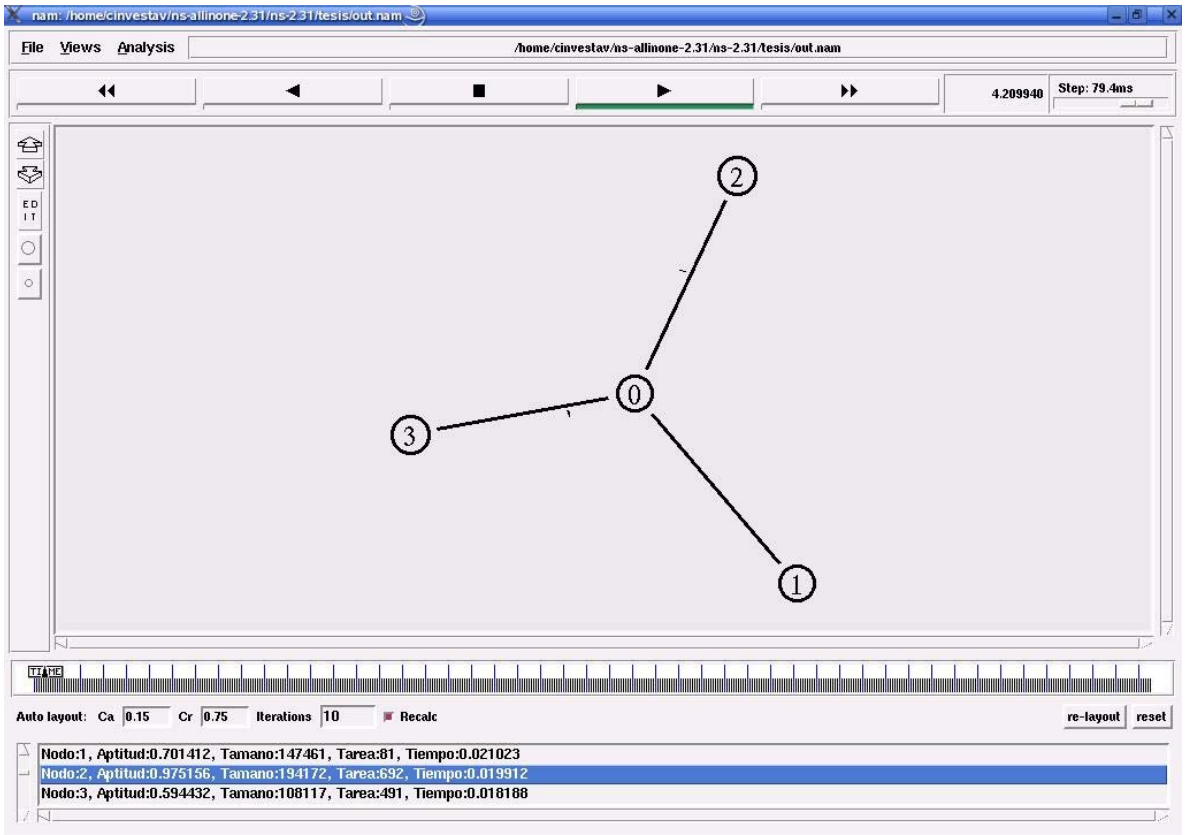


Figura 11. Simulación en el Network Simulator 2 (NS-2) con 3 nodos esclavos.

Tabla 3. Resultados de la ejecución de carga de trabajo con 5 nodos esclavos

Cantidad de tareas totales ejecutadas	**Tiempo de ejecución para cada método [seg]**		
	Aleatorio	**Uniforme**	**Propuesto**
111	5,116441	5,733315	4,094091
211	8,219201	10,459	7,776574
237	10,1707	11,9458	8,644418
270	11,35625	13,49152	9,80757
347	14,72112	17,31851	12,63169
434	18,53125	21,16988	15,39683
756	31,32903	37,28634	27,18736
1080	45,49806	52,90641	38,53823

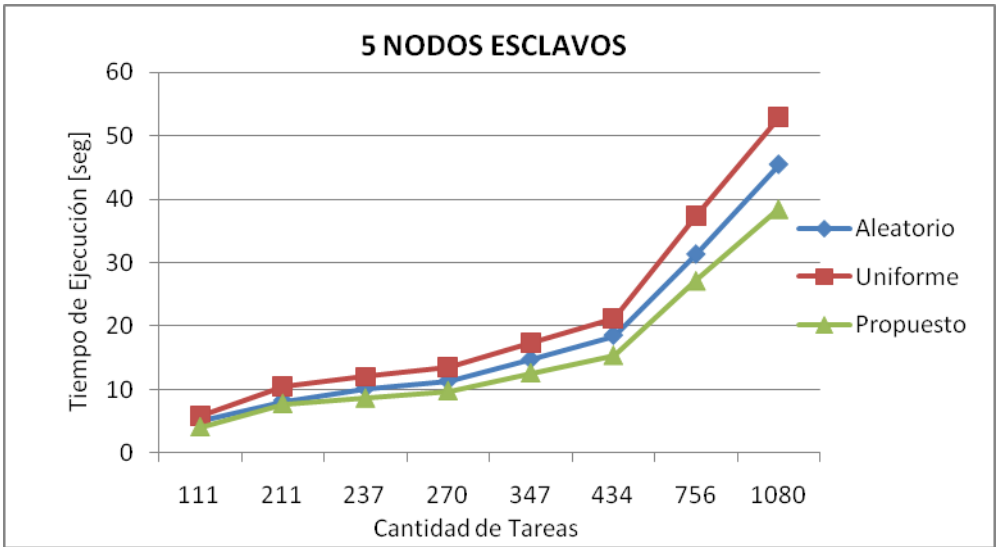


Figura 12. Resultados de la ejecución de carga trabajo con 5 nodos esclavos

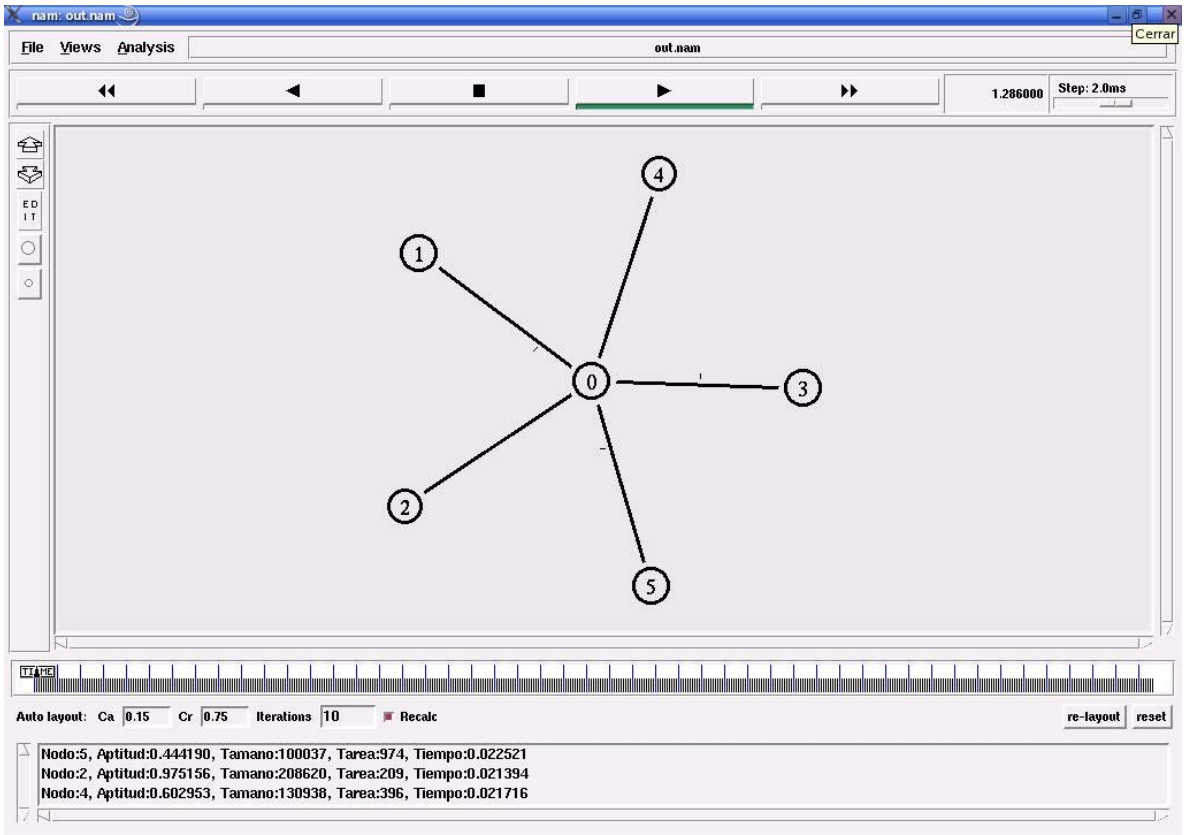


Figura 13. Simulación en el Network Simulator 2 (NS-2) con 5 nodos esclavos.

Como se puede apreciar en las tablas 2 y 3 y en las figuras 10 y 12, el mejor desempeño es presentado con el método propuesto, ya que optimiza el tiempo de ejecución mucho más

que los otros dos métodos probados (aleatorio y uniforme). Sin embargo el método aleatorio se comporta mucho mejor que el uniforme debido a que aunque en el aleatorio la cuestión de la distribución de las tareas es realizada inicialmente al azar, el hecho de que los nodos esclavos exijan nuevas tareas al culminar la que se encuentran realizando hace que los nodos no queden ociosos nunca, o por lo menos casi en la mayoría del tiempo, porque al final quizás habrá algunos esclavos finalizando las tareas pendientes y otros en espera de la terminación de estas para verificar el tiempo total de ejecución. Este método (aleatorio) es más óptimo que el uniforme por el hecho de que el uniforme lo que pretende es hacer la repartición de la carga de la manera más ecuánime posible para todos los esclavos sin importar sus capacidades reales, por esto, siempre habrá nodos más veloces que otros y en algunas ocasiones estarán ociosos.

Tabla 4. Resultados de la ejecución de carga de trabajo con 10 nodos esclavos

Cantidad de tareas totales ejecutadas	**Tiempo de ejecución para cada método [seg]**		
	Aleatorio	**Uniforme**	**Propuesto**
46	1,981356	2,010801	1,210137
121	4,891438	6,420832	2,871366
222	6,575336	11,58503	5,290643
273	9,287934	14,64966	6,429258
365	11,01577	18,82134	8,330664
496	15,14604	26,10157	11,38643
832	26,34462	43,12187	18,79545
1209	36,80888	62,41019	27,19923

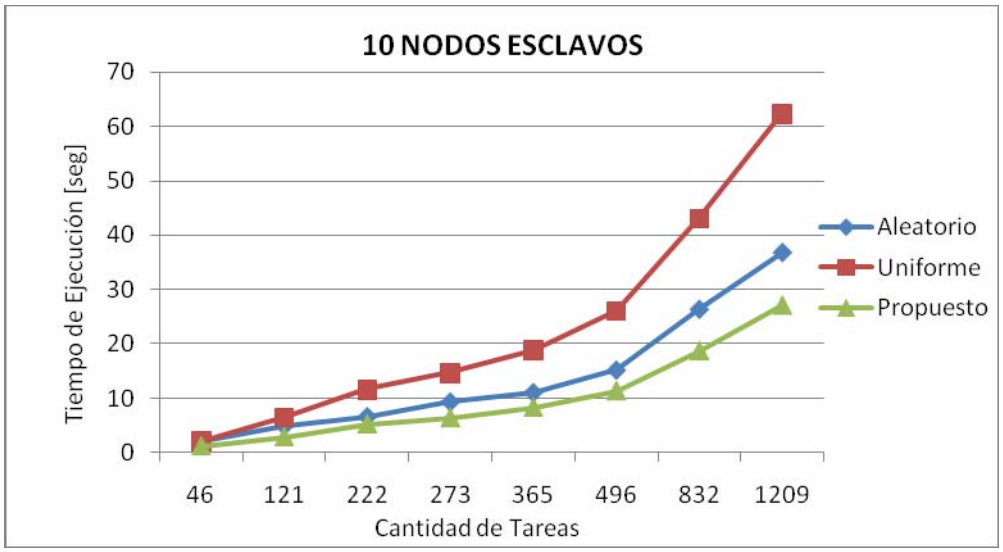


Figura 14. Resultados de la ejecución de carga trabajo con 10 nodos esclavos

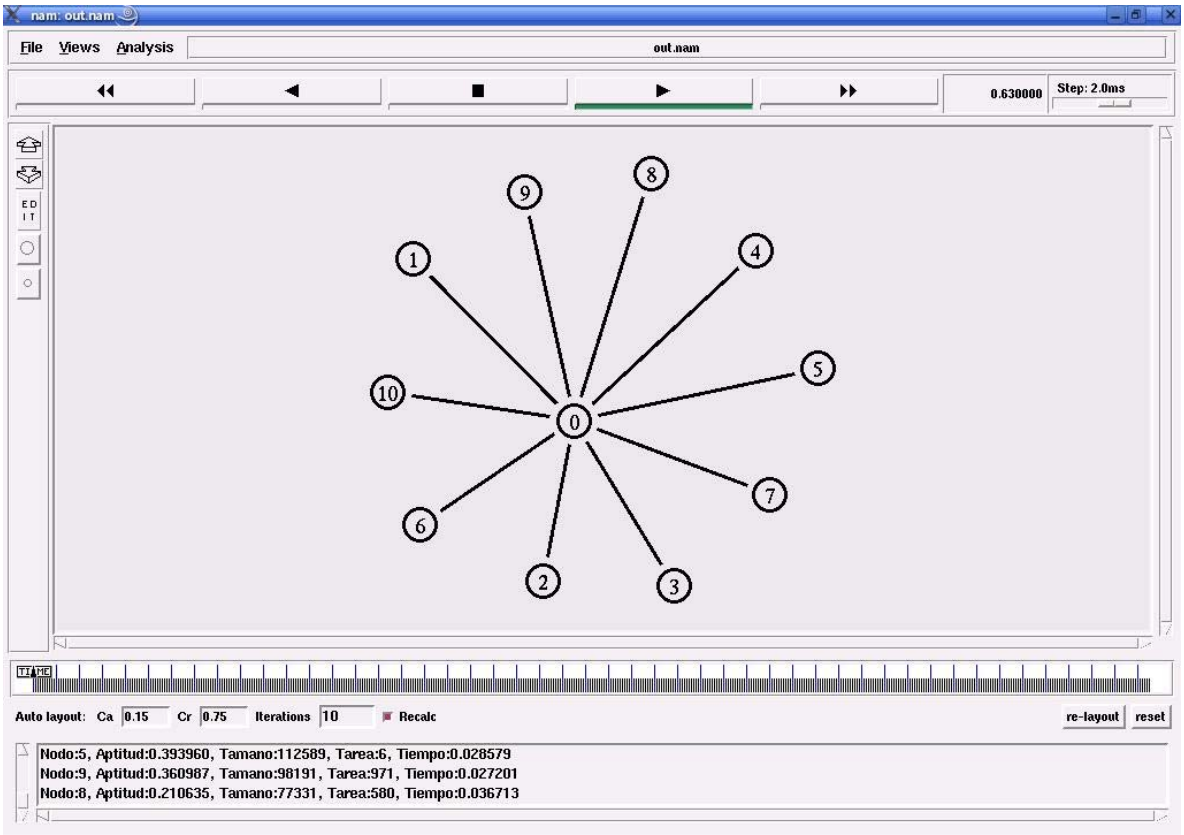


Figura 15. Inicio de simulación en el NS-2 con 10 nodos esclavos.

Los resultados de la simulación al trabajar con 10 nodos esclavos son observados en la tabla 4 y en la figura 14, donde también se observa que el método propuesto da resultados más pequeños en los tiempos totales de simulación con cualquier cantidad total de tareas ejecutadas siendo mas eficiente que el método aleatorio y el uniforme respectivamente.

Por los resultados vistos previamente en las simulaciones con 3, 5 y 10 nodos esclavos se puede observar que cuando se aumenta la cantidad de nodos esclavos la diferencia en los tiempos de ejecución en los tres métodos es más notoria. Esto se debe a que la cantidad total de tareas ha ejecutar por el cluster es muy similar en las diferentes simulaciones, por lo tanto a mayor cantidad de nodos la repartición de trabajo en los nodos se distribuye de mejor manera y la cantidad de trabajo por nodo disminuye, por ende el tiempo de ejecución total es menor cuanto mas nodos haya.

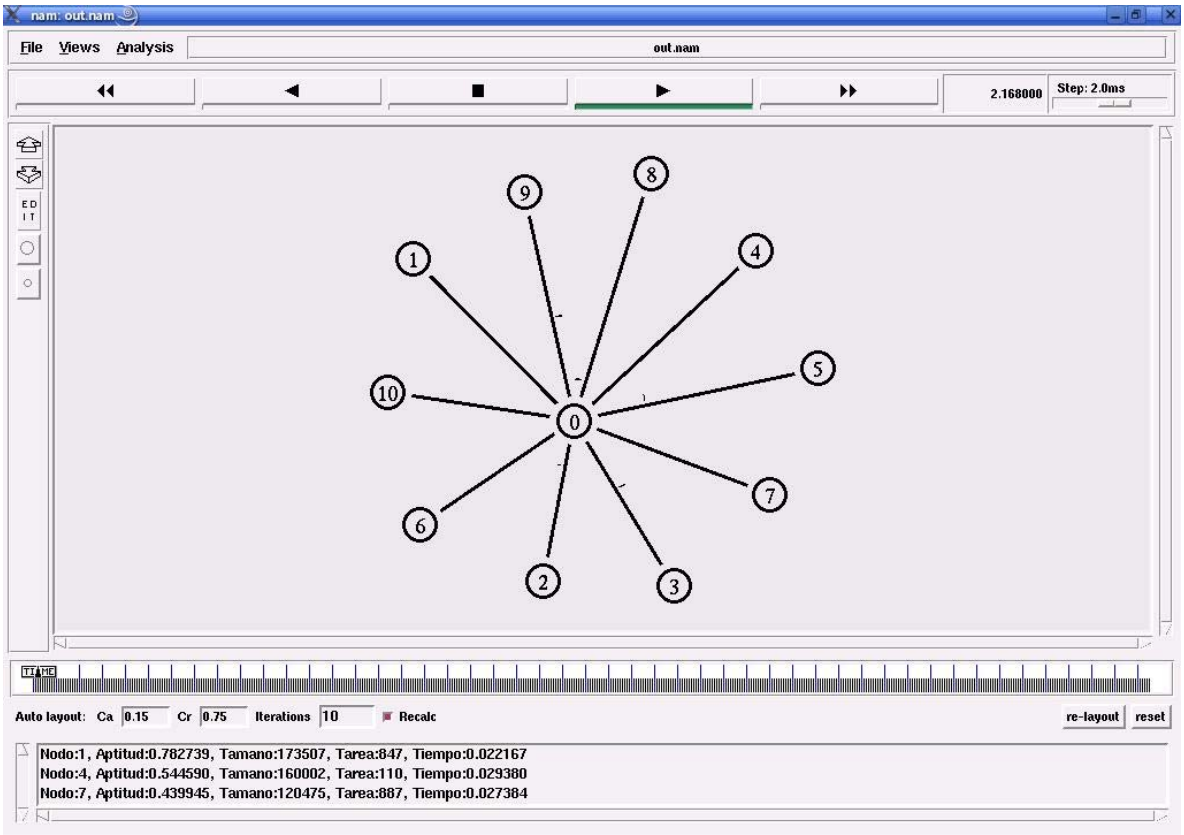


Figura 16. Simulación en el Network Simulator 2 (NS-2) con 10 nodos esclavos.

Las figuras 11, 13, 15 y 16 presentan el proceso en las simulaciones utilizando el NS-2, donde se aprecia visualmente como se encuentran ubicados los nodos del cluster (maestro y esclavos) con la topología estrella de acuerdo a la cantidad de nodos seleccionados en el cluster desde el principio (enviados como parámetro inicial) y cómo se realiza el envío de paquetes a través de estos.

Cuando se habla de envío de paquetes se refiere a la comunicación uno a uno que tiene cada nodo esclavo con el maestro para el envió de atributos, tareas y notificaciones.

Tabla 5. Resultados de la ejecución de carga de trabajo con 20 nodos esclavos

Cantidad de tareas totales ejecutadas	Tiempo de ejecución para cada método [seg]		
	Aleatorio	**Uniforme**	**Propuesto**
53	1,711478	1,33216	0,9655169
91	2,624431	2,463285	1,390105
225	4,249504	7,258994	2,810762
288	4,977993	9,038145	3,362952
292	5,378446	9,823807	3,403296
477	8,855315	15,63398	5,510817
772	13,22734	25,35582	8,870587
1135	18,91291	36,77995	12,84017

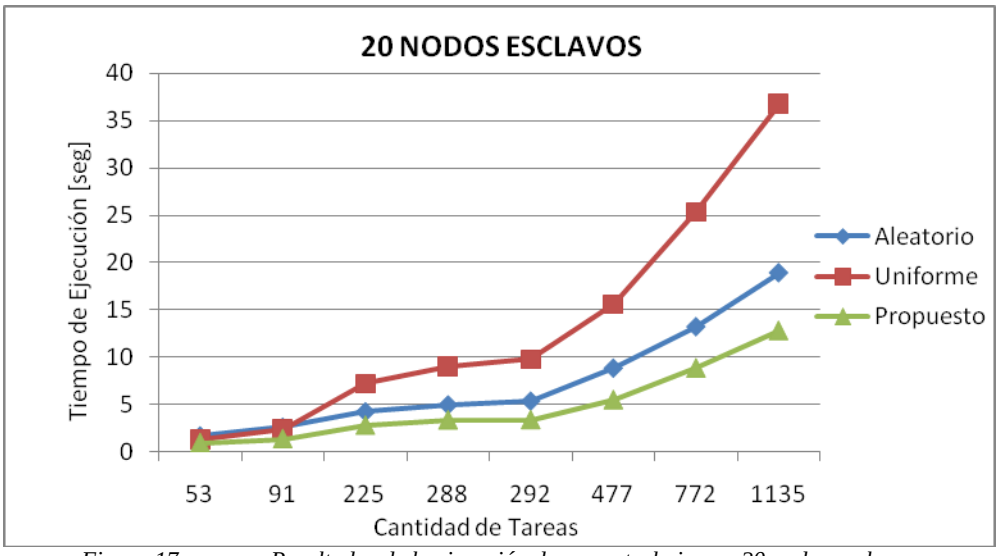


Figura 17. Resultados de la ejecución de carga trabajo con 20 nodos esclavos

Tabla 6. Resultados de la ejecución de carga de trabajo con 50 nodos esclavos

Cantidad de tareas totales ejecutadas	Tiempo de ejecución para cada método [seg]		
	Aleatorio	**Uniforme**	**Propuesto**
100	1,324659	0,784898	0,7829418
198	1,844983	1,995441	1,179421
218	2,034206	2,474166	1,325206
318	2,713001	3,965946	1,483486
387	3,428265	4,141913	1,762839
387	3,492864	4,120583	1,778959
785	5,918553	9,403513	3,50783
1166	7,719521	13,71055	5,286702

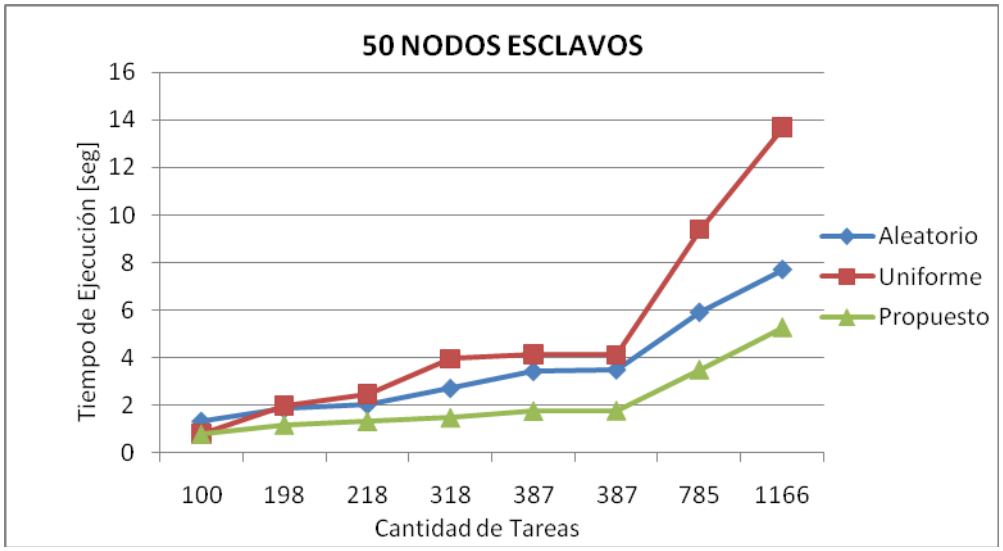


Figura 18. Resultados de la ejecución de carga trabajo con 50 nodos esclavos

6.3 ANÁLISIS

De manera general se puede observar que para todos los casos el método propuesto mejora los tiempos de ejecución, sin embargo se puede apreciar que los mejores resultados ocurren cuando se trabaja con mayor cantidad de tareas y mayor cantidad de nodos, por lo que se podría concluir que el método es altamente escalable y contribuye al tiempo total de ejecución logrando minimizarlo, cumpliendo así, muchas de las expectativas de esta investigación (figuras 15, 17 y 18).

La forma en cómo funcionan cada uno de los métodos en cuanto a la distribución de las tareas puede verse en las ilustraciones siguientes en donde para cada uno de los métodos la distribución es diferente y el tiempo de ejecución también se ve afectado tal y como se vio en los resultados experimentados.
Se verán los resultados con 10 nodos esclavos y 10 tareas iniciales, de las cuales al final se generan dos tandas más de tareas una 165 y otra de 190 haciendo un acumulado total de 365 tareas totales (tabla 4, renglón 5).

Tabla 7. Funciones de aptitud de 10 nodos esclavos

Nodo	Función de Aptitud
1	0,641703
2	0,875157
3	0,542113
4	0,54459
5	0,39396
6	0,782739
7	0,439945
8	0,210635
9	0,360987
10	0,514381

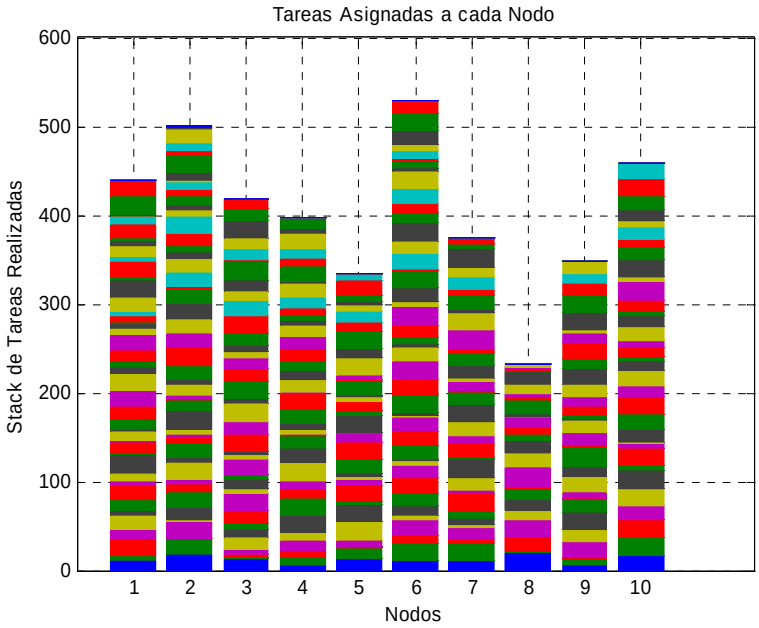


Figura 19. *Método aleatorio con 10 nodos esclavos para 10 Tareas Iniciales como condición inicial.*

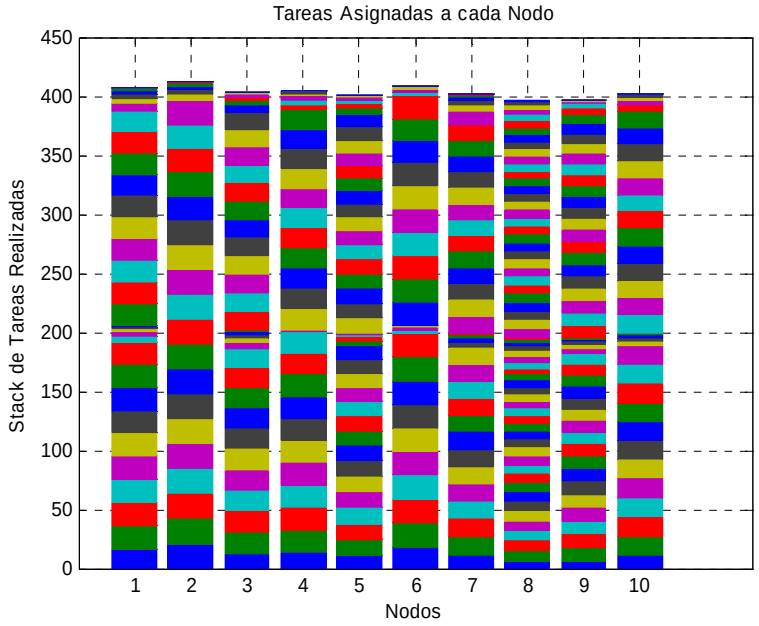


Figura 20. *Método uniforme con 10 nodos esclavos para 10 Tareas Iniciales como condición inicial.*

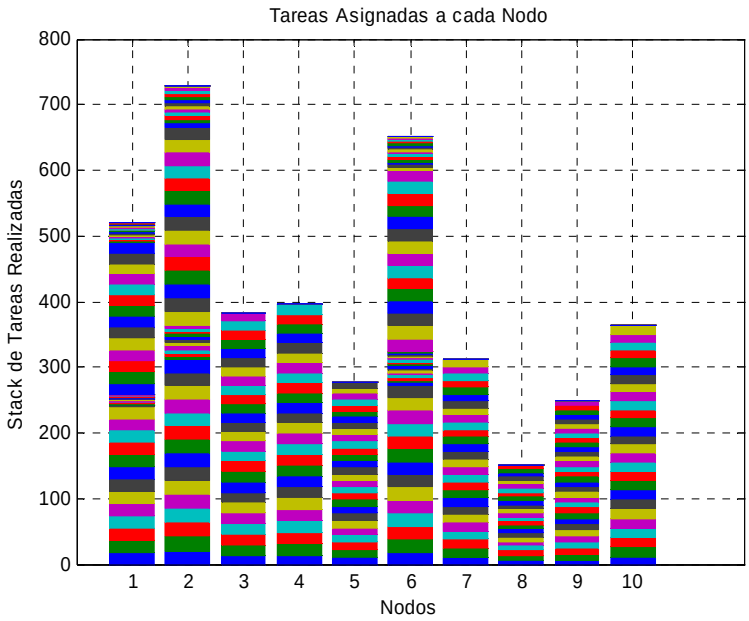


Figura 21. *Método propuesto con 10 nodos esclavos para 10 Tareas Iniciales como condición inicial.*

De la tabla 7 y la figura 21 se puede apreciar que el nodo con mayor eficiencia por su velocidad es el nodo 2 y el de menor aptitud es el nodo 8 en el método propuesto corroborando que la asignación de carga se hace en base a la capacidad de cada nodo.

Por otro lado, en la figura 20 se observa que la asignación en el método uniforme en cuanto a carga es similar para cada nodo esclavo, por lo que el tiempo de ejecución total en este caso es determinado por el nodo más lento, en este caso por la función de aptitud asociada es el nodo 8, dato que es fácilmente observado en la tabla 4 y figura 14 donde el tiempo de ejecución para 365 tareas es de 18,82134 segundos.

Para el método aleatorio se aprecia que al momento de hacer peticiones de tareas y asignarlas por parte del maestro, sólo se hace si un nodo culminó su tarea actual y en la pila de tareas pendientes aún quedan por asignar. De la figura 19 se observa que el nodo con menor asignación de carga es el nodo 8, y esto se debe a que su velocidad (función de aptitud = 0,210635) no alcanza para cubrir con mayor capacidad de carga y por esto no puede pedir gran cantidad de tareas.

A continuación en la tabla 8 se observarán los porcentajes de mejora del método propuesto con respecto a los dos métodos de prueba (aleatorio y uniforme).

Tabla 8. Porcentajes de mejora de los diferentes métodos

Aleatorio (A)	**Uniforme (U)**	**Propuesto (P)**	**Propuesto vs Aleatorio**	**Propuesto vs Uniforme**	**Aleatorio vs Uniforme**
			TE(i) = (A(i) - P(i)) / A(i)	**TE(i) = (U(i) - P(i)) / U(i)**	**TE(i) = (U(i) - A(i)) / U(i)**
6,361607	6,449856	5,71228	0,10206965	0,114355421	0,013682321
8,585431	8,912146	7,595445	0,115309994	0,147742306	0,036659521
8,767952	9,413576	8,042756	0,082709851	0,1456216	0,068584351
14,42972	15,71695	13,50117	0,064349828	0,140980279	0,08190075
23,14886	24,54513	21,01382	0,092230892	0,143870087	0,056885826
27,21146	29,05732	25,03099	0,080130577	0,138565084	0,063524785
31,74227	34,24748	29,50111	0,070604906	0,138590343	0,0731502
66,19169	71,70027	61,83991	0,065745111	0,137521937	0,076827884
5,116441	5,733315	4,094091	0,19981663	0,285912077	0,107594646
8,219201	10,459	7,776574	0,053852802	0,256470599	0,214150397
10,1707	11,9458	8,644418	0,150066564	0,276363408	0,148596159
11,35625	13,49152	9,80757	0,136372482	0,273056705	0,158267564
14,72112	17,31851	12,63169	0,141934173	0,270624898	0,149977683
18,53125	21,16988	15,39683	0,169142395	0,272701121	0,124640763
31,32903	37,28634	27,18736	0,132199114	0,270849324	0,159771916
45,49806	52,90641	38,53823	0,152969819	0,2715773	0,140027456
1,981356	2,010801	1,210137	0,389237976	0,39818162	0,014643418
4,891438	6,420832	2,871366	0,412981213	0,552804683	0,238192496
6,575336	11,58503	5,290643	0,195380586	0,543320734	0,432428228
9,287934	14,64966	6,429258	0,307783841	0,561132613	0,365996617
11,01577	18,82134	8,330664	0,243751095	0,557381993	0,414719143
15,14604	26,10157	11,38643	0,248223958	0,563764555	0,419726859
26,34462	43,12187	18,79545	0,286554522	0,564131843	0,389065919
36,80888	62,41019	27,19923	0,261068796	0,564186073	0,410210416
1,711478	1,33216	0,9655169	0,435857837	0,275224523	-0,28473907
2,624431	2,463285	1,390105	0,470321376	0,435670253	-0,065419146
4,249504	7,258994	2,810762	0,338567042	0,612789045	0,414587751
4,977993	9,038145	3,362952	0,324436173	0,627915684	0,449224039
5,378446	9,823807	3,403296	0,367234328	0,653566484	0,452508992
8,855315	15,63398	5,510817	0,377682556	0,647510295	0,43358537
13,22734	25,35582	8,870587	0,32937484	0,650155783	0,478331208
18,91291	36,77995	12,84017	0,321089668	0,65089213	0,485782063
1,324659	0,784898	0,7829418	0,408948416	0,002492298	-0,687682986
1,844983	1,995441	1,179421	0,360741535	0,408942184	0,075400876
2,034206	2,474166	1,325206	0,348538939	0,464382746	0,177821537
2,713001	3,965946	1,483486	0,453193714	0,625943974	0,315925885
3,428265	4,141913	1,762839	0,485792668	0,574390143	0,172299128
3,492864	4,120583	1,778959	0,490687585	0,568274926	0,152337424
5,918553	9,403513	3,50783	0,407316282	0,626966007	0,370601923
7,719521	13,71055	5,286702	0,315151549	0,614406278	0,436964892

			Promedio de TE(i) P vs. A	Promedio de TE(i) P vs. U	Promedio de TE(i) A vs. U
			0,259735532	0,400730734	0,19341888

Porcentaje de mejora del método propuesto con respecto al método aleatorio	**25,97%**
Porcentaje de mejora del método propuesto con respecto al método uniforme	**40,07%**
Porcentaje de mejora del método aleatorio con respecto al método uniforme	**19,34%**

La tabla anterior recopila todos los resultados obtenidos de los tiempos de ejecución de cada una de las pruebas realizadas.

Como se pudo observar en todos los casos, el método propuesto obtuvo un mejor desempeño con respecto a los otros dos métodos (menor tiempo de ejecución para cada prueba realizada). Por lo tanto se calculó la diferencia del tiempo de ejecución de cada uno de los métodos con respecto al propuesto sobre el tiempo de ejecución del método con el que se está comparando al propuesto (aleatorio o uniforme). Posteriormente se realizó el cálculo del promedio para cada uno de los casos para hallar el cálculo del porcentaje de mejora.

Se puede apreciar claramente que el método propuesto es el que mejor comportamiento presenta, ya que mejora en aproximadamente un 40% al método uniforme y en aproximadamente un 26% al método aleatorio.

Si se quisiera comparar el método propuesto con los sistemas de encolamiento de tareas más utilizados hoy en día (anexos 9.4 y 9.5, donde se presentan los entornos de computación y los sistemas de encolamiento respectivamente) se podría realizar un mapeo comparativo entre los métodos de validación utilizados en esta investigación (Aleatorio y Uniforme) y estos sistemas (Condor, openMosix, PBS y SGE).

En base a [LIARTE, 2007], la funcionalidad básica de Condor es la siguiente: un usuario lanza una tarea sobre Condor. Condor busca una máquina adecuada en su red, y comienza a ejecutar la tarea en dicha máquina. Si por algún motivo un usuario comienza a utilizar dicha máquina, entonces Condor migrará la tarea (checkpointing) hacia otra máquina que esté libre, continuando allí la ejecución, incluso si cambia el entorno de ejecución de Condor.

En [NIEeS, 2006], se comenta sobre los servicios de administración de los recursos de Condor, en donde un maestro (nodo central) es el responsable de recolectar la información necesaria de los nodos que forman el sistema. Esta información contiene las prioridades del usuario, además las tareas que deberán ejecutarse.

Como se puede apreciar, Condor se podría comparar con el método aleatorio debido a que ambos poseen un nodo central, el cual guía al resto de los nodos conectados a la red asignándoles trabajo cada vez que estos lo exijan.

La diferencia entre Condor y al método aleatorio radica en la capacidad que tiene Condor para migrar trabajo a nodos ociosos una vez que el nodo que se encuentre realizando alguna tarea esté siendo ocupado por un determinado usuario. En el método aleatorio el maestro envía trabajo a sus esclavos y estos deben realizarlo inmediatamente; una vez que finalicen dicho trabajo notificaran al maestro su estado de disponibilidad.

Por otro lado y según [VERDEJO, 2009], el sistema openMosix es capaz de balancear automáticamente la carga entre los diferentes nodos del cluster permitiendo la adición o sustracción de nodos durante la ejecución de las tareas sin necesidad de interrumpir el servicio.

La carga se distribuye entre los distintos nodos atendiendo a parámetros como tipo de conexión, memoria disponible y velocidad de CPU.

De acuerdo a [CATALÁN, 2004], en OpenMosix las tareas se dirigen a los nodos que poseen menos carga evitando así la sobrecarga del sistema. Se procura entonces que los nodos que conforman el cluster nunca se encuentren inactivos ó sobrecargados.

El sistema openMosix también es comparado con el método aleatorio, por el hecho de que cada nodo ejecuta lo que puede hacer y en el momento apropiado. Cuando un nodo está bastante cargado envía su trabajo a uno que se encuentra en estado ocioso. Sin embargo, en openMosix no existe un maestro constante que este dirigiendo la carga de los nodos esclavos. Basado en las capacidades propias un nodo saturado pide ayuda a sus compañeros. En el método aleatorio existe un maestro único encargado sólo de la asignación y distribución de tareas. Este maestro nunca efectúa ejecución de trabajo.

El sistema de colas Portable Batch System (PBS) utiliza para el momento en que arriban nuevas tareas la política FIFO. Usa un servidor de tareas y un administrador de tareas encargados de la distribución y control de estas; aparte de estos componentes existen los ejecutores de las tareas que trabajan en la realización de cada una de las tareas que se les fue asignada.

Al comparar el PBS con uno de los métodos de validación, se observó que podría asemejarse al método aleatorio al igual que el openMosix y el Condor por el hecho de que el sistema posee máquinas administrativas (servidor y administrador de tareas) y máquinas ejecutoras (ejecutores de tareas), que en relación con el método aleatorio las primeras serían equivalentes al nodo maestro y las segundas a los nodos esclavos. En ambos casos (sistema PSB y método aleatorio) las tareas que llegan primero serán las que se ejecutarán primero. El PBS tiene como falencia el hecho de que si en algún momento un nodo falla, las tareas que se encontraba realizando se pierden y la posibilidad de recuperarse es casi nula.

En la actualidad existe también el sistema Sun Grid Engine (SGE), el cual debido a su gran capacidad de escalabilidad, facilidad de uso y configuración, es el sistema clustering de mayor éxito.

Como ventaja posee la posibilidad de suspender y reanudar las tareas que se encuentra ejecutando.

Para la realización del balanceo de carga se calcula un valor de carga significativo por medio de una expresión algebraica, para cada uno de los nodos que conforman el cluster. Después de que se calcula dicho valor, el administrador analiza la disponibilidad que tiene cada nodo en base a su capacidad (valor de carga) con el fin de realizar la asignación de tareas. Los nodos ejecutores se organizan de acuerdo a su capacidad de carga, para cada uno se determina un umbral que determina la cantidad de trabajo que realizará, que esta denotado por la media del parámetro de carga de la CPU [SUN, 2007].

El método que se asocia a SGE es el método uniforme por el hecho de que la carga de trabajo para los nodos es determinada por un umbral además de que se usan parámetros similares para cada nodo esclavo (el tiempo de CPU, la memoria y el espacio de nombres System IO). La diferencia radica en que se ajusta un tiempo de ejecución límite para que los esclavos realicen su trabajo, si no efectúan su trabajo en dicho tiempo las tareas serán migradas a los nodos que sí lograron culminar, esto se hace con el fin de obtener resultados razonables.

Luego de haber realizado el anterior análisis comparativo entre los métodos de validación y los sistemas de encolamiento, se puede decir que el método propuesto en el momento de ser escalado a un sistema real, puede llegar a ser aproximadamente 26% mejor que los sistemas Portable Batch System (PBS), openMosix y Condor, por el hecho de que de alguna manera estos fueron similares al método aleatorio y solo difirieron de este en algunos aspectos. Por otro lado, el método propuesto sería aproximadamente 40% mejor que el sistema Sun Grid Engine (SGE), porque este se asemejó mucho más al método uniforme con pequeñas variantes de este.