SISTEMAS DISTRIBUIDOS

Memoria práctica 1

Sergio García Esteban 755844
Irene Fumanal Lacoma 758325

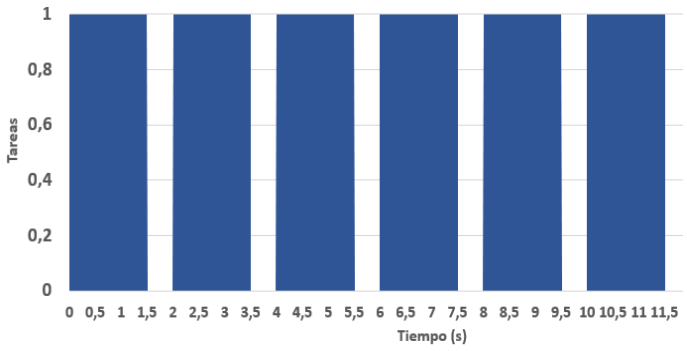
Descripción de las máquinas

Las máquinas con las que trabajamos poseen las siguientes características: procesador i5-4570 con **4 núcleos** y 3,30 GHz de frecuencia y una memoria de 11,6 GiB.

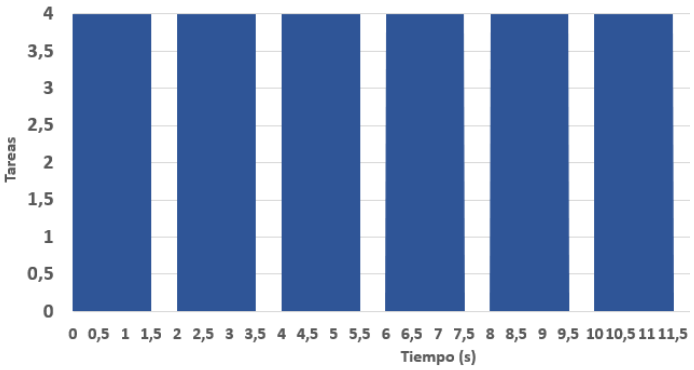
Análisis de la carga de trabajo

A continuación, presentamos la carga de trabajo para cada uno de los escenarios mediante unas gráficas que muestran el número de tareas en función del tiempo.

Escenario 1:

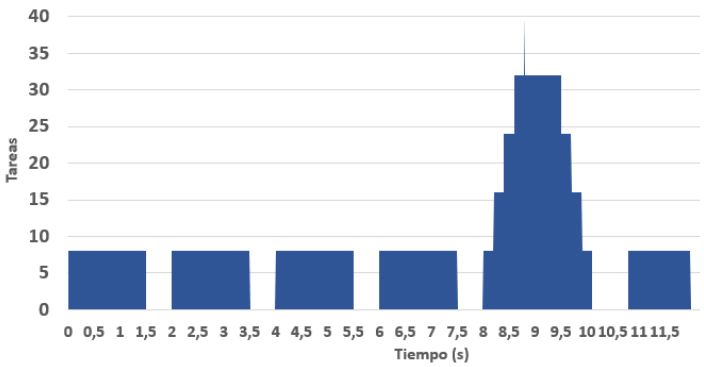


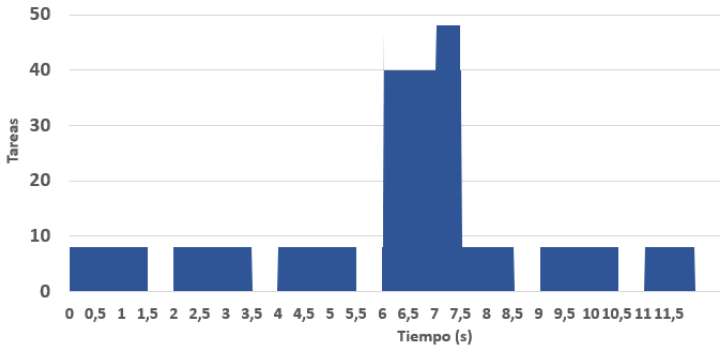
Escenario 2



Escenario 3:

Las siguientes graficas muestran el caso mejor y peor respectivamente:





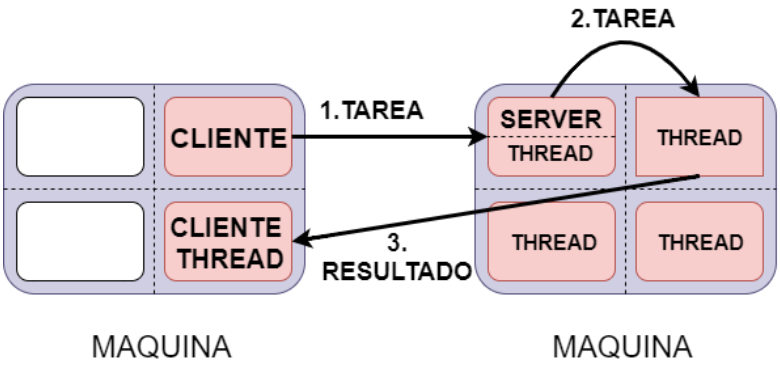
Análisis del tiempo de ejecución

Hemos realizado 10 mediciones de cada algoritmo de Fibonacci y la media de las mediciones nos ha dado: 1 404 720 us (fibonacci) y 26 us (fibonacci_tr).

Patrones arquitecturales

Cliente/Servidor:

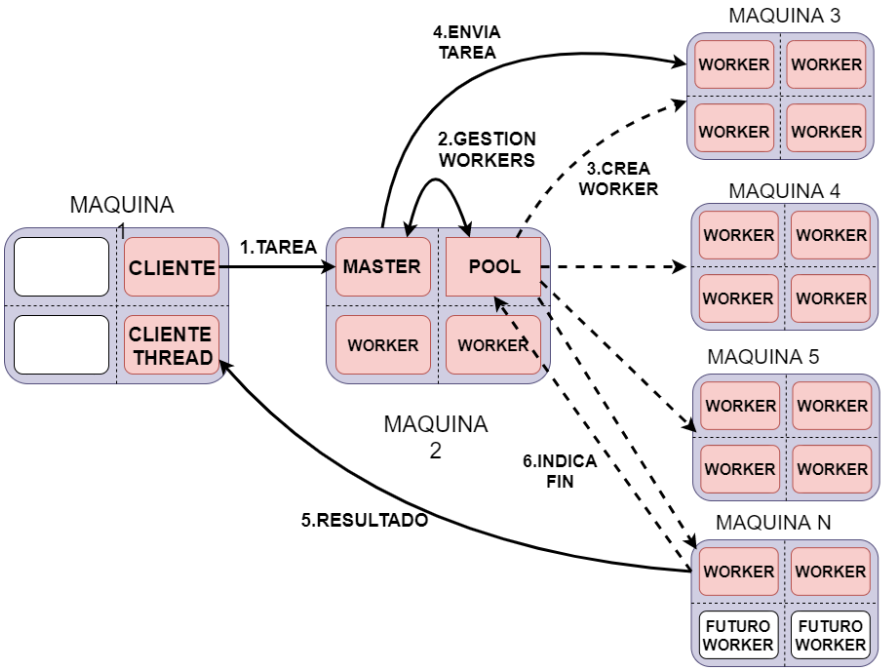
La carga de trabajo de los dos primeros escenarios es inferior a la del tercero y soportan una arquitectura de este tipo. El cliente enviará una tarea al servidor, el servidor creará un proceso donde se realizará dicha tarea y enviará el resultado directamente al thread del cliente, que estará siempre esperando resultados. En el siguiente dibujo mostramos el esquema correspondiente a dicha arquitectura.



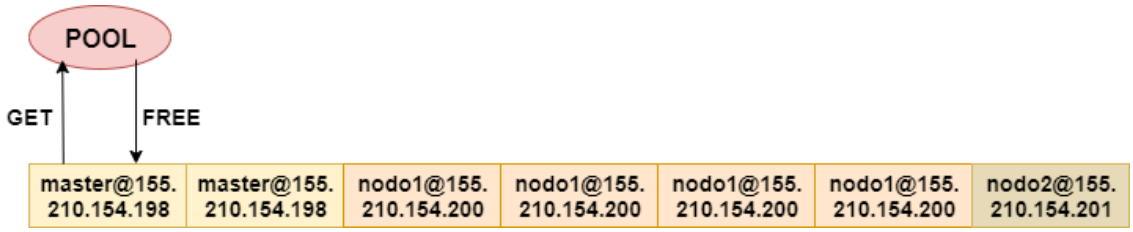
Master/Worker:

El escenario 3 tiene una carga de trabajo superior a los anteriores y por lo tanto necesita una arquitectura de este tipo. El cliente enviará las tareas al master y este solicitará recursos al “Pool de recursos”, componente que creará los workers en función de la demanda de trabajo. En el momento que un worker termina de realizar la tarea, envía el resultado directamente al cliente (thread) que se encarga de recibir los resultados, y además le comunicará al pool que ha terminado.

En el siguiente esquema mostramos nuestra arquitectura.



El pool es el encargado de gestionar los workers y asignárselos al master en función de la demanda de trabajo. Para ello el pool contiene un buffer en el que almacena los workers disponibles. El buffer sigue una política LIFO (Last In First Out). Para facilitar la implementación, en el buffer almacenamos 4 veces cada nodo, indicando de esta forma los 4 workers que puede haber por máquina, incluyendo también dos más en la propia máquina del master y del pool. El siguiente dibujo muestra el funcionamiento del buffer.



Análisis de los resultados

Elegimos utilizar Cliente-Servidor en el escenario 2 a pesar de sufrir secuencialización en la cuarta tarea de cada tanda, ya que cumple el QoS.

En esta medición, con 30 workers observamos que al final no se cumple el QoS.

En esta medición realizada con 38 workers no se nos cumple el QoS por muy poco. Depende de cada medición.

En esta medición con 42 workers si que cumple el QoS.

En esta medición con 46 workers también se cumple el QoF.

Remarcamos que el QoS podría cumplirse con 40 workers (caso mejor) y podría no cumplirse con 48 – 1 workers (caso peor).

Arquitectura para los 3 escenarios a la vez.

La carga de trabajo sería igual al escenario 3 con 5 tareas adicionales cada 2 segundos. Por lo tanto, necesitaríamos entre 45 y 53 workers en la arquitectura master-worker utilizada en el escenario 3.