Introducción a las arquitecturas de Sistemas Distribuidos

Asignatura: Sistemas Distribuidos
Práctica: Práctica 1
Fecha: 17-10-2019
Autores: José Manuel Vidarte Llera, 739729
Diego Marco Beisty, 755232

Índice

1.INTRODUCCIÓN

En esta práctica se ha razonado sobre tres casos reales en los que un cliente genera una carga de trabajo en un servidor.

Para ello, se ha tenido en cuenta el cumplimiento del Quality of Service basado en el tiempo de respuesta. Esto implica que el tiempo total de respuesta de una tarea tiene que ser menor que 1, 5 veces el tiempo de ejecución de la misma de forma aislada.
Como consecuencia, dependiendo de la carga que genera el cliente, se han tenido que diseñar e implementar distintas arquitecturas de comunicación con una escalabilidad adecuada para cumplir el requisito del tiempo de respuesta.

2. Características técnicas

A continuación se presentan las características de los componentes hardware de los ordenadores donde se han implementado las distintas arquitecturas:

Por cada ordenador:

- Núcleos CPU: 4 cores
- Capacidad de la memoria RAM : 12 GB
- Espacio en disco: 931.5GB

Estas características hardware y en especial los núcleos de procesador, han sido fundamentales a la hora de tomar las decisiones de diseño.

3. Análisis gráfico carga de trabajo

Se han realizado diez mediciones del coste en tiempo para cada una de las operaciones que intervienen en la carga de trabajo demandada por el proceso cliente. A continuación se muestra el tiempo de ejecución (en milisegundos) de las operaciones fibonacci y fibonacci_tr de manera aislada.

En el caso de Fib.fibonacci(x) los resultados son los siguientes:

EJECUCIÓN	1	2	3	4	5	6	7	8	9	10	MEDIA	DESVIACIÓN
Tiempo (ms)	1408	1435	1480	1496	1506	1483	1451	1482	1428	1403	1457	37

En el caso de Fib.fibonacci_tr(x) los resultados son los siguientes:

EJECUCIÓN	1	2	3	4	5	6	7	8	9	10	MEDIA	DESVIACIÓN
Tiempo (ms)	14	8	11	6	9	12	10	13	12	7	10	3

Se podría concluir que en los dos primeros escenarios la carga de trabajo es lineal en tiempo. En el primero el servidor recibe una tarea a realizar, la cual de media le cuesta 1457ms, cada 2000ms. En el segundo escenario el servidor recibe cuatro tareas a realizar y

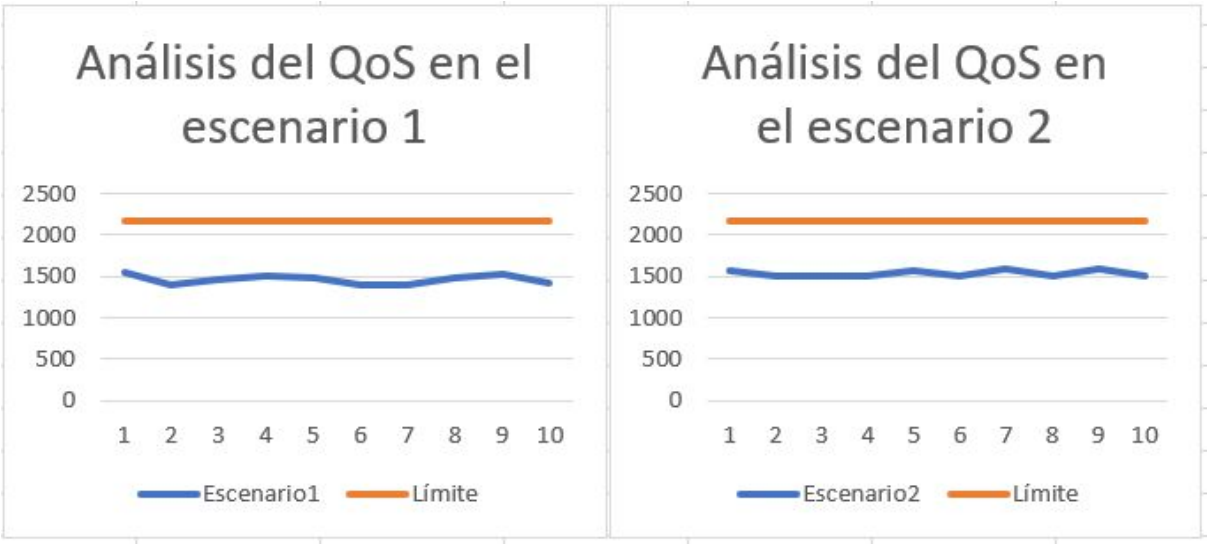
el tiempo medio de ejecución es equivalente al del caso anterior, 1457ms, ya que el servidor dispone de cuatro núcleos a los que asignar las cuatro tareas que recibe para ejecutarlas en paralelo. Éste también recibe el lote de tareas cada 2000ms, luego la carga de trabajo también es lineal en el tiempo.

En el tercer escenario la carga de trabajo es muy variante en el tiempo, ya que el cliente manda ocho tareas cada cierto tiempo, el cual en la mayoría de los casos, está definido por un cálculo que incluye una función aleatoria. Además, el cliente no siempre manda a ejecutar la función fibonacci, sino que en algunos casos también manda la ejecución de la función fibonacci_tr, cuyo coste en tiempo es muy inferior al coste la anterior comentada. Esto provoca que el porcentaje de uso de las máquinas del servidor y los tiempos totales de ejecución de las tareas sean muy variables a cada instante.

4. Análisis tiempo ejecución

Para cada escenario se han realizado mediciones respecto al tiempo total que tarda una tarea en ejecutarse, este se compone del tiempo de transición, el tiempo de ejecución y el tiempo de espera. En los dos primeros escenarios se obtiene un tiempo de espera nulo debido a que se da la situación ideal de que cada núcleo de la única máquina usada en este caso, ejecuta únicamente una tarea, es decir, los núcleos no realizan tareas en paralelo, reciben una petición una vez han acabado la anterior. Esto es apreciable observando que, en los dos primeros escenarios, las tareas tienen un tiempo medio de ejecución de 1457ms y el cliente realiza peticiones cada 2000ms, luego nunca se dará el caso en el que el servidor reciba una petición mientras está ejecutando las que ha recibido anteriormente.

En estos dos primeros escenarios bastaría con disponer de una máquina del laboratorio en la que ejecutar el servidor para garantizar el QoS propuesto (el tiempo total de ejecución de una tarea es siempre menor que 1,5 veces el tiempo de ejecución de una tarea de forma aislada en el peor caso, es decir, la que más tiempo requiere). A continuación se muestran los tiempos de ejecución de diez peticiones al escenario uno y al escenario dos respecto al límite del QoS.



Para el escenario tres no se ha podido analizar los tiempos de ejecución de la carga de trabajo debido a las dificultades encontradas en su implementación. Para calcular el número de máquinas que se necesitan para que el sistema cumpla con el QoS planteado bastaría con seleccionar el caso en el que el tiempo de ejecución de la tarea sea mayor. Teniendo el peor caso, se puede ajustar el número de máquinas necesarias para que el tiempo total de ejecución sea menor a 1,5 veces el tiempo de ejecución aislado, ya que añadiendo más máquinas el tiempo de espera se reduciría considerablemente, siendo cero en el caso ideal.

5. Patrones arquitecturales

Como se ha introducido al inicio de esta memoria, se han abordado distintos patrones arquitecturales dependiendo de la carga de trabajo del cliente.

Escenario uno

Hemos observado que la carga de trabajo del cliente en este escenario es constante. Cada dos segundos manda una tarea al servidor.
En este contexto hemos escogido una arquitectura cliente-servidor, en el que el servidor procesa las peticiones secuencialmente.
Tras implementar este patrón arquitectural y obtener mediciones, hemos observado que se cúmple la condición de tiempo de respuesta.

Escenario dos

En este escenario la carga de trabajo sigue siendo constante, pero se han incrementado el número de peticiones del cliente. Concretamente, manda cuatro tareas cada dos segundos. Ante esta variación, y teniendo en cuenta que cada ordenador tiene cuatro nucleos, hemos adoptado la misma arquitectura que en el escenario uno, pero esta vez con un servidor concurrente. De esta forma logramos aprovechar el procesamiento en paralelo de hasta cuatro procesos en la máquina servidor.

Hemos probado en este escenario el servidor del escenario uno, para confirmar que no es capaz de gestionar la carga y se necesita un servidor más escalable.
Hemos probado la arquitectura cliente-servidor, con servidor concurrente y se cumplen los requisitos del tiempo de respuesta.

Escenario tres
En este escenario, la principal diferencia respecto a los anteriores es que la carga del cliente pasa de ser constante a ser variable en el tiempo. Esto implica que en un instante de tiempo el cliente puede incrementar notablemente el número de peticiones. Las métricas obtenidas manteniendo la arquitectura anterior son nefastas, en ningún caso experimental hemos logrado satisfacer el tiempo de respuesta. Por ello hemos decidido usar un patrón arquitectural mucho más escalable. Este es el patrón master-worker.

6. Tres escenarios simultáneos

Para poder gestionar los tres escenarios explicados en el apartado anterior de una forma simultánea y eficiente, se tendría que pensar en una arquitectura que pudiese gestionar dinámicamente el número de máquinas que se emplean para atender la carga del cliente.

Esto es un patrón master-worker con pool de recursos dinámico, al cual se le pueden añadir o quitar máquinas según las necesidades de cada momento.