Universidad de los Andes
Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional
2020-10

Caso de Estudio 3 – Análisis de Desempeño
Logística y Seguridad Aeroportuaria

Objetivos

- Evaluar las características de desempeño de la infraestructura computacional que soporta el despliegue de una aplicación.
- Simular condiciones de carga y configuraciones de software con el fin de analizar el comportamiento de una aplicación.
- Evaluar el efecto que tienen los mecanismos de seguridad sobre el desempeño de una aplicación.

Descripción General

En esta tercera parte del caso se aborda la problemática del desempeño de la infraestructura computacional para el despliegue de la aplicación trabajada en el caso dos: el sistema Time & Attendance. En el caso dos se trataron los requerimientos de canales seguros. En esta parte del caso se explorarán las relaciones entre el desempeño de la aplicación y las características del software, ante diferentes niveles de carga. Además, se estudiarán los efectos que tiene el uso de mecanismos para ofrecer canales seguros en el desempeño de las máquinas.

Problemática

En el despliegue de una solución computacional es de vital importancia tener en cuenta la infraestructura subyacente con el fin de ofrecer un desempeño adecuado durante el ciclo de vida de la solución.

Actividades

1. **Preparación.** Vamos a simular escenarios para analizar el comportamiento del servidor y de los recursos de la máquina ante diferentes niveles de carga. Para ello deberá instalar el servidor y el cliente de la aplicación en máquinas diferentes; podría usarse la misma máquina, pero esto introduce distorsiones en las mediciones.

   El código del servidor le será suministrado y el cliente será el que usted mismo desarrolló para el Caso 2.

   1.1. Usted debe modificar el código del servidor para manejar un pool de threads en vez de manejar servidores delegados creados al llegar una conexión. A diferencia del primer proyecto, para implementar un servidor con pool de threads no usaremos las primitivas de java, usaremos las librerías de Java, java.util.concurrent.Executors, clases ExecutorService y Executors. Revise el código del servidor y haga los cambios necesarios para implementar un pool de threads de tamaño definido y asigne threads delegados para atender por conexión.

   1.2. Ajuste el código del servidor para garantizar que los mensajes asociados con un delegado quedan escritos en bloque (líneas seguidas) en el log. Tenga en cuenta que necesitará un mecanismo de control de concurrencia.

   1.3. Diseñe e implemente monitores de desempeño para medir los siguientes indicadores en el servidor: (1) tiempo de respuesta de una transacción (desde la lectura de la lectura y procesamiento del certificado del cliente, hasta leer del socket el OK o ERROR de terminación), (2) uso de CPU (explore el método getSystemCpuLoad, al final del enunciado, para medir uso de CPU) y (3) número de transacciones perdidas.

   1.4. En Sicua+ encontrará los tutoriales de Gload, la herramienta que usaremos para generar carga sintética corriendo un número determinado de clientes en una ventana específica de tiempo. Instale y explore la herramienta.

2. **Identificación de la plataforma**. Defina la máquina en la que correrá el servidor e identifique las siguientes características. Tenga en cuenta que el servidor deberá correr en la misma máquina para todos los experimentos (si cambia de máquina, los resultados no serán comparables).

   - Arquitectura (32 o 64 bits)
   - Número de núcleos (*cores*)
   - Velocidad del procesador
   - Tamaño de la memoria RAM

- Espacio de memoria asignado a la JVM

3. **Comportamiento de la aplicación con diferentes estructuras de administración de la concurrencia**. Este escenario tiene como objetivo evaluar cambios en el comportamiento del servidor ante diferencias en la carga:
   - Genere escenarios diferentes cambiando el valor de las variables número de threads en el pool y carga. Para la primera variable (número de threads) use los valores 1 y 2. Para la segunda variable (carga) use 400 transacciones iniciadas con retardos de 20 ms, 200 transacciones iniciadas con retardos de 40 ms y 80 transacciones iniciadas con retardos de 100 ms. (*)
   - Para cada uno de los 6 escenarios posibles (2 tamaños de pool * 3 tipos de carga) mida los siguientes indicadores: tiempo de la transacción, número de transacciones perdidas y porcentaje de uso de la CPU en el servidor. Repita cada experimento 10 veces.
   - Consolide los datos en un archivo Excel (una hoja por escenario) y evalúe la calidad de los datos en cada escenario.
   - Cree las siguientes gráficas:
     a. Fije Carga y genere: # threads vs. tiempo de transacción (por cada carga genere una gráfica = 2 gráficas)
     b. Fije Carga y genere: # threads vs. # de transacciones perdidas (mismo número de gráficas)
     c. Fije Carga y genere # threads vs. porcentaje de uso de la CPU (mismo número de gráficas)
   - Analice cuidadosamente las seis gráficas y determine cómo varía el comportamiento del servidor (en términos de los indicadores tiempo de transacción, número de transacciones perdidas y uso de CPU) con respecto a carga y número de threads.

4. **Comportamiento de la aplicación ante diferentes niveles de seguridad**.
   - A partir del servidor con seguridad, construya un servidor y un cliente sin seguridad. (Al final de este enunciado encuentra el protocolo correspondiente).
   - Como en el punto anterior, el log que se genera debe mantener todos los mensajes asociados con un servidor delegado en bloque (líneas seguidas).
   - Mida porcentaje de uso de CPU para un servidor y clientes que ejecuten el protocolo sin seguridad en los siguientes dos escenarios: pool de 1 y pool de 2 threads, para una carga de 400 transacciones iniciadas con retardo de 20 ms.
   - Recuerde repetir cada experimento 10 veces. Consolide los datos en un archivo Excel (una hoja por escenario) y evalúe la calidad de los datos en cada escenario.
   - Genere una gráfica (una sola) con # threads vs. porcentaje de uso de la CPU.
   - Responda: ¿Cuál es el resultado esperado sobre el comportamiento de una aplicación que implemente funciones de seguridad vs. una aplicación que no implementa funciones de seguridad?
   - Compare la gráfica generada en este punto (# threads vs. porcentaje de uso de la CPU), con la gráfica 3c y revise si ellas confirman los resultados esperados (punto anterior). Justifique brevemente su respuesta.

Entrega. Archivo zip o tar con:

- Código fuente de los dos servidores y los dos clientes (con las modificaciones para medida de indicadores).
- Dos archivos Excel con los datos recopilados (uno con datos con seguridad, y uno con datos sin seguridad).
- Informe con: (a) Descripción detallada de la implementación de los monitores. Indique el código (cliente o servidor) y líneas dónde hizo cambios, (b) identificación de la plataforma, (c) respuestas al punto 3 (todas las gráficas y la conclusión), y (d) respuestas al punto 4 (la gráfica y la conclusión).
- Los datos y el informe **deben** estar en el subdirectorio docs del servidor. **Al comienzo del informe, escriba los nombres y carnés de los integrantes del grupo**. Si un integrante no aparece en el documento entregado, el grupo podrá informarlo posteriormente. Sin embargo, habrá una penalización: la calificación asignada será distribuida (dividida de forma equitativa) entre los integrantes del grupo.
- El trabajo se realiza en grupos de **2** personas. No debe haber consultas entre grupos. El grupo responde solidariamente por el contenido de todo el trabajo, y lo elabora conjuntamente (no es trabajo en grupo repartirse puntos o trabajos diferentes). Se puede solicitar una sustentación a cualquier miembro del grupo sobre cualquier parte del trabajo. Dicha sustentación será parte de la calificación de todos los miembros.
- El proyecto debe ser entregado por Sicua+ por uno solo de los integrantes del grupo.
- **La fecha límite de entrega es el 12 de mayo de 2020 a las 11:50 p.m.**

Referencias

- *Java Threads (Third Edition)*. Oaks, S., & Wong, H. O'Reilly, 2004.
- *Presentaciones del curso (Análisis de desempeño).*

Medida de CPU

https://stackoverflow.com/questions/18489273/how-to-get-percentage-of-cpu-usage-of-os-from-java/21962037)

```
public double getSystemCpuLoad() throws Exception {
      MBeanServer mbs    = ManagementFactory.getPlatformMBeanServer();
      ObjectName name    = ObjectName.getInstance("java.lang:type=OperatingSystem");
      AttributeList list = mbs.getAttributes(name, new String[]{ "SystemCpuLoad" });

      if (list.isEmpty())     return Double.NaN;

      Attribute att = (Attribute)list.get(0);
      Double value  = (Double)att.getValue();

      // usually takes a couple of seconds before we get real values
      if (value == -1.0)      return Double.NaN;
      // returns a percentage value with 1 decimal point precision
      return ((int)(value * 1000) / 10.0);
  }
```

Protocolo sin seguridad: