Juan Pablo Rocha – 201611835
Carlos Eduardo Robles - 201617129

Contenido

Juan Pablo Rocha – 201611835
Carlos Eduardo Robles - 201617129

Caso 3: Análisis y Entendimiento del Problema

1. Descripción detallada de la implementación de los monitores:
   Para hacer las determinadas mediciones de los indicadores, se utilizaron distintas variables, las cuales están descritas a continuación:

```
public static long timeVertificacion;
public static long timeRespuesta;
public static long startTime;
public static long endTime;
public static long nTransaccionesAtendidas;
public static long nTransaccionesPerdidas;
public static long cpu;
```

   Los atributos timeVertificacion, timeRespuesta, nTransaccionesPerdidas y cpu contienen la información de los indicadores. Los atributos startTime y endTime son utilizados para el cálculo de tiempo de ejecución de algunos indicadores.

   Para calcular el uso de la CPU se utilizó el siguiente método:

```
public static double getProcessCpuLoad() throws Exception {

    MBeanServer mbs    = ManagementFactory.getPlatformMBeanServer();
    ObjectName name    = ObjectName.getInstance("java.lang:type=OperatingSystem");
    AttributeList list = mbs.getAttributes(name, new String[]{ "ProcessCpuLoad" });

    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);
}
```

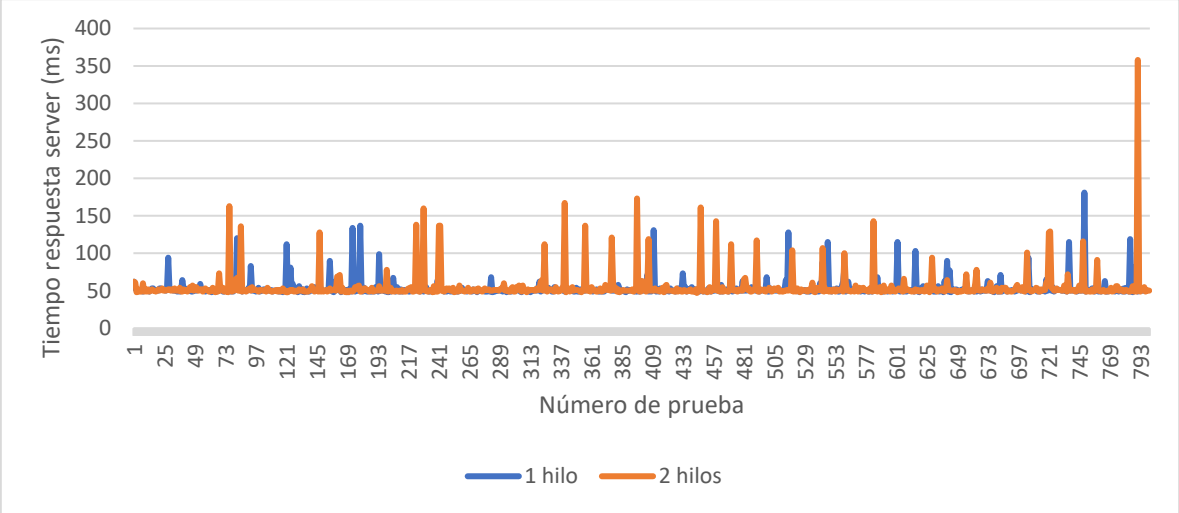
   Para la ejecución del generador, se creó un archivo properties llamado ‘config.prop’ ubicado en la carpeta Data en el proyecto del cliente. Ahí se podrá determinar el numero de threads, tasks y el tiempo entre tareas, así como el puerto y la ip del host.

2. Identificación de la plataforma:
   Para la ejecución del servidor se utilizó una máquina con las siguientes especificaciones:

Atributo	Medida
Arquitectura	64 bits
Número de núcleos	4
Velocidad del procesador	2.2 GHz

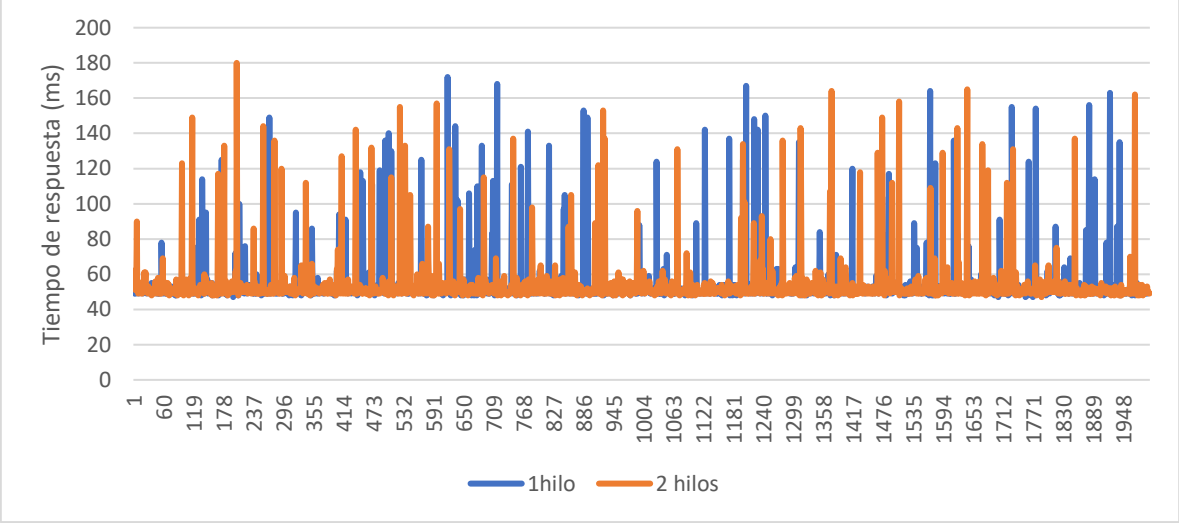
Tamaño de memoria RAM	8 GB
Espacio de memoria asignado a la JVM	4 GB

3. Comportamiento de la aplicación con diferentes estructuras de administración de la concurrencia:
   A. Número de threads vs Tiempo de verificación.
      a) Carga 80:



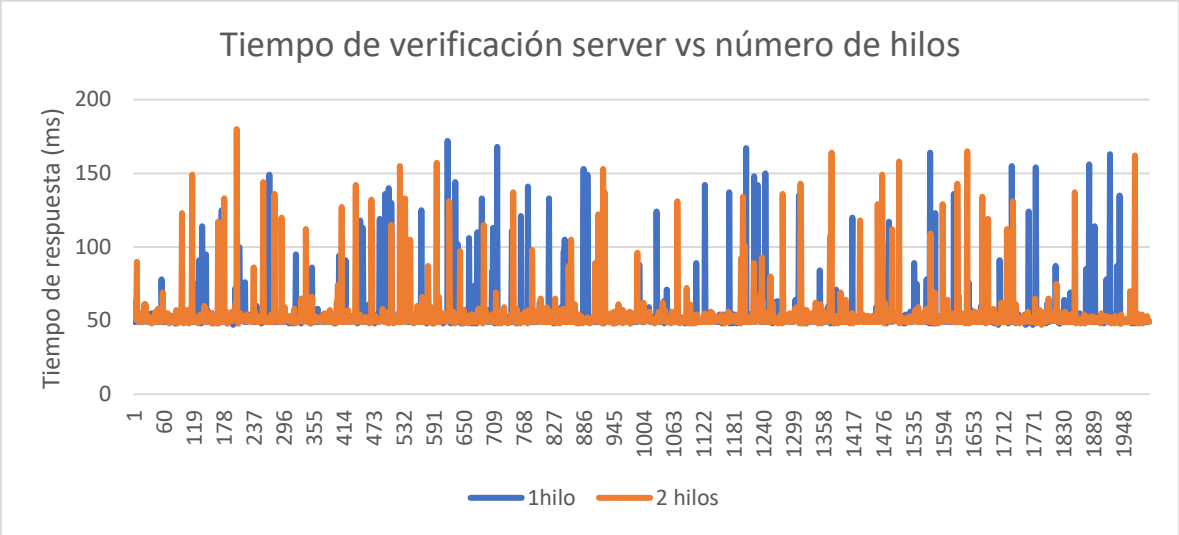
Los tiempos actuaron como era de esperarse, los picos ocurren con mayor frecuencia en la implementación de 2 hilos.

      b) Carga 200:



Según va aumentando la carga, los tiempos de verificación se mantienen en el mismo rango, pero la probabilidad de que haya picos aumenta proporcionalmente. Asimismo, los tiempos de respuesta para la implementación de 2 hilos crecieron a comparación de una carga baja.

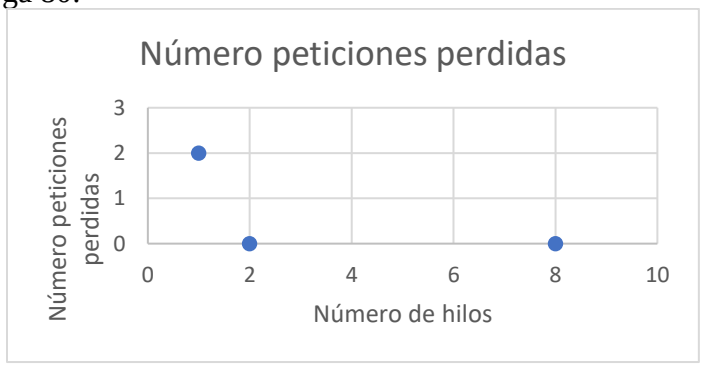
c) Carga 400:



A medida que se aumenta la carga de 80 a 200 solicitudes, los tiempos de verificación fueron aumentando en promedio, haciendo que la probabilidad de que existan picos sea mayor.

B. Número de threads vs Número de peticiones perdidas.

a) Carga 80:

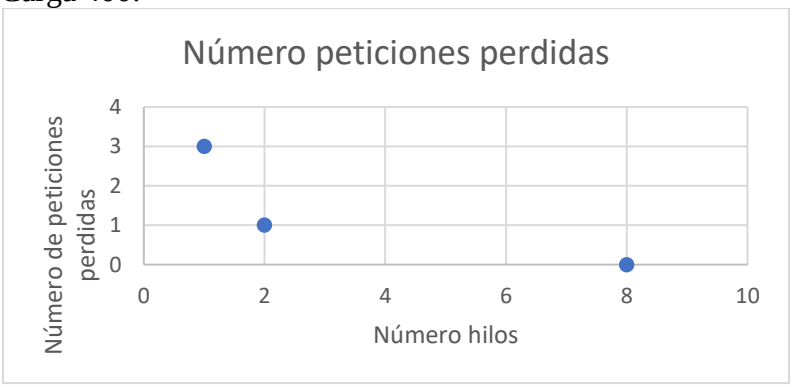


Si se tiene tan solo un thread disponible hay más posibilidades de que hayan transacciones perdidas, en ese caso, sólo se pierden transacciones para ese thread, dejando los otros ejecutando sin problemas.

b) Carga 200:

Aquí se presenta el mismo caso que en el anterior literal.

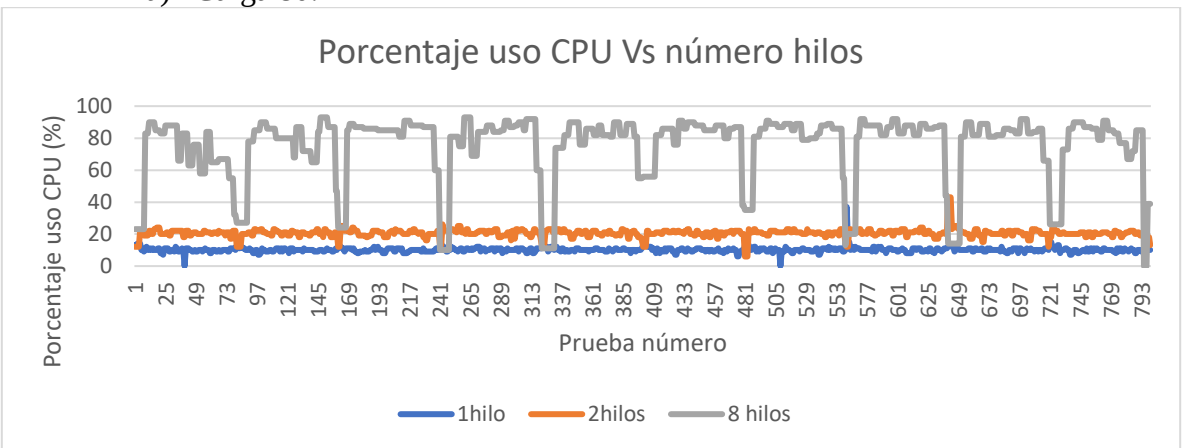
c) Carga 400:



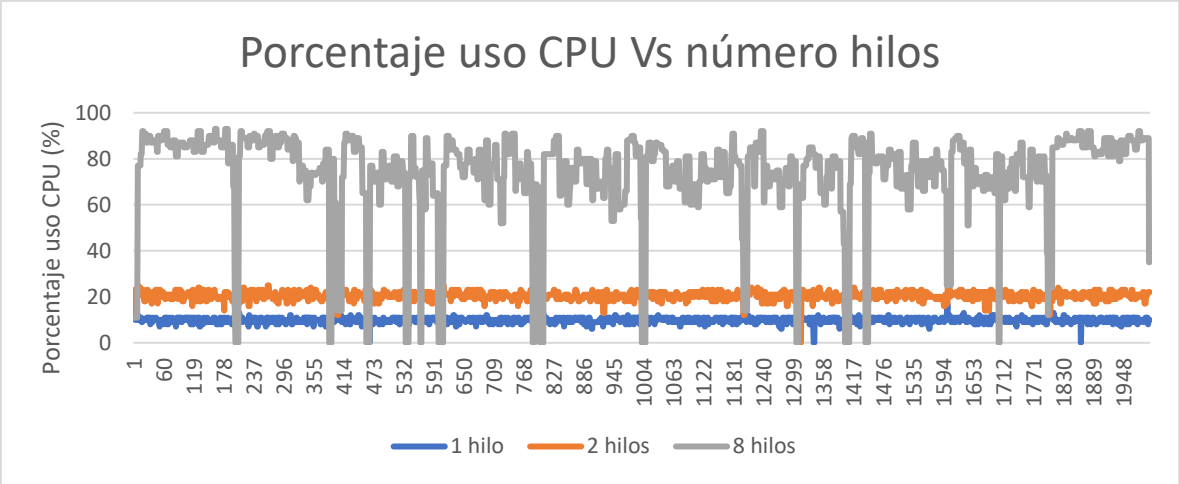
En este caso, como se tienen muchas solicitudes el numero de threads es insuficiente para completar todas estas, haciendo que se pierdan así sea con 2 threads. Sin embargo este índice baja comparado con solo un thread.

C. Número de threads vs Porcentaje de uso de CPU

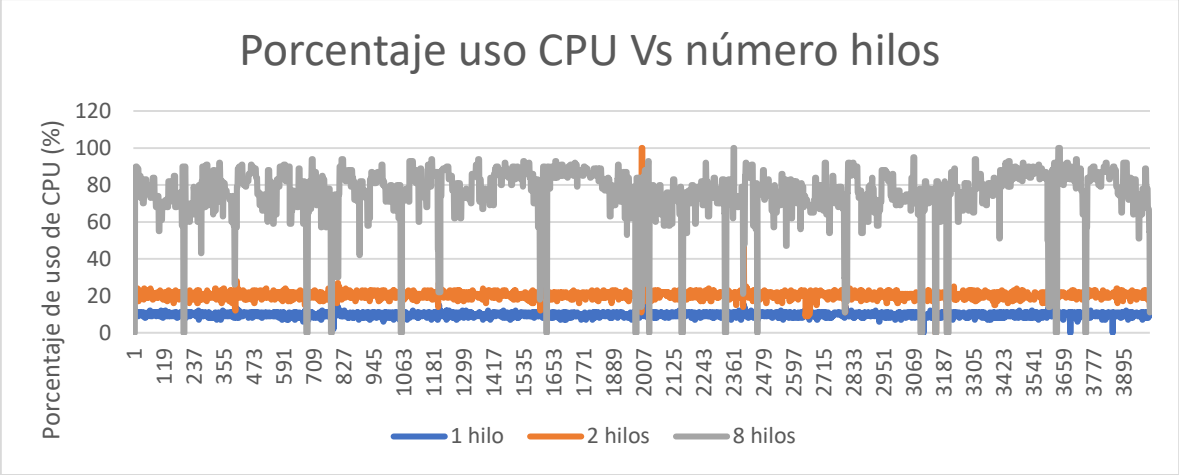
a) Carga 80:



b) Carga 200:



c) Carga 400:



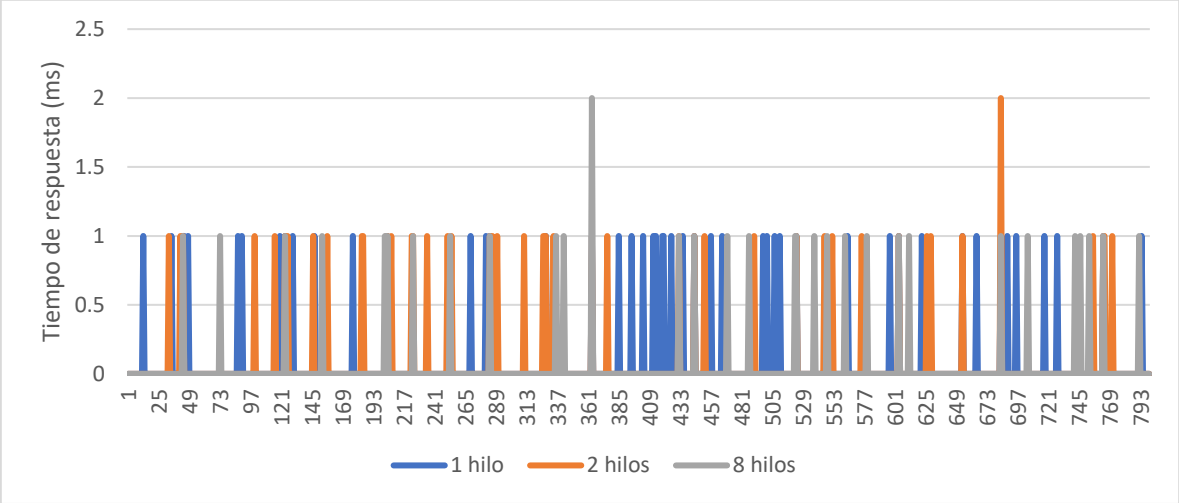
El porcentaje de uso de la CPU aumenta proporcionalmente a la cantidad de threads, esto debido a que se necesitan más recursos de la infraestructura para manejar más hilos.

Juan Pablo Rocha – 201611835
Carlos Eduardo Robles - 201617129

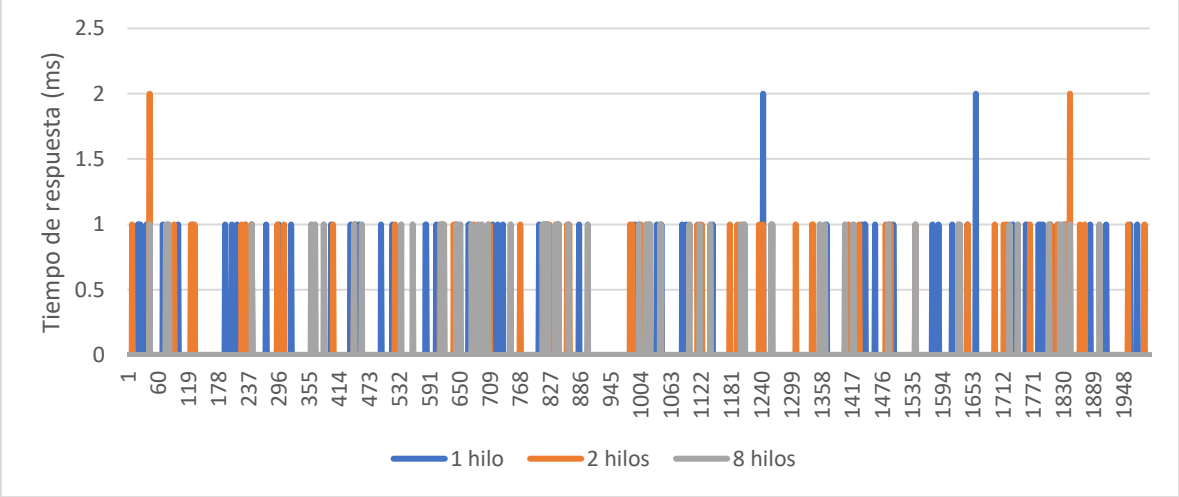
4. Comportamiento de la aplicación ante diferentes niveles de seguridad.

A. Número de threads vs Tiempo de verificación.

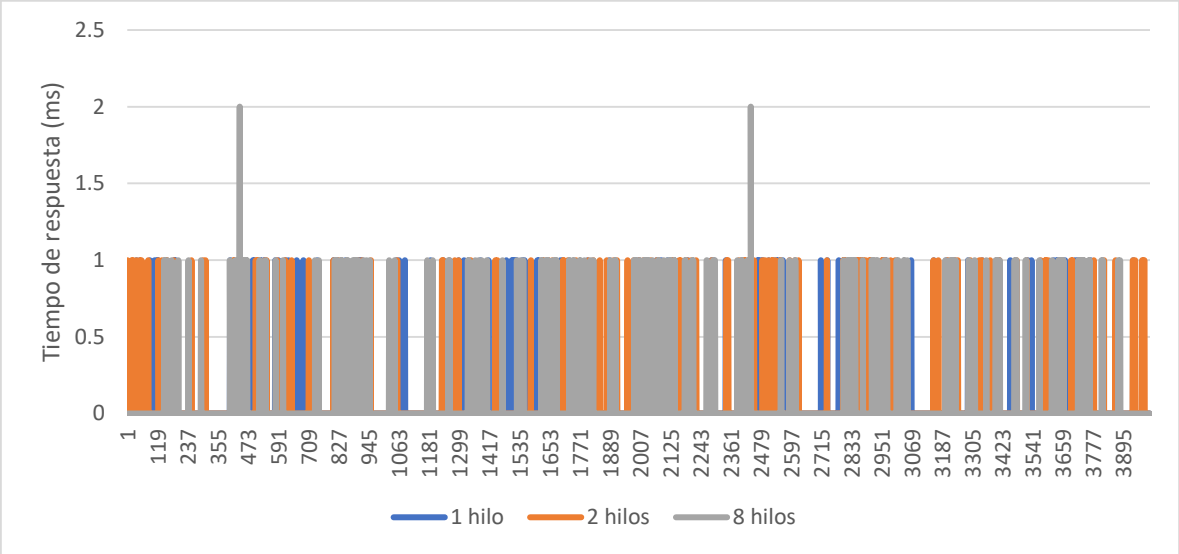
a) Carga 80:



b) Carga 200:

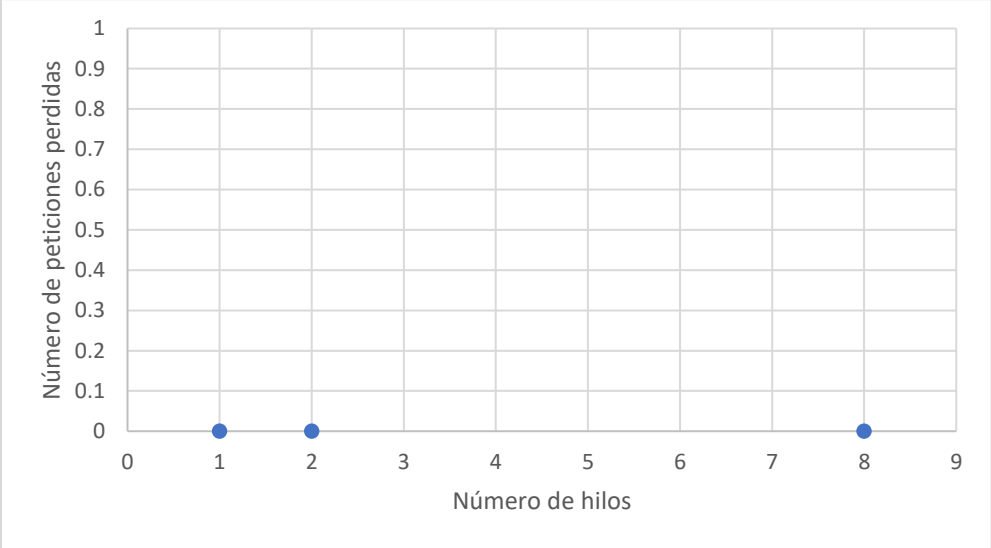


c) Carga 400:

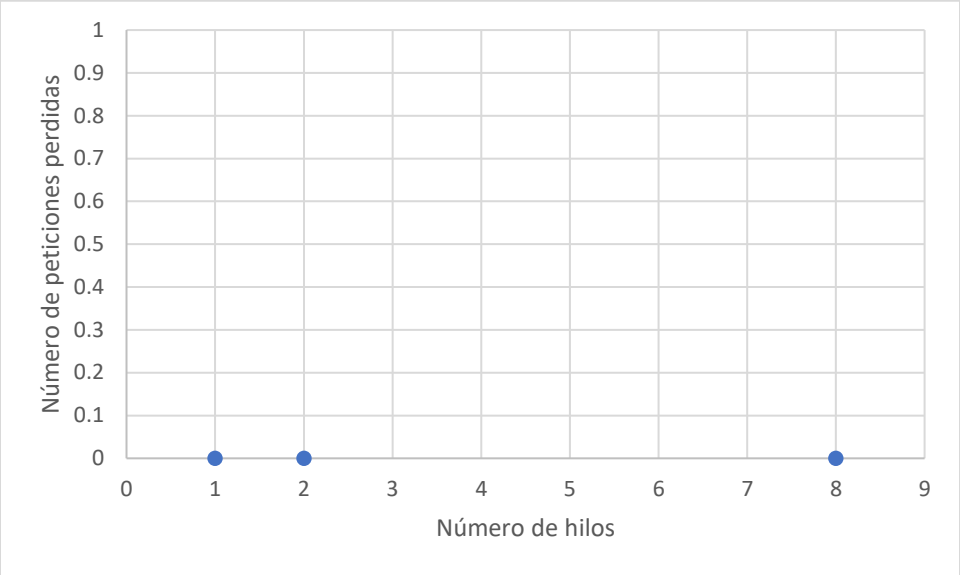


B. Número de threads vs Número de peticiones perdidas.

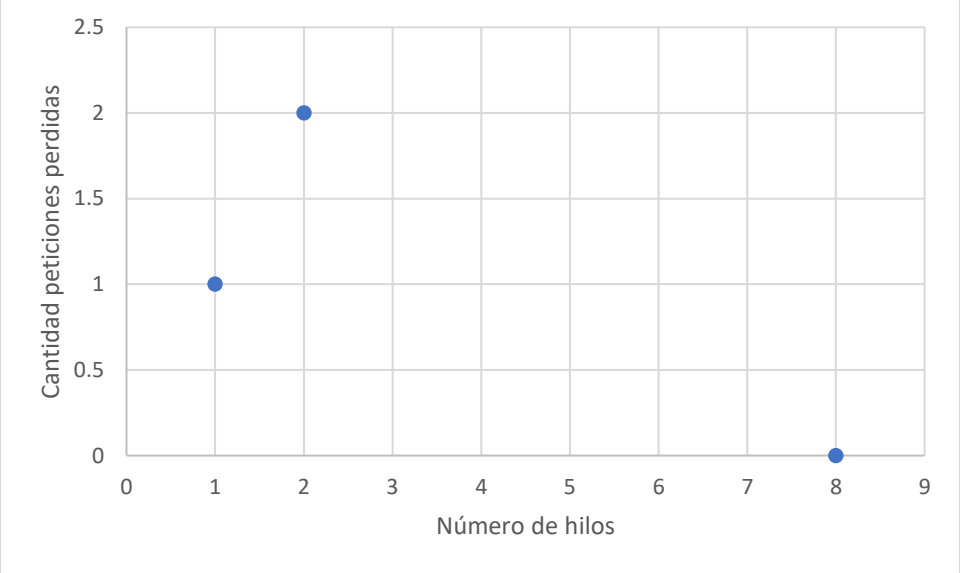
a) Carga 80:



b) Carga 200:



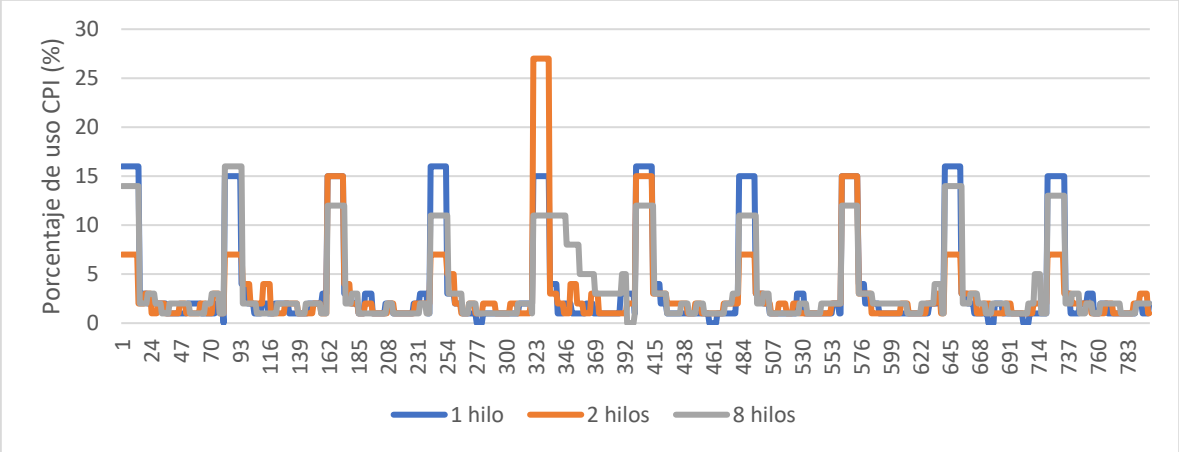
c) Carga 400:



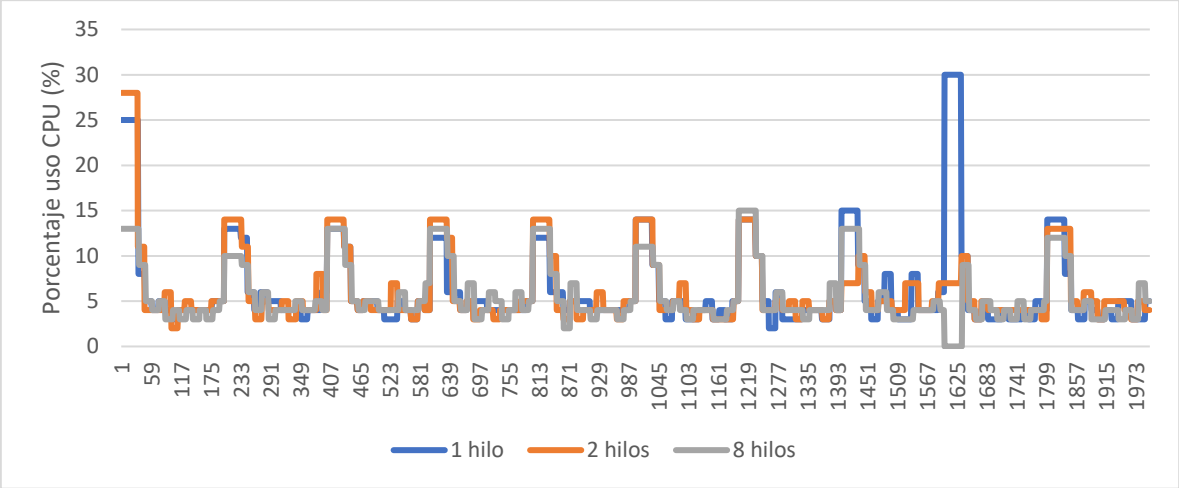
Juan Pablo Rocha – 201611835
Carlos Eduardo Robles - 201617129

C. Número de threads vs Porcentaje de uso de CPU

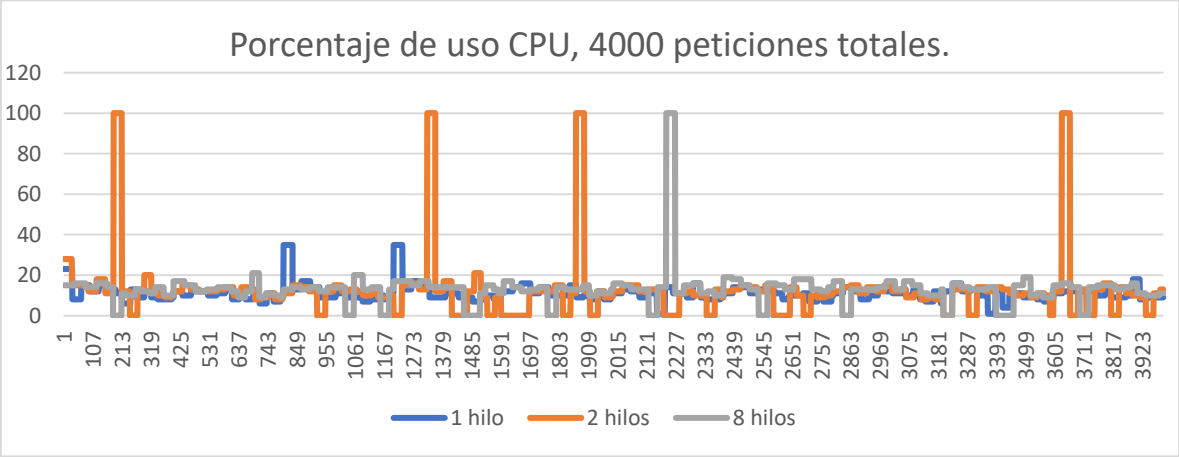
a) Carga 80:



b) Carga 200:



c) Carga 400:



Juan Pablo Rocha – 201611835
Carlos Eduardo Robles - 201617129

5. Respuesta a la pregunta:
   ¿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?

   En términos de tiempo esperamos que la aplicación que no implementa seguridad sea más rápida que una aplicación con seguridad. Esto debido a que la aplicación con seguridad tiene que hacer más procesos para la misma iteración. Y en términos de uso de CPU esperamos que la aplicación sin seguridad use menos porcentaje de CPU ya que tiene que hacer menos procesos.