**Presentado por:** Diana Carolina Gutierrez

**Mail**: [**dianacaro46@gmail.com**](mailto:dianacaro46@gmail.com)

**Cel**: 3054282429

**Reto Performance**

Defina en sus palabras los siguientes conceptos

Teoría

1. **Que son las pruebas de performance y porque son necesarias**: Son las que permiten evaluar el comportamiento que tendrá una aplicación y los recursos que la soportan, cuando se genera una transaccionalidad esperada o en situaciones anormales, midiendo la satisfacción del usuario en requerimientos no funcionales como son el tiempo de respuesta y disponibilidad. Estas pruebas son necesarias porque nos permiten evaluar la eficiencia de que tiene el software y si nuestros recursos de hardware son usados de manera correcta o su por otro lado tienen una configuración que permita la transaccionalidad esperada, antes de que puedan generar una caída inesperada del sistema por indisponibilidad de recursos o por problemas de software.

**2. Que es la concurrencia:** es un tipo de prueba que permite evaluar el efecto en el sistema cuando se quiere acceder a un recurso de software en el mismo instante de tiempo.

**3. Que es la simultaneidad:** se refiere a las instancias / hilos que acceden al mismo recurso paralelamente.

**4. Que son los requisitos no funcionales:** es la necesidad que tiene un producto de software respecto a sus características no funcionales o de performance. Estos requisitos pueden ser, el numero de transacciones espera sean soportadas en un segmento de tiempo crítico para el negocio, tiempo en que se espera dar una respuesta al usuario final, que tanta información quiere que viaje por la infraestructura, entre otros. Estos requisitos se convierten en criterios de aceptación para los resultados de las pruebas de performance.

5. Cuando hacer pruebas de performance: puede haber varias situaciones que nos indiquen la necesidad de una prueba de performance.

* A funcionalidades con alto flujo transaccional o de concurrencia de usuarios.
* Cuando se tienen funcionalidades con volúmenes altos de información en bytes (no necesariamente debe haber muchas transacciones).
* Cuando se tenga un software que este presentando problemas de rendimiento (no debería llegarse hasta este punto si se detectan los requerimientos No Funcionales oportunamente)

6. Nombre los tipos de pruebas de performance y defina cada uno:

* Carga: Busca evaluar el comportamiento del sistema en condiciones normales de transaccionalidad para el sistema o negocio.
* Estrés: Busca evaluar el comportamiento del sistema en condiciones que superan la carga esperada, verificando que al crecer la cantidad de las transacciones el sistema tiene una respuesta aceptable respecto a los criterios de aceptación.
* Concurrencia: permite evaluar el efecto en el sistema cuando se quiere acceder a un recurso de software en el mismo instante de tiempo.
* Capacidad: en esta prueba se buscan evaluar los limites y/ la capacidad de crecimiento del sistema, especialmente cuando no se tiene claridad de los requerimientos no funcionales.
* Prueba de Volumen: esta prueba puede estar contenida en los demás tipos de prueba, pero su objetivo principal es evaluar la cantidad de información en Bytes (u otra unidad) que el sistema es capaz de soportar en un segmento de tiempo. Este tipo de pruebas generalmente se aplica a funcionalidades que generan reportes o que procesan en una sola transacción gran volumen de datos.

7. Cuáles son las etapas de una prueba de performance

* Análisis y planeación
* Diseño de prueba
* Construcción de script de pruebas
* Ejecución de pruebas diseñadas
* Generación de informe

8. Cuáles y que componentes de la arquitectura se recomienda monitorear: se recomienda monitorear Disco, CPU, Memoria y Red. Esto en nodos de aplicación, Base de datos, balanceadores de carga y Bus de servicios si aplica.

**Practica 1**

1. Diseñe un plan de pruebas donde especifique los tipos de prueba a realizar, tiempo de ejecución y componentes a monitorear.

**PLAN DE PRUBAS**

|  |  |
| --- | --- |
| **Alcance funcional de prueba** | Funcionalidad: Reserva de vuelo |
| **Requerimientos No Funcionales** | Transacciones esperadas en 10 min: 100  Usuarios esperados- máxima concurrencia: 10  Tiempo de respuesta: menor a 1Seg por transacción  Error permitido: 0%  Throughput: No informado |
| **Alcance técnico – Componentes a monitorear** | Servidores de aplicación (2)  Servidor de base de datos (1)  Balanceador de carga (1)  Se evaluará componentes de hardware: CPU, Memoria, Red, Disco. Se coordinada con DBA la activación de logs propios de Base de datos para capturar información adicional sobre el comportamiento de las transacciones. |
| **Tipos y escenarios de pruebas** | Carga: 1 agente de pruebas lanzando una carga escalonada, iniciando con un hilo, sumando un hilo cada 1 min hasta llegar a la concurrencia máxima de 10 hilos buscando completar 100 transacciones totales.  Estrés: 1 agente de pruebas lanzando una carga escalonada, iniciando con un hilo, sumando un hilo cada 3 min hasta llegar a la concurrencia máxima de 5 hilos, generando una carga adicional al sistema desde el enfoque de usuarios. Se busca completas hasta 150 transacciones totales.  Concurrencia: 1 agentes de pruebas lanzando 10 transacciones con 10 hilos concurrentes en 1 Seg. |
| Estrategia de ejecución | Se realizará la etapa de análisis durante el sprint 1 buscando definir los requerimientos no funcionales y la contextualización de las funcionalidades paralelo a la certificación de la funcionalidad. Este sprint tendrá como entregable plan y diseño de prueba.  En un sprint 2 se realizará una prueba de concepto que permitirá identificar los requerimientos técnicos de construcción de script de prueba.  En un sprint 3 se realizará la construcción de script de pruebas de acuerdo con planeación y diseño. Esta actividad tendrá como requisito estabilidad del ambiente de prueba y una calidad mínima según estándar (80%).  En la última etapa se realizará ejecución de pruebas. De manera temprana se entregarán resultados parciales.  Finalmente se realizará informe de ejecución. Se coordinará socialización de resultados con el cliente. |

2. Diseñe el script en jmeter 4.0 de la página https://blazedemo.com/ donde se realice un flujo de la compra de un tiquete.

R/ Se adjunta script de pruebas VuelosBlazaDemo.jmx

3. Genere un set de datos con la información del pasajero y medio de pago. Utilice esta para información para variabilizar el script.

R/ se adjunta archivo de datos usado para la prueba datos.csv

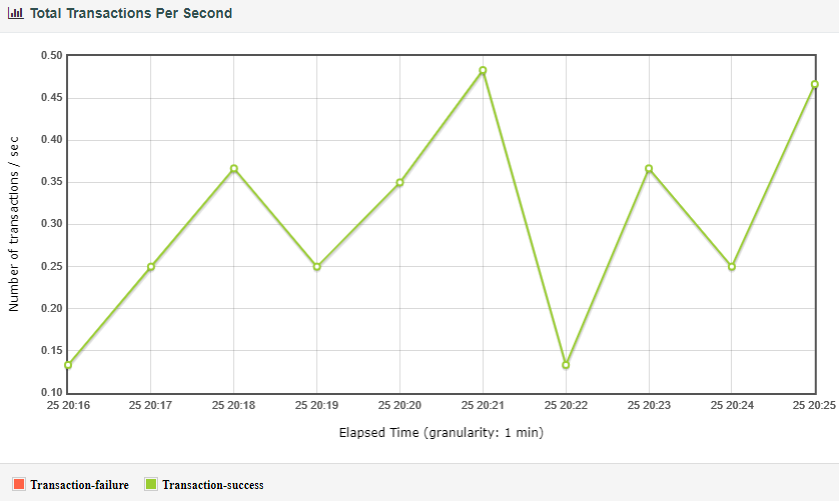
4. Realice un análisis de los resultados de la ejecución:

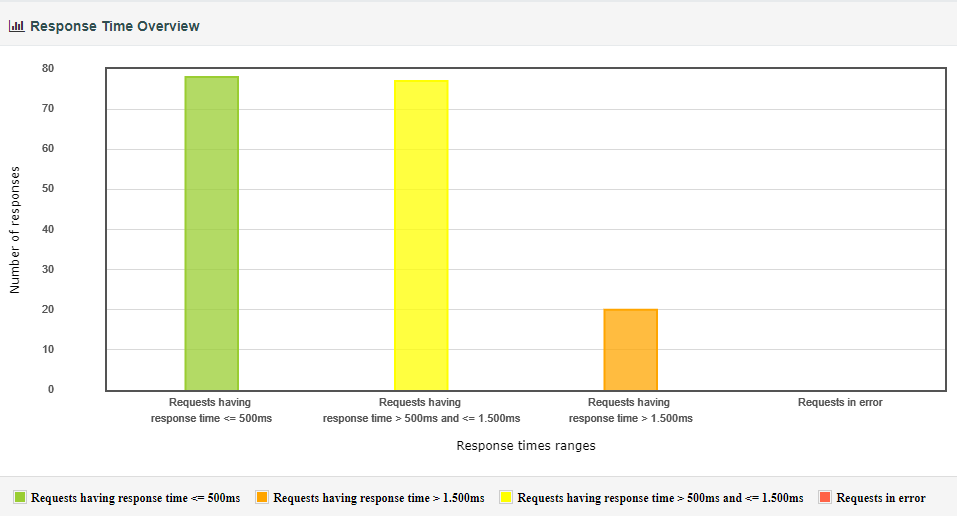


Durante la ejecución no se evidenciaron errores de transaccionales por lo que para la carga de generada el sistema tiene alta disponibilidad.

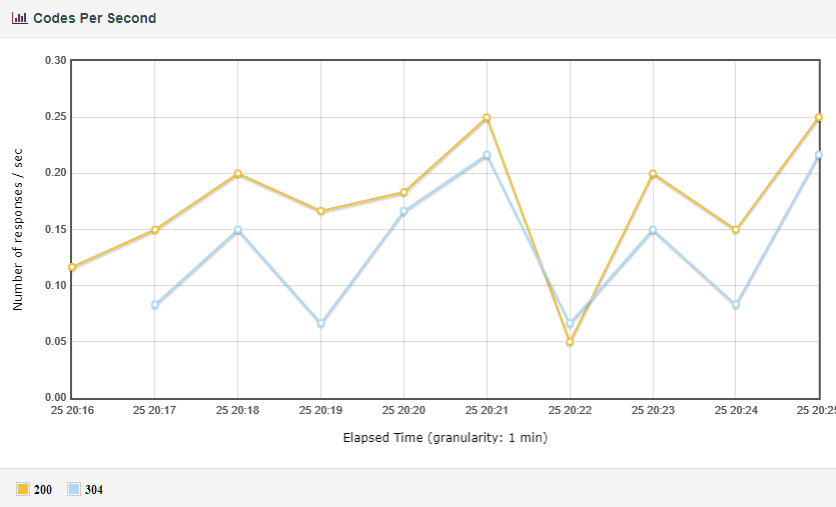
En general se evidencias tiempos con percentil 90 de 1.6 segundos lo que es un tiempo aceptable y que no afecta la percepción del usuario respecto a la respuesta de la aplicación.

La transacción con mejor comportamiento es la “confirmation” que corresponde a la selección del vuelo con alrededor a 2 segundo de respuesta como percentil identificado. Por el contrario, las transacciones de selección de vuelo y confirmación de reserva tienen tiempos oscilantes en los 5 seg, lo que corresponde a una alerta de rendimiento que puede generar una percepción de demora al usuario.





Como se ve en la gráfica de Througput (trn/seg) y la gráfica de tiempos de respuesta, se ve que a pesar de que en general los tiempos son relativamente bajos, estos son inestables y no tienen consistencia en su comportamiento. Se recomienda validar que está ocasionando este comportamiento.



Los códigos de respuesta de las transacciones que no hay errores, pero si hay un uso importante de cache en las transacciones y reúso de transacciones, lo que puede de cierta forma estar mejorando el rendimiento de la aplicación.

5. Presente los resultados de la ejecución en formato html. Utilizando los comandos de jmeter

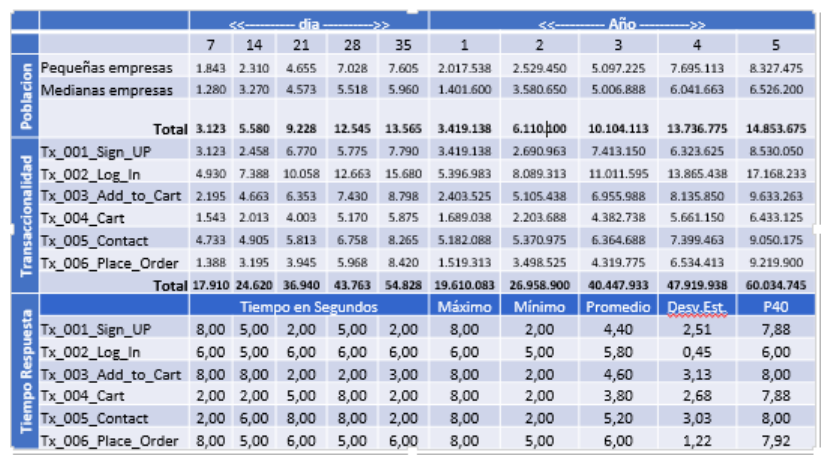
R/ Se adjunta carpeta con reporte

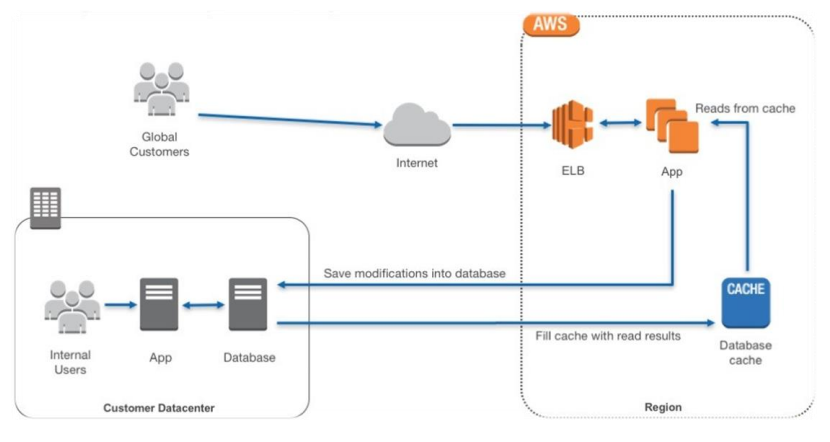
6. Diseñe una matriz de riesgos de la prueba

|  |  |
| --- | --- |
| Riesgo | Acciones preventivas |
| Desviación en los tiempos estimados en el desarrollo del script dado el desconocimiento de validaciones de seguridad y datos de prueba requeridos. | Documentación detallada de las validaciones de seguridad en la funcionalidad de alcance y de los datos necesarios para las pruebas |
| Estabilidad de las funcionalidades a probar. La prueba puede fallar por inconvenientes funcionales. | Pruebas funcionales previas con correctitud y fiabilidad superior al 85% del alcance. |
| Cambios a funcionalidades paralelo a la actividad de construcción de script o ejecución de pruebas, que pueden afectar la ejecución y/o el rendimiento o desempeño de esta. Anulando los resultados obtenidos. | Planear pruebas continuas.  Ambiente de pruebas controlado. |
| Nodos de arquitectura no identificados y que no se contemple su análisis para el diagnóstico del performance. | Entrega y contextualización de diagrama de arquitectura por personal del cliente capacitado. |
| Cambios de la infraestructura después de realizar las pruebas, anularía los resultados obtenidos | Planear pruebas continuas |
| No Acceso VPN para las pruebas, afectando cronograma. | Disponibilidad constante de quien pueda dar solución para para garantizar el acceso. |
| Alcance de funcionalidades y/o requerimientos no funcionales incompletos. | Reuniones tempranas con cliente para identificación de alcance. |

**Practica 2**

En el proyecto ShopMyPymes inicio hoy la construcción del nuevo Portal Pymes se tiene como objetivo dar cobertura a nivel Latinoamérica a todas las pequeñas y medianas empresas. La aplicación cuenta con una solución web responsive y una app para IOS y Android. URL: https://demoblaze.com/index.html Se cuenta con la siguiente volumetría y arquitectura





**1. Plantee los riesgos. (Se sugiere máximo 3 - Los más importantes)**

**R1:** La transaccionalidad presentada esta levantada en segmentos de tiempo mínimo de un día, por lo que no es fácil identificar el pico transaccional y de máxima concurrencia correspondiente al momento más crítico del sistema / negocio, lo que podría ocasionar un riesgo al calcular las transacciones para el tiempo de prueba que durara la ejecución y arrojar resultados errados, alarmas falsas o por el contrario no se llegue a identificar hallazgos de performance.

**R2:** Los tiempos de respuesta presentados pueden no corresponder a tiempos aceptables para los usuarios finales y de tomarse estos como referencia, pueden generar resultados errados y alarmas no presentadas a los dueños de la aplicación.

**R3:** En la arquitectura se evidencia un riesgo en la ubicación del nodo de base de datos teniendo en cuenta que es más susceptible a concurrencia y mayor transaccionalidad, por lo que pueden colapsar fácilmente durante la prueba, más teniendo en cuenta que esta es on premise y no escalable según se identifica en el diagrama.

**2. Plantee los tipos de prueba y escenarios de prueba con sus volumetrías correspondientes.**

|  |  |
| --- | --- |
| **Tipos y escenarios de pruebas** | **Carga Funcionalidad Sign\_Up:** 1 agentes de pruebas lanzando 1.266 transacciones Aprox en 3.600 Segundos. La carga se hará en 8 escalones con 5 seg de espera entre cada escalón (tiempo de ocio del grupo de hilo antes del inicio de las transacciones).  Cada agente iniciará la prueba con 1 usuarios y se agregará 1 usuario en cada escalón, hasta llegar a una concurrencia máxima de 8 Usuarios. Cada usuario lanzara aproximadamente 34 transacciones en cada escalón.  Total transacciones enviadas: 1.266  **Carga Funcionalidad Log\_In:** 1 agentes de pruebas lanzando Aprox 2.549 transacciones en 3.600 Segundos. La carga se hará en 15 escalones con 5 seg de espera entre cada escalón (tiempo de ocio del grupo de hilo antes del inicio de las transacciones).  Cada agente iniciará la prueba con 1 usuarios y se agregará 1 usuario en cada escalón, hasta llegar a una concurrencia máxima de 15 Usuarios. Cada usuario lanzara aproximadamente 22 transacciones en cada escalón.  Total transacciones enviadas: 2.549 |
|  | **Estrés Funcionalidad Sign\_Up:** 1 agentes de pruebas lanzando cada uno aprox. 1.934 transacciones en 3.600 Segundos. La carga se hará en 13 escalones con 5 seg de espera entre cada escalón (tiempo de ocio del grupo de hilo antes del inicio de las transacciones).  Cada agente iniciará la prueba con 1 usuarios y se agregará 1 usuario en cada escalón, hasta llegar a una concurrencia máxima de 13 Usuarios. Cada usuario lanzara aproximadamente 22 transacciones en cada escalón.  Total transacciones enviadas: 1934  **Estrés Funcionalidad Log\_In:** 1 agentes de pruebas lanzando cada uno 3.813 transacciones en 3.600 Segundos. La carga se hará en 12 escalones con 5 seg de espera entre cada escalón (tiempo de ocio del grupo de hilo antes del inicio de las transacciones).  Cada agente iniciará la prueba con 1 usuarios y se agregará 2 usuario en cada escalón, hasta llegar a una concurrencia máxima de 22 Usuarios. Cada usuario lanzara aproximadamente 28 transacciones en cada escalón.  Total transacciones enviadas: 3.813 |
|  | **Concurrencia transaccional Pequeña empresa**: 2 agentes de pruebas lanzando cada uno 618 transacciones en 1 escalones de 1 Seg cada uno, para una duración máxima de las pruebas de 1 Seg  Total usuarios activos durante la prueba 1.237 con una transacción cada uno.  **Concurrencia transaccional empresa Mediana**: 2 agentes de pruebas lanzando cada uno 484 transacciones en 1 escalones de 1 Seg cada uno, para una duración máxima de las pruebas de 1 Seg  Total usuarios activos durante la prueba 969 con una transacción cada uno. |

**Nota**: En la prueba de estrés se realizó un aumento del 50% de las transacciones para estresar el sistema en referencia a la carga transaccional esperada en el periodo de prueba.

Se presentan ejemplos de escenario por funcionalidad y de acuerdo con la estrategia se ejecutarán los escenarios por funcionalidad para detectar problemas específicos.

Al final se ejecutará un escenario de sistema con todas las transacciones respetando el diseño por funcionalidad.

**3. Plantee el alcance transaccional y 4. Realice un cálculo de concurrencia donde soporte los hilos a ejecutar**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **Dia 35** | **Trn. promedio por hora (8 horas día)** | **Aumento por hora pico 30%** | **Tiempo de respuesta de referencia P40** |
| **Población** | Pequeña empresa | 7.606 | 951 | 1.236 | - |
| Mediana Empresa | 5.960 | 745 | 969 | - |
| **Total** | | **13.566** | **1.696** | **2.204** | **-** |
| **Transaccionalidad** | Sign\_Up | 7.790 | 974 | 1.266 | 7,9 |
| Log\_In | 15.680 | 1.960 | 2.548 | 6,0 |
| Add\_to\_Cart | 8.798 | 1.100 | 1.430 | 8,0 |
| Cart | 5.875 | 734 | 955 | 7,9 |
| Contact | 8.265 | 1.033 | 1.343 | 8,0 |
| Place\_Order | 8.420 | 1.053 | 1.368 | 7,9 |
| **Total** | | **54.828** | **6.854** | **8.910** |  |

**5. Realice el script en jmeter el uso de cualquier plugin debe estar especificado en el plan de pruebas.**

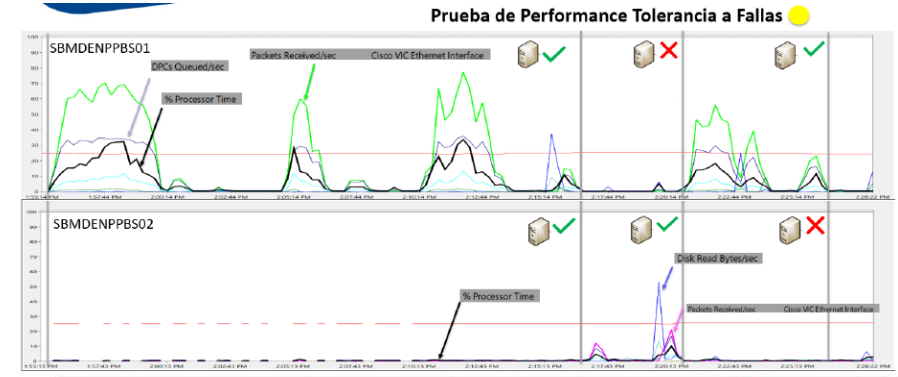
Para la prueba se utiliza el plugin de BlazeMeter para Chrome para la grabación del script de prueba.

Par la configuración de los escenarios se usan los complementos “Ultimate Thread Group”, “Timer Constante” (configuración de los tiempos de ocio).

**Análisis 1.**

Realice un análisis de las siguientes graficas de monitoreo a una aplicación donde se ejecuta una prueba de tolerancia a fallas.

Servidor Online - Servidor Offline

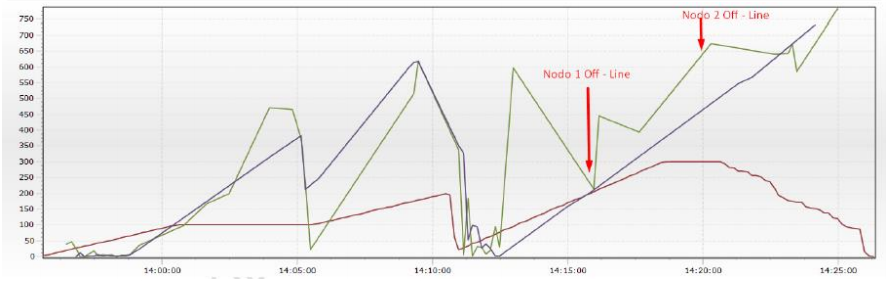


Ilustración

En la gráfica anterior se pueden ver los segmentos de ejecución de prueba para el SBMDENPPBS01 donde se ven un volumen transaccional similar por unidad de tiempo, pero con segmentos o tiempos de prueba diferentes. Durante todos los escenarios se ve un consumo de procesador estable respecto al recurso total sin alertas de disponibilidad de este, sin embargo, se evidencia aumentos en los DCP/Seg a medida que avanza la prueba, encolamiento que en situaciones de transaccionalidad superior a la de la prueba podría ocasionar saturación en la CPU, se recomienda validar que puede estar ocasionando ese comportamiento desde el Software.

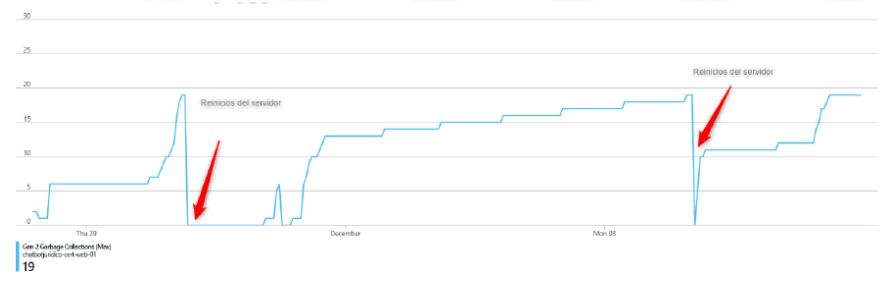
Se evidencia que el uso de CPU es consistente a la carga generada.

Para el servidor SBMDENPPBS02 recibe una carga menor de los procesos ejecutados durante la prueba lo que puede ser consistente con una configuración donde el servidor 02 es de contingencia con un consumo mínimo del recurso de CPU. En el segmento de validación de disponibilidad, se evidencia un aumento en el consumo de disco y CPU, sin embargo, este aumento de uso de recurso no es consistente y no responde a la disponibilidad y desempeño del hardware como si lo ha dado el servidor 01, se recomienda validar la configuración y/o apuntamiento de los recursos de software en situación de contingencia.



Ilustración

Teniendo en cuenta el comportamiento de la Ilustración 1, la gráfica de la ilustración dos ratifica que al tener el nodo 01 fuera de línea, se procesan menos transacciones ya que el nodo 2 no tiene una respuesta optima a la carga. Consistentemente al esta el nodo 2 offline, se procesa mayores transacciones lo que deja ver que el nodo 1 tiene mejor respuesta.



Ilustración

En la ilustración 3, se evidencia reinicios de maquina que liberan el recurso de memoria dejándola disponible para el uso por el software/ sistema. Sin embargo, se puede ver que previo a los reinicios y posterior al último, el consumo de memoria aumenta de con comportamiento similar a escalonado, lo que puede ser consistente a un uso ineficiente de la memoria, con contención del recurso de las aplicaciones que corren en este nodo. Se recomienda validar el manejo de memoria por las funcionalidades en el alcance de prueba, ya que están reteniendo el recurso y solo liberado por reinicios de máquina, qué de no existir estos reinicios puede llegarse a una caída del nodo, y mientras la memora este en los momentos de baja disponibilidad, puede generar mal rendimiento de la aplicación y del SO que lo soporta.