Ejercicio 01 Conceptos básicos

CARRERA: Ingeniería en Computación

NOMBRE: *Efrain Robles Pulido*

CÓDIGO: 221350095

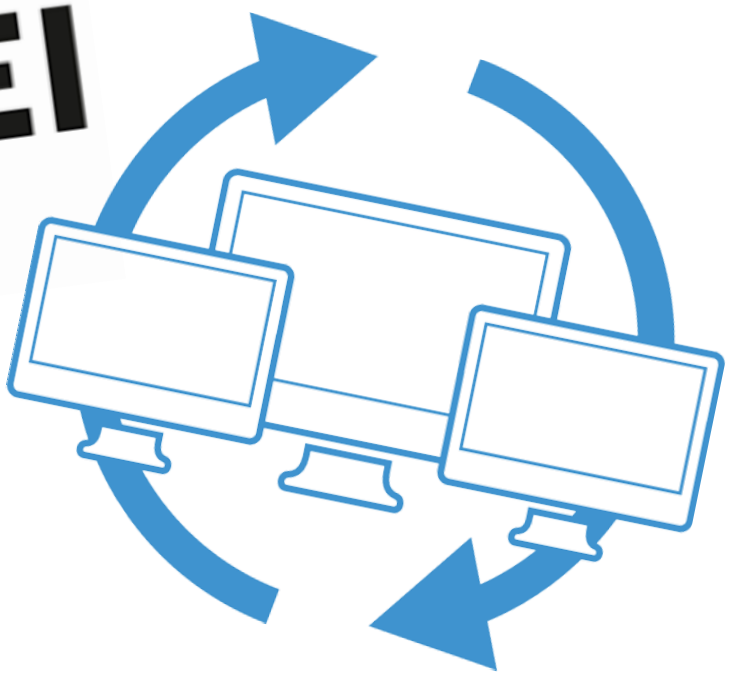
MATERIA: Computación Tolerante a Fallas

MAESTRO: Michel Emanuel Lopez Franco

SECCIÓN: D06 **CALENDARIO:** 2023B

UNIVERSIDAD DE GUADALAJARA

CUCEI



Objetivo:
Conocer los conceptos básicos en sistemas tolerantes a fallas.

Introducción:
En un mundo cada vez más dependiente de la tecnología y los sistemas complejos, la capacidad de mantener la disponibilidad, confiabilidad y seguridad de estos sistemas se vuelve esencial. Los sistemas tolerantes a fallos emergen como una respuesta fundamental para garantizar que, incluso en situaciones adversas o en presencia de fallas imprevistas, los sistemas puedan seguir funcionando de manera efectiva y proporcionar resultados confiables. Para lograr esto, es crucial comprender los conceptos básicos en sistemas tolerantes a fallas, los principios que los rigen y las estrategias que permiten mantener su funcionamiento adecuado.

Desarrollo:
Contesta las siguientes preguntas:

¿Qué son los sistemas tolerantes a fallos?
Son sistemas que pueden continuar funcionando correctamente, a pesar de los problemas que se encuentren en su hardware o del software. Teniendo como objetivo garantizar la disponibilidad, confiabilidad y seguridad del sistema sin importar las fallas a pesar de que la mayoría son inevitables debido a los factores como los componentes, el desgaste, los errores humanos entre otros. Por otro lado, tener un diseño tolerante a fallas puede causar una *reducción en el nivel de productividad o un mayor tiempo de respuesta (rendimiento)*, pero nos garantiza que el mecanismo de afrontamiento del sistema tenga la estabilización.

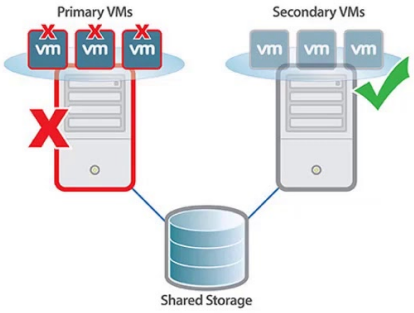
Al principio de la tecnología, los sistemas tolerantes a fallas se diseñaron para dar alarmas al usuario o al operador sobre la posible falla. Se suponía que el operador actuaría sobre la alarma y aclararía las cosas antes de que ocurriera una avería importante. Esto implicó la interferencia humana. Sin embargo, hoy las cosas han cambiado. Los sistemas, ya sean hardware o software, están diseñados para resolver problemas de forma independiente sin mucha interferencia humana, a menos que sea un problema importante que requiera atención inmediata.

La **tolerancia a fallas** se refiere a la capacidad de un sistema (computadora, red, clúster de nube, etc.) para continuar funcionando sin interrupción cuando uno o más de sus componentes fallan.

Existiendo diferentes estrategias para el software como el hardware:

Se utilizan estas técnicas para nivel software:

- **Programación de la versión N.** En esta técnica, n desarrolladores desarrollan n versiones de un programa. Todas estas copias se ejecutan simultáneamente y se selecciona la que tiene la mayor tolerancia a fallos. Esta es una técnica de detección de fallas que se utiliza en la etapa de desarrollo del software.
- **Bloques de recuperación.** Esta técnica es algo similar a la anterior, excepto que las copias redundantes no se ejecutan simultáneamente. Se ejecutan una por una y se generan con un conjunto diferente de algoritmos. Esta técnica se utiliza cuando los plazos de las tareas superan el tiempo de cálculo.
- **Recuperación de comprobación y retroceso.** Mediante esta técnica, el sistema se prueba cada vez que se necesita realizar un cálculo.
- **Computación ajena a las fallas.** Esta técnica permite que los programas informáticos continúen ejecutándose a pesar de los errores. Maneja lecturas de memoria inválidas devolviendo el valor fabricado al programa donde, a su vez, el programa considera este nuevo valor e ignora el valor anterior en su memoria. Esto es algo poco probable en las comprobaciones de memoria anteriores, que abortaban los programas por entradas no válidas.
- **Pastoreo de recuperación.** Esta técnica funciona con el pin de marco binario justo a tiempo. Se adjunta al proceso de aplicación, analiza el error, toma nota de las reparaciones y rastrea los efectos de la reparación y se desconecta del programa de aplicación una vez que todos los efectos de reparación se eliminan del programa. Todo esto ocurre en el back-end mientras el programa funciona en su estado normal y no obstaculiza su ejecución habitual.



Por ejemplo, los datos almacenados en el disco se replican en sistemas redundantes. Enfoques más avanzados del software también copian los datos no comprometidos o en memoria en un sistema de respaldo redundante. En caso de que el sistema principal falle, un sistema secundario de respaldo retoma sus funciones desde el instante preciso en que el sistema principal falló. Esto evita la duplicación o pérdida de transacciones o información.

A continuación, se utilizan estas técnicas para nivel hardware:

- **BIST (autoprueba integrada)**. Esta técnica permite al sistema realizar pruebas a intervalos específicos para evaluar cualquier propagación defectuosa. Siempre que señala una falla, se configura para cambiar el componente defectuoso y cambiar su redundante en su lugar.
- **TMR (redundancia modular triple)**. En esta técnica, se generan tres copias redundantes de un componente defectuoso y se ejecutan simultáneamente. Se vota por su desempeño y se selecciona la mayoría de votos. Puede tolerar una sola falla a la vez.

- **Cortacircuitos**. Este es un diseño de circuito que permite romper el circuito para evitar fallas catastróficas en sistemas distribuidos.

Y con el hardware, los sistemas redundantes operan de manera simultánea. Por ejemplo, los servidores paralelos llevan a cabo tareas idénticas, de manera que si uno de los servidores falla, el otro continúa procesando transacciones o prestando servicios. Este enfoque se sustenta en la baja probabilidad estadística de que ambos sistemas fallen al mismo tiempo. En realidad, solo se requiere un servidor para ofrecer las aplicaciones, pero contar con dos servidores ayuda a garantizar que al menos uno siempre esté operativo.

También se encuentran las fuentes de energía que se hacen tolerantes a fallas utilizando fuentes alternativas. Por ejemplo, muchas organizaciones tienen generadores de energía que pueden asumir el control en caso de que falle la línea principal de electricidad.

Algunos requisitos para que un sistema sea considerado como tolerante a fallas son:

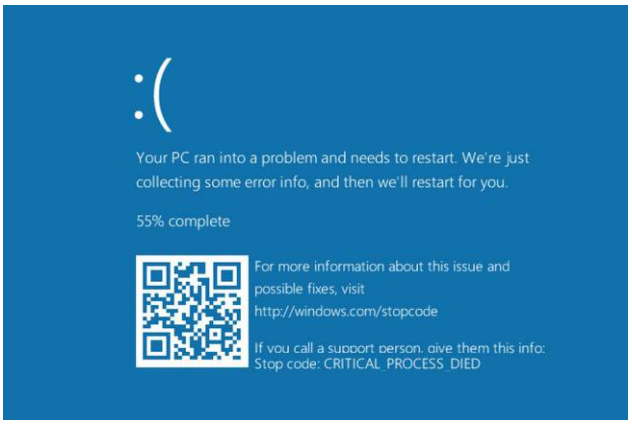
- **Ningún punto de falla:** si el sistema falla, debe continuar funcionando durante la reparación sin interrupciones.
- **Aislamiento de fallas de los componentes defectuosos:** en caso de falla, el sistema debe poder aislar la falla del componente en cuestión. Esto requiere la adición de mecanismos de detección de fallas dedicados que existen solo para el aislamiento de fallas. La recuperación de un estado de falla requiere la clasificación de fallas o componentes defectuosos.
- **Contención de fallas para evitar la propagación de la falla:** Algunos mecanismos de falla pueden causar fallas en el sistema por la propagación de fallas al resto del sistema. El «transmisor no autorizado» es un ejemplo de tal falla que conduce a una comunicación legítima en el sistema y causa una falla completa del sistema. Un transmisor malintencionado o un componente defectuoso debe aislarse para proteger el firewall del sistema u otros mecanismos.
- **Disponibilidad de modos de reversión.**

¿Qué es un fallo?

Es la ocurrencia de un evento que provoca que un sistema, dispositivo, componente o proceso no funcione de acuerdo a su diseño o a las expectativas esperadas. Siendo causados por varias razones, como defectos en el diseño, errores humanos,desgaste,factores ambientales, interferencias, problemas de software, entre otros. Al encontrar un fallo en una ejecución de pruebas, dicho componente contará con un defecto. Existen diferentes fallos:

- **Fallo funcional:** El componente o sistema deja de funcionar por completo y no realiza su tarea prevista.
- **Fallo parcial:** El componente o sistema sigue funcionando, pero lo hace de manera incorrecta o con un rendimiento degradado.
- **Fallo catastrófico:** Un fallo que causa daños graves o la destrucción del sistema o equipo, y potencialmente pone en peligro la seguridad de las personas o el medio ambiente.
- **Fallo intermitente:** El fallo ocurre de manera esporádica o bajo ciertas condiciones específicas, lo que dificulta su detección y diagnóstico.
- **Fallo de software:** Se produce cuando el software del sistema se comporta de manera incorrecta o produce resultados incorrectos debido a errores de programación.
- **Fallo de comunicación:** Los sistemas que se comunican entre sí no pueden intercambiar datos de manera efectiva o no pueden establecer conexiones.
- **Fallo de energía:** La falta de suministro eléctrico o fluctuaciones en la energía pueden causar que los sistemas se apaguen o funcionen de manera inadecuada.
- **Fallo de hardware:** Los componentes físicos, como discos duros, tarjetas de memoria, procesadores, etc., pueden dejar de funcionar correctamente debido a diversos factores.

Entonces, un fallo es la revelación del defecto al utilizar el sistema, o cualquier tecnología. Ya que entre fallo y defecto hay similitud, pero la diferencia está en donde el fallo es la representación visual de un defecto y el defecto es la manifestación de un error.



¿Qué es un error?

Es la equivocación o acción humana en el diseño, desarrollo o implementación de un programa de software, hardware o sistema que produce un resultado incorrecto, inesperado o no deseado. De otra manera, el error es la razón de que algo no funcione de acuerdo a lo esperado. Los errores pueden ser causados por diversos

factores, incluidos errores humanos, defectos en el código de programación, malentendidos de los requisitos, entre otros. Los tipos de errores son:

- **Errores de sintaxis:** Son errores en la estructura del código fuente, donde se violan las reglas del lenguaje de programación utilizado.
- **Errores de lógica:** Se refieren a situaciones en las que el flujo de lógica en el programa no produce el resultado deseado debido a un error en la secuencia de instrucciones.
- **Errores de tiempo de ejecución:** Son errores que ocurren cuando un programa se está ejecutando y se encuentra con una situación inesperada, como una división por cero o un acceso a una ubicación de memoria no válida.
- **Errores semánticos:** Ocurren cuando el código se compila y ejecuta correctamente, pero produce un resultado diferente al esperado debido a una interpretación incorrecta de las instrucciones.
- **Errores de diseño:** Están relacionados con el diseño general del sistema, donde ciertas decisiones de diseño pueden llevar a resultados ineficientes o incorrectos.
- **Errores de entrada/salida:** Estos errores se producen cuando hay problemas al leer o escribir datos desde o hacia dispositivos o archivos.
- **Errores de red:** Se refieren a problemas de comunicación en redes, donde los datos no se transfieren correctamente entre dispositivos.

¿Qué es un bug?

Los bugs informáticos, que significa insecto, se refiere a un error o defecto en el software que hace que un programa no funcione correctamente. Así que un bug, también llamado como defecto es la manifestación de un error de código en la tecnología.

La palabra *bug* tiene su origen desde 1889, cuando Thomas Edison utilizó la palabra bug para referirse al mal funcionamiento de un aparato. Sin embargo, no se popularizó hasta 1947 cuando los operadores del ordenador Mark III lo utilizaron para referirse a los fallos del ordenador. Al revisarlo se dieron cuenta que uno de los relés tenía un comportamiento extraño y decidieron sustituirlo. Al hacerlo, se

encontraron con una polilla atascada en su interior. De ahí viene el término que utilizamos hoy en día para referirnos a los errores de software.
Así que simplemente un bug significa que el resultado no será el esperado. Aunque los bug pueden ser un poco peligrosos, ya que pueden poner en peligro tanto a los usuarios como a las empresas, ya que pueden aprovecharse de esos defectos como vulnerabilidades para atravesar o atacar al sistema de manera malintencionada.

¿Qué es la latencia de un fallo?
La latencia de un fallo es el período de tiempo en el que un componente o sistema experimenta una falla, pero esa falla aún no es detectada o percibida por los usuarios o el sistema en su conjunto. Normalmente la latencia de un fallo se mide en unidad de tiempo (milisegundos o microsegundos).
Cada elemento informático funciona a través de estímulos eléctricos, por lo que el tiempo que se tarda en realizarse todas las conmutaciones eléctricas y lógicas necesarias desde el comienzo de la acción a través de un periférico, hasta que el ordenador ejecuta la acción y muestra los resultados.
La latencia de un fallo puede variar según el tipo de sistema y el contexto. En algunos casos, la latencia de un fallo puede ser muy corta, lo que significa que la falla se detecta y se toman medidas inmediatamente. En otros casos, la latencia puede ser más larga, lo que significa que la falla podría pasar desapercibida durante un período de tiempo antes de que sea identificada y abordada.
Siendo importante detectarlas en sistemas críticos y en aplicaciones donde la detección y corrección temprana de problemas es esencial para evitar consecuencias adversas.

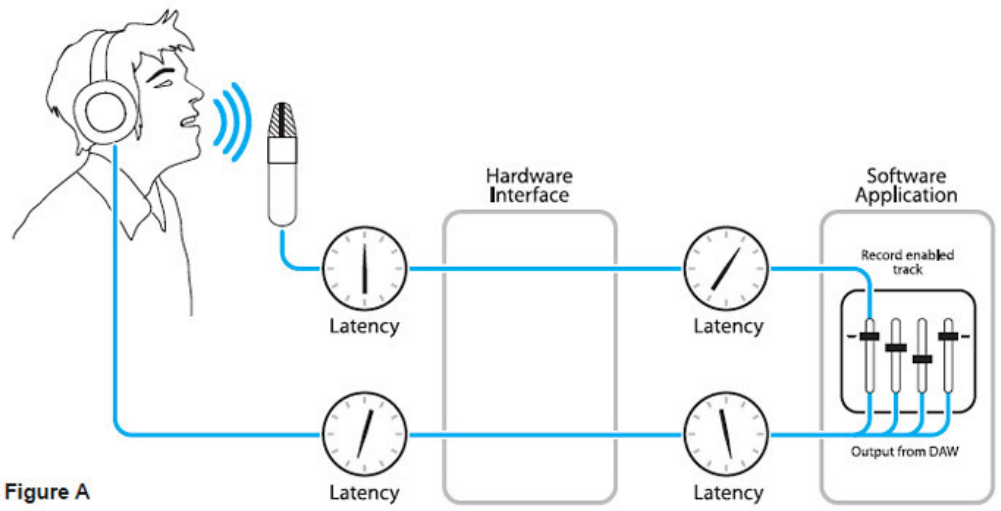


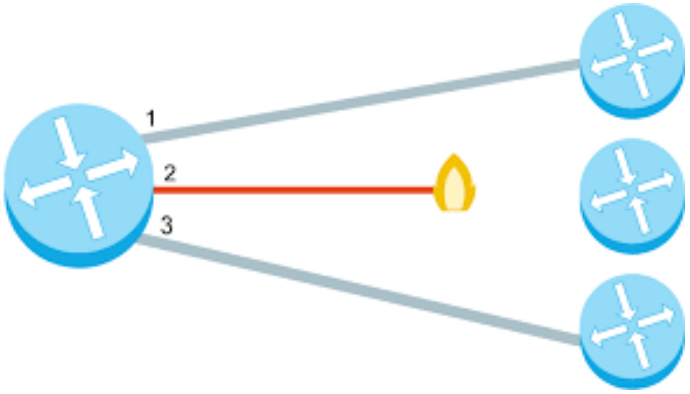
Figure A

¿Qué es la latencia de un error?

La latencia de un error es el tiempo entre la percepción del error y el resultado fallido. Cuando el error produce un fallo, se vuelve efectivo; antes de esto, el fallo es latente. En otras palabras, es el tiempo que pasa desde que se comete un error en el diseño, desarrollo o funcionamiento de un sistema hasta que ese error genera un impacto visible, medible o problemático en el sistema en sí o en sus resultados.

La latencia de un error puede variar según la naturaleza del error, el contexto en el que ocurre y la velocidad a la que los efectos del error se propagan y se hacen evidentes. En algunos casos, un error puede tener una latencia muy corta, lo que significa que sus consecuencias negativas son inmediatas y fácilmente identificables. En otros casos, la latencia puede ser más larga, lo que significa que el error puede permanecer latente durante un tiempo antes de causar problemas notables.

La detección temprana y la corrección de errores pueden ayudar a evitar impactos negativos y a garantizar que el sistema funcione de acuerdo con las expectativas.



Conclusión:

Esta actividad me ayudó a diferenciar los términos usados para identificar algún mal funcionamiento de un sistema o programa. Por lo que me ayudará a entrar en materia, permitiendo identificar las principales causas de que se me pueden presentar. Siendo que el error sea la causa del defecto o bug del sistema, que provoca la falla en el momento en que se ejecuta el sistema o programa.

Bibliografía:

Conozca los servidores con tolerancia a fallos. (2017, marzo 10). Stratus | Zero-touch Edge Computing; Stratus. https://www.stratus.com/es/fault-tolerant/

Necesidades, S., & de solución que ofrecen., L. y. las P. (s/f). Tolerancia a fallas. Udlap.mx. Recuperado el 18 de agosto de 2023, de http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/garcia_r_ra/capitulo2.pdf

Global Business IT. (s/f). Error, Defecto y Fallo ¿Iguales o Diferentes? - Global Business IT. Gbitcorp.com. Recuperado el 18 de agosto de 2023, de https://gbitcorp.com/blog/posts/error-defecto-y-fallo/

Sistemas Distribuidos - Tolerancia a fallos. (2018, enero 7). Blogspot.com. https://programacion-js.blogspot.com/2018/01/sistemas-distribuidos-tolerancia-fallos.html

Sistemas operativos tolerantes a fallos. (2023, julio 11). Computerworld.es. https://www.computerworld.es/archive/sistemas-operativos-tolerantes-a-fallos

Tolerancia a fallos, qué es y técnicas. (2021, diciembre 16). Ciberseguridad. https://ciberseguridad.com/guias/prevencion-proteccion/tolerancia-fallos/

(S/f). Uva.es. Recuperado el 18 de agosto de 2023, de https://www.infor.uva.es/~bastida/Arquitecturas%20Avanzadas/Tolerant.pdf