

**Tecnológico Nacional de México**

**Instituto Tecnológico de Reynosa**

Materia: Sistemas Operativos 2

ACTIVIDAD07: Investigación exhaustiva del Tema03: Consistencia y replicación con sus 4 subtemas.

Tema 3: Consistencia y replicación

Alumno: Castillo Jr. Gregorio

Numero de control: 19580589

Correo electrónico: L19580589@reynosa.tecnm.mx

7mo Semestre Matutino Salón 7

Docente: Mario José Santiago Sánchez

Fecha de entrega: 25/10/2022



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Consecutivo** | **Numero de**  **control** | **Apellido Paterno** | **Apellido Materno** | **Nombres(s)** | **Correo electrónico**  **Institucional** | **Firma de que está**  **autorizando que se**  **entregue esta Publicación** | **Fotografía del rostro de cada**  **Integrante del equipo (selfie)** |
| **1** | **19580589** | **Castillo Jr** |  | **Gregorio** | **L@19580589@reynosa.tecnm.mx** | Abrir foto |  |
| **2** | **19580595** | **Flores** | **Acosta** | **Sheila Lizeth** | **L@19580595@reynosa.tecnm.mx** | No hay descripción disponible. |  |
| **3** | **19580867** | **Morales** | **Calixto** | **Daniel Alexander** | **L@19580867@reynosa.tecnm.mx** | No hay descripción disponible. |  |

Equipo #12

# Tabla de contenidos

[Tabla de contenidos 3](#_Toc117436537)

[Contenido 4](#_Toc117436538)

[3.1 Introducción de modelos de consistencia centrada en los datos 4](#_Toc117436539)

[3.2 Modelos de consistencia centrada en el cliente 7](#_Toc117436540)

[3.3 Administración de replicas 10](#_Toc117436541)

[3.4 Protocolos de consistencia 15](#_Toc117436542)

[Conclusiones 23](#_Toc117436543)

[Herramientas y recursos 24](#_Toc117436544)

[Bibliografía 25](#_Toc117436545)

# Contenido

## **3.1 Introducción de modelos de consistencia centrada en los datos**

*Modelo de consistencia*

Cuando hablamos de los modelos de consistencia centrada en los datos, estos datos tienen que contar con la característica de obedecer ciertas reglas, de modo que permita al almacenamiento trabajar correctamente y que, con ello en mente, evitemos la replicación de dato, de este modo, el proceso que realiza una operación de lectura espera que esa operación devuelva un valor que refleje el resultado de la última operación de escritura sobre el dato.

*Modelos de consistencia centrada en los datos*

*Consistencia estricta:* Ese modelo de consistencia esta definido por la siguiente condición: “Cualquier lectura a una localidad de memoria x regresa el valor guardado en la operación de escritura más reciente en x”. Esto es dado asi ya que es necesario que estos modelos no entren en casos de ambigüedad al tener que evaluar cuales son los datos mas recientes, haciendo asi que esta consistencia no se repita dos veces en el mismo evento, prácticamente es como si las escrituras existieran en carpetas separadas y por cada escritura usamos una sola carpeta la cual ya esta destina en determinado momento, guardar un valor x por ejemplo, si queremos volver a escribir dicho valor en la carpeta y este proceso se da cuenta que la carpeta correspondiente al valor no está vacía, este descarta el valor ya que es necesario que no se repita

*Consistencia secuencial:* Este modelo consiste en una versión mas amigable al ya antes mencionado y cumple con la siguiente norma:” El resultado de una ejecución es el mismo sin las operaciones (lectura y escritura) de todos los procesos sobre el dato fueron ejecutadas en algún orden secuencial.

  -Un dato almacenado secuencialmente consistentemente.

  -Un dato almacenado que no es secuencialmente consistente.

*El resultado de cualquier ejecución es el mismo que si las operaciones de todos los procesos fueran ejecutadas en algún orden secuencial, y las operaciones de cada proceso individual aparecen en esta secuencia en el orden especificado por su programa*”

Como nos podemos dar cuenta y gracias otra vez al ejemplo de las carpetas, en este caso no se “elimina” dicho dato repetido si no mas bien se le agrega “una nota” la cual nos permite ubicar el dato en otra carpeta o como “otra versión”

*Consistencia Causal*

Con esta consistencia sucede algo curioso, ya que necesitamos tener conocimiento del resto de escrituras por lo que la norma a cumplir es la siguiente: “Escrituras que están potencialmente relacionadas en forma causal deben ser vistas por todos los procesos en el mismo orden.

Escrituras concurrentes pueden ser vistas en un orden diferente sobre diferentes máquinas.

Es permitida con un almacenamiento causalmente consistente, pero no con un almacenamiento secuencialmente consistente o con un almacenamiento consistente en forma estricta.”

Con ello en mente y volviendo al ejemplo de las carpetas, nos encontramos que al momento de revisar o modificar una carpeta con el valor de x, tendremos que revisar todas aquellas con las cuales cuenten con una nota de que provienen de equis, ya que con ello evaluamos cual carpeta es la causar de la carpeta que estamos agregando

Este tipo de modelos de distribución están pensados mas que nada en sistemas informáticos en los cuales podemos tener diversos almacenamientos y no solamente en un espacio físico sino que también estos estén distribuidos, ya no solo a lo largo de una ciudad o país, si no alrededor de todo el mundo, con lo cual podemos conectarnos a información (siempre y cuando esta sea publica) como lo son paginas de internet, que no son mas que nada interconexiones entres diversos dispositivos de almacenamiento, o incluso información privada (siempre y cuando se nos den los accesos necesario para ello). Es importante conocer sobre esto, ya que son herramientas que nos permiten trabajar de formas mas optimas y fluidas, ya no solo a niveles de equipo si no también a niveles personales o con proyectos, permitiendo abrir una gamma mayor de posibilidades al desarrollo de nuevas tecnologías y a la comprensión de que es lo que consiste un sistema de estos, como es que administran la información, como la estructuran, como la envían, como la reciben, etc.

Con ello cada uno de estos modelos de consistencia, cumplen con un propósito, para suplir una determinada necesidad, la cual se va corrigiendo con el paso del tiempo, generando mas mejoras y mejores modelos, para cierto propósito, de este modo se vuelve un circulo en el cual tratamos de mejorar un modelo para cumplir con una necesidad y después de suplir esta necesidad se genera otra, a lo cual generamos un nuevo modelo

Dentro de las ventajas de este tipo de modelos podemos encontrar:

* Todos los procesos observan los accesos compartidos en el mismo orden
* Los datos se replican para aumentar la disponibilidad o el rendimiento
* Un aspecto fundamental es la consistencia entre las réplicas, cuando se actualiza una copia se deben actualizar también las demás
* La replicación está muy relacionada con la escalabilidad

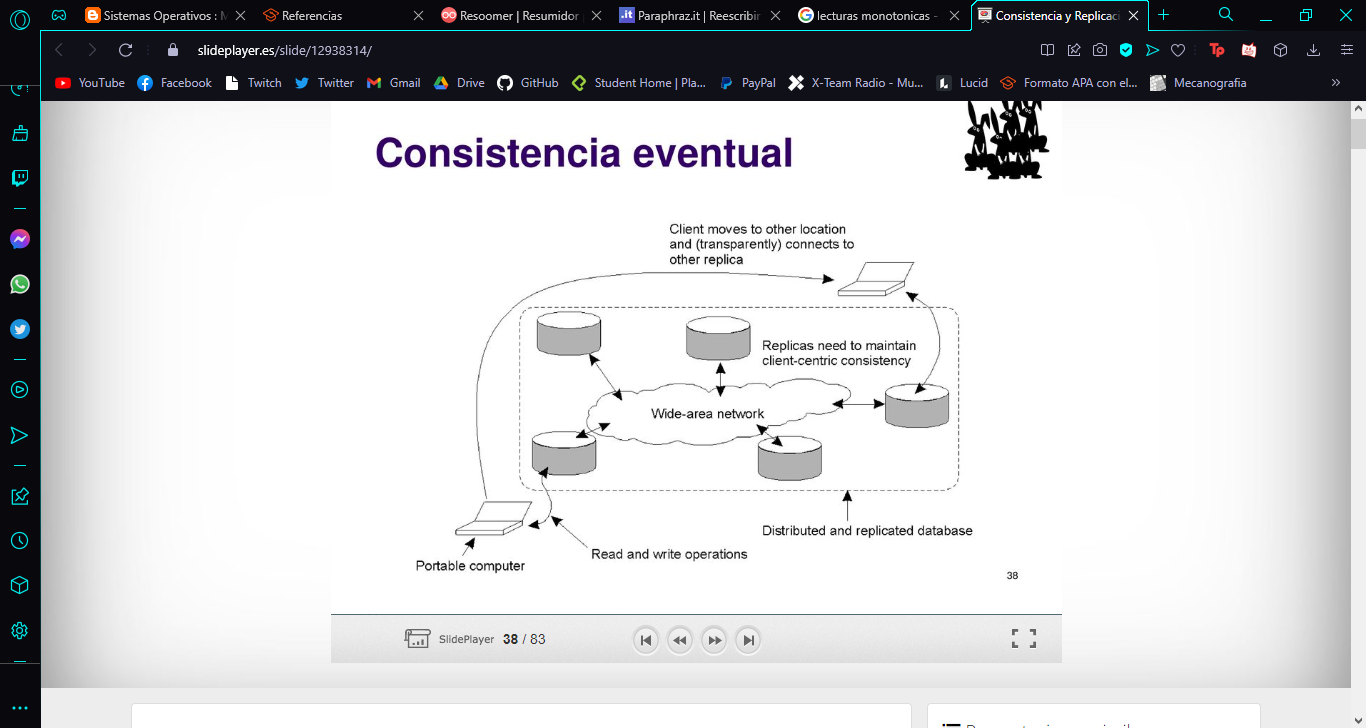
Sin embargo, como todo modelo también cuenta con sus desventajas:

* Carece de reloj global
* Condiciones estrictas
* Seguir instrucciones paso a paso
* No se permiten operaciones de escritura o lectura sobre datos hasta que no se hayan completado operaciones precias sobre variables de sincronización

## **3.2 Modelos de consistencia centrada en el cliente**

Estos modelos cuentan con ciertas características como lo son:

* El bajo número de actualizaciones simultaneas
* Fácil resolución entre actualizaciones concurrentes
* Generalmente con operaciones de lectura
* Modelo de consistencia bastante débil (Consistencia eventual o de eventos)
* Consistencia garantizada para un único cliente, por lo tanto, muchas inconsistencias son ocultadas fácilmente



Para los modelos de consistencia centrada en el cliente estos son muy similares a lo que corresponde como los centrados a datos, con la diferencia de que en general, estos se dedican principalmente a la lectura, con ello evitamos problemas que podemos tener como lo es con el modelo de consistencia centrada en los datos, donde estos pueden ser modificados, pues aquí es donde no hacemos uso de la manipulación de estos datos y por ello es mas sencillo que contemos con este tipo de modelos que solo permiten leer al usuarios, permitiendo esconder efectos que de ser también de escritura, estarían presentes, haciendo más fáciles el resolver dichos problemas

*Lecturas monotónicas*

Un dato ofrece consistencia de lecturas monotónicas si y solo si se cumple la siguiente condición:

“Si un proceso le el valor de un documento x, cualquier operación de lectura sucesiva sobre x por el mismo proceso siempre retornará el mismo valor o un valor más reciente. La consistencia de lecturas monotónicas garantiza que, si un proceso ha visto un valor de X al tiempo T, no se verá una versión más vieja de X en un tiempo posterior.”

Dicho esto, podemos entender que cuando requiramos leer valores en cierto dispositivo este se hará de forma secuencial ya que son realizadas por un único proceso, de este modo pasaremos de leer el proceso numero 1 a el proceso numero 2, esto porque se debe a la realización de 2 copias las cuales son almacenadas en el mismo punto de guardado

*Escrituras monotónicas*

Las escrituras monotónicas deben de ser correctamente propagadas en el orden correcto a todas las copias del almacenamiento se debe cumplir:

Una operación de escritura por un proceso sobre un elemento de datos x debe de ser completada antes que cualquier otra operación siguiente de escritura sobre x por el mismo proceso, esto quiere decir que si una operación de escritura sobre una copia de un elemento X se realiza solo si la copia se ha actualizado mediante cualquier operación de escritura previa

En este apartado lo que sucede cuando tenemos múltiples tareas por hacer es que se enlistan, dando paso a la realización de estas de forma secuencial, es decir que no podemos estar trabajando en varias tareas al mismo tiempo ni tampoco agregar tareas entre estas, esto para llevar a cabo un orden para la realización de estas

*Lea sus escrituras*

Un almacén de datos provee consistencia lea sus escrituras si se cumple que:

Es cuando se sigue el mismo proceso un ejemplo de ello es cuando yo escribo un documento X, con el mismo proceso leeré el mismo documento. El efecto de una operación de escritura por procesos sobre un elemento de datos X será siempre visto por las operaciones sucesivas de lectura sobre X por el mismo proceso.

Dentro de esto, lo único que se nos menciona es que podemos hacer lectura de todos y cada unos de nuestros archivos, sin importar el lugar ni la hora, esto siempre y cuando contemos con que nuestros archivos han sido escritos correctamente

*Las escrituras siguen a lecturas*

El objetivo de este modelo es que, si se va a modificar el valor de un dato, es que haya sido leído antes de la última actualización.

Un almacén de datos provee consistencia de escrituras siguen lecturas solo si se cumple que:

Una operación de escritura de un proceso sobre un elemento de datos x realizada luego de leer ese dato x, se realizó sobre el valor más reciente de x.

De esta forma nos podemos asegurar que de necesitar sobrescribir algún dato, este cliente lo hará sobre el dato mas reciente, ya que es el dato de lectura el que se le muestra y con ello después podemos modificar el de escritura

## **3.3 Administración de replicas**

**Replicación**

Es una clave que sirve para proporcionar alta disponibilidad y tolerancia a los fallos en un sistema distribuido, al igual que proporciona una mejora en servicios como:

* La mejora del rendimiento
* Incremento de disponibilidad
* Tolerancia a los fallos

Se defina como el conjunto de técnicas que tratan proporcionar un mayor rendimiento, disponibilidad y escalabilidad mediante el mantenimiento de copias.

**Administrador de replicas**

En sistemas operativos el administrador de replicas son subsistemas que tiene la responsabilidad de administrar la sincronización de replicas. Este administrador de replicas es responsable de:

* Serialización y deserialización de las réplicas en cada homólogo
* Reenvío de réplicas de un homólogo a otro
* frotación de cambios de propiedad de réplicas
* Administración de la vida útil de las réplicas

Uno de los puntos clave que tienen los sistemas distribuidos que soportan las réplicas es la forma de decidir, donde cuando y por quien deben de ser ubicadas las réplicas y cuales mecanismos se utilizaron para mantener consistentes a estas.

El problema de la ubicación en la administración de réplicas es dividido en dos “subproblemas”:

* Ubicación de servidoras de replicas
* Ubicación de contenido

La diferencia de estos es muy importante y con frecuencia las dos cuestiones no están claramente separadas. La ubicación del servidor de réplicas esta en encontrar los mejores lugares para colocar un servidos que sea parte de un almacén de datos, mientras que la ubicación de contenido se relaciona en encontrar a los mejores servidores para poner el contenido.

**Ubicación del servidor de replicas**

No es considerado un problema estudiado intensivamente ya que a menudo se trata de un asunto más administrativo y comercial que un problema de optimización. Aun así, el análisis de las propiedades de los clientes y la red es útil para llegar a tomar decisiones informadas.

Hay muchas formas de calcular la mejor ubicación de los servidores de réplicas, esto se resume en un problema de optimización en el que se necesita seleccionar la mejor “K” entre “N” (ubicaciones K<N). Sabemos que estos problemas son considerados de cómputo complejo y solo se resuelven mediante la heurística. Se dice que Qiu y sus colaboradores en 2001 toman la distancia entre los clientes y ubicaciones desde el punto de partida. La distancia se mide en términos de latencia o ancho de banda.

Para la solución se selecciona un servidor a un tiempo que tal distancia promedio entre ese servidor y sus clientes sea mínima, ya que “k” servidores ya han sido ubicados, significando que hay N-K ubicaciones que fueron descartadas.

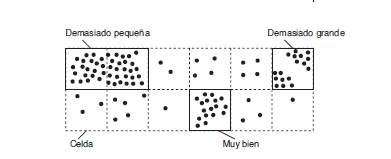
Como otra alternativa tenemos a Radoslavov y sus colaboradores (desde 2001) propusieron ignorar la posición de los clientes, solo se toma la topología de internet como si esta estuviera formada por sistemas autónomos. El sistema autónomo se puede considerar como una red en la que todos los nodos son ejecutados al mismo protocolo de enrutamiento y es administrada por una sola organización. En enero 2006, ya existían más de 20,000 sistemas autónomos. Radoslavov y los colaboradores consideran que el primer sistema autónomo más grande y colocan un servidor en el ruteador que tenga el número mayor de interfaces de red. Después el algoritmo es repetido con el segundo sistema autónomo más grande y así se va sucesivamente.

El resultado de la ubicación de un servidor sin conocimiento del cliente, hace que logre resultados similares a los que se obtuvieron con conocimiento del cliente, clero que bajo la suposición de que los clientes están distribuidos de forma uniformemente a través de internet dependiendo de la topología existente.

Un problema sobre estos algoritmos es que son de cómputo caro. Por ejemplo, los dos algoritmos previos tienen complejidad mayor a “O”\*N^2, donde “N” es el número de las ubicaciones por inspeccionar. Ya en práctica suele significar que incluso para algunas de las ubicaciones un cálculo puede necesitar ejecutarse por decenas en unos minutos. Puede resultar inaceptable y más cuando se presentan “flash crowds” (se le conoce como un súbito de peticiones para un sitio especifico). En estos tipos de casos es básico determinar rápidamente en donde se necesitan los servidores de réplicas, de la cual después se puede seleccionar un servidor para la ubicación de contenido.

En 2006 Szymaniak y colaboradores desarrollaron un nuevo método donde se puede identificar rápidamente una región para ubicar replicas. La región es identificada para ser una colección de nodos que acceden al mismo contenido, pero para la cual la latencia internodal es baja. Tiene como objetivo del algoritmo es seleccionar primero las regiones con más demanda, es decir que aquellas con más nodos. Y después de dejar uno de los nodos de esa región actúe como un servidor de réplicas.

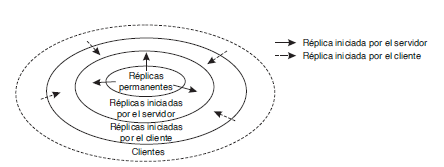
Lo importante aquí es el tamaño de la celda. Si estas se eligen demasiado grandes, entonces varios de los clústeres de nodos se pueden encontrar en la misma celda. Si es ese caso, entonces se elegirían muy pocos servidores de réplicas para esos clústeres. De lo contrario, si se eligen celdas pequeñas puede ocasionar que un solo clúster se propague a través de varias celdas, lo que ocasiona la elección de muchos servidores de réplicas.



De resultado vamos a tener un tamaño adecuado de celda que puede calcularse como una simple función de la distancia promedio existente entre nodos y el número de réplicas requeridas.

**Ubicación y replicación de contenido**

Cuando se habla de la replicación y ubicación de contenido es posible diferenciar como tres tipos de replicas lógicamente organizadas.



* Réplicas permanentes

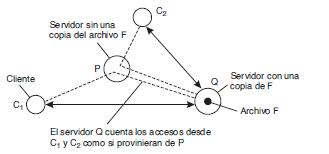
Se consideran como el conjunto inicial de réplicas que constituyen un almacén de datos distribuido. En varios casos el número de las réplicas permanentes es pequeña. Por ejemplo, si consideramos un sitio web, la distribución por lo general viene en una de dos formas. La primera es aquella en la cual los archivos que constituyen un sitio se replican a través de un número limitado de servidores en una ubicación. Cada que llega una petición, se reenvía a uno de los servidores.

La segunda se conoce como espejar. El sitio web es copiado en un numero limitado de servidores, son conocidos como sitios espejo, donde están geográficamente distribuidos en internet.

* Réplicas iniciadas por servidores

En contraste de las réplicas permanentes, las iniciadas por servidores son copiadas de un almacén de datos que existe para mejorar el rendimiento, estas son creadas por iniciativa del propietario del almacén de datos.

El problema que hay en colocar replicas dinámicamente se está tratando también en los servicios web del hosting. Ofrecen una colección de servidores que están dispersos a través de internet que pueden mantener y proporcionar acceso a archivos web de terceras personas.



* Réplicas iniciadas por el cliente

La replica iniciada por el cliente es considerada importante ya que son más comúnmente conocidas como cachés. El caché es una herramienta de almacenaje local utilizada por un cliente por un cliente para poder almacenar temporalmente una copia de datos que fueron solicitados. La administración del caché se les deja completamente a los clientes.

Los cachés del cliente solo son utilizados para poder mejorar el tiempo de acceso a los daros. Cuando un cliente quiere acceder a estos datos se conecta con la copia más cercana del almacén de datos desde donde se traen los datos que se requieran leer o a donde almacenan los datos que acaban de ser modificados

## **3.4 Protocolos de consistencia**

Protocolos de Consistencia

Los protocolos de consistencia describen una implementación de un modelo de consistencia específico (incluyen la consistencia secuencial, consistencia débil con variables de sincronización, así como transacciones atómicas).

Básicamente pueden ser separados considerando si hay una copia primaria de los datos donde todas las escrituras son mandadas, en caso contrario las copias pueden ser iniciadas desde cualquier réplica.

**Protocolos basados en primarias**

Estos modelos incluyen a los que limitan la desviación de discontinuidad, y en menor medida a los que limitan las desviaciones numéricas. Cuando se trata de modelos que manejan el ordenamiento consistente de operaciones, o consistencia secuencial, son más populares aquellos en los que las operaciones pueden agruparse mediante candados o transacciones.

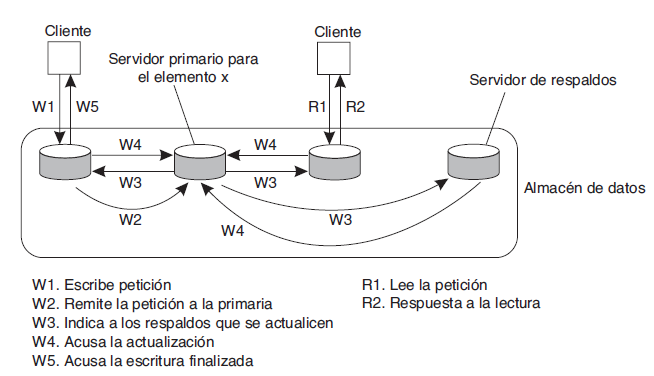
Tan pronto como los modelos de consistencia se vuelven ligeramente difíciles de entender

para los desarrolladores de aplicaciones, vemos que son ignorados aun cuando el rendimiento pudiera mejorarse. El resultado final es que si la semántica de un modelo de consistencia no resulta

intuitivamente clara, los desarrolladores la pasarán mal para construir aplicaciones correctas. La sencillez es muy apreciada (y tal vez justificada). En el caso de la consistencia secuencial, resulta que prevalecen los protocolos basados en primarias. En estos protocolos, cada elemento de datos x del almacén de datos tiene una primaria asociada, la cual es responsable de coordinar las operaciones de escritura sobre x. Es posible diferenciar si la primaria está fija en un servidor remoto o si las operaciones de escritura pueden realizarse localmente después de trasladar la primaria al proceso en donde se inició la operación de escritura. Demos un vistazo a esta clase de protocolos.

**Protocolos de escritura remota**

El protocolo mas sencillo basado en primarias que soporta la repilacion es aquel en el que todas las operaciones de escritura necesitan remitirse a un solo servidor fijo. Tales esquemas también se conocen como protocolos primarios de respaldo.



Un protocolo primario de respaldo funciona de esta forma, un proceso que espera para poder realizar una operación de escritura sobre un elemento de datos *x*, remite esa operación al servidor primario de x.

La primaria realiza la actualización en su copia local de x, y posteriormente remite la actualización a los servidores de respaldo. Cada servidor de respaldo también realiza la actualización, y envía un acuse de recibo de vuelta a la primaria. Cuando finalmente todos los respaldos se han actualizado su copia local la primaria envía un acuse de vuelta al proceso inicial.

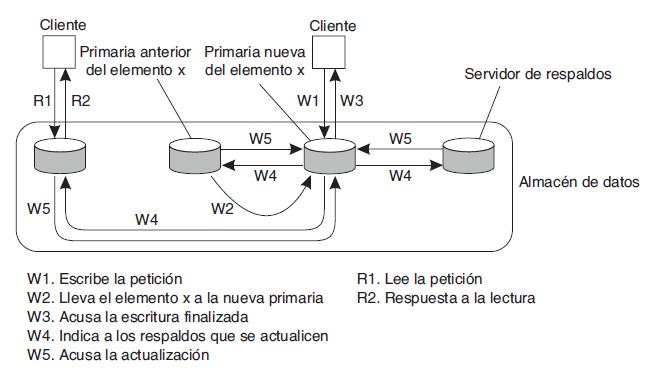
Uno de los problemas es la potencia de rendimiento con este esquema es que puede llegar a pasar demasiado tiempo antes de que el proceso que se inició con la actualización se le pueda permitir continuar. En efecto, una actualización es implementada como una operación de bloqueo. Una alternativa es utilizar un método de no bloqueo. Tan pronto como la primaria se haya actualizado su copia local de *x,* esta devuelve un acuse; después de lo cual indica a los servidores de respaldos que también realicen la actualización.

Uno de los principales problemas con los protocolos primarios de respaldo de no bloqueo tiene que ver con su tolerancia a las fallas. En un esquema de bloqueo, el proceso cliente sabe y tiene la certeza que la operación de actualización es respaldada por varis servidores. Este no es el caso con una solución de no bloqueo. La ventaja, por supuesto es que las operaciones de escritura pueden ser considerablemente rápidas.

Los protocolos primarios de respaldo proporcionan una implementación directa de la consistencia secuencial, ya que la primaria puede llegar a ordenar todas las escrituras enterantes en un orden de tiempo globalmente de lectura. Es evidente que todo el proceso vea a todas las operaciones de escritura en el mismo orden. Con los protocolos de bloqueo, los procesos siempre verán los efectos de su operación de escritura más reciente.

**Protocolos de escritura local**

Una de las variantes de los protocolos primarios de respaldo es un protocolo en el que copia primaria migra entre procesos que desean realizar una operación de escritura. Siempre que un proceso quiere actualizar el elemento *x*, este localiza la copia primaria de *x*, y posteriormente la lleva a su propia ubicación. La principal ventaja de este método es que varias operaciones sucesivas de escritura pueden realizarse localmente, mientras que los procesos de lectura aún pueden acceder a su copia local. Sin embargo, tal mejora solo puede lograrse si un protocolo de no bloqueo es seguido por las actualizaciones que se pongan a las réplicas después de que la primaria ha terminado de realizar localmente las actualizaciones.



Este protocolo primario de respaldo de escritura local también lo podemos llegar aplicar a unas computadoras móviles que son capaces de poder operar desconectadas. Antes de desconectar o apagar, la computadora móvil se vuelve el servidor primario para cada elemento de datos que espera actualizar. Mientras es desconectada, todas las operaciones de actualización se llevan a cabo localmente en lo que otros procesos aún pueden realizar operaciones de lectura (pero no actualizaciones). Después, cuando se conecta de nuevo, las actualizaciones se propagan desde la primaria hacia los respaldos, llevando al almacén de datos de nuevo a un estado consistente.

Como una última variante de este esquema, los protocolos basados en primarias de escritura local y de no bloqueo también son utilizados por sistemas de archivo distribuidos en general. En este caso, puede haber un servidor central fijo a través del cual, normalmente, ocurren todas las operaciones de escritura, tal como en el caso de la primaria de respaldo de escritura remota.

Sin embargo, de modo temporal el servidor permite que una de las réplicas realice una serie de actualizaciones locales, ya que esto puede incrementar considerablemente el rendimiento. Cuando el servidor de réplicas termina, las actualizaciones se propagan hacia el servidor central, desde donde se distribuyen hacia los otros servidores de réplicas.

**Protocolos de escritura replicados**

En los protocolos de escritura replicados, las operaciones de escritura pueden realizarse en varias réplicas en lugar de un solo en una, como en el caso de réplicas basadas en primarias. Es posible diferenciar la replicación activa, en la que una operación se remite a todas las réplicas, y los protocolos de consistencia basados en la mayoría de votos.

Replicación activa

En la replicación activa, cada réplica tiene un proceso asociado que realiza operaciones de actualización. Por contraste con otros protocolos, generalmente las actualizaciones se propagan mediante la operación de escritura que causa la actualización. En otras palabras, la operación se envía a cada réplica. Sin embargo, también es posible enviar la actualización.

el ordenamiento total puede lograrse utilizando un coordinador central, también llamado secuenciador. Un método es remitir primero cada operación al secuenciador, el cual le asigna un número de secuencia único y posteriormente remite la operación a todas las réplicas. Las operaciones se realizan según el orden de sus números de secuencia. Resulta claro que esta implementación de multidifusión totalmente ordenada se parece mucho a los protocolos de consistencia basados en primarias. Observe que utilizar un secuenciador no resuelve el problema de escalabilidad.

Protocolos basados en quorum

Un método diferente para soportar escrituras replicadas es utilizar la votación como la propuso

originalmente Thomas (1979) y fue generalizada por Gifford (1979). La idea básica es requerir a

los clientes que soliciten y adquieran el permiso de varios servidores antes de leer o escribir un elemento de datos replicado.

Como un ejemplo sencillo del funcionamiento del algoritmo, considere un sistema distribuido

de archivos y suponga que un archivo se replica en N servidores. Podríamos implementar una regla

que establezca que, para actualizar un archivo, un cliente primero debe contactar al menos a la

mitad de los servidores más uno (la mayoría) y hacerlos aceptar que se realice la actualización. Una

vez que lo acuerden, el archivo es cambiado y se asocia un nuevo número de versión al archivo

resultante. El número de versión se utiliza para identificar la versión del archivo, y es el mismo

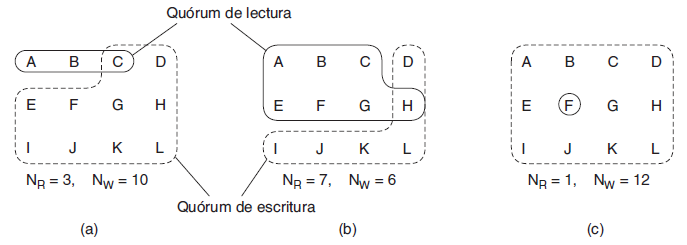
para todos los archivos actualizados recientemente.

Para leer un archivo replicado, un cliente debe contactar también al menos a la mitad de los

servidores más uno y solicitarles le envíen los números de versión asociados con el archivo. Si

todos los números de versión son los mismos, ésta debe ser la versión más reciente porque intentar

actualizar sólo a los servidores restantes fallaría ya que no hay suficientes de ellos.



**Protocolos de coherencia de cache**

Los caches forman parte de un caso en especial de replicación en este sentido entendemos que generalmente, son controlados por los clientes en lugar de por los servicios. Sin embargo, los protocolos de coherencia de cache, los cuales nos garantizan la consistencia del cache con las réplicas iniciadas por el servidor son solo el principio de los protocolos de consistencia.

Mucha investigación sobre el diseño y la implementación de caches, especialmente en el contexto de sistemas multiproceso de memoria compartida. Muchas de las soluciones están basadas en el soporte de hardware subyacente. En los sistemas distribuidos basados en middleware que se construyen en la cima de sistemas operativos de propósito general, las soluciones basadas en software son más interesantes.

Las soluciones de cacheo pueden diferir en su estrategia de detección de coherencia, esto es, cuando se detectan en realidad las inconsistencias. Las soluciones estáticas asumimos que el compilador realiza el análisis necesario previo a la ejecución y para determinar efectivamente cual dato pudiera provocar inconsistencias debido a que pueda ser cacheado. El compilador simplemente inserta instrucciones que evitan inconsistencias.

Con lo protocolos de coherencia de cache es la estrategia de reforzamiento de coherencia, la cual determina la forma en que se mantienen consistencia los caches con el número de copias almacenadas en los servidores.

**Implementación de la consistencia centrada en el cliente**

Hay que enfocarnos en la implementación de la consistencia centrada en el cliente. La implementación de la consistencia centrada en el cliente es relativamente sencilla si ignoramos cuestiones de rendimiento.

Una implementación simplista

Es una implementación simplista de la consistencia centrada en el cliente, a nada operación de escritura *W* se le asigna un identificador global único. Este identificador es asignado por el servidor al que se le solicita la escritura.

La consistencia monotonía de lectura se implementa de esta manera, cuando un cliente realiza una operación de lectura en un servidor, dicho servidor manipula al conjunto de lecturas del cliente para verificar si todas las escrituras identificadas se realizaron localmente, antes de concluir la operación de lectura, hace contacto con los otros servidores para asegurarse de que este el dia. Una manera alternativa, la operación de lectura se reenvía hacia el servidor de donde se llevaron a cabo las escrituras una vez ya efectuada la operación de lectura, las operaciones de escritura realizadas en el servidor seleccionado, y que son relevantes para la operación de lectura, se adicionan al conjunto de lecturas del cliente.

Como mejorar eficiencia

Advertir que el conjunto de lectura y escritura asociado con cada cliente puede volverse muy grande. Para mantener manejables estos conjuntos, las operaciones de lectura y escritura de un cliente se agrupan en sesiones. Una sesión está asociada con una aplicación: esta se abre cuando la aplicación comienza, y se cierra cuando finaliza. Cada vez que un cliente cierra una sesión, los conjuntos simplemente son vaciados. Por supuesto si un cliente abre una sesión que nunca se cierra, los conjuntos de lectura y escritura asociados pueden volverse muy grandes.

El problema principal con la implementación simplista radica en la representación de los conjuntos de lectura y escritura. Cada conjunto costa de un numero de identificadores para operaciones de escritura. Cada vez que un cliente remite una petición de lectura o escritura hacia un servidor, también se transmite al servidor un conjunto de identificadores para ver si todas las operaciones de escritura importantes para la petición se han realizado en dicho servidor.

# **Conclusiones**

A lo largo de la unidad, entendimos en que consisten los procesos de escritura y lectura de un sistema distribuido, entendimos que la complejidad de estos radica esencialmente en cuanto existen diversos usuarios, leyendo y modificando un solo archivos, esto quiere decir que tenemos que desarrollar una forma de darle un orden y secuencia a lo que queramos presentar, ya que de no hacerlo podemos desorganizar todo, o peor aun perderlo todo, no hablamos solamente de archivos, sino que también de bases de datos completas, ya que contamos con los procesos y desarrollos necesarios para esto, aunque siempre se pueden encontrar cosas en las cuales mejorar, estos modelos presentados en el documento son los estándares mínimos con los cuales tiene que cumplir nuestro sistema, esto para generar confianza a la hora de usarlos y no encontremos errores críticos a la hora de trabajar con ello, es importante destacar que la efectividad de estos procesos radica en diversos aspectos y no solo de software, desde el hardware que usamos, hasta las personas con las que conectamos, es fundamental entender que mientras mas complejidad exista en estos sistemas no necesariamente significa que sean mejores, puesto que el objetivo siempre a sido reducir el tiempo de perdida de los usuarios y maximizar el tiempo de producción

# **Herramientas y recursos**

* Imágenes de Google
* Power Point para crear la exposición
* Youtube
* El documento de nuestra investigación del tema
* Scribbr para el APA
* Equipo de cómputo.

# **Bibliografía**

Administrador de réplicas - Guía del usuario de Lumberyard. (s. f.). Recuperado 22 de octubre de 2022, de <https://docs.aws.amazon.com/es_es/lumberyard/latest/userguide/network-replicas-replica-manager.html>

Salguero, E. (2019, 12 octubre). Sistemas distribuidos: replicación (IV) - Edu Salguero. Medium. Recuperado 22 de octubre de 2022, de <https://medium.com/@edusalguero/sistemas-distribuidos-replicacion-14d8f3819c1d>

Ubicación del servidor de replicas - Trabajos de investigación - 475 Palabras. (s. f.). Buenas Tareas. Recuperado 23 de octubre de 2022, de <https://www.buenastareas.com/ensayos/Ubicaci%C3%B3n-Del-Servidor-De-Replicas/1130721.html>

Access denied. (s. f.). Recuperado 23 de octubre de 2022, de <https://techdocs.broadcom.com/es/es/symantec-security-software/endpoint-security-and-management/endpoint-protection/all/Managing-management-servers-sites-and-databases/setting-up-sites-and-replication-v50382461-d15e1412/what-are-sites-and-how-does-replication-work-v52289630-d15e1767.html>

*Access to this page has been denied.* (s. f.). Recuperado 23 de octubre de 2022, de https://www.studocu.com/es-mx/document/instituto-tecnologico-latinoamericano/calculo-integral/modelos-de-consistencia-en-los-datos-sistemas-operativos-ii/19511599

Maldonado, R. (2021, 16 abril). *Modelo de consistencia centrada en datos*. Apuntes - Rodri Maldonado. Recuperado 23 de octubre de 2022, de https://www.clubensayos.com/Tecnolog%C3%ADa/Modelo-de-consistencia-centrada-en-datos/5257907.html

*Modelos de Consistencia Centrada al Cliente*. (s. f.). Blog. http://blogcristoramirez.blogspot.com/2016/09/descripcionde-modelosde-consistencia.html

*MODELOS DE CONSISTENCIA (CENTRADA EN DATOS)*. (s. f.). Recuperado 23 de octubre de 2022, de http://itsrlljcr.blogspot.com/2016/09/modelos-de-consistencia-centrada-en.html