Artículo
Mario Sanchez Macias · 13 mayo, 2021 Lectura de 9 min

Chequeando la integridad de datos: Acelerarla o Ralentizarla.

Aunque la integridad de las bases de datos Caché e InterSystems IRIS está completamente protegida de las consecuencias de un fallo de sistema, los dispositivos de almacenamiento físico sí que pueden fallar, corrompiendo los datos que almacenan.

Por esa razón, muchos sitios optan por realizar chequeos o verificaciones periódicas de integridad de bases de datos, sobre todo en coordinación con las copias de seguridad, para validar que se pueda confiar en una determinada copia de seguridad en caso de que ocurra algún desastre. El chequeo de integridad también puede ser muy necesario para el administrador del sistema, como respuesta a un desastre que implique la corrupción del almacenamiento. El chequeo de integridad ha de leer todos los bloques de los *globals* que están en el proceso de verificación (si actualmente no están en los *buffers*), y en el orden dictado por la estructura del global. Esto lleva mucho tiempo, **pero el chequeo de integridad es capaz de leer tan rápido como el subsistema de almacenamiento pueda soportar**. En algunas situaciones, es aconsejable ejecutarlo de esta manera para obtener resultados lo más rápido posible. En otras situaciones, el chequeo de integridad debe ser más conservador para evitar consumir demasiado ancho de banda del subsistema de almacenamiento.

Plan de Ataque

El siguiente esquema se adapta a la mayoría de las situaciones. El análisis detallado del resto de este artículo proporciona la información necesaria para actuar sobre cualquiera de ellos, o para derivar otras líneas de acción.

1. Si utilizas Linux y el chequeo de integridad es lento, consulta la información que encontrarás más abajo sobre la activación de la E/S asíncrona.
2. Si el chequeo de integridad debe completarse lo más rápido posible (ejecutándose en un entorno aislado, o porque los resultados se necesitan urgentemente), utiliza el chequeo de Integridad Multiproceso para comprobar varios *globals* o bases de datos en paralelo. El número de procesos multiplicado por el número de lecturas asíncronas simultáneas que cada proceso realizará (8 de forma predeterminada, o 1 si se utiliza Linux con la E/S asíncrona deshabilitada) es el límite del número de lecturas simultáneas sostenidas. Considera que el promedio puede ser la mitad de eso y después compara con las características del subsistema de almacenamiento. Por ejemplo, con el almacenamiento dividido en 20 unidades y las 8 lecturas simultáneas predeterminadas por proceso, pueden ser necesarios cinco o más procesos para capturar toda la capacidad del subsistema de almacenamiento (5*8/2=20).
3. Al equilibrar la velocidad del chequeo de integridad contra su impacto en producción, primero ajusta el número de procesos en el chequeo de integridad multiproceso; después, si es necesario, consulta el ajuste SetAsyncReadBuffers. Consulta el chequeo de Integridad de Aislamiento indicado más abajo para obtener una solución a largo plazo (y para eliminar falsos positivos).
4. Si ya está limitado a un solo proceso (por ejemplo, hay un *global* extremadamente grande u otras restricciones externas) y la velocidad de el chequeo de integridad necesita un ajuste hacia arriba o hacia abajo, consulta más abajo el ajuste SetAsyncReadBuffers.

Chequeo de Integridad Multiproceso

La solución general para obtener un chequeo de integridad que se complete más rápido (usando recursos del sistema a un mayor ritmo) es dividir el trabajo entre varios procesos paralelos. Algunas de las interfaces de usuario y APIs de chequeo de integridad lo hacen, mientras que otras utilizan un solo proceso. La asignación a los procesos es una por global, por lo que el chequeo de un único *global* siempre se realiza mediante un solo proceso (las versiones anteriores a Caché 2018.1 dividían el trabajo por base de datos en vez de por *global*).

La API principal para verificar la integridad de varios procesos es **CheckLIst Integrity** (consulta la documentación para más detalles). Recopila los resultados en un *global* temporal para ser mostrados por Display^Integrity. El siguiente es un ejemplo de verificación de tres bases de datos usando cinco procesos. Si se omite el parámetro de la lista de bases de datos, se verifican todas las bases de datos.

```
set dblist=$listbuild("/data/db1/","/data/db2/","/data/db3/")
set sc=$$CheckList^Integrity(,dblist,,,5)
do Display^Integrity()
kill ^IRIS.TempIntegrityOutput(+$job)

/* Note: evaluating 'sc' above isn't needed just to display the results, but...
   $system.Status.IsOK(sc) - ran successfully and found no errors
   $system.Status.GetErrorCodes(sc)=$$$ERRORCODE($$$IntegrityCheckErrors) // 267
                           - ran successfully, but found errors.
   Else - a problem may have prevented some portion from running, 'sc' may have
          multiple error codes, one of which may be $$$IntegrityCheckErrors. */
```

El uso de CheckLIst^Integrity de esta manera es la forma más sencilla de lograr el nivel de control que nos interesa. Tanto la interfaz del Portal de administración como la Tarea de Chequeo de Integridad (incorporada, pero no programada) utilizan varios procesos, pero puede que no ofrezca un control suficiente para nuestros propósitos.*

Otras interfaces de chequeo de integridad, especialmente la interfaz de usuario del terminal, ^INTEGRIT o ^Integrity, así como Silent^Integrity, realizan el chequeo de integridad en un solo proceso. Por lo tanto, estas interfaces no completan el chequeo tan rápido como es posible conseguir, y utilizan menos recursos. Sin embargo, una ventaja es que sus resultados son visibles, se registran en un archivo o se envían al terminal, según se verifica cada *global*, y en un orden bien definido.

E/S asíncronas

Un proceso de chequeo de integridad recorre cada bloque puntero de un *global*, uno cada vez, validando cada uno contra el contenido de los bloques de datos a los que apunta. Los bloques de datos se leen con E/S asíncrona para mantener un número de solicitudes de lectura sostenidos para que el subsistema de almacenamiento las procese, y la validación se va completando cada lectura.

En Linux, la E/S asíncrona solo es efectiva en combinación con la E/S directa, que no está habilitada de forma predeterminada hasta InterSystems IRIS 2020.3. Esto explica un gran número de casos en los que el chequeo de la integridad tarda demasiado tiempo en Linux. Afortunadamente, se puede habilitar en Cache 2018.1, IRIS 2019.1 y posteriores, al establecer **wduseasyncio=1** en la sección [config] del archivo .cpf y reiniciando. Este parámetro se recomienda en general para la escalabilidad de E/S en sistemas con mucha carga y es el predeterminado en plataformas que no son de Linux desde Caché 2015.2. Antes de habilitarlo, asegúrate de que has configurado suficiente memoria de caché de base de datos (global *buffers*) porque con Direct E/S, las bases de datos ya no serán almacenadas en la caché (de forma redundante) por Linux. Si no se activan, las lecturas realizadas por el chequeo de integridad se completan de forma sincróna y no se puede utilizar el almacenamiento de forma eficiente.

En todas las plataformas, el número de lecturas que un proceso de chequeo activa esta fijado en 8 por defecto. Si tienes que alterar la velocidad a la que un solo proceso de chequeo lee del disco, este parámetro se puede ajustar: hacia arriba para conseguir que un solo proceso se complete más rápido o hacia abajo para utilizar menos ancho de banda de almacenamiento. Ten en cuenta que:

- Este parámetro se aplica a cada proceso de chequeo de integridad. Cuando se utilizan varios procesos, el número de procesos multiplica este número de lecturas sostenidas. Cambiar el número de procesos de chequeo de integridad paralelos tiene un impacto mucho mayor y, por tanto, normalmente es lo primero que se hace. Cada proceso también está limitado por el tiempo computacional (entre otras cosas), por lo

que aumentar el valor de este parámetro tiene un beneficio limitado.
- Esto solo funciona dentro de la capacidad del subsistema de almacenamiento para procesar lecturas simultáneas. Valores más altos no tienen ningún beneficio si las bases de datos se almacenan en una sola unidad local, mientras que una matriz de almacenamiento con striping a lo largo de docenas de unidades, puede procesar docenas de lecturas de forma simultánea.

Para ajustar este parámetro desde el namespace %SYS, **do SetAsyncReadBuffers^Integrity(**value**)**. Para ver el valor actual, **write $$GetAsyncReadBuffers^Integrity()**. El cambio tiene efecto cuando se verifica el siguiente *global*. La configuración no persiste tras un reinicio del sistema, aunque se puede añadir a SYSTEM^%ZSTART.

Hay un parámetro similar para controlar el tamaño máximo de cada lectura cuando los bloques son contiguos (o cercanos) en el disco. Este parámetro se necesita con menos frecuencia, aunque los sistemas con alta latencia de almacenamiento o bases de datos con tamaños de bloque más grandes podrían beneficiarse de un ajuste más fino. El valor tiene unidades de 64KB, por lo que un valor de 1 es 64KB, 4 es 256KB, etc. 0 (el valor predeterminado) permite que el sistema seleccione automáticamente,y, actualmente selecciona 1 (64KB). La función ^Integrity para este parámetro, paralela a las mencionadas anteriormente, son **SetAsyncReadBufferSize** y **GetAsyncReadBufferSize**.

Aislamiento del Chequeo de integridad

Muchos sitios realizan chequeos periódicos de integridad directamente en el sistema de producción. Esto es lo más sencillo de configurar, pero no es lo ideal. Además de las preocupaciones sobre el impacto de el chequeo de integridad en el ancho de banda de almacenamiento, la actualización simultánea de la base de datos a veces puede conducir a errores de falsos positivos (a pesar de las mitigaciones incorporadas en el algoritmo de verificación). Como resultado, los errores reportados por una chequeo de integridad ejecutado en producción deben ser evaluados y/o verificados de nuevo por un administrador.

Con frecuencia, existe una mejor opción. Un snapshot del almacenamiento o una imagen de la copia de seguridad se pueden montar en otro servidor, donde una instancia aislada de Caché o IRIS ejecuta el chequeo de integridad. Esto no solo evita cualquier posibilidad de falsos positivos, sino que si el almacenamiento también se aísla de la producción, el chequeo de integridad se puede ejecutar para utilizar completamente el ancho de banda del almacenamiento y completarse mucho más rápido. Este enfoque encaja bien en el modelo donde el chequeo de integridad se utiliza para validar copias de seguridad; una copia de seguridad validada ratifica de forma efectiva la producción, desde el momento en que se hizo la copia de seguridad. Las plataformas en la nube y de virtualización también pueden facilitar el establecimiento de un entorno aislado utilizable a partir de un snapshot.

*La interfaz del Portal de Gestión, la tarea de *Chequeo de integridad* y el método IntegrityCheck de SYS.Database, seleccionan un número bastante grande de procesos (igual al número de núcleos de la CPU), sin que exista un control que puede ser necesario en muchas situaciones. El Portal de Gestión y la tarea también hacen un doble chequeo de cualquier *global* que haya informado de un error, en un intento por identificar falsos positivos, que pueden ser debidos a actualizaciones simultáneas. Este nuevo chequeo va más allá de la mitigación de falsos positivos incorporada en los algoritmos de verificación de integridad, y puede ser molesta en algunas situaciones, debido al tiempo adicional que requiere (el nuevo chequeo se ejecuta en un solo proceso y verifica todo el *global*). Este comportamiento se podrá modificar en el futuro.

#Administración del sistema #Backup #Mejores prácticas #Rendimiento #Caché #InterSystems IRIS

URL de fuente:https://es.community.intersystems.com/post/chequeando-la-integridad-de-datos-acelerarla-o-ralentizarla