Artículo
Jose-Tomas Salvador · 3 sep, 2019 Lectura de 4 min

Evaluación de Sharding #1

¡Hola Comunidad!

Con IRIS nos llegó una INTERESANTÍSIMA funcionalidad - **¡SHARDING!** Sin duda una gran característica.

Pero, ¿cómo puedo descubrir si encaja con mis aplicaciones actuales? ¿Hay alguna funcionalidad práctica para apostar por ello en mi perfecta aplicación transaccional? ¿O es sólo algo para las nuevas aplicaciones que vaya a diseñar?

Hice, y todavía hago, una serie de investigaciones que me gustaría compartir contigo:

Primero:
Implica moverse de un entorno de (prácticamente) un único servidor a un entorno con múltiples servidores sincronizados y limitados en prestaciones.
Esto implica un mayor, y especialmente más cualificado, control sobre tus instancias. No es necesariamente un gran problema pero sí algo que debe considerarse por planificación de recursos. Y por supuesto también para cualquier gestión de alta disponibilidad, desde *backups* a sistemas de *failover* con *mirrors*.

Segundo:
El *sharding* sólo es útil si las tablas en que piensas utilizarlo tienen un tamaño relevante que permita obtener un beneficio del procesamiento paralelo que compense del sobrecoste derivado de las comunicaciones ECP entre servidores. **^GSIZE** es un buen punto de partida para ayudarte a decidir esto.

Y por último, debes saber si tus *result sets* son lo suficientemente grandes para apreciar mejoras entre una alternativa u otra. Buscar sólo unos pocos registros es más bien un ejercicio para aplicar o no un índice apropiado. Por supuesto puedes hacerlo con *sharding* también, pero puede ser difícil identificar y presentar la ganancia de utilizar *sharding* a tu dirección.

Basándome en estas ideas,seleccioné una tabla de 26,5 millones de registros que típicamente tenía *resultsets* de 1000 o 10000 entradas. Con esta tabla, seleccioné un conjunto significativo de consultas para comparar el rendimiento en un único servidor contra un cluster de 4 servidores *Sharding* con su correspondiente *master*.

Aunque el entorno de producción corre en Windows, para los 5 servidores de test decidí utilizar Ubuntu que tiene una huella menor.

Por otro lado, ya que no disponía de ninguna LAN aislada para las conexiones ECP, ejecuté la mayoría de los tests fuera de las horas de trabajo para evitar interferencias con otro tráfico de red.

Sorpresa #1:
El pasar de Windows (12 GB buffer 8 cores) a Linux (400MB buffer 2 cores) resultó en una reducción en el tiempo de ejecución, llegando al 50%, e incluso menor, dado que se trataba de un máquina significativamente menor [Corrección: 75% - ver comentario en artículo original].

Esperaba que el Linux fuera más rápido, pero la magnitud que vi fue increíble (y la verifiqué varias veces). A tener en cuenta también que utilicé una instalación de Linux *out-of-the-box* ya que no soy un experto en Linux.

La siguiente fase fue distribuir los datos en 4 *shards*. Decidí utilizar la *shard key* por defecto. La carga llevó algún tiempo, pero finalizó a lo largo de la noche.

Sorpresa #2:

El sistema en *sharding* fue claramente más rápido que en el servidor único (como esperaba). El tiempo de ejecución para consultas simples cayó al 40-80% comparado con la ejecución en el servidor único.

Menos atractivo: una consulta compleja con varios *joins* tardó en el sistema con *sharding* casi el doble que en el sistema de servidor único (la sorpresa).

Así que volví a la documentación Choose a Shard Key + Evaluate Unique Constraints y descubrí que mi *lazy join* trabajaba sobre una restricción de tipo UNIQUE y no estaba relacionada con la *shard key*!! Manos a la obra, reconstruí la tabla en *shard* de nuevo, utilizando esta vez la parte principal de la restricción UNIQUE como *shard key*.

Sorpresa #3:
- El tiempo de carga se redujo al **60%**
- Las consultas simples se redujeron entre el **15-25%** del tiempo en el servidor único
- La consulta con *joins* engorrosamente compleja ahora terminó su ejecución en el ~**50%** del tiempo que tardó en el servidor único.

Mi aprendizaje personal:
- Hay situaciones donde leer *y* comprender la documentación previamente nos lleva al éxito más rápido.
- Los valores por defecto no son siempre la mejor solución.

Resumen:

- sé consciente del esfuerzo de mantener servidores adicionales
- selecciona cuidadosamente si tu tabla se ajusta a la filosofía de *sharding* en tamaño y estructura
- piensa dos veces al seleccionar tu *shard key*
- ejecuta tests serios para alinear expectativas con realidad

Mis siguientes pasos serán evaluar el impacto del número de shards. Comenzaré con el máximo disponible y, después, comprobaré cómo funciona con 2 y 3 shards en uso.

Os tendré informados cuando obtenga números significativos.

Evaluación de Sharding #2

#Sharding #InterSystems IRIS