Resumen 2

José David Mamani Vilca

I. Entendiendo tu servidor

Entender un servidor de manera más profunda e íntima en donde se busca analizar las capacidades y todas las posibles tareas que se pueda realizar con este implica pasar por un proceso súmamente peligroso. *Benchmarking* o lo que es lo mismo, ubicar puntos de referencias para los análisis que realizaremos. Debe considerarse que si bien este paso es muy importante para ubicar las capacidades del hardware que poseemos y como se desenvuelve este frente a los servidores que posee la competencia, es también un proceso con inclinación a botar errores en caso los *tests* de evaluación hayan sido concretados de manera incorrecta. Debido a esto, se requiere de cierta actitud detectivezca para poder averiguar más del funcionamiento interno del sistema.

Revisada la importancia del Benchmarking podemos pasar al caso de **pgbench** una vieja herramienta incluida en PostgreSQL que tiene por finalidad reflejar cual es el estado del servidor. Con **pgbench** tenemos la capacidad de crear pruebas de estrés en base de datos relativamente gigantescas. De hecho, realizar este tipo de pruebas es una actividad que el mismo encargado de PostgreSQL recomienda hacer cada cierto tiempo y que, además, cada una de estas pruebas debe de ser lo suficentemente extensa (bajo parámetros de tiempo) como para validar la factibilidad del sistema. Benchmarking permite por ejemplo verificar la actividad de los *cores* de nuestro procesador, o de saber si el trabajo se reparte equitativamente entre estos, también permite saber la distribución de los datos y como hay información que se guarda en disco e información que se almacena en caché. De hecho, datos como el número de transacciones por minuto, la cantidad de clientes escribiendo al mismo tiempo y la velocidad con la que la información se guarda es posible de extraer.

Por otra lado y bajo los mismo experimentos, realizar Benchmarking en una máquina más poderosa deberá reflejar resultados que indiquen que aspectos tanto del software como del hardware son mejores (o peores en ciertos casos). El uso de un procesador más poderoso, de discos duros en estado sólido o incluso la disponibilidad mayor de RAM son factores que tienden a repercutir en los resultados del Benchmarking y verificar cuando estos no funcionan apropiadamente tiende a ser una posible causa de fallas posteriores.

II. Adaptando tu servidor

Se necesita resaltar que una vez instalado PostgreSQL, este es incapaz de saber cuál es el tipo de máquina en donde se está ejecutando. Debido a esto, se hace necesario el uso de configuraciones iniciales que vienen por defecto una vez iniciado el sistema. Aunque esta configuraciones no sean óptimas en la mayoría de los casos, si son fáciles de configurar.

Una de las primeras configuraciones a realizar está relacionada con el tamaño de la caché de los datos. Si bien este paso puede realizarse de manera manual, se recomienda por lo general dejar que el sistema escoja el tamaño más óptimo. Visto esto, pasamos a ubicar un documento muy importante dentro de la base de datos, el archivo de configuración. Generalmente, es posible ubicarlo en /utc/PostgreSQL -version aunque el algunos casos, y dependiendo del ambiente de instalación, esta direccion podría variar. Al abrir dicho documento es posible figurar una serie de datos que representan los estados iniciales con los que la base de datos inicia. Algunas de las configuraciones más importantes a realizar son por ejemplo, el número de usuarios conectados de forma simultánea. De forma estándar, se permite que un máximo de 100 usuarios estén conectados simultáneamiente y aunque este valor puede configurarse, se recomienda que su valor no supere los 400 (considerar que cada conexión representa un segmento de memoria RAM

cedido al nuevo usuario). Otra configuración a ver es el tamaño de los *buffers* compartidos. Suele ser común pensar, que mientras más grande sea el tamaño de este *buffer* mayor será la eficacia de la base de datos. Sin embargo, esto no es verdad. PostgreSQL es lo sufucientemente rápido con la cantidad de memoria que dispone y en algunos casos puede verse como un desperdicio el incrementar el tamaño.

Volviendo al caso de la memoria RAM, se ve que por lo general se suele ceder hasta el 75 por ciento de la misma cuando en realidad lo que necesita, es ubicar un cantidad superior al promedio de conexiones simultánea y acorde a este número ubicar la cantidad de RAM correspondiente (Notar que por lo general cada nueva conexión consume aproximadamente 42 mb).

Finalmente, y como otro buen punto a considerar, tenemos los *Wal Buffers* de los únicamente podría mencionar, a más mejor. Es fácil de constatar como el llevar esta memoria a sus límites mejora de forma considerable el funcionamiento del sistema.

III. Hagamos que PostgreSQL te diga que va mal...

En este módulo, vamos a analizar las consultas que se ejecutan en nuestra máquina utilizando un paquete de seguimiento de estadísticas que se envía con Postgres. En lugar de adivinar qué consultas se ejecutan más y cuáles se ejecutan más lentamente, vamos a hacer que Postgres nos diga esto.

III-A. Instalando pg-stat-statements

Es necesario destacar que este complemento no viene incorporado en el sistema, por lo que será necesario instalarlo. *Pg-stat-statements* proporciona un medio para rastrear estadísticas de ejecución de todas las *query* SQL ejecutadas por un servido, además permite optimizar algunas de las consultas realizadas en tabla al modificar el *query* enviado en uno que resulte ser mucho más amigable. Sin embargo, esto conlleva el riesgo de consumir más memoria, más recursos y por ende más tiempo, por lo que dependerá del usuario el querer utilizarlo o no.

III-B. Uso del Caché

Probablemente, el mayor aumento en el rendimiento de Postgres deriva de usar el caché para leer datos de la base de datos. Si se tiene que ir al disco y filtrar los datos, se consume más tiempo. Pero si esos datos están en la memoria, obviamente serán mucho más rápidos. Asegurarse de poseer un caché lo suficientemente grande1 permite garantizar un mejor rendimiento en cuestiones de velocidad y rendimiento.

III-C. Uso de Indices

Lla desnormalización puede ser tu amiga, y verás esto en bases de datos de súper alta capacidad. Denormalizar implica que podría ejecutar consultas de agregación de forma más rápida. Supongamos que desea saber la cantidad total de votos para una publicación determinada. Comentarios totales, vistas totales y demás, puede ejecutar esas consultas de agregación, pero si lo piensa, esas uniones son costosas, ralentizan la consulta. Entonces, lo que harán muchos DBA es que simplemente lo desnormalizarán, y se asegurarán de que las escrituras sucedan de tal manera que puedan actualizar y controlar esos conteos. Echando un vistazo a las tablas de los ejemplos se puede ver comentarios, tipos de publicaciones, publicaciones, etiquetas de publicaciones, etiquetas, una base de datos básica típica. Pero si echamos un vistazo a las publicaciones, y solo usamos un número determinado de publicaciones,se puede ver que las cosas normalizadas están arriba, como una unión autorreferenciada en la tabla. Pero a medida que se baja y se ve el conteo de vistas, conteo de respuestas, conteo de comentarios, conteo de favoritos, etc. se ve un desbordamiento de pila, una desnormalización en la base de datos. En ese caso, en realidad sepueden indexar varias de esas columnas, por lo que cuando ejecutan consultas sobre ellas se puede obtener una mayor velocidad en la base de datos.