Materia Bases de datos avanzadas

PROFESOR: WILLIAM RUIZ

Universidad Iberoamericana

Ingeniería en Ciencia de Datos

Actividad 3 - Conceptos y Comandos básicos del particionamiento en bases de datos NoSQL

Nelson Fernando Nopssa Castro

ID 100169351

Tabla de Contenido

Introducción

Este documento tiene como objetivo especificar los requerimientos no funcionales necesarios para implementar el particionamiento (Sharding) en la base de datos de Fórmula 1 utilizando MongoDB. El Sharding es una estrategia de particionamiento de datos que se utiliza para manejar grandes volúmenes de datos distribuyéndolos en múltiples servidores, mejorando la escalabilidad, el rendimiento y la disponibilidad del sistema.

Descripción del Escenario

La base de datos de la Fórmula 1 almacena información crítica sobre los circuitos, los ganadores, los tiempos y las fechas de las carreras. A medida que aumenta la cantidad de datos debido a la incorporación de nuevas temporadas y carreras, el sistema actual ha empezado a mostrar problemas de rendimiento y capacidad.

Para asegurar que la base de datos pueda manejar este crecimiento continuo, se requiere implementar una solución de Sharding en MongoDB que permita distribuir los datos en múltiples servidores. Esto garantizará que el sistema pueda escalar horizontalmente, manteniendo un rendimiento óptimo y una alta disponibilidad.

Requerimientos No Funcionales

3.1 Escalabilidad

Objetivo: Garantizar que la base de datos pueda escalar horizontalmente al agregar más nodos al clúster de Sharding para manejar el creciente volumen de datos y tráfico.

Implementación:

Configuración de Shards: Implementar Sharding con múltiples shards para distribuir los datos.

Autoescalado: Configurar mecanismos para agregar nodos adicionales de manera automática o manual según sea necesario.

Hash-Based Sharding: Utilizar Sharding basado en hash para distribuir uniformemente los datos sobre los circuitos y carreras.

3.2 Balanceo de Carga

Objetivo: Distribuir la carga de trabajo de manera equitativa entre los nodos del clúster de Sharding para evitar cuellos de botella y mejorar el rendimiento.

Implementación:

Mongos: Configurar routers mongos para gestionar las operaciones de consulta y escritura, distribuyendo las solicitudes entre los shards.

Balanceador de Shards: Utilizar el balanceador interno de MongoDB para redistribuir datos de forma automática cuando un shard se vuelve demasiado grande.

3.3 Disponibilidad y Tolerancia a Fallos

Objetivo: Asegurar que la base de datos permanezca disponible y operativa incluso en caso de fallo de uno o más nodos.

Implementación:

Replicación de Shards: Cada shard debe ser un clúster de réplica para proporcionar redundancia y alta disponibilidad.

Failover Automático: Configurar failover automático para que, en caso de fallo de un nodo, otro nodo secundario sea promovido a primario sin interrupción del servicio.

Monitoreo Continuo: Implementar monitoreo continuo para detectar y manejar fallos rápidamente.

3.4 Rendimiento

Objetivo: Mantener un rendimiento óptimo de la base de datos, asegurando tiempos de respuesta rápidos para operaciones de lectura y escritura.

Implementación:

Índices Eficientes: Crear índices adecuados para optimizar las consultas y mejorar los tiempos de respuesta.

Caching: Implementar mecanismos de caching para reducir la carga en los nodos de datos.

Configuración de Chunk Size: Ajustar el tamaño de los chunks para equilibrar la carga entre los shards.

3.5 Mantenimiento y Monitoreo

Objetivo: Facilitar el mantenimiento y la monitorización de la base de datos para asegurar su correcto funcionamiento y detectar problemas antes de que afecten a los usuarios.

Implementación:

Herramientas de Monitorización: Utilizar herramientas como MongoDB Atlas, Prometheus y Grafana para supervisar la salud y el rendimiento del clúster.

Alertas y Notificaciones: Configurar alertas para notificar a los administradores en caso de problemas como uso elevado de CPU, fallos en la replicación o indisponibilidad de nodos.

Actualizaciones y Mantenimiento: Planificar ventanas de mantenimiento y realizar actualizaciones de software y hardware de manera escalonada para minimizar el impacto en el servicio.

Conclusión

Implementar Sharding en la base de datos de la Fórmula 1 permitirá manejar eficientemente el creciente volumen de datos y tráfico. Los requerimientos no funcionales especificados en este documento aseguran que el sistema será escalable, equilibrado, altamente disponible, con buen rendimiento y fácil de mantener y monitorear.

Referencias Bibliográficas

MongoDB Documentation. Sharding. https://docs.mongodb.com/manual/sharding/

MongoDB Atlas. https://www.mongodb.com/cloud/atlas

Prometheus - Monitoring system & time series database. https://prometheus.io/

Grafana - The open observability platform. https://grafana.com/