Key-value store distribuído

*TP Grupal -IASC 1C2016*

Contexto

El objetivo es diseñar e implementar una base de datos clave/valor simple, no persistente, distribuida, que tenga un grado razonable de tolerancia ante ciertos fallos. El objetivo de diseñar e implementar dicha base de datos es para que el grupo pueda aplicar los conocimientos vistos durante la cursada.

Consignas

El Trabajo práctico se debe realizar entre un máximo de 4-5 personas, y es condición presentar dicho trabajo práctico para aprobar la cursada de la materia. Una vez presentado el trabajo el grupo podrá rendir el final donde se tomará un examen oral sobre el mismo.

La tecnología sobre la que se debe implementar dicho trabajo práctico es libre, es decir que se puede implementar en cualquier lenguaje que el grupo considere apropiado siempre y cuando se cumplan los requerimientos solicitados, que son los que están mencionados en la próxima sección, y son todos obligatorios a menos que indique explícitamente que el requerimiento es opcional.

Con respecto a la resolución, el trabajo práctico debe estar en un repositorio, puede ser git o de otro tipo, y debe enviarse a alguno de los docentes de la materia, los integrantes del grupo, junto con el repositorio en cuestión (puede estar vacío en un comienzo, pero debería tener que tener eventualmente la resolución del trabajo práctico).

Requerimientos

Nuestra base de de datos debe cumplir los siguientes puntos:

* Debe permitir insertar y quitar pares clave-valor
* Tanto las claves como los valores serán strings
* Tanto las claves como los valores tienen un tamaño máximo, configurable.
* El sistema permite realizar consultas de la forma:
  + Obtener el valor de la clave K
  + Obtener todos los valores mayores/menores a X
* Se debe poder manipular una cantidad de datos mayor a la almacenable en un único nodo, distribuyendolos entre varios nodos
* Lo *clientes* de la base de datos son configurados con una lista de IPs de uno o más nodos *orquestadores*, que se encargan de aceptar pedidos de escrituras y lecturas por parte de los clientes. No todos los clientes tienen que tener la misma lista en el mismo orden.
* En todo momento, hay un sólo *orquestador* activo, que se denomina *master.* Todos los clientes hablan con este nodo.
* Si un nodo *master* se cae, otro nodo *orquestador* debe tomar su lugar y volverse *master*
* Al establecer la primera conexión o ante un error de comunicación, un *cliente* probará con todos los *orquestadores* de su lista hasta encontrar el *master* actual.
* La interfaz expuesta por el orquestador debe ser simple: con un API rest y/o CLI alcanza.
* Además de nodos orquestadores, hay nodos de *datos,* que son los que realmente contienen los pares clave-valor.
* Un par puede vivir en un único nodo. **Opcional:** permitir réplicas de los datos.
* Las cantidad de nodos de *datos y orquestación* y sus direcciones pueden estar fijas. **Opcional**: permitir agregar y nodos dinámicamente
* Todos los nodos de *datos* tienen la misma *capacidad máxima*, configurable al iniciarlos. Es decir, que si la *capacidad máxima* es N, y tenemos M nodos de *datos*, la base puede almacenar hasta N\*M claves.
* Cuando un *orquestador* recibe un pedido de escritura, lo transfiere al nodo de *datos* correspondiente. La estrategia para hacer esto no está especificada: se puede utilizar, por ejemplo, un simple round robin o distribución consistente por hash.
* El nodo orquestador debe esperar a que el nodo de datos elegido para escribir confirme la escritura, y el cliente debe esperar al orquestador. Si esta escritura falla o la comunicación falla, la operación toda falla. La operación debe ser atómica: no debe suceder, por ejemplo, que el nodo de escritura efectivamente escriba, pero el orquestador informe lo contrario. **Opcional:** ofrecer una operación adicional de escritura no segura, pero más rápida y asincrónica.
* No es importante los datos sean persistentes: si el nodo de datos correspondiente se cae, esta información simplemente se pierde.
* El sistema debe hacer su mejor esfuerzo para que los nodos de datos tengan una carga similar de datos, pero no es un requerimiento fuerte.