SISTEMAS DE BASES DE DATOS
KEY-VALUE STORES
Prof. María Constanza Pabón

Contenido
• Características comunes de los Key-value stores
• Riak

KV Stores: Objetivo
Asegurar la disponibilidad de los datos frente a fallos del sistema o de la red, aprovechando la mas pequeña posibilidad de proveer el servicio, aun cuando partes del conjunto de datos falten temporalmente

Modelo de datos
Datos almacenado en una tabla hash simple (map)
• Todas las operaciones acceden usando la llave primaria
• Almacena objetos que se componen de (KEY, DATA)
• En el mundo RDBMS: un esquema con una única tabla de dos columnas
• KEY (primary key)
• DATA (BLOB)

Operaciones
Put (key, data)
• Almacenar un valor para una llave
Get (key)
• Recuperar el valor asociado a una llave
Delete (key)
• Borrar la llave y su valo

Key
• La definición de las llaves es vital
• No debería ser solamente un número secuencial
• Lo más eficiente es recuperar los datos por la llave
• Con la llave se define la asignación del dato a un nodo

¿Cómo asignar la llave?
• Dato que provee el usuario (llave natural): userId, e-mail, …
Ejemplos
• Carrito de compras: userID
• Datos de sesiones: sessionID
• Perfiles de usuario: userID
• Generada por algún algoritmo
• Derivada de timestamps u otros datos
• Las llaves pueden expirar
• Ejemplos: llaves de sesiones, carritos de compra

Esquema de los datos
• No se define el esquema de la BD
• Usualmente se da alguna estructura a los objetos. Ej. JSON
• La estructura la maneja la aplicación
• La aplicación maneja la integridad de los datos

• Usualmente la aplicación maneja la resolución de conflictos de lectura
• Es usual tener datos redundantes

Consultas
• La única condición de las consulta es la llave
• No es posible recuperar el valor de un atributo particular
• KV stores no conocen el contenido del “valor”, no conocen el esquema
• Si no conocemos la llave: Es posible obtener la lista de las llaves
• Si se quiere acceder por atributos diferentes a la llave
• En algunos sistemas se pueden usar índices adicionales
• Ejemplo: Full text search (Riak search)
• En algunos motores el índice se mantiene desde la aplicación

Consistencia BASE
• Basically Available
• Soft state
• Eventually consistent

RIAK-KV: Información básica
• Desarrollado por Basho (http://basho.com/)
• Herramienta open source
• Se liberó en 2009
• Licencias: Apache 2 y comercial
• Desarrollado en Erlang
• Erlang es fuerte en manejo de concurrencia, comunicación distribuida, actualización dinámica del software, y tolerancia a fallos.
• Partes en C, C++, JavaScript
• Sistemas operativos: Linux, BSD, Mac OS X, Solaris
• Back-ends: LevelDB, memory, Bitcask (default)

Riak-KV: Buckets
• Agrupa objetos Riak
• Pueden entenderse cómo logical namespaces
• Unicidad de la llave a nivel de bucket (la combinación bucket-key es única)
• La misma llave puede existir en varios buckets
• Todo objeto que se guarde pertenece a un bucket
• Cada bucket tiene una configuración independiente de los otros
• Ej. Propiedades de replicación

Riak-KV: Search
• Integra Solr (para indexamiento y consultas) con Riak (almacenamiento y distribución)
• Solr indexa campos (fields)
• Extractores: toman el valor y lo convierten en una lista de campos que puede ser indexada por Solr
• JSON, XML, texto plano, counter, map, set

RIAK operaciones: PUT / GET
• Parámetros de escritura y lectura según quorum

RIAK: Storage backends
• Bitcask
• LevelDB

• In Memory
• Múltiple backends
• Custom backend API

Referencias

• Basho. Riak KV Basho Docs. En línea. http://docs.basho.com/riak/kv/2.2.3/
• DeCandia et al. Dynamo: Amazon’s Highly Avalaible Key-value Store. SOSP, 2007
• Fowler, M. y Sadalage, P. NoSQL distilled. Addison-Wesley, 2013
• Redmond, E. y Daily, J. A Little Riak Book
• Celko, J. Joe Celko’s Complete guide to NoSQL. Morgan Kaufmann, 2017