Revista Tecnológica ESPOL – RTE, Vol. 28, N. 5, 333-345, (Diciembre 2015)

YCSB-replay: Framework de evaluación realista de bases de datos de clave-valor

Edwin F. Boza, Cristina L. Abad

Facultad de Ingeniería en Electricidad y Computación (FIEC)
Escuela Superior Politécnica del Litoral (ESPOL)
Campus Gustavo Galindo, Km 30.5 vía Perimetral, Guayaquil-Ecuador
eboza@fiec.espol.edu.ec, cabad@fiec.espol.edu.ec

Resumen. En la era de los Grandes Datos, las empresas de servicios de Internet necesitan construir sistemas de alto rendimiento para atender a sus usuarios. Uno de los componentes importantes de dichos sistemas son las caches distribuidas, típicamente implementadas con bases de datos clave-valor como Redis y Memcached.

Un paso importante en la implementación de estos sistemas es el dimensionamiento de sus componentes, realizado a través de estudios previos usando herramientas de evaluación de rendimiento (benchmarks). En la actualidad, el Yahoo Cloud Serving Benchmark (YCSB) es la herramienta más utilizada para la evaluación de las bases de datos clave-valor. Lastimosamente, YCSB solamente puede generar cargas de trabajo sintéticas, basadas en distribuciones probabilísticas configurables que pueden reproducir la popularidad de accesos reales observados, más no su localidad temporal. En este trabajo, se propone *YCSB-replay*, una mejora a YCSB a través de la cual se adiciona a este framework la habilidad para generar cargas de trabajo realistas, basadas en la repetición de trazas de sistemas reales en producción.

Para demostrar la utilidad de las mejoras propuestas, se ha evaluado YCSB-replay a través de un estudio del rendimiento de la base de datos clave-valor Redis, en su función como cache de datos, utilizando una carga de trabajo real de videos de YouTube descargados desde la Universidad de Massachusetts. Esta evaluación demuestra que las cargas sintéticas generadas por YCSB no permiten aproximar los resultados obtenidos con cargas reales (YouTube), por lo que su uso es inadecuado para el dimensionamiento de bases de datos clave-valor. En cambio, YCSB-replay permite obtener una evaluación realista de la base de datos, manteniendo el alto rendimiento en la generación de cargas de YCSB.

Palabras clave: rendimiento, YCSB, clave-valor, cache, trazas, cargas de trabajo, Redis.

1 Introducción

En la era de los Grandes Datos, las empresas de servicios de Internet necesitan construir sistemas de alto rendimiento para atender a sus usuarios. Estos sistemas incluyen subcomponentes distribuidos, como sistemas de archivos, caches, sistemas de procesamiento de datos, entre otros. Este trabajo este enfocado en uno de estos subcomponentes: las caches distribuidas.

Una cache distribuida es una cache que abarca múltiples servidores de tal manera que pueda crecer y así poder aumentar su capacidad transaccional. Su uso busca reducir la latencia de acceso a datos remotos, mejorando así los tiempos de respuesta y la escalabilidad del sistema. Estas caches están típicamente implementadas con bases de

datos clave-valor, las cuales son un tipo de base de datos NoSQL. De las bases de datos de clave-valor existentes, la más popular es Redis [4], usada por empresas como Twitter, GitHub, Pinterest, Flickr, entre otras [9].

Un paso importante en la implementación de estos sistemas es el dimensionamiento de sus componentes, realizado a través de estudios previos basados en cargas de trabajo realistas. En la actualidad, el Yahoo Cloud Serving Benchmark (YCSB) [3] es la herramienta más utilizada para la evaluación de las bases de datos clave-valor, llegando a convertirse en el estándar de facto de la industria, especialmente en entornos de almacenamiento en la nube. Su popularidad esta dada en parte por su arquitectura modular que permite extender el funcionamiento para incluir compatibilidad con nuevas bases de datos, así como también realizar variantes en la generación de las cargas de trabajo sintéticas.

Lastimosamente, YCSB solamente puede generar cargas de trabajo sintéticas, basadas en distribuciones probabilísticas configurables (Uniforme, Latest y Zipf) que pueden reproducir la popularidad de accesos reales observados, mas no su localidad temporal.

En este trabajo se presenta *YCSB-replay*, una mejora a YCSB a través de la cual se adiciona a este framework la habilidad para generar cargas de trabajo realistas, basadas en la repetición de trazas de sistemas reales en producción.

YCSB-replay ha sido validado a través de varios experimentos, en los que: (1) se verifica que la generación basada en trazas no degrada el rendimiento de YCSB y (2) se muestra que existen diferencias de hasta 31% en el rendimiento de la cache cuando es evaluada con cargas de trabajo reales versus cargas sintéticas, demostrando así que el uso de YCSB-replay permite realizar un dimensionamiento más efectivo de los componentes.

El resto del presente trabajo se encuentra organizado de la siguiente manera. En la Sección 2, se plantea el problema de las limitantes de YCSB, especialmente las relacionadas con la utilización de cargas de trabajo aleatorias. En las Secciones 3 y 4 se detallan y evalúan, respectivamente, los cambios realizados al framework YCSB. Las secciones finales contienen un análisis de los trabajos previos (Sección 5) y una exposición de las conclusiones obtenidas a partir de la realización del trabajo, así como una mención a las siguientes etapas de la investigación (Sección 6).

2 Antecedentes y descripción del problema

Como se explicó en la sección anterior, la evaluación de sistemas distribuidos bajo cargas de trabajo realistas es importante para el dimensionamiento correcto de dichos sistemas antes de su puesta en producción, de tal manera que se pueda alcanzar las metas de rendimiento planteadas.

El presente artículo aborda el problema de la evaluación de bases de datos de clave valor bajo cargas de trabajo realistas. Específicamente, se considera el caso de YCSB ya que es la herramienta más utilizada para la evaluación de dichas bases de datos[1].

[1] Por ejemplo, al 5 de agosto de 2015 registra 873 citaciones: `https://goo.gl/LMeHYL`.

El *YCSB CoreWorkload Package* es el paquete de cargas de trabajo que viene incluido en YCSB; esta compuesto por un conjunto de generadores de cargas de trabajo sintéticas adecuadas para microbenchmarks. El usuario puede configurar el porcentaje de lecturas y escrituras, así como la distribución aleatoria utilizada para muestrear los accesos a las claves (Uniforme, Latest y Zipf) [3].

El muestrear los accesos a las claves de una distribución de probabilidad es equivalente a usar el *Modelo de Referencias Independientes* (IRM, por sus siglas en ingles), asumiendo una cierta distribución de popularidad de las claves. Lastimosamente, el IRM reproduce la popularidad de los objetos mas no logra capturar otras dimensiones de la carga de trabajo como la localidad temporal[2]. Como ya se ha demostrado reiteradamente [1, 5, 13], la localidad temporal incide considerablemente en el rendimiento de las caches de todo tipo. Por esta razón, una evaluación basada en cargas de trabajos que no reproducen esta dimensión de la carga, resulta inadecuada y puede significar perdidas de dinero por sobredimensionamiento de la capacidad requerida, o inhabilidad para alcanzar metas de rendimiento planteadas debido a una sobrestimación del rendimiento.

Afortunadamente, las limitaciones de YCSB pueden ser superadas gracias a la disponibilidad de su código fuente. Hasta la actualidad, 630 proyectos han sido derivados de su repositorio en GitHub[3], lo que indica que su extensibilidad ha sido aprovechada por otros investigadores para agregar funcionalidades como la inclusión de tiempos de interarribo en las cargas de trabajo [8], la adición de un coordinador de ejecución de múltiples instancias del cliente YCSB [7], la variación de la popularidad de los objetos mediante el cambio de la inclinación de la distribución de frecuencia [16] o el registro de trazas de las cargas de trabajo generadas para reproducción posterior [14].

Sin embargo, de lo que se conoce hasta la redacción de este artículo, no existe disponible una extensión de YCSB que permita utilizar trazas[4] para la generación de la carga de trabajo. La reproducción de trazas permite el uso de cargas de trabajo reales, lo cual continúa siendo solicitando por la industria [10] debido a las ya mencionadas limitaciones de caracterización y generación sintética de cargas de trabajo. Además, la reproducción de trazas puede ser utilizada por investigadores de caracterización de cargas de trabajo, para la evaluación de sistemas con sus nuevos modelos, sin necesidad de enfocarse en el desarrollo y mantenimiento de la herramienta de benchmarking.

[2] La localidad de la referencia o localidad temporal de los accesos indica que los objetos recientemente accesados tienen mayor probabilidad de ser referenciados en el futuro cercano.

[3] Repositorio de codigo de YCSB: `https://github.com/brianfrankcooper/YCSB`.

[4] Las trazas (“traces” in ingles) son archivos que contienen una captura de una carga de trabajo observada en un sistema en produccion. Dichas trazas pueden ser obtenidas a traves de instrumentacion especial en el sistema o al procesar logs o bitacoras del sistema.

3 Implementación

En esta sección se describe la infraestructura de YCSB y los cambios realizados para implementar la reproducción de cargas de trabajo realistas basada en trazas obtenidas de sistemas en producción. YCSB-replay ha sido publicado en línea como código abierto para que pueda ser aprovechado por otros investigadores y profesionales[5].

3.1 Infraestructura de YCSB

En YCSB una carga de trabajo esta definida por dos componentes, (1) un conjunto de datos que son los registros a ser cargados en la base de datos y (2) el conjunto de operaciones que se ejecutara sobre esos datos. A su vez, el conjunto de datos está definido por un campo "clave" que identifica al registro y un grupo de campos que representa la información que debe ser almacenada en la base de datos evaluada.

En el *CoreWorkload Package* la generación del conjunto de datos para la carga de trabajo se realiza en dos etapas independientes. Primero, se genera la clave del registro a partir de una distribución de frecuencias seleccionada (Zipfian, Latest o Uniform). Segundo, se genera los registros de datos, cuyo número tamaño puede ser especificado dependiendo de la evaluación a realizar. La generación del conjunto de operaciones se realiza de manera aleatoria, pero respetando una proporción entre lecturas y escrituras que ha sido definida por el usuario.

Los para 'metros para la generación de la carga de trabajo, sean estos la distribución utilizada para la generación de las claves, el número y tamaño de los campos de cada registro, el número de operaciones y la proporción de cada tipo de operación se especifican mediante un archivo de configuración. Aquí también se debe especificar la clase de cargas de trabajo que será utilizada.

Para la generación de nuevos modelos de cargas de trabajo se dispone de dos métodos: utilizar un archivo de configuración para variar los para 'metros de los modelos incluidos o implementar una nueva clase en Java que se encargue de la generación de los registros y transacciones. Este último enfoque proporciona más flexibilidad, permitiendo implementar cualquier carga de trabajo arbitraria. Dicha clase debe extender la clase abstracta `com.yahoo.ycsb.Workload`.

3.2 YCSB-replay

Para agregar la capacidad de reproducción de trazas en YCSB, se optó por el método de proveer una nueva clase en Java, llamada *ReplayWorkload*, que pueda coexistir con el `CoreWorkload` original; además, se reutilizo una parte del código fuente de la clase `CoreWorkload`, específicamente, aquella correspondiente a la generación de los registros de datos. Para obtener la clave del registro y las operaciones a realizar, la clase `ReplayWorkload` lee la traza con la carga de trabajo a ser reproducida. La Figura 1 muestra el diseño de YCSB-replay.

[5] `http://github.com/ebozag/YCSB-replay`.

La traza a ser reproducida es un archivo de texto que debe tener el formato "COMANDO,CLAVE". El comando puede ser READ o INSERT, dependiendo de si se desea leer o escribir una clave. En el caso de escrituras, se escribe una cadena aleatoria de bytes, tal y como lo hace YCSB. La Figura 2 muestra las primeras diez líneas de la traza utilizada en los experimentos detallados en la Sección 4.

La selección de la nueva clase se realiza mediante el archivo de configuración, donde además se deberá especificar la ruta y nombre del archivo que contiene las trazas. Es importante notar que la inclusión de este nuevo parámetro no afecta el funcionamiento de la clase original. De esta manera se garantiza que la opción de generación de cargas de trabajo a partir de la reproducción de trazas conviva con las opciones estándar de generación de cargas de trabajo incluidas en YCSB. Esto permite al evaluador elegir

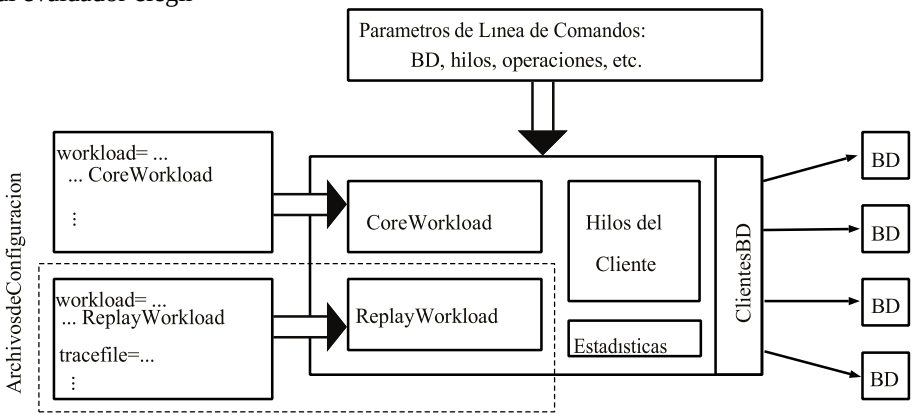

Figura 1: Diseño de YCSB-replay. Lo que esta dentro del recuadro es la funcionalidad añadida; el resto es YCSB sin modificaciones.

```
READ,HSOw4LO05Ws
READ,1bhOE7xcZyg
READ,3TKi92CP-vc
READ,1bhOE7xcZyg
READ,1bhOE7xcZyg
READ,AH47iuMTuC8
READ,Bdc_CWMnPtM
READ,opsMFXnyWag
READ,CTVvok4S_9E
READ,dTzrSDlaJis
```

Figura 2: Ejemplo del formato de la traza utilizado por YCSB. La primera columna detalla la operación sobre la base de datos (leer o escribir una clave). La segunda columna contiene la clave a ser leída o escrita.

entre la reproducción de trazas para evaluaciones bajo cargas de trabajo realistas y la generación aleatoria para microbenchmarks (ej.: probar la tasa máxima de escrituras cuando los accesos a las claves son uniformes).

4 Evaluación

En esta sección se exponen los resultados de la evaluación a YCSB-replay. Primero se describe la carga de trabajo utilizada y luego se presentan los resultados que demuestran la utilidad y rendimiento de YCSB-replay.

4.1 Carga de trabajo utilizada

Los experimentos descritos en esta sección reproducen una carga de trabajo real observada en producción: registros de descargas de videos de YouTube en la Universidad de Massachusetts [17]. En dichas trazas cada línea representa una solicitud de video de YouTube e incluye los siguientes campos: estampa de tiempo, dirección IP del servidor de YouTube, dirección IP del cliente que hizo el requerimiento, tipo de requerimiento, identificador del video y dirección IP del servidor de contenidos.

Para estas pruebas fue seleccionado el archivo de trazas con mayor cantidad de eventos, que corresponde al 29 de enero del 2008, el mismo que contiene 611968 registros. Luego, utilizando un *script de shell*, se separó únicamente las columnas correspondientes al tipo de requerimiento e identificador del video, las cuales fueron almacenadas en el formato requerido por la clase `ReplayWorkload` (ver Figura 2).

La carga de trabajo de descargas de videos de YouTube fue añadida de dos maneras a YCSB: (1) Utilizando las mejoras propuestas a YCSB para realizar una repetición fiel de la carga de trabajo original (“replay”) y (2) también usando la funcionalidad de generar pedidos tipo Zipf de YCSB, modelados a partir de la distribución de RangoFrecuencias de los videos en la traza original.

La Figura 3 muestra la distribución de Rango-Frecuencias observables en la carga de trabajo real y el mejor encaje de la distribución Zipf a dicha distribución (coeficiente de la ley de Zipf = 0,5). Para hallar el mejor encaje de los datos a la distribución Zipf, se utilizó el enfoque de descartar las colas de la distribución [6] y se aplicó el algoritmo publicado en [15].

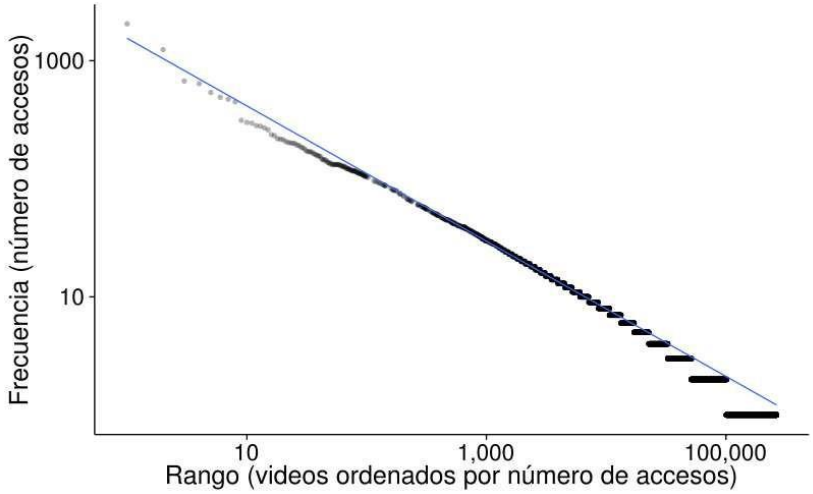

Figura 3: Distribución de rango-frecuencias de los accesos a videos de YouTube en la carga de trabajo real utilizada en este estudio. La línea azul representa el mejor encaje de la distribución Zipf a los datos (coeficiente = 0,5). Cada punto representa un ID de video único observado en la carga de trabajo. Por estética, se ha graficado solamente una muestra aleatoria del 10% de los datos.

4.2 Pruebas de utilidad

Para demostrar la utilidad de YCSB-replay, se diseñaron experimentos que permitan evidenciar que los microbenchmarks de YCSB no proporcionan una aproximación al comportamiento real del sistema y por lo tanto no resultan útiles para el dimensionamiento de sistemas previo a su puesta en producción. Se muestra también como YCSBreplay puede ser utilizado para cubrir esa falencia.

En estos experimentos se evalúa el rendimiento de Redis como cache de datos, para lo cual se utiliza la principal métrica empleada en la evaluación de caches: la tasa de aciertos (“hit rate”) de la cache. Una tasa de aciertos de 0% indica que el contenido buscado nunca se encuentra en la cache. Una tasa de aciertos del 100% indica que el contenido buscado siempre se encuentra en la cache.

Es común utilizar pruebas de cache para dimensionarla misma, ya que una cache más grande permite aumentar la tasa de aciertos (deseable) pero a un mayor costo en la adquisición de memorias RAM (no deseable). Durante el dimensionamiento de una cache, se busca predecir cuál es el tamaño de la cache necesario para alcanzar una cierta tasa de aciertos (por ejemplo, tasa de aciertos del 70%).

En las pruebas de utilidad, se configuro tamaños cada vez mayores de la cache, para determinar la tasa de aciertos esperada. Dada una cantidad fija de claves únicas en todos los experimentos (303332 claves) y el mismo número de transacciones (611968 transacciones) para todas las pruebas, se utilizaron las siguientes cargas de trabajo:

- Latest (YCSB, valores por defecto): Las claves insertadas más recientemente son las más populares. Carga sintética aleatoria que busca añadir algo de localidad temporal. No modelada a partir de cargas de trabajo reales.

- Zipfian (YCSB, valores por defecto: coeficiente = 0,99): Carga sintética aleatoria con accesos modelados de acuerdo a la ley de Zipf en la que unas pocas claves son *muy* populares y hay una larga cola de claves impopulares. No modelada a partir de cargas de trabajo reales.

- Uniform (YCSB, valores por defecto): Carga sintética aleatoria con accesos que siguen una distribución uniforme; todas las claves son igual de populares. No modelada a partir de cargas de trabajo reales.

- Zipfian modificada (YCSB, coeficiente = 0,5): Carga sintética aleatoria con accesos modelados de acuerdo a la ley de Zipf. Modelada a partir de la misma carga de trabajo utilizada en YCSB-replay (ver siguiente ítem); sin embargo, al utilizar el modelo de referencias independiente (IRM), no captura la localidad temporal de los accesos presente en la carga de trabajo original.

- YouTube (YCSB-replay, con traza de descargas de YouTube): Es una reproducción fiel de una carga de trabajo real.

La Figura 4, muestra la comparación de las tasas de acierto entre las cargas de trabajo descritas anteriormente. Se puede observar que ninguna de las cargas sintéticas (ni siquiera Zipfian modelada a partir de la traza real) logra aproximar los resultados reales. De hecho, los errores son muy altos en algunos casos, por ejemplo, las tasas de acierto obtenidas por las cargas sintéticas generadas siguiendo las distribuciones Latest y Uniforme muestran una RMSE[6] del 31% y 20%, mientras que los errores de las cargas generadas con la ley de Zipfian están entre el 9% y el 17%. Esto demuestra el valor de realizar pruebas basadas en la reproducción de trazas reales, sustentando la tesis de que YCSB-replay es un buen complemento a YCSB permitiendo al evaluador realizar pruebas bajo escenarios realistas.

[6] RMSE - Raız del Error Cuadratico Medio

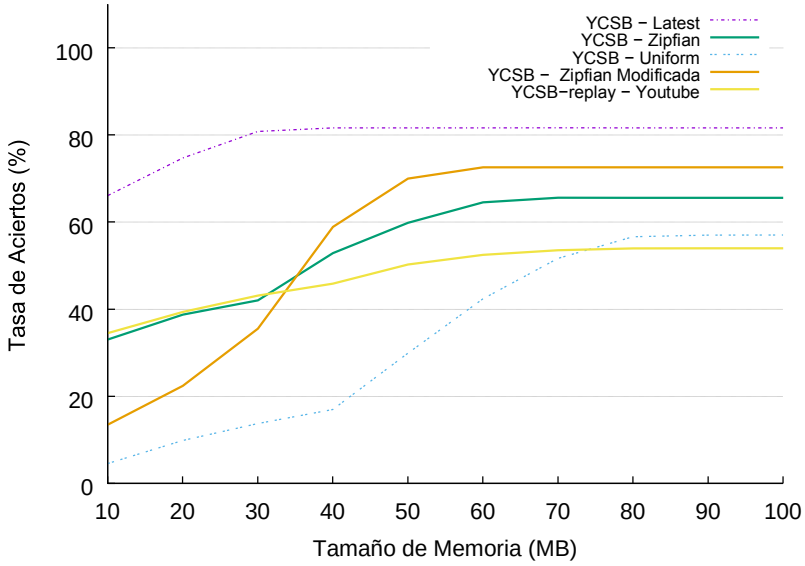

Figura 4: Comparación de tasas de acierto entre las cargas de trabajo sintéticas generadas con YCSB y la carga de trabajo real de YouTube reproducida con YCSB-replay. Ninguna de las cargas sintéticas (ni siquiera Zipfian modelada a partir de la traza real) logra aproximar los resultados reales.

4.3 Pruebas de rendimiento

Se realizaron pruebas de rendimiento para determinar si la reproducción de una carga de trabajo real con YCSB-replay afecta el rendimiento del proceso de generación de transacciones. Es decir, la meta es observar si se puede realizar la misma cantidad de transacciones por segundo con YCSB-replay que con YCSB.

La Figura 5 muestra el número de operaciones por segundo obtenidas durante la ejecución de una carga de trabajo generada aleatoriamente por YCSB (Zipf) y una carga de trabajo real obtenida a partir de la reproducción de un archivo de trazas (YCSB-replay); en cada caso se ejecutaron cinco millones de transacciones. Se realizaron pruebas con un número variable de hilos ya que mientras más hilos se usa, más crítico es tener código de alto rendimiento. Se observa que no existen diferencias significativas entre la ejecución de ambas cargas de trabajo. Esto demuestra que YCSB-replay es capaz de realizar pruebas de estrés igual de efectivas que con la versión original de YCSB.

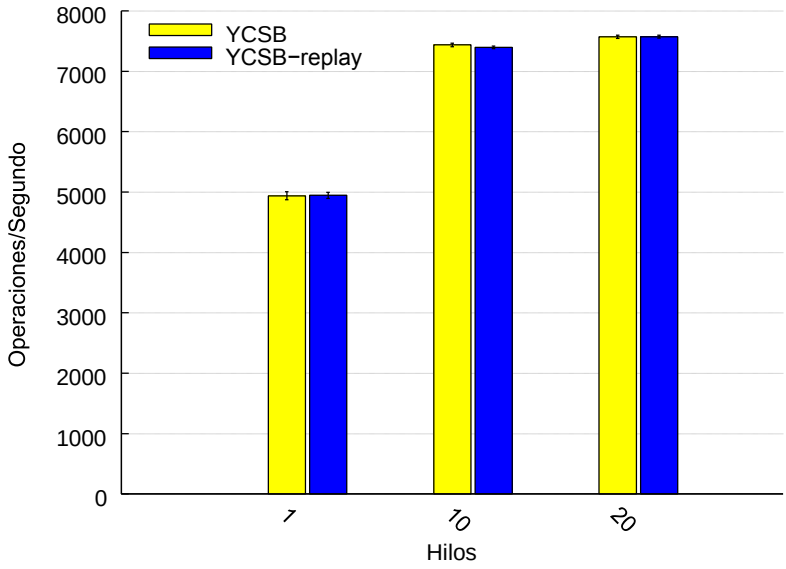

Figura 5: Comparación de tiempos de ejecución de YCSB y YCSB-replay, usando 1, 10 y 20 hilos para generar las cargas de trabajo. Cada barra muestra el promedio de diez corridas. Las barras de error indican la desviación estándar correspondiente. Se puede observar que YCSBreplay tiene un rendimiento equivalente al de YCSB.

4.4 Discusión

Los resultados presentados en esta Sección muestran lo siguiente:

- YCSB-replay permite, de manera efectiva, reproducir cargas de trabajo reales que pueden ser utilizadas por investigadores y profesionales para realizar pruebas de rendimiento realistas y así efectuar un correcto dimensionamiento de bases de datos clave-valor.
- La reproducción sintética de cargas (YCSB) no logra aproximar los resultados de cargas reales, generando errores en la tasa de aciertos de hasta el 31%. Por lo tanto, YCSB resulta útil para realizar microbenchmarks y YCSB-replay extiende esta funcionalidad permitiendo realizar pruebas bajo escenarios de cargas de trabajo realistas.
- Las pruebas de rendimiento realizadas demuestran que YCSB-replay es capaz de generar la misma cantidad de operaciones por segundo que YCSB. Esto demuestra que YCSB-replay es igual de efectivo que YCSB para realizar pruebas de estrés.

Existen otras dos dimensiones de la carga de trabajo real que no han sido consideradas en YCSB-replay: los datos (tamaño y forma) insertados en la base de datos

y los interarribos de las transacciones (es decir, cuanto tiempo transcurre entre una transacción y otra).

En cuanto a los datos, se optó por utilizar el mismo proceso de generación aleatoria presente en YCSB. Esto permite que los resultados de YCSB-replay puedan ser comparados con los de YCSB. Sin embargo, en ocasiones es importante que los datos insertados y recuperados durante las transacciones también sean realistas. La generación de conjuntos de datos o datasets realistas es un área actual de investigación. Se sugiere al lector referirse al trabajo de Tarasov et al. [12] para más información sobre cómo se puede realizar una generación realista de datasets.

Finalmente, hay dos maneras de añadir interarribos realistas a YCSB-replay. La una es utilizar una generación sintética basada en la carga real observada [8] y la otra, es la reproducción fiel de los interarribos observados en la traza original. En el presente trabajo se considera que la segunda alternativa es el mejor enfoque ya que garantiza que cualquier peculiaridad de los interarribos sea reproducida (cosa que ningún modelo, por su naturaleza reductiva, puede asegurar). Al momento se está trabajando en añadir esta funcionalidad a YCSB-replay y la nueva versión de código abierto de esta herramienta será liberada en cuanto los cambios necesarios a la misma estén finalizados.

Cabe recalcar que estas dos dimensiones (datos e interarribos) son ortogonales al trabajo presentado en este artículo. Es decir, estas mejoras pueden convivir con la funcionalidad ya implementada en YCSB-replay.

5 Trabajos relacionados

Los trabajos de investigación en el área de modelamiento y generación de cargas de trabajo requieren comprobar sus resultados mediante la ejecución de pruebas de rendimiento usando las cargas generadas. En un trabajo previo, Abad et al. [2] construyeron un simulador para reproducir trazas de metadatos y calcular tasas de acierto en memorias cache usando diferentes políticas de desalojo. Sin embargo, esta práctica puede contribuir a que los investigadores desvíen su atención hacia los detalles de la implementación de la herramienta de evaluación.

Otro enfoque es utilizar una herramienta disponible para que reproduzca el modelo de carga de trabajo generada. Tarasov et al. [11] utilizaron este método creando componentes que traduzcan su modelo de carga de trabajo en el lenguaje requerido por dos herramientas de evaluación de rendimiento Filebench y IOzone. No obstante, este método también requiere conocimiento de los detalles de funcionamiento de una herramienta de terceros. Además, la carga generada adolece de las limitaciones de modelamiento de la herramienta utilizada para la reproducción.

Un tercer método es la reproducción de trazas de cargas de trabajo, sean estas obtenidas a partir de sistemas reales o generadas sintéticamente por herramienta de modelamiento. Ungureanu et al. [14] modificaron YCSB para grabar las trazas obtenidas con el generador de cargas de trabajo para reproducirlas durante la evaluación. Sin embargo, no han hecho disponible el método de reproducción de dichas trazas. En el presente trabajo se considera a este método de reproducción de cargas a partir de trazas como el más adecuado puesto que no requiere que los investigadores

realicen modificaciones a la herramienta que ejecuta la evaluación, sino que únicamente deben generar y guardar las trazas de la carga de trabajo, la misma que es leída por YCSB-replay durante la evaluación.

6 Conclusiones y trabajo futuro

En este trabajo se ha presentado YCSB-replay, una extensión del ampliamente utilizado framework YCSB que permite reproducir cargas de trabajo mediante la lectura de trazas desde un archivo. Se aprovechó la extensibilidad del producto original para adicionar la solución propuesta como un complemento a los generadores de carga ya disponibles y mediante pruebas de rendimiento se demostró que este no se ve afectado por la lectura de las trazas en lugar de la generación aleatoria de las claves. Como prueba de la utilidad de YCSB-replay, se realizaron experimentos en los que se analizó a la base de datos de clave-valor Redis, en su funcionalidad de cache de datos. Los resultados muestran que al evaluar a Redis con cargas sintéticas (YCSB) no se logra aproximar los rendimientos de una carga de trabajo real, observando errores promedio (RMSE) de hasta el 31% en la tasa de aciertos de la cache. Esto demuestra que el uso de YCSB-replay permite realizar un mejor dimensionamiento de los recursos requeridos. En el futuro, se planea incluir marcas de tiempo en las trazas para que YCSB-replay genere transacciones con tiempos de arribo realistas, en lugar de la tasa constante implementada actualmente.

Referencias

1. Abad, C., Roberts, Lu, Campbell: A storage-centric analysis of MapReduce workloads: File popularity, temporal locality and arrival patterns. In: Proceedings of the IEEE International Symposium on Workload Characterization (IISWC) (2012)

2. Abad, C.L., Luu, H., Roberts, N., Lee, K., Lu, Y., Campbell, R.H.: Metadata tracesand workload models for evaluating Big storage systems. In: Proceedings of the IEEE/ACM Utility and Cloud Computing Conference (UCC) (2012)

3. Cooper, B., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarkingcloud serving systems with YCSB. In: Proceedings of the ACM Symposium on Cloud Computing (SoCC) (2010)

4. DB-engines ranking of key-value stores.http://db-engines.com/en/ranking/key-value+store (Aug 2015), fecha de ultimo acceso: Agosto 7 de 2015

5. Mahanti, A., Eager, D., Williamson, C.: Temporal locality and its impact on Webproxy cache performance. Performance Evaluation 42(2–3) (2000)

6. Motwani, R., Vassilvitskii, S.: Distinct values estimators for power law distributions. In: Proceedings of the Third Workshop on Analytic Algorithmics and Combinatorics, ANALCO 2006 (2006)

7. Patil, S., Polte, M., Ren, K., Tantisiriroj, W., Xiao, L., Lopez, J., Gibson, G., Fuchs, A., Rinaldi, B.: YCSB++: Benchmarking and performance debugging advanced features in scalable table stores. In: Proceedings of the ACM Symposium on Cloud Computing (SoCC) (2011)

8. Pitchumani, R., Frank, S., Miller, E.L.: Realistic request arrival generation in storage benchmarks. In: Proceedings of the IEEE Symposium on Mass Storage Systems and Technologies (MSST) (2015)

9. Who's using redis? `http://redis.io/topics/whos-using-redis` (Aug 2015), fecha de ultimo acceso: Agosto 7 de 2015

10. Reiss, C., Wilkes, J., Hellerstein, J.L.: Obfuscatory obscanturism: making workload traces of commercially-sensitivesystems safe to release. In: CloudMAN. Maui, HI, USA (2012), http://www.e-wilkes.com/john/papers/2012.04-obfuscation-paper.pdf

11. Tarasov, V., Kumar, S., Ma, J., Hildebrand, D., Povzner, A., Kuenning, G., Zadok, E.: Extracting flexible, replayable models from large block traces. In: Proceedings of the USENIX Conference on File and Storage Technologies (FAST) (2012)

12. Tarasov, V., Mudrankit, A., Buik, W., Shilane, P., Kuenning, G., Zadok, E.: Generating realistic datasets for deduplication analysis. In: Proceedings of the USENIX Annual Technical Conference (ATC). USENIX ATC'12 (Jun 2012)

13. Traverso, S., Ahmed, M., Garetto, M., Giaccone, P., Leonardi, E., Niccolini, S.: Temporal locality in today's content caching: Why it matters and how to model it. SIGCOMM Comp. Comm. Rev. 43(5) (Nov 2013)

14. Ungureanu, C., Debnath, B., Rago, S., Aranya, A.: TBF: A memory-efficient replacement policy for flash-based caches. In: Data Engineering (ICDE), 2013 IEEE 29th International Conference on (2013)

15. Zemlys, V.: Answer to: How to calculate zipf's law coefficient from a set oftop frequencies? http://stats.stackexchange.com/questions/6780/how-to-calculatezipfs-law-coefficient-from-a-set-of-top-frequencies(Feb 2011),fecha de ultimo acceso: Agosto 8 de 2015

16. Zhang,H., Chen, G., Ooi, B.C., Wong, W.F., Wu, S., Xia, Y.: "Anti-Caching"-basedelastic memory management for big data. In: Data Engineering(ICDE), 2015 IEEE 31st International Conference on. IEEE (2015)

17. Zink, M., Suh, K., Gu, Y., Kurose, J.: Characteristics of youtube network traffic at a campus network - measurements, models, and implications. Computer Networs 53(4) (Mar 2009)