Búsquedas Selectivas sobre Flujos de Documentos

Santiago Ricci[1], Pablo Lavallén[1], Gabriel H. Tolosa[1,2]
{sricci, plavallen, tolosoft}@unlu.edu.ar
[1]Departamento de Ciencias Básicas, Universidad Nacional de Luján
[2]CIDETIC, Universidad Nacional de Luján

Resumen

La cantidad de información digital que se genera día a día impone restricciones a los usuarios en cuanto a la facilidad de acceso. Considerando la necesidad de acceder a información relevante, la alta tasa de aparición de nuevo contenido genera la necesidad de contar con herramientas de búsqueda que puedan manejar el tamaño, complejidad y dinamismo de las fuentes de información digital actuales. Este problema no puede ser resuelto en el ámbito de un solo equipo de cómputo por lo que requiere de una arquitectura que involucra procesamiento paralelo y distribuido, la cual incluye diseñar y optimizar estructuras de datos y algoritmos eficientes que las gestionen.

Esta arquitectura es desafiada cuando los documentos aparecen en flujos en tiempo real como, por ejemplo, las publicaciones en las redes sociales. Un caso paradigmático son las publicaciones en Twitter, en la cual millones de usuarios[1] alrededor del mundo publican "documentos cortos"(*tweets*) desde diferentes tipos de dispositivos (generalmente, móviles), los cuales deben estar disponibles casi de inmediato (segundos) por lo que las estructuras de datos deben soportar un alto dinamismo. Esto contrasta con la búsquedas web clásicas, donde el índice invertido se actualiza en modo batch ya que existe un tiempo entre actualizaciones debido a la necesidad de recolectar los nuevos documentos a indexar.

Un abordaje actual a este problema es la partición de la colección en porciones (*shards*) de acuerdo a algún criterio (por ejemplo, temático) de manera tal de enviar las consultas solamente a un número reducido n de nodos ($n \ll P$) que contengan particiones de la colección que potencialmente pueden satisfacer la consulta. Este problema se lo conoce como "búsquedas selectivas"(*selective search*) e incluye métodos que permiten seleccionar los recursos adecuados, algoritmos de fusión de resultados parciales y estrategias adaptadas de caching.

Este trabajo presenta las líneas de investigación en el contexto de las búsquedas en tiempo real utilizando una arquitectura basada en búsquedas selectivas. Las propuestas abarcan el estudio, diseño y evaluación de los criterios de actualización del índice invertidos por partición, las estrategias de cache a implementar y el algoritmo de búsqueda final y cómo estos impactan en la performance que se pretende optimizar (eficiencia y/o efectividad).

Palabras clave: algoritmos de búsqueda, estructuras de datos, búsquedas selectivas.

Contexto

Esta presentación se encuentra enmarcada en el proyecto de investigación "Búsquedas Selectivas sobre Flujos de Documentos", de la Secretaría de Ciencia y Tecnología de la Universidad Nacional de Luján. Convocatoria: Proyectos de Investigación Articulados con Centros de Investigación, Docencia y Extensión y/o Centros Regionales, Delegaciones y Sede CABA (RESREC:213-19).

Introducción

Las búsquedas de escala web son llevadas a cabo por motores de búsqueda que se encuentran preparados para gestionar eficientemente aspectos de escalabilidad, eficiencia y efectividad [2]. Esto se puede plantear como el problema de diseñar sistemas distribuidos de gran escala que permitan satisfacer las expectativas de los usuarios, utilizando eficientemente los recursos disponibles [5]. Hoy en día, este problema es considerado uno de los más atrayentes tanto en aspectos teóricos como prácticos en ciencias de la computación.

[1]De acuerdo a https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/, más de 300 millones de usuarios. Última consulta: 10/03/2020.

La arquitectura típica de un motor de búsqueda está compuesta por un nodo front-end (broker) que proporciona la interfaz de usuario y un conjunto (grande) de nodos *workers* (P) que almacenan los datos y soportan las búsquedas. Para alcanzar una alta eficiencia, se han desarrollado estrategias sofisticadas para recorrer las estructuras de datos (índice invertido) en los *workers* como los algoritmos de poda dinámica (*dynamic pruning*) Maxscore, WAND y Block-Max-Wand [8]. A partir de una consulta de usuario (*query*), éstos permiten "navegar" por las estructuras de datos sin evaluar todos los documentos de modo de acelerar el procesamiento sin disminuir la calidad de las respuestas. Complementariamente, se implementan varios niveles de cache: a nivel del *broker* se mantiene generalmente un cache de resultados[4], el cual almacena la lista final de los resultados correspondientes a las consultas más frecuentes o recientes. Además, un cache de listas de posteo (*posting list cache*) [14] se implementa en cada nodo de búsqueda. Este cache almacena en memoria las listas de posteo de términos populares o valiosos, para evitar el acceso al disco.

Cuando esta arquitectura es aplicada sobre flujos de documentos de escasa extensión generados en tiempo real, como el caso de Twitter, aparecen nuevos problemas, siendo necesario revisar esta arquitectura y diseñar y aplicar nuevas estrategias, políticas, estructuras de datos y algoritmos.

Diversos trabajos se enfocan en este tema, ya sea, basándose en indexación adaptativa o en control de la evolución del tamaño de las estructuras de datos. En el primero de los casos, Chen et al. [6] implementa un sistema de indexación orientado incorporar al índice solamente aquellos tweets que tengan mayor probabilidad de formar parte de los resultados de búsqueda (llamados "distinguidos"), retrasando la indexación de los restantes (llamados "ruidosos"). En el otro caso, Rissola et al. [12] propone estrategias de poda dinámica del índice invertido (invalidación) que controlan su tamaño y permiten aumentar la eficiencia de la búsqueda al recorrer estructuras más compactas.

Por otro lado, se han propuesto estrategias alternativas para acelerar el proceso de búsqueda en grandes colecciones de documentos. Una de ellas son las "búsquedas selectivas"(*selective search*), la cual implica particionar la colección en subcolecciones que luego son asignadas a diferentes nodos de acuerdo a algún criterio con el objetivo de consultar un número reducido de ellos para satisfacer una consulta. Como es de suponer, un aspecto clave aquí es el criterio empleado para particionar y distribuir la colección entre los nodos. Dicho criterio es llamado política de asignación. En relación a esto, Kulkarni et al. [11] estudiaron tres tipos de asignación (aleatoria, basada en origen y basada en el tema) e investigaron cómo realizar la búsqueda sólo en un subconjunto de *shards* para cada consulta sin sacrificar calidad. Sus resultados muestran que los costos de búsqueda se reducen al menos un orden de magnitud utilizando la estrategia de dividir por tema, esto es, mediante una técnica de clustering para la asignación de documentos a conjuntos disjuntos temáticos. En cuanto a la precisión, al menos el 86 % de las consultas obtuvo una performance más alta con esta estrategia versus la aleatoria y la basada en origen.

Por su parte, Hafizoglu et al. [9] plantean la utilización de un índice denominado *cluster-skipping* por cada *shard* que permite mejorar la latencia en la resolución de las consultas, llegando a mejoras del orden del 55 % respecto de la búsqueda selectiva basada en tópicos. Además, muestran que la arquitectura mejora el imbalance producido al utilizar índices basados en tópicos, permitiendo un balanceo uniforme a través de los diferentes *shards*.

Sin embargo, ninguno de los trabajos previos considera la posibilidad que los documentos arriben en flujos, como es el caso de las publicaciones en redes sociales.

Finalmente, entre los trabajos sobre caching y búsquedas selectivas solamente se cuenta con el aporte de Dai y Callan [7]. En el mismo, se estudia si los *shards* basados en temas reducen la efectividad del *caching* basado en *QtfDf* [3] y propone una serie de variantes que tienen en cuenta esta arquitectura. Además, explora si el uso de un *log* de consultas relacionadas a los *shards* puede mejorar los resultados del *caching* de *posting lists*. Los resultados muestran que la distribución sesgada de las *posting listss* pueden causar una disminución en la tasa de aciertos. Sin embargo, este efecto puede ser eliminado utilizando los logs de consultas en la política de asignación de los shards, distribuyendo los más populares en diferentes núcleos de los procesadores, lo cual a su vez, mejora la performance.

Líneas de I+D

Las líneas de I+D del grupo se basan, principalmente, en mejorar la eficiencia en la recuperación de información de gran escala sobre flujos de documentos en tiempo real, con énfasis en los siguientes temas:

a. Diseño de políticas de asignación

Como se mencionó anteriormente, en este trabajo se parte de la hipótesis de que una arquitectura basada en búsquedas selectivas pueden mejorar la búsqueda sobre flujos de documentos cortos en tiempo real sobre todo en eficiencia. Así, dentro de este tipo de arquitecturas, los criterios empleados para la asignación de los documentos en los diferentes *shards* juega un rol fundamental, dado que la misma impacta fuertemente en el logro de una distribución de carga lo más equitativa posible entre los nodos que componen el *cluster*. Aquí no hay que perder de vista que dicha política será empleada sobre un flujo de documentos en tiempo real, lo cual implica una indexación incremental lo que supone desafíos desde el punto de vista del rendimiento a la hora de asignar documentos a las particiones. Los objetivos aquí perseguidos entonces son el diseño y evaluación, respecto al estado del arte, de políticas y técnicas de asignación de documentos a las diferentes particiones utilizando, entre otras, técnicas de clasificación de aprendizaje automático. En particular, se propone tomar como base los algoritmos/métodos, considerados estado del arte, utilizados para seleccionar recursos, concretamente Rank-S [10] y Taily [1]. A partir de éstos, se propondrán variantes que consideren que los documentos son cortos y que aparecen en flujos. Lo cual, además, lleva a que consideren que las estadísticas[2] no son tan robustas y que deben recalcularse de forma on line.

b. Métodos de selección de los nodos

Luego de haber distribuido los documentos entre las particiones, otra cuestión de suma importancia en el contexto de la búsqueda selectiva, es cuáles y cuántos *shards* consultar a la hora de resolver una *query*. Aquí se pueden distinguir dos grandes clases de algoritmos de selección que modelan a los *shards* de diferentes formas: los basados en vocabulario y los basados en muestreo [1]. Los primeros, representan las particiones mediante una serie de estadísticas de los términos del vocabulario manejado por el motor de búsqueda. En cambio, los basados en muestreo son aquellos que utilizan un índice central que contiene muestras representativas de cada *shard*. En este proyecto, específicamente, en base a escenarios diferentes, con requerimientos diversos planteados para el diseño de las políticas de asignación, se propone estudiar el *tradeoff* entre la cantidad de particiones y número seleccionado para resolver la consulta.

c. Técnicas de caching en los nodos y en el broker

Para el estudio de las políticas de caching, se propone partir del trabajo de Dai et al. [7], quienes trabajan con caches estáticos, luego extenderlo a dinámicos y estudiar el impacto de diferentes políticas de reemplazo, con la posibilidad de combinarlas entre *posting lists* y resultados. En general, estas pruebas de evaluación se realizan habitualmente con datos reales provenientes de archivos de log de motores de búsqueda comerciales. Existen datasets disponibles para investigación que son de dominio público y – además – se cuenta con vínculos académicos con grupos en la temática con quienes se comparten datos de prueba..

Finalmente, para poder estimar la performance de diferentes arquitecturas y configuraciones se propone complementar la experimentación real con modelos de simulación. Éstos posibilitan evaluar un sistema sobre escenarios para los cuales no se disponen recursos (por ejemplo, un cluster de 2000 o más nodos) y determinar costos, performance, escalabilidad y otras métricas.

Resultados y objetivos

Dentro del escenario en el cual se cuenta con flujos de documentos que ingesta el sistema y se deben incorporar a una o varias particiones para soportar la búsqueda, la hipótesis de trabajo que aparece es que es posible aumentar la performance del sistema de búsqueda a partir de la utilización de métodos ad-hoc para la asignación de documentos,

[2]Con el término "estadísticas" se pretende indicar el conjunto de indicadores que se utilizan para la descripción de una colección (o partición) y que son necesarios también para la selección de recursos (por ejemplo, distribución de a frecuencia de los términos en los documentos y en la colección, tamaño de la misma, tasa de actualización, etc.)

junto con estrategias de selección de particiones que consideren este mismo criterio.

De aquí, se define como objetivo general de esta propuesta el desarrollo de técnicas y algoritmos para la asignación de flujos de documentos a diferentes particiones de un índice para luego soportar búsquedas sobre un subconjunto de éstas, maximizando la performance (tanto eficiencia como efectividad).

En cuanto a los objetivos específicos, se consideran:

- Desarrollar un método online para rutear los documentos a los nodos (o particionar una colección) para la asignación a las diferentes particiones (*shards*) basado en parámetros de la arquitectura (por ejemplo, número de procesadores, núcleos, etc.).
- Proponer y evaluar nuevos métodos de selección de los nodos a consultar para responder una consulta a partir de considerar la dinámica de ingreso de los documentos y la estrategia de ruteo.
- Estudiar el *tradeoff* entre cantidad de *shards* y número de nodos a consultar (k) para obtener la respuesta a una consulta dada para las métricas de interés (eficiencia/efectividad).
- Incorporar y evaluar técnicas de *caching* en los nodos y en el *broker* para incrementar el *throughput* del sistema (*queries/segundo*).
- Desarrollar un método que permita especificar una métrica a maximizar para un *query* dado (por ejemplo, para soportar aplicaciones que requieren una aproximación cercana a una búsqueda exhaustiva).
- Desarrollar un modelo de estimación de la performance de la arquitectura en base a las variables más determinantes para un hardware dado (por ejemplo, un *cluster* particular).

Los resultados de este proyecto poseen aplicación práctica directa en la industria. Hoy en día la mayoría de los sistemas que gestionan información implementan estrategias de búsquedas, incluso sobre documentos de texto o flujos de tweets. En este último caso, principalmente debido a que Twitter se utiliza como una plataforma multipropósito, tanto de diseminación de información social como para negocios (marketing), publicidad, entre otras. Cualquiera de estas aplicaciones que requieran soportar búsquedas en tiempo real deben contar con estrategias eficientes que optimicen el uso de recursos ofreciendo parámetros de efectividad aceptables (que dependen del problema). Además, teniendo en cuenta que se propone realizar la experimentación sobre plataformas distribuidas (por ejemplo, Spark [13]), se genera conocimiento para implementar los modelos/servicios a diferentes escalas, permitiendo, por ejemplo, que una pequeña/mediana organización acceda a desplegar un cluster con hardware commodity.

Formación de Recursos Humanos

En cuanto a formación de recursos humanos se propone que los integrantes del proyecto se incorporen a tareas de investigación de forma regular. Además, se espera que al menos uno de los integrantes inicie estudios de postgrado. Complementariamente, durante los dos años del proyecto se espera incorporar un pasante (estudiante de la Lic. en Sistemas de Información) para que inicie tareas de investigación y ofrecerle la posibilidad de seguir esta línea para su trabajo final (tesina de grado).

Referencias

[1] R. Aly, D. Hiemstra, and T. Demeester. Taily: shard selection using the tail of score distributions. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 673–682, 2013.

[2] R. Baeza-Yates, C. Castillo, F. Junqueira, V. Plachouras, and F. Silvestri. Challenges on distributed web retrieval. In *2007 IEEE 23rd international conference on data engineering*, pages 6–20. IEEE, 2007.

[3] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Murdock, V. Plachouras, and F. Silvestri. The impact of caching on search engines. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 183–190, 2007.

[4] R. Blanco, E. Bortnikov, F. Junqueira, R. Lempel, L. Telloli, and H. Zaragoza. Caching search

engine results over incremental indices. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 82–89, 2010.

[5] B. B. Cambazoglu and R. Baeza-Yates. Scalability challenges in web search engines. In *Advanced topics in information retrieval*, pages 27–50. Springer, 2011.

[6] C. Chen, F. Li, B. C. Ooi, and S. Wu. Ti: an efficient indexing mechanism for real-time search on tweets. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 649–660, 2011.

[7] Z. Dai and J. Callan. Inverted list caching for topical index shards. In *European Conference on Information Retrieval*, pages 577–583. Springer, 2018.

[8] S. Ding and T. Suel. Faster top-k document retrieval using block-max indexes. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 993–1002, 2011.

[9] F. Hafizoglu, E. C. Kucukoglu, and I. S. Altingovde. On the efficiency of selective search. In *European Conference on Information Retrieval*, pages 705–712. Springer, 2017.

[10] A. Kulkarni. *Efficient and effective large-scale search*. PhD thesis, Carnegie Mellon University, 2013.

[11] A. Kulkarni and J. Callan. Selective search: Efficient and effective search of large textual collections. *ACM Transactions on Information Systems (TOIS)*, 33(4):1–33, 2015.

[12] E. A. Ríssola and G. H. Tolosa. Improving real time search performance using inverted index entries invalidation strategies. *Journal of Computer Science & Technology*, 16, 2016.

[13] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, et al. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

[14] J. Zhang, X. Long, and T. Suel. Performance of compressed inverted list caching in search engines. In *Proceedings of the 17th international conference on World Wide Web*, pages 387–396, 2008.