Propuesta de caché semántico en un sistema de interrogación P2P
Semantic Caching Proposal in a P2P Querying System

Carlos-Hernan Prada-Rojas,
Claudia Roncancio, Cyril Labbé
Laboratoire d'Informatique de Grenoble
681 rue de la Passerelle - Domaine Universitaire
BP 72 - 38402 St Martin d'Hères. Francia
Nombre.Apellido@imag.fr

María del Pilar Villamil
Universidad de los Andes
Carrera 1 N. 18A 10 Bogota, Colombia
mavillam@uniandes.edu.co

Abstract

Distributed systems as Peer to Peer systems use resource optimization strategies to improve computing and communication performance. In the context of structured P2P systems based on DHT, the query caching, represented by inverted indexes, can be considered as a solution alternative. The present paper shows a query cache optimization for the Data Location Service PinS, through the proposition of a semantic retrieval strategy of data contained inside PinS Query Cache[1].

Resumen

Los sistemas P2P como sistemas distribuidos utilizan estrategias de optimización de recursos de comunicación y cálculo para mejorar su desempeño. Dentro del contexto de los sistemas P2P estructurados basados en DHT, el caché de consultas representadas por medio de índices inversos constituye una alterativa de solución. Este artículo presenta una propuesta de caché para optimizar la evaluación de consultas de desigualdad brindadas por el servicio de localización PinS, mediante el uso de una estrategia de recuperación semántica de datos contenidos en el caché de consultas[2].

1. Introducción

El desarrollo de alternativas para la distribución de la información sobre Internet ha generado nuevas propuestas como los sistemas *Peer to Peer* (P2P). Estos sistemas se caracterizan en su mayoría por su equidad en la distribución de los contenidos almacenados, igualdad de roles y capacidades entre los nodos y resistencia a los posibles fallos causados por la volatilidad de sitios participantes o por problemas en la comunicación [Sch01]. Para ello, estos sistemas implementan de manera no trivial, estrategias de duplicación de datos junto con políticas para el manejo de problemas de coherencia debidos a la misma duplicación.

Así mismo, los sistemas *P2P* de gran tamaño han presentado varias arquitecturas, que van desde las que manejaban algún tipo de centralización, hasta las completamente distribuidas en recursos y funcionalidades. Estas últimas se dividen en las no estructuradas y las estructuradas. Su diferencia radica en la estrategia para conectar los nodos entre ellos y en los mecanismos de almacenamiento y recuperación de datos al interior del sistema.

Dentro del marco de alternativas para mejorar el rendimiento y como derivado de una estrategia de duplicación, se encuentra el caché de objetos y/o consultas, dependiendo del recurso del que se requiera optimizar su uso.

Gran cantidad de trabajos relacionados con la utilización de sistemas de caché basados en arquitecturas *P2P* han sido publicados, dentro de los cuales es posible diferenciar dos grupos: sistemas P2P que trabajan como caché para otros sistemas, y los sistemas P2P que usan una solución de caché implementada dentro de su propia infraestructura para mejorar su propio rendimiento.

Dentro del primer grupo, se encuentran trabajos como *PeerOLAP* [KNO+02] que utiliza una red P2P no estructurada como soporte de caché de un sistema *OLAP*. Este caché está orientado principalmente al almacenamiento de consultas o segmentos de resultados de consultas (particionamiento vertical u horizontal). Los segmentos aquí mencionados buscan evitar reprocesamien-

[1]Work made with the support of the ECOS action C07M02
[2]Trabajo realizado con el apoyo de la acción ECOS C07M02

to de cálculos previamente realizados dentro del sistema. Otros trabajos buscan disminuir el uso de los recursos de red en términos de ancho de banda y cantidad de mensajes. Tal es el caso de *Squirrel* [IRD02], utilizado como solución de caché para evitar accesos a red por parte de clientes Web a partir de la cooperación entre los cachés clientes de su propia red, y *PierSearch* [LHH+04], usado como caché de elementos poco solicitados dentro de una red *P2P* no estructurada, pero con un caché implementado sobre una red *P2P* estructurada.

Los trabajos como *PeerOLAP*, buscan de alguna manera aminorar costos de acceso o cargos producidos por el uso del canal de comunicación, porque están construidos sobre redes P2P que usan la inundación como método de acceso a los datos. Como solución a esta problemática los esquemas planteados buscan realizar copias de los objetos a lo largo del sistema.

En el segundo grupo de trabajos, la mayoría funcionan sobre redes P2P estructuradas que usan *Distributed Hash Table (DHT)* como soporte de almacenamiento[3]. Su enrutamiento está basado en llaves *(KBR - Key Based Routing)* para entregar información sobre la ubicación de los nodos y proveer, al mismo tiempo, resistencia a la salida de nodos del sistema. De lo anterior se puede concluir que la necesidad potencial de un caché para mejorar el rendimiento no buscaría aminorar el número de saltos para encontrar un recurso[4], sino evitar costos de procesamiento por consultas o conjuntos de consultas que hayan sido calculadas previamente dentro del sistema. El caché de consultas, en este caso, es una opción válida.

Distributed Cache Table - DCT [SA06] propone una estrategia de unión entre la indexación y el caché vistas como un solo tipo de entradas a lo largo del sistema *P2P*. Utiliza un mecanismo de *ranking* distribuido para decidir los resultados a almacenar. *PinS* [Vil06] es un servicio de localización de datos. Estos pueden ser solicitados por medio de consultas de igualdad y desigualdad sobre valores de atributos. *PinS* implementa una estrategia de caché de consultas de igualdad.

Por otra parte y fuera del contexto *P2P*, existen soluciones que buscan sacar un mejor provecho del contenido del caché. Tal es el caso de las estrategias que usan semántica dentro del caché [CRS99][RDK03]. Estas estrategias consiguen una mejor reutilización, a un nivel de granularidad más fino, del contenido del caché.

Nuestra propuesta utiliza conceptos de cachés semánticos para extender las funciones del caché de *PinS*. Esta propuesta busca optimizar la evaluación de consultas de desigualdad que son particularmente costosas. El caché se integra al sistema sin introducir puntos de centralización y sin agregar costos de manejo elevados.

El artículo está organizado de la siguiente manera: La sección 2 introduce las nociones generales propias de los cachés de consultas y semánticos. La sección 3 presenta *PinS*, un servicio de localización de datos para sistemas *P2P* basado en *DHT*. La sección 4 presenta la propuesta de un caché semántico para *PinS*. Por último, la sección 5 presenta las conclusiones y perspectivas del presente trabajo.

2. Cachés de consultas y semántico

El caché, como estrategia de mejoramiento del desempeño de sistemas informáticos, es muy utilizado en entornos distribuidos. Las estrategias de caché propuestas han intentado respetar las arquitecturas de los sistemas de los que hacen parte. Por ejemplo las arquitecturas de tipo servidor centralizado usan caché del lado del servidor para evitar el procesamiento de consultas ya calculadas y cachés locales para evitar incurrir en costos de comunicación del lado del cliente [Vil06]. Los sistemas *P2P* por su parte, han implementado diferentes técnicas de duplicación que van desde la selección de nodos que poseen condiciones operativas superiores para administrar los datos duplicados, hasta los que usan una estrategia coordinada y distribuida para el manejo de la duplicación y el caché.

A continuación introducimos las nociones generales de caché de consultas y caché semántico.

2.1. Caché de consultas

Un caché de consultas permite conservar en una memoria de acceso rápido las respuestas a consultas evaluadas por un sistema. El objetivo es simplemente reutilizar esa respuesta en caso de que la misma consulta se presente.

Una consulta es una proposición lógica en la que se solicita, por medio de una sentencia de tipo *Atributo* ***operador*** *Valor*, un elemento o conjunto de elementos que cumplen con la condición representada por la consulta. El ejemplo ilustrado en la Figura 1 muestra un conjunto de objetos almacenados en el nodo P_{234}. Los objetos representan libros que tienen asociados algunas características como el nombre del escrito, el autor, el año de publicación, el idioma de publicación. Si se propone una consulta C_x: *Año* $>$ *1800*, la respuesta que se espera según los objetos que existen es *Libro 1, Libro 4*.

[3] Aunque los sistemas P2P estructurados basados en *DHT* son los más difundidos, existen otros sistemas *P2P* estructurados basados en topologías lógicas como *d-dimension*, grafos y arborescencia

[4] La localización de un nodo dentro de una red *P2P* estructurada basada en *DHT* tiene como número máximo de saltos el *log(n)*, siendo *n* el número de sitios participantes

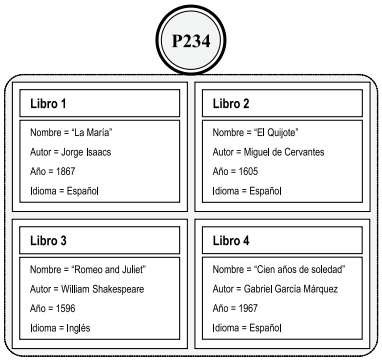

Figura 1. *Objetos ejemplo caché de consultas*

Ahora asumamos que C_x es propuesta a un sistema P2P que contiene objetos que pueden satisfacer a esa solicitud. C_x es demandada frecuentemente y de alguna manera, el sistema toma la decisión de almacenar algún tipo de entrada que le permita ahorrar costos en la evaluación de C_x. Las alternativas posibles en este caso son almacenar los objetos que satisfacen a C_x, o la consulta C_x seguida de algún tipo de identificación de objeto que evite reevaluar esa consulta.

Si en este caso se opta por almacenar la consulta C_x, junto con la lista de identificadores de los objetos que cumplen con C_x, la entrada a almacenar luciría de esta manera:

$$< Año > 1800 : Libro\ 1, Libro\ 4>$$

A la pareja compuesta por consulta y lista de llaves de identificación de objeto se les puede denominar *Índice Invertido* [JFY05]. Estos índices invertidos, pueden ser utilizados para crear entradas dentro de una estrategia de duplicación para mejorar el desempeño de un sistema distribuido. Su justificación es el ahorro de tiempo en la ejecución de una consulta dentro del caché.

A continuación se presenta la noción de caché semántico que permité un mejor aprovechamiento del contenido del caché por medio del empleo de regiones semánticas.

2.2. Caché semántico

Los cachés semánticos administran su contenido como un conjunto de regiones semánticas. Una región semántica reagrupa un conjunto de datos conexos [RDK03]. Un caché semántico tiene la particularidad de poseer conocimiento de una descripción semántica de los datos que almacena, que le permite manejar y usar ese contenido según criterios semánticos. Cuando una consulta llega al caché, es descompuesta en dos partes: el *Probe Query*, que recupera la porción del resultado disponible en el caché local y el *Reminder Query (RQ)* que recupera el conjunto de tuplas que no están presentes dentro del caché. Si *RQ* no es nula, se envía para ser evaluada en la fuente de datos [Lau06].

Si R es una región contenida en el caché semántico y C es una consulta para ser comparada contra el contenido del caché, existen varios casos posibles, a saber:

1. Coincidencia exacta: $C = R$ (*RQ* vacía),
2. Inclusión de la consulta: C contenida completamente en R,
3. Coincidencia parcial: R contenida completamente en C pero no a la inversa (*RQ* no es nula),
4. Intersección: cubrimiento parcial de la consulta sobre la región (*RQ* no nula),
5. Disjunción: R y C no se intersectan (*RQ* = *C*).

La evaluación de la presencia de uno de los casos dentro de las entradas de un caché permiten reutilizar de manera parcial o total el contenido de una entrada en el caché para resolver una consulta o decidir que el caché no es de utilidad en determinadas ocasiones. Nuestro interés se encuentra en el contexto de las consultas de desigualdad sobre sistemas *P2P* que, como se verá más adelante en *PinS* [Vil06], poseen un costo elevado de cálculo.

3. PinS

Una de las grandes ventajas de la utilización de redes P2P estructuradas basadas en *DHT* es la capacidad de recuperar objetos con una complejidad logarítmica, sin importar la ubicación del cliente que propone la consulta. Esta complejidad se respeta únicamente si se conoce el identificador del objeto solicitado sobre la *DHT*. Existen muchos trabajos interesados en aumentar el nivel de expresividad de consultas de estos sistemas P2P [SA06][VRL06], respetando las propiedades de dichos sistemas tales como la escalabilidad en términos de objetos compartidos en el sistema y la equidad entre los nodos o *peers* participantes. Un ejemplo de tales sistemas corresponde a *PinS*. *PinS*, es un servicio de localización de datos para redes *P2P* estructuradas basadas en *DHT*. PinS utiliza técnicas de indexación distribuidas que le permiten encontrar objetos sobre la *DHT*, a partir de metadatos del objeto. Es importante mencionar que dentro de lo que se podrían llamar las capas de un sistema *P2P* [Vil06], con respecto a la gestión de la red lógica, la gestión de almacenamiento y la gestión de datos, *PinS* se ubica funcionalmente en la capa más alta. Esta capa permite la inserción de datos descritos por medio

Query :: Term [and | or Term]*

Term ::= EqTerm | IneqTerm | LikeTerm

EqTerm ::= AttributeName = value

IneqTerm ::= AttributeName op value

op is <, >, =< or >=

LikeTerm ::= AttributeName like [% | + char + | %]

Figura 2. *Gramática de PinS*

de metadatos, de tipo *Atributo = Valor*, y la formulación de consultas que respetan la gramática de la Figura 2. Adicionalmente garantiza su portabilidad con respecto al sistema *DHT* subyacente, representado por las capas gestión de la red lógica y de almacenamiento.

A continuación se presenta una descripción del mecanismo de caché propuesto por *PinS* para la optimización de consultas de igualdad y el proceso de evaluación de consultas de desigualdad.

3.1. Caché para consultas de igualdad

PinS propone una estrategia de caché de consultas para reducir el número de mensajes involucrados en la evaluación de consultas de igualdad, al igual que el tamaño de los mismos y así mejorar el rendimiento general del sistema. La estrategia de caché se basa en el almacenamiento del resultado de consultas solicitadas de forma frecuente al sistema. La frecuencia de una consulta se calcula gracias a un registro de históricos que almacena el número de solicitudes de una consulta. Cada nodo responsable de indexar un metadato mantiene un histórico y define un umbral sobre ese parámetro para seleccionar las consultas que guardará el caché [Vil06].

El caché es distribuido y está compuesto por los catálogos distribuidos de caché y las entradas en el caché. Los catálogos distribuidos poseen información de las consultas almacenadas en el caché y del sitio donde se encuentran. Las entradas en el caché por su parte contienen el identificador del término que administra el nodo responsable de la entrada y relacionada con la consulta almacenada en una entrada, la consulta dada por el usuario y la lista de identificadores de objetos que la satisfacen.

Sea la consulta C_1 : *año=1867 and nombre='La María'*. C_1 es la conjunción de dos términos que llamaremos t_1 y t_2. La evaluación de consultas de igualdad, del estilo de C_1, se ve beneficiada con la existencia del caché (ver Figura 3). Supongamos la existencia de C_1 en el caché. Para evaluar la consulta se contactan los nodos responsables de los términos t_1 y t_2. Cada nodo antes de enviar la lista de identificadores de objetos asociados al término solicitado, busca en el catálogo de caché si existe una o más consultas que puedan aportar a la consulta que se está evaluando. En este caso el nodo responsable de t_1 busca en su catálogo de caché y encuentra que C_1 está en el caché, de esta manera contacta al nodo responsable de su almacenamiento identificado con $idC_1 = hash(C_1)$, consulta los objetos asociados a C_1 y envía la respuesta al nodo coordinador de la consulta. Esta estrategia de evaluación se puede extender para resolver consultas que no están totalmente almacenadas en el caché. En ese caso, los identificadores de objetos relacionados con los términos ya calculados en el caché serán considerados una respuesta parcial y los demás identificadores serán evaluados de la forma tradicional: un acceso por medio de su identificador de término será realizado para consultar los identificadores de objetos que lo satisfacen. Al final, el nodo coordinador será el responsable de calcular la respuesta total.

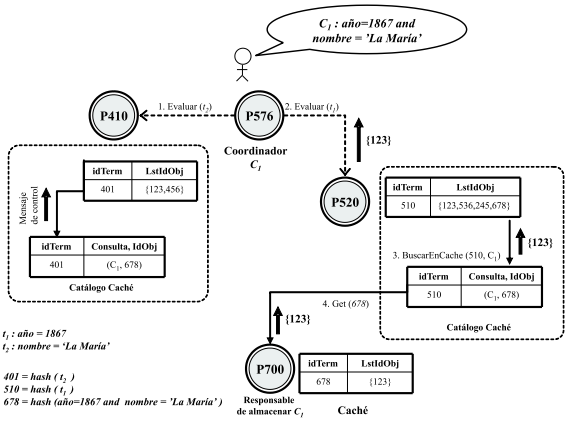

Figura 3. *Caché RM PinS*

El caché incluye un análisis semántico sencillo que corresponde a una forma de inclusión de consulta. Por ejemplo si se tiene en el caché una entrada E_1 : *Año = 2000 and Autor = Mistral*, y se solicita una nueva consulta C_2 : *Año = 2000 and Autor = Mistral and Idioma = Español*, es posible reutilizar E_1 para la primera parte de C_2, el último término deberá ser evaluado. En este artículo se propone extender el uso del caché a las consultas con términos de desigualdad.

PinS propone un modelo de coherencia de consultas en el caché fuerte para garantizar la comprensividad de las respuestas a las consultas. Esto significa que una vez que un nuevo objeto es ingresado al sistema, las entradas relacionadas con consultas almacenadas en el caché que el nuevo objeto cumple son actualizadas [Vil06].

3.2. Consultas de desigualdad

Nuevos retos para la evaluación eficiente de consultas de desigualdad surgen en los sistemas P2P basados en *DHT*. Estos retos son el resultado de la dificultad para identificar de forma eficiente los valores involucrados en una consulta de desigualdad y contactar únicamente los nodos que posean información para satisfacer esas consultas. PinS propone varias estrategias para la evaluación de consultas de desigualdad. Entre ellas se destacan la estrategia basada en catálogos locales (CL), la estrategia basada en índices distribuidos por atributos (IX) o fragmentados en grupos (IG) [Vil06]. En el contexto de este artículo se presenta la estrategia IX que permite ilustrar la dificultad para evaluar consultas de desigualdad y la oportunidad de utilizar técnicas de caché para su optimización.

La estrategia IX utiliza índices para resolver consultas de desigualdad. Los índices contienen los valores del atributo indexado y los identificadores de términos que permiten contactar al nodo responsable de un metadato. Tales nodos almacenan los identificadores de objetos que satisfacen el valor solicitado. El proceso utilizado para la evaluación de consultas de desigualdad, estilo *año* $<$ *1990* se muestra en la Figura 4.

1. El nodo coordinador de la consulta contacta al nodo responsable del índice asociado al atributo año, denominado nodo de atributo.

2. El nodo de atributo consulta localmente su índice y retorna al nodo coordinador todos los identificadores de términos (IdT_i) que satisfacen el criterio de búsqueda.

3. Posteriormente el nodo coordinador utiliza los identificadores de términos para obtener todos los objetos que contienen el metadato asociado a cada IdT_i, calcula la respuesta y la entrega al solicitante. De esta manera solo los nodos que contienen información relevante son contactados.

Varios riesgos se identifican en la evaluación de consultas de desigualdad con la estrategia IX. El nodo responsable de almacenar el índice puede ser sobrecargado por el tamaño del índice, al igual que por el número de consultas que le son solicitadas. Adicionalmente, si una consulta de desigualdad involucra un gran número de valores del dominio del atributo esto aumenta el número de mensajes enviados de forma paralela a los nodos responsables de los identificadores de términos. Estos riesgos evidencian la necesidad de plantear nuevas estrategias de evaluación de consultas para lograr su optimización.

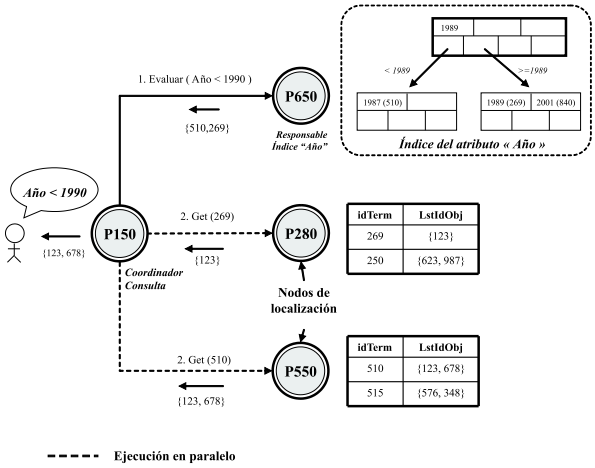

Figura 4. *Estrategia IX para evaluar la consulta año $<$ 1990*

4. Caché semántico para consultas de desigualdad en PinS

La resolución de consultas dentro de una *DHT* es un proceso costoso en términos de cálculo. Para ello, existen estrategias que implementan el almacenamiento de metadatos de los objetos contenidos en el sistema, dentro de la *DHT*. Aun así, si se requiere adicionalmente realizar operaciones entre eventuales conjuntos resultantes de la respuesta de una consulta, el proceso puede tener un alto costo de procesamiento. De hecho, la evaluación de consultas de desigualdad tiene un costo bastante elevado en términos de mensajes y cálculo [VRL06].

La sección 3 introdujo brevemente la evaluación de consultas de desigualdad en PinS. La presente sección presenta una propuesta de uso de caché para optimizar la evaluación de consultas de desigualdad. Se aborda por una parte el primer caso de coincidencia semántica, mencionada en la sección 2, y por otra parte, casos de intersección de consultas que necesitan un análisis fino. La solución extiende el uso de índices previsto por PinS.

4.1. Consultas de desigualdad en el caché

La primera parte de la propuesta consiste en generalizar la política del caché de consultas enunciada en PinS, y adicionarle el caso de coincidencia exacta para las consultas que incluyen términos tipo *Atributo [operadorDesigualdad] Valor*. Esto permite reutilizar directamente las consultas almacenadas cuando se solicitan varias veces (caso de coincidencia exacta).

Las entradas en el caché son homogéneas para todo tipo de consultas. Están compuestas de tres elementos: la consulta, su identificador idConsulta obtenido vía la función de hash del sistema *hash(Consulta)* y los identificadores de objetos que satisfacen la consulta. La forma general es:

<idConsulta, Consulta, IdObjetosSatisfacenConsulta>

Normalización de consultas: La generalización de materializar cualquier tipo de consulta necesita una estrategia de reescritura que permita normalizar las consultas para facilitar su manejo. Consideramos el caso general de consultas con términos de igualdad y de desigualdad. La versión normalizada de una consulta ubica primero los términos de desigualdad y luego los de igualdad. En cada "grupo"se ordenan los términos por orden alfabético del atributo que use. A continuación se ilustra la normalización con un ejemplo de reordenamiento independiente de términos de igualdad y desigualdad aplicados sobre la consulta C_i:

1. C_i : *d = 5 and c < 45 and b = 34 and a > 3* (consulta original - sin normalizar),
2. C_i : *a > 3 and c < 45 and b = 34 and d = 5* (después de normalizar).

Las entradas del caché usan la versión normalizada de las consultas. Sin pérdida de generalidad en lo que sigue consideraremos que las consultas se representan siempre en forma normalizada.

Estrategia de decisión para la introducción en el caché: el caché es totalmente distribuido lo que implica que cualquier nodo es susceptible de poseer un caché. La toma de la decisión de materializar una consulta en el caché debe ser hecha de forma que no se tenga un ente centralizador y que no introduzca costos elevados en mensajes u otras acciones. Para alcanzar este objetivo, proponemos una estrategia predefinida que permite al sistema tomar la decisión en forma autónoma. Esta estrategia se basa en el uso del nombre del atributo utilizado por el primer término (de desigualdad) de la consulta. Sea a el nombre de ese atributo. El nodo que tomará la decisión de la materialización de una consulta C_i es el responsable de la llave *hash(a)*. Notemos que dicho nodo debe participar de todas formas en la evaluación de C_i porque es quien se encarga del índice del atributo *a* (ver Sección 3.2). Ese nodo lo notaremos P_a en lo que sigue. P_a se basa en un histórico de ocurrencias de consultas para la decisión. No obstante, esta política es susceptible de cambio. Una vez que se decide que la consulta debe ser almacenada, P_a envía un mensaje al nodo que conservará C_i en su caché. Dicho nodo, que denominamos P_{rc}, se determina dinámicamente pues es quien se encarga de *hash(Ci)*.

Coherencia del contenido del caché: La política de reemplazo de entradas en los cachés es LRFU [LCK+01]. Además cada P_{rc} actualiza el contenido de su caché periódicamente. El periodo refleja un tiempo de validez. Si una consulta permanece en el caché dicho tiempo, P_{rc} lanza la ejecución para actualizar la respuesta.

La materialización de consultas con términos de desigualdad permite evitar cálculos y envíos de mensajes. No obstante, no es posible utilizar una consulta C_1, almacenada en el caché, para responder a una consulta C_2, que se encuentre parcialmente contenida en C_1. Por ejemplo, una consulta C_1 : *Año > 1995* en el caché, no servirá para evaluar C_j : *Año > 2005*. La razón es que en el caché se tienen los identificadores de los objetos que forman parte de la respuesta, pero no se tiene el valor del atributo *Año* de cada uno de ellos. Por lo tanto no se pueden filtrar los objetos para eliminar los de *1995 < Año <= 2005*. A continuación se presenta una propuesta que permite mejorar este aspecto.

4.2. Consultas de desigualdad y análisis semántico

En esta sección proponemos una extensión que permite cierta forma de análisis semántico de las consultas para reutilizar las respuestas en el caché. Concretamente, nos interesa manejar los casos de intersección como el de *Año > 1995* y *Año > 2005* que citamos en la sección anterior.

Nuestra propuesta se combina con el uso de índices, estrategia IX (ver sección 3.1), propuestos en *PinS* para evaluar los términos de desigualdad. La idea es aprovechar la información y el rol de tales índices para efectuar el análisis semántico. Presentaremos nuestra propuesta para el caso de un término de desigualdad. La generalización a consultas con varios términos no se desarrollará en este artículo.

Información en el índice: la Figura 4 ilustra parte de la estrategia IX con la que vamos a trabajar. P_{650} es el nodo encargado del índice del atributo año. El nodo P_{150} le envía la consulta C_1 : *año < 1990*. El índice de P_{650} contiene los valores válidos *1987*, *1989* y *2001*. P_{650} envía a P_{150} la lista de los identificadores de los términos *año = 1987* y *año = 1989* que son L_{IDs} : *< 510, 269>*. Esta respuesta le permite a P_{150} evaluar C_1 :. Se observa que en este proceso, P_{650} es informado de todas las consultas de desigualdad sobre el atributo *año* y que él conoce el dominio real de este atributo.

Definición del caché para uso semántico: proponemos una nueva forma de entradas para el caché que per-

mite consultar el valor del atributo de los objetos en la respuesta. Se trata de una variación de las entradas presentadas en 4.1. Su forma es:

$$<idConsulta, Consulta, [Atributo_1 = Valor_1 : idObjeto_{11}, idObjeto_{12}, ...,idObjeto_{1m}], [Atributo_2 = Valor_2 : idObjeto_{21}, idObjeto_{22}, ...,idObjeto_{2m}], ..., [Atributo_n = Valor_n : idObjeto_{n1}, idObjeto_{n2}, ...,idObjeto_{nm}]>$$

La diferencia radica en el último elemento. Los identificadores de objetos están agrupados según el valor del atributo usado en la consulta. En el ejemplo precedente sería *[año = 1987: (idObjetos)], [año = 1989: (idObjetos)]*

Para que se puedan crear tales entradas de caché, se requiere modificar un poco la estrategia IX y el uso de los índices. La respuesta enviada por un nodo encargado de un índice deberá incluir los valores válidos para una consulta dada y no solamente las llaves *(hashcode)* de los términos como se hacia hasta ahora. Cuando una consulta es almacenada en el caché, el nodo N_i, encargado del índice anota esa información. Para esto marca los nodos del índice que participan a la respuesta de dicha consulta. Con esta información es posible contemplar los casos de coincidencia exacta y coincidencia parcial, presentados en la sección 2.2.

Intersección de consultas de desigualdad: consideremos el caso de la intersección de consultas introducido previamente. En el ejemplo que se presenta a continuación se hace referencia a P_a y P_c, quienes corresponden a roles de responsables de índice del atributo y caché de consulta respectivamente. En este caso particular, P_a es asumido por P_{450} y P_c por P_{6850}, pero es importante anotar que su elección dentro del sistema es realizada en forma dinámica.

Sea la consulta C_1 : *a>12* presente en el caché. La entrada es de la forma:

$$E_1 : <6826, a > 12, [a=13:7987, 3223, 5432], [a=22:321, 2342, 4356], [a=420:979, 290, 546]>$$

Sea la $C_2 : a > 20$. El objetivo es evaluar C_2 aprovechando la presencia de C_1 en el caché. La respuesta de C_2 se encuentra contenida en la respuesta de C_1. En este caso la evaluación se inicia consultando el nodo P_a responsable del índice del atributo a. P_a tiene un registro de las consultas de desigualdad sobre a que se encuentran en el caché. *Pa* encuentra en el índice que las hojas marcadas con *a=22* y *a=420* son respuesta para C_2, tal como lo muestra la Figura 5 y de igual manera, conoce que ya se encuentran materializadas en E_1. P_a envía esta información al nodo que solicitó C_2.

Consideremos ahora el caso de una inclusión parcial, que en el contexto del ejemplo anterior, aplicaría para

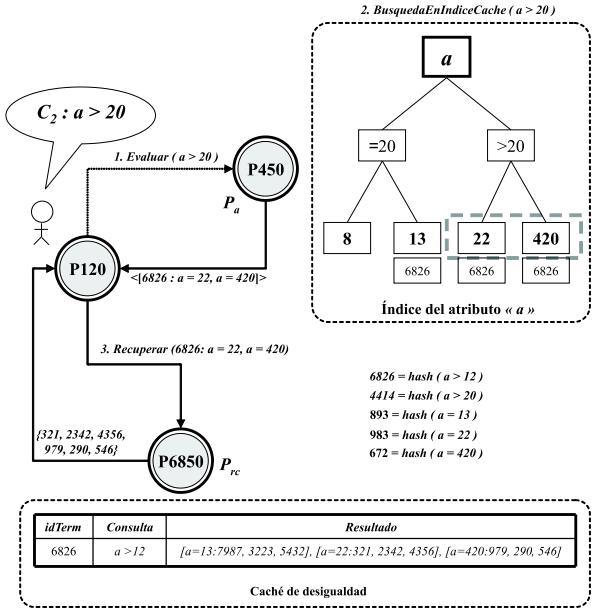

Figura 5. *Ejemplo búsqueda en el caché de consultas de desigualdad*

una consulta $C_3 : a < 25$. La entrada en el caché E_1 que corresponde a $a > 12$, contiene una parte de la respuesta.

Asumamos que el índice posee entradas en a también para los valores *5* y *8*. En este caso, P_a buscará dentro su registro los valores que sean solución para C_3. P_a encuentra que *5*, *8*, *13* y *22* son respuesta y que los dos últimos ya están materializados en E_1. La respuesta de P_a hacia el nodo que solicita C_3, contiene tanto las consultas que se encuentran en el caché, como las que faltan por calcular. Para el ejemplo de la Figura 5, en las nuevas condiciones, el mensaje será:

$$< [null : a = 5], [null : a = 8], [6826 : a = 13, a = 22]>$$

Por lo tanto, el caché puede enviar la parte de la respuesta correspondiente a *13* y *22*. La evaluación para *5* y *8* deberá hacerse de la forma general.

Una optimización posible para una entrada en el caché es el reemplazo de los términos *Atributo=Valor* por el resultado de aplicar la función *hash(Atributo=Valor)*. Esto evita un nuevo cálculo de la función de hash en el nodo que recibe la respuesta, en el caso que haya consultas que no se encuentran en el caché, y permite eventualmente tener entradas en el caché de talla más reducida, dependiendo de la proporción *Consulta / LongResultadoHash*. Esta política provee privacidad al enviar mensajes y al almacenar en el caché entradas que no contienen las consultas de igualdad en claro.

De esta manera, reescribiendo E_1 y con *hash(a=13) = 893*, *hash(a=22) = 983* y *hash (a=420) = 672*, se tiene:

E_1 : <*6826, a* > *12, [893:7987, 3223, 5432], [983:321, 2342, 4356], [672:979, 290, 546]*>

La política de actualización de esta propuesta es la misma que se menciona en la sección 4.1, adoptando un tiempo (T_e) de expiración de consultas. El tiempo T_e representa el tiempo de gracia o de ventaja de utilizar el caché sobre la ejecución de consultas sin la ayuda de un caché.

5. Conclusiones y perspectivas

Los sistemas de caché para sistemas distribuidos son concebidos con propósitos diversos. El desarrollo del presente trabajo, ha permitido encontrar algunos puntos importantes con respecto al uso de los sistemas de caché: su uso no es necesariamente por motivos de disminución del uso del ancho de banda, dado que en sistemas *P2P* basados en *DHT*, el costo del acceso a los nodos es similar, así que los costos que podrían disminuirse son los relacionados con las capas superiores a las capas de enrutamiento y almacenamiento, según el modelo propuesto en [Vil06]. La capa de evaluación de consultas tiene costos de cálculo y cantidad de mensajes que pueden ser mejorados con el uso de un caché de consultas. Precisamente en este tipo de sistemas es una solución bastante más efectiva que el caché de objetos en términos de mejora del desempeño total.

Por otra parte, la adopción de una estrategia de caché con una aproximación semántica permite una explotación más fina del contenido del caché con respecto a las consultas de desigualdad a través del concepto de regiones que pueden ser reutilizadas. Estrategias como el particionamiento y la fusión de regiones dentro del caché pueden convertirse en una mejora para la reutilización del contenido del caché. Sin embargo, es importante llegar a un equilibrio entre el costo de cálculo y el tiempo ganado con respecto al uso de un caché no semántico, dado que finalmente es el nodo encargado del caché quien tendrá que realizar una buena cantidad de procesamiento sobre sus entradas. Estos procesos pueden sobrecargar dicho nodo, perdiéndose así la filosofía de redes *P2P* donde los nodos deberían tener una carga de almacenamiento y procesamiento equitativa.

Hasta el momento según el conocimiento de los autores, no hay trabajos que utilicen un caché semántico para mejorar la eficiencia de consultas de desigualdad en sistemas *P2P* basados en *DHT*. Sin embargo, se considera como trabajo a seguir la implementación de las estrategias descritas junto con sus respectivas pruebas de rendimiento, a nivel teórico y practico, en términos de cálculo, tamaño y cantidad de mensajes, y funcionamiento general del sistema, con el fin de realizar una comparación entre el desempeño de nuestra proposición con respecto a otras estrategias existentes que tengan como objetivo un mejor aprovechamiento del contenido del caché en sistemas *P2P*.

Como perspectivas de trabajo posterior, se contempla la posibilidad de proponer una estrategia de caché semántico independiente del servicio de indexación de datos, con el fin de adaptar la propuesta a entornos heterogéneos en arquitectura e independientes de los sistemas *P2P DHT*. De la misma manera, nuevas alternativas de caché semántico orientadas a contextos móviles, corresponden a una línea de investigación de mucho interés dentro del equipo de trabajo.

Referencias

[CRS99] Boris Childlovskii, Claudia Roncancio, and Marie-Luise Schneider. Semantic cache mechanism for heterogeneous web querying. In *WWW '99: Proceeding of the eighth international conference on World Wide Web*, pages 1347–1360, New York, NY, USA, 1999. Elsevier North-Holland, Inc.

[IRD02] Sitaram Iyer, Antony Rowstron, and Peter Druschel. Squirrel: a decentralized peer-to-peer web cache. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 213–222, New York, NY, USA, 2002. ACM Press.

[JFY05] Yuh-Jzer Joung, Chien-Tse Fang, and Li-Wei Yang. Keyword search in dht-based peer-to-peer networks. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 339–348, Washington, DC, USA, 2005. IEEE Computer Society.

[KNO+02] Panos Kalnis, Wee Siong Ng, Beng Chin Ooi, Dimitris Papadias, and Kian-Lee Tan. An adaptive peer-to-peer network for distributed caching of olap results. In *SIGMOD Conference*, pages 25–36, 2002.

[Lau06] Laurent d'Orazio and Olivier Valentin and Fabrice Jouanot and Yves Denneulin and

Cyril Labbé and Claudia Roncancio. Services de cache et intergiciel pour grilles de données. In *Proceedings of BDA 2006, conférence sur les Bases de Données Avancées*, Lille, oct 2006.

[LCK+01] D. Lee, J. Choi, J. H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim. Lrfu: A spectrum of policies that subsumes the least recently used and least frequently used policies. *IEEE Trans. Comput.*, 50(12):1352–1361, 2001.

[LHH+04] "Boon Thau Loo, Joseph M. Hellerstein, Ryan Huebsch, Scott Shenker, and Ion Stoica". Enhancing p2p file-sharing with an internet-scale query processor, "2004".

[RDK03] Qun Ren, Margaret H. Dunham, and Vijay Kumar. Semantic caching and query processing. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):192–210, 2003.

[SA06] Gleb Skobeltsyn and Karl Aberer. Distributed cache table: efficient query-driven processing of multi-term queries in p2p networks. In *P2PIR '06: Proceedings of the international workshop on Information retrieval in peer-to-peer networks*, pages 33–40, New York, NY, USA, 2006. ACM Press.

[Sch01] R. Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *P2P '01: Proceedings of the First International Conference on Peer-to-Peer Computing (P2P'01)*, page 101, Washington, DC, USA, 2001. IEEE Computer Society.

[Vil06] M. Villamil. *Service de localisation des données pour les systèmes P2P*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, Francia, 2006.

[VRL06] M. Villamil, C. Roncancio, and C. Labbé. Range Queries in Massively Distributed Data. In *Proc. Int'l WS on Grid and Peer-to-Peer Computing: Impacts on Large Scale Heterogeneous Distributed Database Systems*, Cracovie, Pologne, Septembre 2006.