Esteban Meneses. University of Illinous at Urbana-Champaign

Tablas de Hash Distribuidas

CARLOS MARTÍN FLORES GONZÁLEZ, Carné: 2015183528

Instituto Tecnológico de Costa Rica
Maestría en Computación
Sistemas Operativos Avanzados
Profesor: Francisco Torres Rojas, Ph.D

Los sistemas *peer-to-peer*(P2P) conforman una de las tendencias más importantes en la actualidad en el campo de la computación distribuida. En contraposición con los sistemas distribuidos tradicionales, donde los clientes recurren a los servidos para obtener alguna funcionalidad, en los sistemas P2P algunas o todas las funciones de los servidores son reemplazadas por funciones en los clientes. Esto hace que los sistemas P2P disfruten, en general de dos características deseable: la escalabilidad y la tolerancia fallos.

De acuerdo a la estructura que subyace dicha implementación, podemos tener varios tipos de sistemas P2P. Algunos sistemas se denominan *centralizados*, debido a que los clientes envían sus consultas a servidores centralizados. Otros sistemas son *descentralizados* porque no tiene un servidor central y más bien los clientes forma una red *ad hoc* y envían sus consultas a sus otros pares. Entre estos sistemas existen los *estructurados*, en donde hay una clara relación entre la topología del sistema P2P y la ubicación de los datos, y también existen los *no estructurados* que no imponen ninguna condición para ubicar datos de acuerdo a la topología.

Sistemas P2P No-estructurados. Se dicen no estructurados porque no hay restricción alguna del lugar donde un archivo particular debe ser almacenado. La red de nodos que subyace el sistema se puede modelar como un grafo dirigido $G = (N, E)$ donde N es el conjunto de nodos o computadoras que forman el sistema, E se refiere a los enlances en las computadoras. Asi pues, si $e_{ij} = (n_i, n_j) \in E$ entonces el nodo n_j está en la lista de pares del nodo n_i. Se denomina P_i como dicha lista. Cada nodo $n_i \in N$ contiene una lista de llaves L_i que representa los objetos contenidos en ese nodo. Una tabla *hash* distribuida H asocia una llave l_x con una serie de nodos en N. El sistema *Gnutella* elimina la necesidad de servidroes dedicados para implementar una tabla de *hash* distribuida. La especificación original permite a una red de nodos compartir archivo y hace que cada nodo tenga la dosble funcionalidad de ser cliente y servidor al mismo tiempo (un *servent*). Todo nodo n_i almacena sus llaves L_i, así como una lista de pares P_i. La colección de nodos forma entonces un grafo dirigido. La colección de nodos forma entonces un grafo dirigido. La manera en que se crean los enlaces entre nodos no debe seguir una estructura particular. Si un nodo nuevo k quiere ingresar a la red, típicamente contactará algún nodo conocido y le pedirá una serie de pares para formar P_k. Con el paso del tiempo el conjunto P_k irá cambiando de acuerdo a al fallo de nodos y la dinámica de la red. En *Gnutella* es posible enviar cinco tipod de mensajes entre nodos: (1) `Query`: para inciiar la búsqueda de una llave l_x. (2) `Query hit`: responde a una consulta. (3) `Ping`: para determinar si otro nodo de la red está vivo. (4) `Pong`: respuesta al mensaje de *Ping*. Normalmente, este mensaje también contien la dirección de otro par, de forma que el nodo destindo pueda actualizar su conjunto P_i. (5) `Push`: utilizado para iniciar la transferencia de un archivo. **¿Cómo se resuelve una consulta?** Cuando un nodo está interesado en obtener un archivo, envía un mensaje de `Query` a sus paresy de allí en adelante el mensaje será esparcido por la red. Sin embargo, todo mensaje `Query` tiene un valor que limita el numero de *hops* que puede visitar. Este

valor es conocido como **TTL** y es una parte fundamental de *Gnutella*. Entre mayor sea este valor más probable es encontrar el archivo, pero se le ipoene mucha más carga a la red. La selección del **TTL** debe ser una balance entre esos dos factores. Cuando un el mensaje `Query` llega al nodo que tiene el archivo que se está buscando, este empezará la ruta de vuelta con el mensaje `Query hit` hasta llegar al nodo que inició la búsqueda. Una vez que esto sucede, el nodo que inició la búsqueda, de nuveo envía un mensaje de `Push` al nodo destindo para el envío del archivo. Los mensajes `Ping` y `Pong` sirven para determinar cuáles de los pare de un nodo particular siguen con vida en el sistema. Tipos de nodos: (1) *freeloaders:* solamente están disponibles en el sistema por una pequeña fracción de tiempo, mientras bajan algunos archivos de otros nodos. (2) *supernodos:* que muestran más estabilidad en su participacio4n y por ellos tienen asociadas responsabilidades especiales. Muchas redes presentan en cierto grando la *ley de la potencia*. ESto quiere decir que la probabilidad de que un nodo tenga un número de vecinos igual a x es directamente proporcional a $\frac{1}{n^k}$, donde k es el parámetro de distribución. Por otro lado las redes *Gnutella* siguen el patrón de redes sociales que muestran un alto *coeficiente de clustering*. Este coeficiente determina para cada nodo n_i qué tan cerca están él y sus vecinos de forma un subgrafo totalmente desconectado. Dado el grafo dirigido $G = (N, E)$ y un nodo particular $n_i \in N$, definimos su vecindario V_i como el conjunto de nodos n_j tales que (n_i, n_j) o (n_j, n_i) pertenece a E. Ahora, si el nodo n_i tiene k_i vecinos, entonces habrá a lo sumo $k_i(k_i - 1)$ enlaces entre ellos. El coeficiente de clustering CC del nodo n_i es la fracción del número de enlaces que existen en el vecindario V_i entre el máximo número posible de enlaces en ese mismo vecindario:

$$CC(n_i) = \frac{|\{e_{j,k}\}|}{k_i(k_i - 1)}$$

donde $\{e_{l,k}\}$ es el conjunto de enlaces (n_j, n_k) tales que los nodos n_j y n_k pertenecen a V_i y $(n_j, n_k) \in E$. Finalmente para calcular el coeficiente de clusterin del grafo, se debe tomar el promedio de cada nodo:

$$CC(G) = \sum_{n_i \in N} \frac{CC(n_i)}{|N|}$$

.

Sistemas P2P Estructurados. Una de las principales desventajas de los sistemas P2P no-estructurados es el alto nivel de tráfico que genera en la red el proceso de esparcir una consulta. Los sistemas P2P estructurados logran minimizar el número de mensajes requerido para satisfacer una búsqueda, pero imponen un régimen menos flexible en cuando a la ubicación de los objetos. Para implementar eficientemente la tabla de *hash* distribuida H, los sistemas estructurados recurren a la asignacio4n de identificadores (*IDs*) tanto a los nodos de la red, como a los objetos. Por medio de una función consistente de *hash*[1] se puede asociar cada nodo en la red con un número 160 bits, por ejemplo. La función de *hash* debería recibir como parámetro el IP y el puerto del nodo para obtener su *ID* respectivo. Sin embargo, para el caso de los sistemas P2P estructurados, los identificadores se truncan en m bits, así que los identificadores de los nodos tienen un valor entre 0 y $2m - 1$. Un valor apropiado de m hará improbable la colisión por los IDs. Finalmente, todos los IDs pueden representarse como puntos en un círculo, de manera que se tiene un espacio virtual circular donde se representan todos los nodos. Uno de los sistemas P2P estructurados más conocidos es *Chord*. Este sistema utiliza el espacio virtual circular y le asocia a cada nodo y a cada objeto un identificador de m bits. Con $m = 5$, cada nodo tiene un *ID* de 5 bits, lo que deja un espacio para 32 posibles nodos del sistema. La estructura de *Chord* proviene de los apuntadores que cada noso contiene (o sea, de P_i). Primeramente, todo nodo mantiene la dirección del nodo *sucesor* y el nodo *antecesor* en el espacio particular. Además cada nodo almacena una *tabla de dedos* de tamaño m. Dado un nodo i con $IDid_i$, la $j-$ésima entrada de su tabla de dedos, denotada por $d[j]$, contiene el nodo con el menor ID tal que sea mayor o igual a $n + 2^j$ (mód 2^m). La tabla empieza a indexarse desde 0. Nada previene que varias entradas se repitan en la tabal. Si un

[1] como SHA-1 o MD5

objeto tiene un *ID* igual a x, entonces ese objeto será almacenado en el nodo cuyo *ID* sea el menor *ID* mayor o igual a x. **¿Cómo resuelve la tabla H de *Chord* una consulta?** Bien, pues si un nodo está interesado en un archivo entonces lo primero que hace es revisar si él mismo contiene un archivo con ese *ID*. Si no, se utiliza la tabal de dedos para hacer *routing* de la consulta. Se busca en la tabla de dedos el mayor dedo que sea *menor* o igual al *ID* del archivo.Cuando no es posible encontrar en la tabla de dedos una entrada con la propiedad deseada, simplemente se envía la consulta al sucesor del nodo. Por otro lado, el sistema *Chord* ejecuta periódicamente el *protocolo de estabilización*, que está destinado a lidiar con la dinámica del sistema, en el que los nuevos nodos se integran y otros falla o dejan el sistema. Este protocolo actualiza las entradas en las tablas de dedos y garantiza que no habrá ciclos en el sistema de apuntadores.

Resultados. *Operación de búsqueda:* Napster emplea un tiempo constante, debido a que la consulta solamente necesita dirigirse a un grupo de servidores. *Gnutella* va a requerir N unidades de tiempo, utilizando un valor de **TTL** mayor o igual a N. *Chord* limita logarítmicamente el costa de una búsqueda. *Número de mensajes:* se repite el mismo comportamiento. *Cantidad de memoria:* Napster requiere potencialmente $O(N)$ en el servidor. *Gnutella* podría almacenar todos los otros nodos en el sistema. *Chord* solamente requiere de recordar las entradas de las tablas de dedos.

Sistema	Tiempo de búsqueda	Mensajes por Consulta	Memoria
Napster	$O(1)$	$O(1)$	$O(1)$ en nodo $O(N)$ en servidor
Gnutella	$O(N)$	$O(N)$	$O(N)$
Chord	$O(\log N)$	$O(\log N)$	$O(\log N)$

Cuadro 1. Eficiencias de Operaciones en distintas tablas de *hash* distribuidas

Coeficiente de clustering: Gnutella aparece consistentemente en segundo lugar y la estrategia aleatoria es la última. El grafo G generado por *Chord* provee la versión más ordenada, debido a que existe una forma de conectar los nodos. Por su parte, *Gnutella* muestra cierto nivel de orde, porque los nodos tienden a tener vecinos qeu también sus vecinos poseen, pero, aún asi, hay un factor aleatoria en la conexión. Finalmente, el grafo aleatorio no provee ningún tipo de ordenamiento. Por otro lado, a mayor número de nodos menor valor del coeficiente de *clustering*. Esto resulta claro después de considera que el número de enlaces es constante, pero hay más nodos en el sistema. *La longitud promedio del camino más corto:* la estrategia aleatoria ofrece la mejor opción (menos longitud), mientras que *Chord* está de último en esta métrica. *Gnutella* conserva una posición intermedia, pero más cercana de la estrategia aleatoria. La explicación de este fenómeno proviene del hecho que conexiones aleatorias ayudan a a minimizar la distancia entre nodos en un grafo. Los resultados dejan a *Gnutella* como una estrateia que provee un valor aceptable de *clustering* y también un valor pequeño en la longitud del camnio más corto. Sin embargo, estas características no son aprovechadas por el protocolo *Gnutella* y mas bien son características emergentesde las redes sociales que las redes en *Gnutella* parecen emular.

Sistemas estructurados: Número total de mensajes enviados pro el sistema cuando se hizo una consulta aleatoria. Chord logar un número ínfimo de mensajes requerido para ubicar un objeto, si se compara con el número de mensajes que se necesitan en *Gnutella.*

1. ¿CUÁL ES EL PROBLEMA QUE PLANTEA EL *PAPER*?

Los sistemas P2P modernos están basados en la abstracción de una tabla de *hash* distribuida. Esta estructura de datos permite asociar una llave particular con varios nodos en la red. En general, la implementación de las tablas *hash* distribuidas se pueden clasificar en dos grupos: sistemas P2P estructurados y no-estructurados. El artículo presenta una revisiòn de estas dos técnicas basadas en casos representativos de cada una de ellas.

2. ¿POR QUÉ EL PROBLEMA ES INTERESANTE O IMPORTANTE?

Aunque la mayoría de los sistemas P2P de la actualidad son utilizados para compartir archivos o para ofrecer canales de televisión, existen muchas otras aplicacioens: sistemas de administración de almacenamiento, juegos interactivos con múltiples jugadores, sistemas de referencias bibliográficas, sistemas de *caching* para la Web, distribución de contenido, entre otras.

3. ¿QUÉ OTRAS SOLUCIONES SE HAN INTENTADO PARA RESOLVER ESTE PROBLEMA?

Aunque el inicio de la computación P2P se remonta a la creación del sistema de servidores de noticias *Usenet* en 1979, fue Napster quien revivió el tema 20 años después. Napster fue uno de los primero sistemas de uso masivo para compartir música y llegó a tener millones de usuario en sus dos primeros años de existencia. El sistema original debió ser clausurado debido a serios problemas legales por violación de los derechos de copia. Sin embargo su legado ha permitido el desarrollo de muchos otros sistemas basados en ideas P2P. Napster permitía resolver consultas sobre títulos de canciones. En su forma más básica, la funcionalidad era encontrar una lista de nodos en el sistema que tuvieran el archivo de música buscado. El sistema original de Napster no era puramente P2P, porque mucha de la funcionalidad todavía dependía de los servidores.

La mayoría de sistemas P2P comerciales (*Kazaa, iMesh, BitTorrent*) corresponden a implementaciones de tablas de *hash* distribuidas no-estructuradas.

Muchos sistemas P2P de la academia son estructurados. Por ejemplo, *Pastry* y *Tapestry* también están basados en un espacio circular, pero utilizan técnicas diferentes para hacer el *routing* de la consulta. Estos dos sistemas usan variantes inspiradas en la búsqueda de prefijos. Otros sistemas, como *Content Addressable Network*, más bien recurren a un plano para asociar a un plano para asociar nodos y objetos a punto en el plano. Otro sistema es *Kelips* que crea grupos de nodos y asocia objectos a los grupos.

4. ¿CUÁL ES LA SOLUCIÓN PROPUESTA POR LOS AUTORES?

Los sistemas P2P se sustentan en una abstracción particular llamada *tabla hash distribuida.* Así como una tabla *hash* asocia llaves con objetos y permite insertar, buscar y borrar eficientemente, una tabla *hash* distribuida tiene las mismas funcionalidades y puede entonces modelar una tabla *hash* en una red de computadoras. Por ejemplo Napster, es una tabla *hash* que asocia nombres de archivos con nodos de la red que tienen el contenido de esos archivos.

5. ¿QUÉ TAN EXITOSA ES ESTA SOLUCIÓN?

En *Gnutella* hay menos restricciones para construir las redes y almacenar objetos. Se ha demostrado que estos sistemas siguen las características de las redes sociales, lo que significa qeu se pueden aprovechar estas propiedades para mejorar mucho de los procesos que se siguen en *Gnutella.* En *Chord*, los objetos del sistema deben ser colocados en los nodos de acuerdo al identificador del objeto. Esto hace que el número de mensajes que se necesita para resolver una consulta sea mucho menor que en sistemas no-estructurados.