CS 559: Algorithmic Aspects of Computer Networks **Fall 2007**

Lecture 14 — October 24

Lecturer: John Byers *BOSTON UNIVERSITY* *Scribe: Yin Chen*

In today's lecture, we continued our discussion on **Consistent Hashing & its applications**, which appears in *Karger et.al.'97*. The key ideas from this paper are:

- Random trees
- Consistent hashing

14.1 Random Trees

We consider a model with full information (static), i.e., users know about the structure, and every cache is present. For each page, we associated a K-ary tree of size C (C =total # of caches in system). Then we conduct a random assignment (permutation) that maps each cache to a node in the abstract tree, as shown in the Figure 14.1 and 14.2.

Access protocol for users:

When a user needs a page, he does the following:

1. Pick a random[1] leaf-root path.
2. Inquire bottom-up: if a cache has the page - return; otherwise increment counter for the page, inquire parent.
3. Associated with each cache there is a threshold q. If the a cache sees too many requests for a certain page it didn't have, i.e., the counter reaches q, it caches a copy of the page[2].

* *Note: (1)The choice of leaf also depends on views of different users. (2)In either case the cache passes the page on to the requester when it is obtained. (3) An advantage of the tree structure: a page doesn't have to be replicated too many times before its requests are met.*

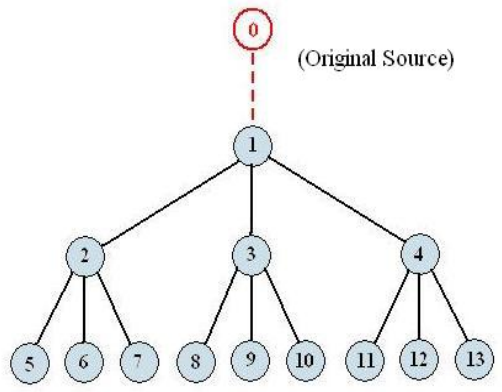

Figure 14.1. Example of a 3-ary abstract tree. The numbers in the nodes are labels for cache positions in the tree.

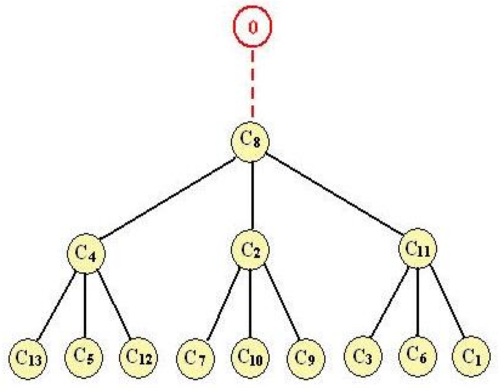

Figure 14.2. The tree formed by actual caches.

Views:

A *view* is a subset of caches a particular user knows. For example a user's view can be $\{C_6, C_3, C_8\}$. A user does the following:

$$\text{hash}\{\text{position, page}\} \rightarrow \{\text{caches in his view}\}$$

**Note: try to pick a leaf-root path that contains as much as possible the caches in his view may not be a good idea for a user, since if many users know the same cache, this cache naturally becomes a hotspot.*

14.2 Consistent Hashing

Let $\mathcal{I}$ be the set of items and $\mathcal{B}$ be the set of buckets. A view $\mathcal{V}$ is any subset of the buckets, i.e., $\mathcal{V} \subseteq \mathcal{B}$. The hash function conducts: Items $I \rightarrow position \times URLs$

14.2.1 Key Definitions

We now state and relate some resonable notions of consistency:

1. ***Balance***: For a particular view, with high probability: Pr[an item is hashed to a bucket in the view] $\sim O(\frac{1}{|V|})$.

2. ***Monotonicity***: For $V_1 \subseteq V_2 \subseteq B$ if $h_{V_2}(i) \to b, b \in V_1$, then $h_{V_1}(i) \to b$ (This is an iff)

3. ***Spread*** (with respect to views that are substantial - $|V_j| \geq \frac{|B|}{t}$): Spread of item i is

$$\sigma(i) = |\{h_{V_j}(i)\}_{j=1}^{|V|}|$$

Example: for a specific item i, different views may associate it with different buckets. In this case V_1 maps it to bucket 10 while other views map it to bucket 9. Therefore $\sigma(i) = 3$.

View	V_1	V_2	V_3	V_4
Bucket	10	9	9	9

4. ***Load*** (with respect to bucket): Set of items assigned to b in a view. It keeps a count of items that a bucket is responsible for a view (any view).

$$| \cup h_V^{-1}(b)|$$

Example:

	V_1	V_2	V_3
i_1	3	3	4
i_2	4	5	3

Load $(b_3) = 2$
Load $(b_4) = 2$
Load $(b_5) = 1$

14.2.2 Techniques

Recall from last lecture where we hashed the items and buckets to $[0, 1]$ and map each item to the nearest bucket. Now we slightly modify the strategy:

1. Each item is hashed once $\to [0, 1]$.
2. Replicate buckets d times, then hash b buckets $\to [0, 1]$.
3. For each replica, map each item to nearest bucket.

* *Note: A bad view may have different buckets each have a replica close to an item.*

Figure 14.3 and 14.4 show a simple example for 3 items and 2 buckets, with replication $d = 2$.

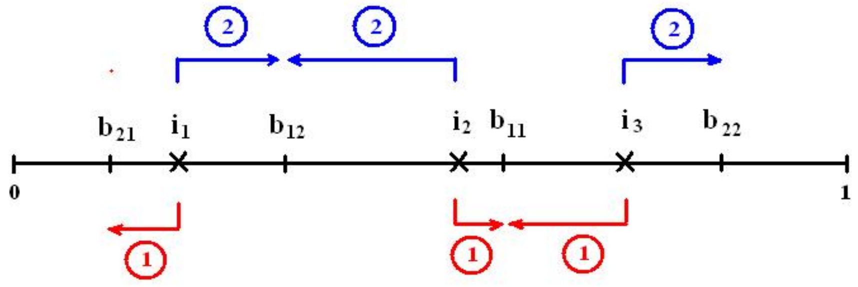

Figure 14.3. Example of hashed items and buckets for $d = 2$, where ①,② denote the first and second rounds of mapping items to buckets; i_1, i_2, i_3 are the items; $b_{ij}, i, j \in \{1, 2\}$ is bucket i hashed in the j^{th} round.

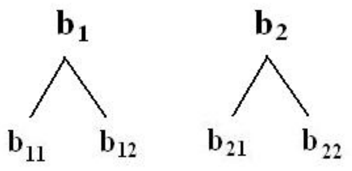

Figure 14.4. For $d = 2$, each bucket is hashed twice.

1. look at b_{11}, b_{21} : $i_1 \rightarrow b_{21}, i_2 \rightarrow b_{11}, i_3 \rightarrow b_{11}$
2. look at b_{12}, b_{22} : $i_1 \rightarrow b_{12}, i_2 \rightarrow b_{12}, i_3 \rightarrow b_{22}$

Analysis: Assume every view is substantial, i.e., the size of set $\mathcal{V}$ is at least $1/t$ of the size of set $\mathcal{B}$. Then for a particular user, an item fails to fall in any bucket in his view is:

$$Pr[miss] = \left(\frac{t-1}{t}\right)^d = \left(1 - \frac{1}{t}\right)^d \sim e^{-d/t}$$