An Implementation and Analysis of the Skip Graph Data Structure

Jonathan Kirsch
Advisor: Arvind Krishnamurthy
September 25, 2003

1 Background

Over the past four or five years, Peer-to-Peer (P2P) systems have become increasingly popular and widespread. While there is some debate regarding the precise definition of “peer-to-peer,” we define such a system as a collection of loosely coupled nodes which interact for the purpose of sharing some resource. This resource could be anything from a data file to CPU processing time.

The first surge of P2P systems arrived around 1999, and included Napster, Gnutella, and Freenet. All three systems used different models and protocols for efficiently finding the location of a given resource: Napster maintained a centralized indexing server, Gnutella used a flooding algorithm in a decentralized environment, and Freenet used routing-based lookups. Each model possessed considerable limitations. For example, Napster’s centralized scheme limited scalability and created a single point of failure – if the server went down, the system was unable to process queries. These first systems help us to define some of the goals in the design of a P2P system:

**Scalability.** The system should not be limited to a threshold number of nodes. This ties in closely with the notion of performance, in that as the number of nodes participating in the system increases, the system should still be able to achieve good performance.

**Efficiency.** A query should be handled in such a way that it minimizes message costs, response time, and the use of system resources.
**Decentralization**. A system where each node is, in some sense, “equal,” is more robust and can handle node failures gracefully. This is especially important in a distributed environment in which nodes can crash, or more generally leave and enter the system, in a dynamic and unpredictable fashion.

Around 2001, the “next generation” of P2P systems was developed, including Chord, CAN, and Tapestry. These systems are based on the notion of a distributed hash table, where the location of a given resource can be found via a known hashing function. For example, Chord hashes both nodes and keys (representing the shared resource) to the same ID space. To perform a lookup query, a node hashes the value of the desired resource to obtain the location of the node in the system who has that resource (or knows where to find it). Chord provides an excellent example of the tradeoff between the amount of state that a given node must maintain, and the efficiency of the lookup procedure. It handles this tradeoff by using a cached routing scheme, so that both are $O(\log N)$, where $N$ is the number of nodes in the system.

While the distributed hash table scheme provides efficiency and a low degree of required state, one of the primary drawbacks is that any resemblance among keys is destroyed by the hashing function. This makes complex queries, such as range queries, inefficient because the system does not use any notion of “key locality,” where two similar keys would hash to nearby locations. Furthermore, the level of load-balancing is dependent on the uniformity of the hashing function.

With these limitations in mind, James Aspnes and Gauri Shah developed a novel data structure known as a “skip graph,” which is a generalization of the skip list. Like a skip list, a skip graph is a randomized balanced tree data structure. The primary difference is that each “level” can now have several skip lists. As a result, resource location and the dynamic addition and deletion of nodes in the system can be achieved in logarithmic time, with each node (like in Chord) requiring only logarithmic state about its neighbors. Such redundancy also allows for node connectivity amidst the failure of a considerable ratio of nodes. Furthermore, since nodes and keys are not hashed, similar keys can be stored close to each other – thus, skip graphs support complex queries. Aspnes has also proposed a load-balancing scheme which affects the way in which nodes are inserted or deleted from the system.
2 Proposal

I propose to undertake a detailed study of the skip graph data structure and its associated algorithms. This includes reading background papers on the distributed hashing scheme, and analyzing the contents of Aspnes' paper. After understanding the intricacies of the data structure, I intend to write a simulator to test the sequential performance capabilities of skip graphs. I then intend to implement a prototype system which utilizes the skip graph algorithms using the PlanetLab system. Finally, I will write a paper describing my results. The deliverables therefore include:

- A list of papers read
- Source code for the simulator and prototype system
- Paper describing the results