Nathan Matsuda
CS443
2/5/07

1. Paper title and its author(s).

Efficient Replica Maintenance for Distributed Storage Systems
Byung-Gon Chun,† Frank Dabek,? Andreas Haeberlen,‡ Emil Sit,? Hakim Weatherspoon,†
M. Frans Kaashoek,? John Kubiatowicz,† and Robert Morris?

2. Brief one-line summary.

The paper describes a strategy for maintaining durable replicas of data across a wide internet storage system while reducing the overhead of bandwidth used in moving replicating data.

3. A paragraph of the most important ideas: perhaps a combination of their motivations, observations, interesting parts of the design, or clever parts of their implementation.

The overall goal of the paper is to develop a protocol, given an average failure rate and amount of burstiness for failures, that will maintain durable copies of immutable data. That is, there must exist a copy of the data somewhere on the system, even if the storage node is powered down or otherwise disconnected. To do this, copies must be made faster than their hosting nodes fail. Other implementations incur more than two times overhead compared to a theoretical optimum system that replicates exactly when needed. This overhead is incurred by mistaking transient failures, where a node disconnects or reboots, but the data is not lost, for an actual failure.
Using the planetLab data, the authors show that with the bandwidth cap imposed by planetLab, one nodes data can be replicated roughly three times per year. As a result, if all three nodes containing that data fail in a year, the data will be lost, because replication cannot keep up. However, it is unlikely that this would happen - this would be an example of a burst. The authors point out that on a wide internet storage system, failures can be assumed to be unconnected given geographical and hardware differences. A replication rate has to be chosen to take into account the bandwidth limitations and the maximum concurrent failure rate (burstiness).
A transient failure cannot be distinguished from an actual failure, so replication procedures cannot be altered to fit both types. Instead, the authors propose that the data on transiently-failed nodes, when they come back online, will be counted as a normal replica, instead of being forgotten. The system then must track the location of all nodes containing a replica, even if those nodes are offline. When a node goes offline a replica is made just in case it was a true failure. If the node comes back, then there is one more replica so the priority of making replicas for that data goes down. Data with more replicas than necessary have low priority and data with not enough replicas to ensure durability are given high priority. The system uses a DHT where returning nodes will pop up at the same virtual location, eliminating the need for explicit tracking of data replicas. However, they note that the monitoring overhead for the DHT can limit scalability. As a countermeasure, Carbonite nodes keeps track of some information about other local nodes that don’t have copy of their data so that when a failure occurs, suitable hosts for a new replica are at hand without the cost of rediscovering these nodes.

The authors compare Carbonite to several other implementations. The test used planetLab node failure traces to drive an event-based simulation. Synthetic traces were also used for higher failure probabilities. It performs better in simulation than systems that use simpler methods for dealing with transient failures (like timeouts - if a node is not available for a certain period, it must have failed), but does not perform better than systems that actually know when a node has failed or not (Oracle). Other systems are mentioned in the related work, but the authors use them to point out how reintegrating replicas from transient failures has not been explored and eliminates the need for birth-death rate estimations. This paper also only looks at immutable data, so other back systems are not applicable because they deal with constantly changing data. Some other systems are designed for large-scale correlated failures, which Carbonite is not concerned with.

4. A paragraph of the largest flaws; maybe an experiment was poorly designed or the main idea had a narrow scope or applicability. Being able to assess weaknesses as well as strengths is an important skill for this course and beyond.

While the authors' tests do compare against other systems, no tests are done on the actual planetLab test bed. Moreover, the simulations themselves are the same size as planetLab, so there is no indication on how well the scheme scales. The authors themselves note that some of the implementation details, particularly surrounding the use of DHTs, might pose problems with large scales.
The analysis data shows no indication about how burstiness and failure rate actually affect the percentage of data lost. This seems like an incredibly important set of numbers to evaluate if this system works well. It would have been good to include graphs showing percent data lost v.s. different average and burst failure rates.

5. A last paragraph where you state the relevance of the ideas today, potential future research suggested by the article, etc.

As the authors note in their conclusion, the low cost and ready availability of both storage hardware and internet connections make it seem like wide-scale storage is going to be increasingly useful. However, until systems are developed that can ensure that user data will not be lost while using bandwidth effectively, widespread use of internet storage systems may not catch on.