Efficient Replica Maintenance for Distributed Storage Systems

The area of focus of the Carbonite algorithm is durably and cost-efficiently storing of immutable object in a system that aggregates the disks of many Internet nodes. Hence, Carbonite is a replication algorithm for large storage systems. More.

To ensure availability (immediate access to data) of data, replication systems responds to disk failure by creating new replicas. When availability is key, replication is immediately used regardless if the failure was a disk failure or just a transient (temporary) failure. The problem of this approach is that constantly making new replicas is costly, and do not scale to large systems. The developers of Carbonite realized that Internet users can tolerate some unavailability as long as they eventually will be able to view what they requested. This knowledge is the main motivation behind Carbonite. With main focus on durability, not availability, they could build an Internet scale replication system. Need to talk about reintegration and replication faster than failure, but I don't know if it should be done here or in the technique-paragraph.

As mentioned above, being able to maintain large storage application systems require sophisticated techniques for when and how replication is performed. Carbonite only makes replications in respond to failure if the number of replicas for a given object is below the threshould r_L, where r_L is the number of replicas needed to survive a burst of failures. This property is enforced by reintegration of failed nodes when they come back up again. By remembering which replicas was stored on a node, they can be reused, and thereby make sure that replication only is performed when the number of replications is below r_L, this technique drastically reduces the number of replications required. Another technique to reduce the number of replications made as responds to transient failures is timeouts. By waiting a given amount of time before creating a new replica, the node might come back, and cancel the planned replication.

Carbonite is tested using the PlanetLab testbed a synthetic trace and the results are compared against Cates and TotalRecall as well as an oracle that is knows if a failure is transient and thereby only creates replications in case of disk failure. For both test cases, Carbonite outperform Cates and TotalRecal, and is only beaten by the oracle, the perfect hypothetical system. Carbonite creates 44% more data then the oracle when when keeping 1T of data durable. Results also shows that as long as the target number of replicas r_L is well chosen, the system is durable. However, they do not have any results on availability which would be interesting to look at, this is justified based on the observation that users can tolerate some unavailability, but should be included because users will not tolerate too much unavailability either.Think I should remove this remark on availability to the last section.

Carbonite can be applied to systems that require durably storing of large amounts of data. As they mention, the systems must be able to handle some unavailability. For instance, they mention that news paper readers can tolerate some unavailability as long as they are able to read the news article at some point.

Carbonite in compared against different systems and the results in the paper looks promising when it comes to durability in a large scale system. However, as mentioned above, the paper does not present any results of availability of the system. Even though durability might be a more practical and useful goal than availability for this kind of system, it will be useless if non of the object are ever available. While the research presented is promising it is yet to be tested on a real-world application. A few assumptions, that might not be that realistic, such as assuming that the disks fail independently, no shared bottlenecks, and that available nodes are reachable from all other nodes, should be tested. The good think is, that the authors have acknowledge the need for real-world testing and have developed prototypes of different applications for which they promise to report on long-term experience with.