CSC2231 – Internet Systems and Services

Paper Review #2 – Lessons from Giant-Scale Services
Name: Alex Wun
Date: Sep. 14, 05

The prevailing message in Brewer's paper is that failures are unavoidable. Hence, Giant-Scale Services must always be designed with fault tolerance and graceful degradation in mind in order to continue providing a reasonable user experience in spite of failures. Brewer introduces the DQ (Data per query x Queries per second) value, which represents the throughput demand on a system. Various fault-tolerant techniques used by industry can be described in terms of balancing D and Q in different ways. The paper argues that replicated data storage is superior to partitioned storage in systems that service mostly read queries (the majority of Giant-Scale systems) by preserving D, the available data per query. Additionally, maximizing uptime is best achieved by focusing efforts on improving the Mean-Time-To-Failure rather than the Mean-Time-Between-Failures. Finally, arguments are made for using smart-clients as a simple means to deal with fail-over situations.

The key strength of this paper is the introduction of several terms that can be used to describe the benefits of various fault-tolerant techniques. By clearly defining MTTR, MTBF, uptime, yield, harvest, and the DQ value, Brewer is able to concisely analyze methods for maintaining availability and performance in the face of failures. The concept of system design using the DQ principle clearly extends beyond the example techniques presented in the paper. Another strength of the paper is that the presented techniques are highly practical – either grounded by experience or a careful DQ analysis. The most prominent technique is the "big flip" system upgrade process that incurs a 50% DQ loss to facilitate complex system-wide changes.

Unfortunately, as Brewer admits, the DQ principle is only appropriate for data-transfer heavy systems. Systems involving little external I/O must be analyzed using different techniques altogether. Another drawback of the paper is the tendency towards using "smart-clients". Putting fault-tolerance awareness behaviour into clients increases the coupling between the clients and the service system – a generally undesirable result since users will be required to install and upgrade custom software. Simplifying fault-tolerant behaviour within the systems comes at the expense of burdening the clients. Finally, the paper could be improved if the concept of a "big flip" upgrade was presented as a special case of a "rolling group upgrade". Whereas a "rolling upgrade" upgrades one node at a time, a "rolling group upgrade" would upgrade a certain group of nodes at a time. This is especially useful if the architecture of the system is such that the nodes are already organized as a collection of sub-clusters. The "big flip" is simply a special case when the system is thought of as two joined clusters.

Overall, Brewer introduces a powerful technique for evaluating system design under fault-tolerance considerations. The techniques presented are fairly intuitive from a systems viewpoint, but the distillation of concepts into a common terminology successfully formalizes the process of designing systems to handle failures.