Lessons from Giant-Scale Services

Submitted by Ravi Prakash Giri (rgiri8)

The author of the paper has defined several giant-scale services and the tools for designing and analyzing the giant-scale services. The author further proposed the use of harvest, yield and DQ principle to quantify availability. According to the author, combinations of replication and partitioning are tools useful for graceful degradation, disaster tolerance, and online evolution. The paper discusses principles for design and management of high availability of giant-scale services. The paper surveys the existing methods to manage giant-scale internet services that are remote from the user and are driven by read-queries. The main idea of the paper is `failures are unavoidable` hence giant scale services are always designed with fault-tolerance and graceful-degradation in order to provide unparalleled user experience.

The key strength of this paper is the systematic approach to discuss and evaluate the availability of the giant-scale services by clearly explaining MTTR, MTBF, uptime, yield, harvest, and the DQ value. The author has defined `yield` and `harvest` as more significant factor than the `uptime` as they measure more accurately the user experience than the former. Another strength of the paper is the use of DQ principle for assessment and planning of failure and maintenance work on cluster systems. The author also addresses the online evolution and growth by comparing the existing approaches with the DQ analysis. The key weakness of this paper is the author's inability of producing the experimental results to support his experience with giant-scale services. The proposed idea of graceful degradation sounds good, but how well is it doing real time is an another question. The author in the paper also claims that the performance obtained by replicating the node and partitioning is same, however he hasn't given any evidence to prove his claim. This can be true for the cases with high utilization because under low utilization, replication yields better performance as multiple queries execute independently with concurrency.

One of the assumptions made by the author is that the queries here are small. In todays scenario, we can have multiple connections all the time for ex. video streaming, and it would be hard to wait for inactivity before unplugging a node for updates. Also, In my view there is no specific rule that states what values the harvest and yield metrics should be to indicate if a service is with high availability. These metrics are complex in themselves than the uptime metric. For example, deploying an extra replica is supposed to improve the service's availability but it actually lowers the yield value. Besides this, the model presented can be used for any web service today. Also the author's idea of online evaluation can be used in any data center that need upgrades with minimal harm to availability.