<u>REPORT</u>

By Sandeepkrishnan

<u>Overview</u>:

This report is summarized version of the papers “Building a Database on S3” and “Consistency Rationing in the cloud: Pay only when it matters”.

The authors of the first paper explore the possibility and limitations involved in building a general-purpose database system in the cloud upon the services (Amazon S3) already offered. Amazon S3 is an online storage which makes an unlimited amount of capacity available to application programmers by means of a simple web service interface. The author proposes client-server architecture to implement the system. Applications on the client side access the database by means of a Record based interface. Then author discusses how traditional database techniques to implement tables, pages, B-trees, and logging can be applied to implement a database on top of S3. The authors have developed a working prototype which implements protocols to introduce transactions and to mitigate the problem of lost updates in the case of concurrent updates. Then he shows how durability can implemented on top of S3. This is shown with the help of commit protocols (which are required for solving issues arising out of concurrent updates) and checkpoints. Also the implementation of checkpoints for Data pages and B-trees and alternative checkpoint strategies (condition when a checkpoint is performed) are also discussed. Then authors goes on describing how additional transactional properties like Atomicity, Isolation etc can be implemented on top of S3. The main goal of the architecture described in the paper is to preserve the ideal properties of the services offered by Amazon while implementing a database system that shows as much as possible the ACID properties of a traditional database system.

It is clear from the discussion of First paper that Cloud storage solutions provide high scalability at low cost. But existing solutions, however, differ in the degree of consistency they provide and there is a non-trivial trade-off between cost, consistency and availability. High consistency implies high cost per transaction and, in some situations, reduced availability. Low consistency is cheaper but it might result in higher operational cost because of, e.g., overselling of products in a Web shop. In the second paper, authors present a new transaction paradigm, which allows designers to define the consistency guarantees on the data instead at the transaction level and also allows in automatically switching consistency guarantees at runtime. Number of techniques that let the system dynamically adapts to the consistency level by monitoring the data and gathering temporal statistics of the data have been discussed. Here data has been mainly categorized in to three groups based on the level of consistency which need to be applied. Data belonging to Category A requires maximum consistency levels and belonging to Category C requires least consistency. Category B comes in between the above mentioned categories and is mainly discussed in this paper. The adaptive techniques for switching consistency levels mainly apply to this category of data. The author then presents certain policies and models to adapt the consistency guarantees for data items in category B. The feasibility of the ideas presented in the paper is shown by conducting experiments on the architecture implemented on Amazon's S3.

The results of experiments indicate that the adaptive strategies presented in the paper result in a significant reduction in response time and costs including the cost penalties of inconsistencies.

Comments:

The paper addresses in depth various technical issues in implementing a database over a cloud. Also technical description of the services sounds perfect. The proposed system provides an infinite amount of storage, is always available and is very scalable and in the same time puts much emphasis in retaining the characteristics of a general-purpose database like support of concurrent user access and transactions. Preserving the benefits of utility computing is considered more important than ensuring strict ACID properties. Although the proposed approach is quite promising, however it remains uncertain if such architecture can actually be used for a Web application. Security concerns like what will happen if one of the clients is compromised are mentioned but not completely resolved. Aggregation functions like summation, average or maximum are not covered in this paper. Reading from S3 and writing to S3 are slower compared to local disks. These performance characteristics cannot be tolerated in many application domains. Also for some applications more emphasis should be on ACID properties than on Scalability and Availability. It has not been discussed how to address this issue. One of the main questions which came during the presentation was how the architecture proposed in the paper scores over Amazons SimpleDB. Amazon SimpleDB indexes all data attributes thus resulting in quick querying of the data. Also SimpleDB uses less dense drives for data storing and stores data as small bits which results in greater data access speed.

The second paper gives an in depth analysis of how consistency rationing can be applied to cloud data and shows how this can provide big performance and cost benefits. This paper introduces the notion of statistical policies which is the first step towards probabilistic consistency guarantees. Although the paper proposes a new concept called consistency rationing to optimize the runtime cost of a database system in the cloud, more insights needed to be provided in the following areas and should be addressed in future: providing better and faster statistical methods, implementing automatic optimizations with regards to other parameters (e.g., energy consumption), adding budget restrictions to the cost function, and relaxing other principles of the ACID paradigm (e.g., durability) etc. One of the questions that were asked during the discussion of the paper how this paper is related to the paper “Building a Database on S3”. The authors have done experiments on the architecture on Amazon S3 proposed in the first paper to prove the feasibility of this idea.