Reminiscences on Influential Papers

Kenneth A. Ross, editor

I continue to invite unsolicited contributions. See `http://www.acm.org/sigmod/record/author.html` for submission guidelines.

Flip Korn, AT&T Labs-Research, `flip@research.att.com`.

[Joseph M. Hellerstein, Peter J. Haas, Helen Wong. Online Aggregation. Proc. ACM SIGMOD 1997, pages 171–182 Tucson, Arizona, 1997.]

Sometimes the simplest ideas are the best. Hellerstein et al. introduced the idea of extending the relational database engine to compute basic SQL aggregates in an online fashion, recognizing that OLAP queries are often posed to get rough estimates (e.g., decision support) and thus do not require "painstaking precision". While there had been previous work on approximate query answering as well as so-called "fast-first" query processing, this is the first paper to consider online, interactive query processing in an RDB. The key implementation ideas include random access to data, non-blocking strategies, and round-robin fetching, all based on leveraging standard database tools. However, more important than these specifics is the paradigm shift away from traditional batch processing. Although the techniques in this paper are inherently for data stored in a DBMS, this general paradigm has influenced the way I think about processing queries over data streams.

Renee J. Miller, University of Toronto, `miller@cs.toronto.edu`.

[R. Hull. Relative Information Capacity of Simple Relational Database Schemata. SIAM Journal of Computing 15(3), pages 856–886, 1986. (Extended abstract in PODS 1984.)]

Like many, I find the most influential papers are those that change my intellectual biases. When I was a graduate student searching for a thesis topic, my advisor, Yannis Ioannidis, suggested information integration as an important problem that would only become more important. He was right of course, but I was having a hard time getting my head around this challenging problem. After several months of reading, I had learned that information integration was a messy problem. The common wisdom was that it took messy, *ad hoc* solutions to handle this problem. I was uninspired and ready to turn to a cleaner problem when Yannis handed me Hull's 84 PODS paper. The main goal of this paper was to provide a theoretical foundation for studying the relative capacity of schemas to store information. Hull went beyond earlier work that had considered schema equivalence in the context of data design. In doing so, he provided a set of tools for studying the integration of independent schemas and databases. Perhaps most importantly, he demonstrated that reasoning about independently designed databases was indeed a data management problem to which our arsenal of management tools could profitably be applied.

Kaladhar Voruganti, IBM Almaden Research Lab, `kaladhar@us.ibm.com`.

[Michael J. Franklin, Michael J. Carey, Miron Livny. Transactional Client-Server Cache Consistency: Alternatives and Performance. ACM TODS, Vol 22. No. 3, September, 1997, pp 315–363.]

As a doctoral student at the University of Alberta, Edmonton, Canada, I began looking for an interesting DBMS systems thesis topic. A lot of research was done on client caching database management systems in 1990s with respect to their data transfer, client and server buffer management, cache consistency and recovery algorithms. I was first motivated to work on the cache consistency problem, after I read this paper by Michael Franklin et al. on cache consistency (content of this paper was also part of Michael Franklin's

PhD dissertation). I had previously read 7 papers on transactional cache consistency algorithms but the area of DBMS client cache consistency was still confusing because there was not a systematic way of classifying, and thus, differentiating the different algorithms. The terms optimistic and pessimistic (borrowed from the centralized DBMS nonmenclature) were still being used for client-server DBMSs and I felt that this was not a good way of classifying the client-server algorithms. Furthermore, I also wondered why the caching research performed in other areas of computing science such as file systems and shared-memory multi-processor systems could not be adapted and utilized by client-server systems.

In this light, this paper did an excellent job of addressing both of my concerns. It first compared the DBMS client-server caching problem with the caching problems present in distributed file systems, shared-disk systems, and distributed shared-memory multi-processor systems, and therefore, highlighted the issues that are unique to the client caching DBMS area. It then presented an elaborate and elegant cache consistency classification mechanism which classified all of the existing client caching DBMS cache consistency algorithms. The key notion of avoidance-based and detection-based cache consistency algorithms was introduced in this paper, which I felt was a much better way of classifying client-server cache consistency algorithms. Furthermore, this paper also provided a systematic way of looking at the different components of a cache consistency algorithm.

Currently, a lot of cache consistency research is being conducted on non-transactional caching systems (multi-tiered web systems). I strongly recommend to the developers of algorithms in these emerging areas to read this paper, so that they too can develop a similar rigourous classification mechanism for the non-transactional cache consistency algorithms, and to also potentially modify and reuse some of the techniques that have been developed for transactional cache consistency algorithms.
