Reflecting on CAP

David Alan Grier, Djaghe, LLC

Eric Brewer's influential 2012 Computer *article encouraged designers to think on a broader scale.*

We don't tend to think of contributions to the field as being reflective. We prefer ideas that are novel, rigorous, and innovative. Yet, Eric Brewer's 2012 article, "CAP Twelve Years Later: How the 'Rules' Have Changed,"[2] is very much a reflection on a fundamental result for database design. This article was part of a special issue on the CAP theorem. In the seven years since its publication, it has become one of the more influential articles from *Computer*. It ranks 21st, with 4,251 downloads and 155 citations. Yet, it presented nothing new. Instead, it tried to get us to think differently about shared databases and spur system designers to articulate the requirements of their work.

At first glance, the CAP theorem seems to be one of those grand impossibility theorems in the spirit of Kurt Gödel's incompleteness theorem, Alan Turing's uncomputablity theorem, and Ken Arrow's impossibility theorem. Those three results demonstrate the impossibility of achieving some obvious and simple goals. Gödel showed that a mathematical system can't be complete and consistent.[6] Turing demonstrated that some numbers cannot be computed.[7] Finally, Arrow proved that almost any voting mechanism can be dominated by a dictator.[1]

The CAP theorem also asserts an impossibility, but that impossibility is not as surprising. It shows that a distributed database cannot simultaneously have three properties: consistency of data across the entire structure (C), immediate availability of all data (A), and tolerance against partitions (P). One can have two of the three properties, but one cannot have all of them. Unlike the theorems of Gödel, Turing, and Arrow, CAP has a fairly straightforward proof. Those first three impossibility results require a substantial amount of intellectual machinery and lengthy arguments. By contrast, the proof to CAP can be written in a paragraph. One can easily present it to a class without notes and fearing that one will make a confounding mistake that destroys the lecture.

Part of its simplicity is obvious upon reflection. No one wants to build a partitioned database in which one part is not accessible to another. Some of the big Internet databases, such as those belonging to the Domain Name System and the search-engine database, fall into geographic clumps. They record a lot of data from local areas, store that information in regional servers, and provide it to users who reside in the geographic region. They are still are unified, unpartitioned databases designed to provide data to any query from any part of the globe.

Digital Object Identifier 10.1109/MC.2019.2958449
Date of current version: 12 February 2020

0018-9162/20©2020IEEE

EDITOR **DAVID ALAN GRIER**
Djaghe, LLC; grier@gwu.edu

If no one wants to build a partitioned database, CAP really describes how database designers should react to the possibility of a partition. They should consider how to preserve consistency and mitigate the loss of availability or how they would preserve availability and address the loss of consistency. The purpose of the theorem was not to tell us that certain things were impossible, explained Brewer. It was "to open the minds of designers to a wider range of systems and tradeoffs." He argued that designers should "not blindly sacrifice consistency or availability when partitions exist." Instead, they "could optimize both properties through careful management" of their design.[5]

Brewer presented the CAP principle, as he then called it, at the 1999 Hot Topics in Operating Systems conference, which had a track devoted to file systems and another that was identified as "Potpourri." "I was participating in both networking and databases," Brewer explained. "I could see the two fields had different values due to differences in assumptions about the likelihood of network partitions." The network group assumed that partitions were likely and wanted to ensure that their systems were available in the presence of partitions. By contrast, the database community discounted the possibility of partitions and designed systems that relied on full connectivity to provide consistency and availability. The first approach he called "AP" and the second "AC."

Brewer's writing combined the practical and theoretical, a blend that should be the goal of anyone trying to produce an influential article for *Computer*. This approach reflected the fact that he was working on both the practical and theoretical sides of the field. At the time that he wrote the original CAP paper, "I was delivering real-world distributed systems for Inktomi," he explained, and "teaching the graduate operating-systems class at Berkeley." The paper makes a substantial effort to establish a research agenda and engage the research community. "We would like to motivate a broader research effort," he and his coauthor wrote at the end of the paper, "that extends these observations, resulting in a set of design guidelines for the construction of large-scale robust applications."[5]

Brewer's writing combined the practical and theoretical, a blend that should be the goal of anyone trying to produce an influential article for *Computer*.

To understand how the field of computing works, it is useful to compare Brewer's 1999 paper with his 2012 *Computer* article. His original paper quickly established itself in the database literature and acquired 35 citations from other authors. However, the bulk of those citations occurred after the publication of the 2012 article. It is easy to theorize that the *Computer* article brought attention to the original because it was a tutorial and easier to comprehend. The 2012 version was well written, to be sure, but it was far from a tutorial. It is better described as a reflection. It looked at what the original paper did, considered how the community received those ideas, and how the community needed to advance from that point. In particular, it noted that designers of distributed systems have a tendency to preserve consistency in their databases when they might be better served by working toward availability.

There is a parallel between the two publications and two other canonical texts from the database literature: E.F. Codd's 1970 paper on relational databases and his 1981 Turing lecture. The 1970 paper was the seminal work of Codd's career and developed a logical theory of databases.[3] It remains highly influential in the computing literature, with more than 1,700 citations and 28,000 downloads. The second paper, written 11 years after the first, was more reflective. While a Turing Award should make any author reflective, Codd's award pushed him to ask which elements of his model worked and which did not. While the paper was not as technical as his first, it did present a detailed description of data and their relation to application programs. "We have presented a series of arguments to support the claim that relational database technology offers dramatic improvements in productivity," Codd wrote. The arguments center around the data independence, structural simplicity, and relational processing of the relational database model. All "three of these features simplify the task of developing application programs and the formulation of queries and updates."[4]

By definition, reflective papers bask in borrowed light. They would not be published had their precursors not been printed. However, when well done, they add new insights to an idea and enable authors to explain their work more fully and precisely because the community has grown in response to the original. For Codd, his reflective paper cemented the position for his ideas. His 1970 paper has been cited 35 times a year for nearly 50 years. Few papers remain current for that long. For the CAP theorem, a reflective paper has expanded

its audience and encouraged readers to its note its value.

REFERENCES

1. K. J. Arrow, "A difficulty in the concept of social welfare," *J. Political Econ.*, vol. 58, no. 4, pp. 328–346, 1950. doi: 10.1086/256963.
2. E. A. Brewer, "CAP twelve years later: How the 'rules' have changed," *Computer*, vol. 45, no. 2, pp. 23–29, 2012. doi: 10.1109/MC.2012.37.
3. E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, June 1970. doi: 10.1145/362384.362685.
4. E. F. Codd, "Relational database: A practical foundation for productivity," *Commun. ACM*, vol. 25, no. 2, pp. 109–117, Feb. 1982.
5. A. Fox and E.A. Brewer, "Harvest, yield and scalable tolerant systems," in *Proc. 7th IEEE CS Workshop on Hot Topics in Operating Systems (HotOS 99)*, 1999, pp. 174–178. doi: 10.1109/HOTOS.1999.798396.
6. K. Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I," *Monatshefte für Mathematik und Physik*, vol. 38, pp. 173–198, Dec. 1931. doi: 10.1007/BF01700692.
7. A. Turing, "On computable numbers, with an application to the entscheidungsproblem," *Proc. London Math. Soc.*, vol. s2-42, no. 1, pp. 230–265, 1937. doi: 10.1112/plms/s2-42.1.230.

DAVID ALAN GRIER is a principal at Djaghe, LLC, and former editor in chief of *Computer*. He is a Fellow of the IEEE. Contact him at grier@gwu.edu.

Digital Object Identifier 10.1109/MC.2020.2967209