Revisiting RISC-style Data Management System Design

Danica Porobic
Oracle
danica.porobic@oracle.com

Microprocessors have followed the CICS (Complex Instruction Set Computer) path of increasing complexity where new functionality required increasingly complex instructions, that made processors costly to design and hard to debug, in addition to being hard to program. In the early 1980s, David Patterson and John Hennessy proposed a radically simpler RISC (Reduced Instruction Set Computer) approach that is now used in almost all processor designs. In the lecture following their 2018 Turing award for RISC design, Hennessy and Patterson reflect on the RISC journey and argue that we are currently at a new crossroads where dark silicon [4] and modern applications require more effective hardware-software codesign that is already bearing fruit for hyperscalers [3, 7]. Modern domain-specific languages are the key enabling technology in the development of such specialized hardware designs [1].

We have witnessed similar trends in the data management community: relational databases became the dominant type of databases in the 1990s due to their dominance in transaction processing. Analytical (OLAP) processing, object oriented databases, and semi-structured (XML) databases all challenged relational databases, however, they all failed to gain dominance in the marketplace. At the same time relational databases gained capabilities of these specialized systems, thus increasing their complexity and leading to arguments in favor of specialization [9].

Specialization has been an attractive approach for addressing the challenge of the ever increasing size and diversity of data. Furthermore, the value of many internet companies is measured by the amount of data they have and the ability to extract actionable information in a timely manner. Both the data deluge and the exponential rise of types of questions asked about that data are leading to a large number of data management systems specialized for different types of data (graphs, JSON, spatial, time series...), different types of applications (reporting, machine learning, streaming, operational...) and different platforms (enterprise class servers, virtualized cloud environments, heterogenous compute and storage hierarchies...).

In modern software architectures based on microservices and functions, data is rarely produced and consumed by the same system. Instead, it is generated in one system, cleaned in another, transformed in the third, integrated in the fourth, and analyzed in the fifth. While each of components of a complex data pipeline is very efficient on its own, actual question answering represents a small fraction of the overall time and energy spent by the system. Additionally, the system complexity requires a number of experts to develop, optimize, and operate such data pipelines.

However, if we step back and examine architectures of both traditional general and modern specialized data management systems, they are remarkably similar: they all have similar components, including data storage, data consistency, access method selection, and query processing. This is not surprising as they are all fundamentally doing the same thing: storing, accessing, and processing data. Thus, data management systems are very suitable to "RISC" architecture [2]. Furthermore, modern compiler technology [8] and machine learning [5] are more than capable of enabling data management systems tuned to specific workloads. By making all major components of data management systems composable and self-tuning, we can go even further and dynamically specialize data management systems to the applications, as we are already doing for hardware in engineered systems [6].

9th Biennial Conference on Innovative Data Systems Research (CIDR '17) January 13-16, 2019 , Asilomar, California, USA.

1. REFERENCES

[1] J. Bachrach et al. Chisel: constructing hardware in a scala embedded language. In *DAC*, 2012.

[2] S. Chaudhuri and G. Weikum. Rethinking Database System Architecture: Towards a Self-Tuning RISC-Style Database System. In *VLDB*, pages 1–10, 2000.

[3] J. Dean, D. Patterson, and C. Young. A new golden age in computer architecture: Empowering the machine-learning revolution. *IEEE Micro*, 38(2), 2018.

[4] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki. Toward dark silicon in servers. *IEEE Micro*, 31(4):6–15, 2011.

[5] S. Idreos, K. Zoumpatianos, B. Hentschel, M. S. Kester, and D. Guo. The data calculator: Data structure design and cost synthesis from first principles and learned cost models. In *SIGMOD*, pages 535–550, 2018.

[6] J. Loaiza. Engineering database hardware and software together. *PVLDB*, 8(12):2052, 2015.

[7] A. Putnam et al. A reconfigurable fabric for accelerating large-scale datacenter services. In *ISCA*, pages 13–24, 2014.

[8] E. Sedlar. How I Learned to Stop Worrying and Love Compilers. In *SIGMOD*, pages 1–2, 2014.

[9] M. Stonebraker and U. Cetintemel. "one size fits all": an idea whose time has come and gone. In *ICDE*, 2005.