Efficient Hard Real-Time Transaction Processing – Unachievable in Software

Matheus A. Nerone
CWI Amsterdam, NL
matheus.nerone@cwi.nl

Stefan Manegold
CWI Amsterdam, NL
Stefan.Manegold@cwi.nl

Holger Pirk
Imperial College London, UK
holger.pirk@imperial.ac.uk

ABSTRACT

Many practical data management applications are subject to *real-time* constraints, i.e., upper limits on response time of each operation. In the past, database research has mostly focused on what is known as "soft" real-time applications, i.e., those that expect results in a timely manner but only suffer minor degradation if the constraints are violated.

However, data management systems are becoming increasingly common in applications with "firm"[1] or even "hard"[2] real-time requirements. State-of-the-art cars, for example, are engineered as collections of semi-autonomous components that are connected using a real-time Bus such as CAN. The components (sensors, actuators, steering devices, ...) update a centralized database which forms the basis for hard real-time decisions (such as controlling the Anti-lock Brake System or activating the airbag) [2].

Hard real-time systems are difficult to implement on top of general purpose CPUs due to hardware design decisions that favor throughput over performance predictability: out-of-order execution, cache-line prefetching, pre-emptive multi-threading and many more. Similarly, most high-performance transaction processors focus on throughput (i.e., average latency) rather than deterministic latency. This makes them inherently unsuited for hard real-time applications.

To illustrate the difficulty of providing daterministic latencies in throughput-oriented transaction processing systems, we evaluate the YCSB write transactions on the state-of-the-art transaction processor Cicada [1]. To minimize variance caused by contention, we limit Cicada to a single worker thread and present the histogram of response times in Figure 1. Note that there is almost two orders of magnitude difference between the average latency and that of the slowest transaction. While this is not problematic for throughput-oriented systems, this violates the requirements of efficient hard real-time applications: consider an (arbitrary) constraint of 60 us (illustrated by the purple vertical line in Figure 1). Any transaction to the right of this line is considered a failure while any transaction that finishes earlier than the constraint is an indication of misprovisioned resources, i.e., those that could be used to improve secondary objectives such as energy efficiency or throughput. As apparent from the figure, *a constraint cannot be set such that it satisfies the real-time requirement and uses resources efficiently*.

We argue that efficient hard real-time transaction processing is not achievable in software but requires the use of customizable hardware such as Field-Programmable Gate Arrays (FPGAs), which have very low and predictable latency. In fact, FPGAs have been used in automotive applications for exactly that reason [2]. However, the inherently parallel nature of FPGAs makes them tricky to program correctly.

To address that problem, we are developing an FPGA-based real-time transaction processor. While data mangement on FPGAs has received interest from researchers in the past, the focus was almost exclusively on data analytics. We conjecture that this is, to a large extent, due to persistence being an unsolved problem. However, the emergence of non-volatile memory is likely to change that which opens a window of opportunity for research in that space.

While we concede that there are significant challenges in developing such a system (among them concurrency control, indexing, buffer management, scaling), we believe that now is the right time to study transaction processing on FPGAs and exploit their efficient hard real-time guarantees.

[1]operations that violate the time constraint count as "failed"
[2]even one violated constraint is considered a system failure

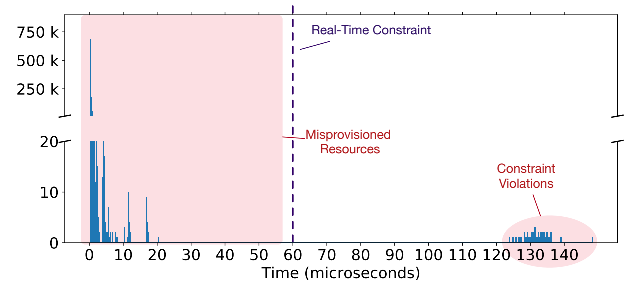

Figure 1: Histogram of Cicada's per query latency, executing 1 million YCSB writes (theta = 0.1) on 10 million rows.

1. REFERENCES

[1] H. Lim et al. Cicada: Dependably fast multi-core in-memory transactions. In *SIGMOD*, '17.

[2] S. Shanker. *Enhancing automotive embedded systems with FPGAs*. PhD thesis, NTU, 2016.