Implementing High-Performance Resilient Data Management: What we Learned

Jelle Hellings
jhellings@mcmaster.ca

Nancy Kansal
kansan1@mcmaster.ca

Celine Sana
sanay@mcmaster.ca

Department of Computing & Software
McMaster University

1 Background

Recently, we have seen a lot of interest in, research on, and development of *resilient systems* that can manage data and process transactions even in the presence of network, software, or hardware failure. Unfortunately, the practical usage of resilient systems technologies remains a niche in data-based applications: existing systems are costly, hard to use, highly complex, and have limited performance potential.

Typical resilient systems are operated by a *consensus protocol* such as Raft [4] (which can deal with crashes) or PBFT [2] (which can deal with arbitrary failures) that ensures agreement among all participating replicas on which requests to process and in which order. Although consensus is *inherently complex*, several works have illustrated how one can implement consensus with very high throughputs [1, 3]. Unfortunately, these works describe highly complex implementations that mainly focus on providing the highest possible throughput at the cost of ease-of-use and latency.

2 Problem Statement

To further push resilient systems and their usage in data management and transaction processing into the mainstream, we believe there is a strong need for improved resilient system technologies that promote simplicity in use and implementation, while still guaranteeing great performance (i.e., high throughput with low latency).

Our research aims at developing these improved resilient system technologies by designing and implementing a novel resilient system framework that is easy to use, can support high performance, and has limited complexity. In this talk, we will present what we learned developing and implementing this framework. In specific, we will look at three orthogonal approaches toward improving resilient system technologies:

1. the development of a high-performance framework that *simplifies* the implementation of consensus protocols in a highly multi-threaded environment;
2. *revisiting and redesigning* existing consensus consensus protocols toward enabling simpler and more-efficient implementations; and
3. the development of new *fault-tolerant primitives* that can manage resilient systems at lower cost than existing consensus-based approaches.

Finally, we provide a practical evaluation of how these approaches influence the design and implementation of our novel high-performance resilient system framework. In this evaluation, we analyze the potential impact of compute power, network bandwidth, message delay, and protocol architecture on both practical and theoretically-achievable resilient system performance.

References

[1] Johannes Behl, Tobias Distler, and Rüdiger Kapitza. Consensus-oriented parallelization: How to earn your first million. In *Proceedings of the 16th Annual Middleware Conference*, pages 173–184. ACM, 2015. doi:10.1145/2814576.2814800.

[2] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002. doi:10.1145/571637.571640.

[3] Suyash Gupta, Sajjad Rahnama, and Mohammad Sadoghi. Permissioned blockchain through the looking glass: Architectural and implementation lessons learned. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 754–764. IEEE, 2020. doi:10.1109/ICDCS47774.2020.00012.

[4] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, pages 305–320. USENIX, 2014.