Research Statement — Nicolas Schiper

Cloud-based applications are everywhere around us. They range from performance sensitive services such as web search, e-commerce, and social networking to mission-critical applications such as air-traffic control and the power grid. These applications tend to be complex, involve a large number of computers, and must face all sorts of failures ranging from power outages and software bugs to malicious attackers.

My research interests are centered around making these applications reliable, trustworthy, and scalable. Designing protocols that meet all these requirements is not only a science but also an art and it is a very exciting topic. As such, I design, build, and evaluate protocols to make these services fault-tolerant and scalable, and I use formal methods to provide bug-free code for critical components. As fault-tolerant storage systems have become pervasive in today's data centers, I have also started investigating the issue of energy-efficient cloud storage. In this line of research, I strive to maximize the performance obtained per Watt by adapting the replication protocol to the workload.

My approach to research consists in identifying a problem, precisely stating what properties the solution should guarantee, and exploring the space of solutions. I carry out this exploration both at a theoretical level to distinguish what can be done from what cannot, and at a practical level by building the complete system for it to be extensively evaluated. As a theoretical model cannot fully encompass the complexity of a real system, the initial formal model and algorithms may have to be adapted. This can take the form of refining the fault-model, e.g., in a single tenant cloud hardware and software failures may be more common than malicious attacks; or it may mean trading off some performance for the sake of simplicity. Empirical evaluations often reveal protocol behaviors not foreseen in the design process and are thus essential. To date, I have successfully applied this approach to several projects, various of which I will describe below.

Scalable Fault-Tolerance. In the context of performance-sensitive multi-datacenter applications, we have developed several atomic multicast protocols to build fault-tolerant applications [8, 9]. These protocols ensure agreement on the sequence of messages delivered and an extensive empirical evaluation shows that these protocols offer good scalability even with modest inter-datacenter communication links [11]. This project showed a trade-off between the best achievable latency to deliver a message and the scalability of the atomic multicast protocol. We devised a protocol that can deliver a message addressed to multiple datacenters in a single inter-datacenter message latency but at the cost of involving machines that are not addressed by the message. To get better scalability, we presented a *genuine* protocol that only involves nodes addressed by the message. We characterized such protocols by showing that *genuineness* has a cost: it requires a minimum of two message delays. Our proposed protocol achieves this minimal latency.

To demonstrate the ease of building multi-datacenter fault-tolerant applications using the multicast service, we proposed P-Store [12], one of the first replicated key-value stores that offers full transactional support. P-Store allows data to be sharded among datacenters in a flexible way to allow applications to scale with the number of available machines, and can be used as an effective transactional storage system for the cloud. The evaluation of P-Store helped us discover a convoy effect where synchronization between single datacenter and multiple datacenter transactions would reduce the overall performance. We proposed a technique to reduce this synchronization while maintaining strong data consistency.

We also addressed the issue of high-performance replication inside the datacenter. In [5], we proposed Ring Paxos, an atomic broadcast service optimized for *throughput efficiency*: the rate between

the maximum achieved throughput of a protocol and the nominal transmission capacity of the network. This project showed that it is possible to totally order messages at close to wire-speed. Ring Paxos has been open-sourced on Sourceforge and, as of October 2013, has been downloaded more than 1,700 times. My work in [13] proposed a leader-election service that is required by consensus protocols such as Paxos to ensure progress. The service provides robustness in the face of poor network conditions and allows the user to specify the quality of service contract required from the service. This contract is expressed in terms of how fast leader failures need to be detected and bounds, to the extent possible, the probability of spuriously suspecting the leader to have crashed and the duration of such erroneous suspicions.

Bug-Free Replication Protocols using Formal Methods In an effort to develop more robust fault-tolerant software we have tried to bridge the gap between formal methods and distributed systems. As a result, we have built ShadowDB, a replicated database whose failure-handling code has been proven correct [10]. The failure-handling code relies on the first formally-verified implementation of *Paxos*, a widely used protocol in today's datacenters. This critical code is automatically generated from a language with formal semantics called EventML. The generated code is run inside an interpreter or translated to a programing language such as Lisp to be compiled.

Our methodology produces bug-free code that can be integrated with performance-sensitive hand-written programs. Hence, normal case execution is hand-coded in ShadowDB. Recovery using the correct-by-construction code takes a few tens of milliseconds with an interpreter and about 5 milliseconds with Lisp code. To reduce the chances of correlated failures among the replicas, our methodology eases the generation of diversity in a way that preserves correctness. The failure-handling code is translated into executable machine code using different compilers. These interact with their hand-written environments (e.g., the operating system and libraries) in different ways and this reduces the chances of replicas failing at the same time. ShadowDB also offers the possibility to run different databases (e.g., BerkeleyDB, MySQL, Derby) at the different replicas. The failure-handling code can also be changed at runtime either for performance reasons (e.g. when the workload changes) or to better shield it against adversaries that attempt to attack the system (so-called *moving target defense*). We currently have an implementation of Paxos and one-third Consensus [1].

Enhancing the Dependability of Distributed Applications A substantial proportion of application bugs lead to non-crash failures: data corruption, silent errors, and unauthorized user access are some of the consequences of these bugs. Although formal methods can help mitigate these problems, we will, and for some time, continue to rely on manually written code. Formal methods help in the design of robust code but cannot replace the designer. In this project, we proposed byzantine-tolerant replication protocols that are simpler to implement, reduce hardware costs, and offer good performance. Such protocols not only mask crash failures but they also hide bugs that lead replicas to behave arbitrarily.

Based on the observation that most datacenters contain a configuration service, Byzantine Chain Replication [7] presents a simple protocol to mask arbitrary failures with only $2t + 1$ machines, where t is the maximum number of Byzantine failures. This is a reduction of t machines compared to the state of the art. Pipelined execution and the avoidance of expensive public-key cryptography allows this protocol to exhibit excellent performance in practice: it imposes a mere 30% overhead compared to a non-replicated version of BerkeleyDB. Another project [6] demonstrated how to build an optimistic database replication protocol using a broadcast service. Surprisingly, this protocol allows read-only transactions to execute at a single node despite a potentially malicious database, making this protocol well-fitted for read-dominated workloads.

Future Work

As Internet services become more prevalent in our daily lives, designing trustworthy, reliable, and scalable distributed applications will only become more important. To meet these goals, there is a need to define modular abstractions from which distributed protocols can be built. In doing so, we simplify the reasoning about their correctness and allow the use of formal methods for critical components. Simultaneously, technology and applications are evolving. In the cloud, machines are equipped with more and more cores, SSDs are becoming common-place, volatile memory is becoming cheaper, and datacenters are migrating towards network topologies that enable virtually identical communication performance regardless of the position of the machines in the cluster. Additionally, the worldwide energy consumption of datacenters is becoming significant. In 2010, this consumption represented more than 1% of the worldwide electricity use [4]. To make storage highly-available, most data centers rely on a single monolithic replication protocol [2, 3], a solution that cannot provide maximal energy efficiency for the wide range of applications that exist.

More energy-efficient datacenters can be obtained by developing replication protocols that *adapt* to the application: often accessed data items will be replicated with protocols that handle contention well, items that are less often accessed can be replicated with optimistic techniques that allow to extract more parallelism from the application workload. Such adaptation must be provided by the replication layer and will automatically adapt to workload changes. In doing so, we provide energy efficiency by maximizing the performance extracted out of each server. Further savings can be obtained by turning off computing resources that are not absolutely necessary. To gain confidence in these replication protocols, the effort to bridge the gap between formal methods and systems design needs to be pursued.

In the short-term, I intend to study the performance and energy needs of existing replication solutions. Given this initial study, I plan on developing workload-aware replication protocols that provide better energy efficiency. In the context of formal methods applied to the design of distributed protocols, I intend on continuing my collaboration with colleagues at Cornell. Further optimizing the efficiency of correct-by-construction programs and providing provably-correct Byzantine-resilient replication solutions are two important goals.

References

[1] B. Charron-Bost and A. Schiper. The Heard-Of model: computing in distributed systems with benign failures. *Distributed Computing*, 22(1):49–71, 2009.

[2] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford. Spanner: Google's globally-distributed database. In *Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation*, OSDI'12, pages 251–264. USENIX Association.

[3] P. Hunt, M. Konar, F.P. Junqueira, and B. Reed. ZooKeeper: Wait-free coordination for Internet-scale systems. In *USENIX Annual Technology Conference*, 2010.

[4] J. G. Koomey. Growth in data center electricity use 2005 to 2010.

[5] P.J. Marandi, M. Primi, N. Schiper, and F. Pedone. Ring Paxos: A high-throughput atomic broadcast protocol. In *Proceedings of the 30th IEEE/IFIP International Conference on Dependable Systems and Networks*, DSN'10, pages 527–536. IEEE Computer Society.

[6] F. Pedone and N. Schiper. Byzantine fault-tolerant deferred update replication. *Journal of the Brazilian Computer Society*, pages 3–18, 2012.

[7] R. Van Renesse, C. Ho, and N. Schiper. Byzantine chain replication. In *Proceedings of the 16th International Conference on Principles of Distributed Systems*, OPODIS'12, pages 345–359. Springer.

[8] N. Schiper and F. Pedone. On the inherent cost of atomic broadcast and multicast in wide area networks. In *Proceedings of the 9th International Conference on Distributed Computing and Networks*, ICDCN'08, pages 147–157. Springer.

[9] N. Schiper and F. Pedone. Solving atomic multicast when groups crash. In *Proceedings of the 12th International Conference on Principles of Distributed Systems*, OPODIS'08, pages 481–495. Springer.

[10] N. Schiper, V. Rahli, R. Van Renesse, M. Bickford, and R. L. Constable. ShadowDB: A replicated database on a synthesized consensus core. In *8th Workshop on Hot Topics in System Dependability (HotDep'12)*.

[11] N. Schiper, P. Sutra, and F. Pedone. Genuine versus non-genuine atomic multicast protocols for wide area networks: An empirical study. In *Proceedings of the 28th IEEE Symposium on Reliable Distributed Systems*, SRDS'09, pages 166–175. IEEE Computer Society.

[12] N. Schiper, P. Sutra, and F. Pedone. P-Store: Genuine partial replication in wide area networks. In *Proceedings of the 29th IEEE Symposium on Reliable Distributed Systems*, SRDS'10, pages 214–224. IEEE Computer Society.

[13] N. Schiper and S. Toueg. A robust and lightweight stable leader election service for dynamic systems. In *Proceedings of the 28th IEEE/IFIP International Conference on Dependable Systems and Networks*, DSN'08, pages 207–216. IEEE Computer Society.