TECHNISCHE
UNIVERSITÄT
DRESDEN

Faculty of Computer Science Institute of Systems Architecture, Professorship of Systems Engineering

Short version of the dissertation to achieve the academic degree Doktor-Ingenieur

ERROR ISOLATION IN DISTRIBUTED SYSTEMS

Diogo Behrens

Submitted on 2015-01-26

Supervisor
Prof. Christof Fetzer, Ph.D.
Technische Universität Dresden, Germany

Large-scale distributed systems are at the core of every successful online service on the Web. Operating such systems at scale comes with a number of challenges related to fault tolerance. Since machine and process crash failures are common in production, many systems critical to Web-scale services are using techniques such as state-machine replication [Sch90] to guarantee availability and integrity despite crash failures. Real-world examples of such systems are Chubby [Bur06], ZooKeeper [Hun+10], and Megastore [Bak+11].

Although occurring less often than crash failures, *hardware errors* also occur in production [SPW09; HSS12] and are expected to happen more frequently in future hardware generations [BC11]. A non-negligible number of these errors is uncorrectable by hardware techniques such as error-correcting codes (ECC) [SPW09] and manifest to the application as *state corruptions*. As consequence, a process might commit a so-called *value failure* by externalizing corrupted state, for example, by sending a corrupted message out.

In spite of crash-tolerant systems being successfully deployed and extensively used in practice, their underlying crash-failure model does not include value failures. Consequently, one may expect value failures to propagate across the system, bringing the system to inconsistent states and even causing outages. Recent events have shown large distributed services being disrupted by failures of this nature, resulting in long periods of unavailability, *e.g.*, the Amazon S3 case in 2008 [Das08].

In this work, we argue that preventing end-to-end error propagation due to state corruption should be a first-class requirement for large-scale fault-tolerant distributed systems. We design, formalize, proof correct, implement, and evaluate *hardening* techniques that enable existing crash-tolerant protocols to handle state corruption with minimal developer intervention. In particular, we define *hardening* as any transformation of distributed system processes that allows each process to locally detect its own faults and "virtualize" them into benign failures such as process crashes and message omissions.

In this short version of the dissertation, we start by discussing the causes and effects of hardware errors in distributed systems. Next, we briefly describe our contributions.

1 HARDWARE ERRORS IN THE WILD

Hardware errors can be caused by several reasons such as cosmic rays, aging, heat and production failures [SPW09] as well as electromagnetic coupling [Kim+14], hardware/software incompatibility [Ker07], and even incorrect use of dynamic voltage/frequency scaling. The cause of an error is called fault. Hardware faults can be transient or permanent, depending on whether they are bounded or continuous in time [Avi+04]; and they can affect different hardware components, in particular, sequential-logic circuits used in memory elements as well as combinational-logic circuits used in computation units.

Regardless of the specific hardware faults, the caused errors manifest at the application level as state corruptions, which in turn may propagate as crash failures or, in the worst case, as value failures.

Unfortunately, the consequences of state corruption in distributed systems are rather unpredictable due to the inherent complexity of communication patterns among processes. Consider the following anecdotal examples.

- Chandra *et al.* have observed hardware errors corrupting the state of processes in some instances of the Google's Chubby locking service [CGR07], resulting in at least one complete failure of the service; the duration of the outage was however not reported.
- In July 2008, a single flipped bit in a machine propagated via messages to other processes, contaminating them and finally causing an 8-hour outage of Amazon S3 service in US and Europe [Das08]; a severe service disruption to most clients of the S3.

- Ma.gnolia, a social bookmarking website, had a major date corruption episode in January 2009 [Mag09], losing all its data from all its users; an event that finally led to the company's shutdown at the end of 2010.

Hardware errors might not only disrupt the availability and integrity of systems. Hardware errors have also been successfully used to compromise the security of systems. Govindavajhala and Appel have have shown how to gain control over a Java Virtual Machine using a heat lamp pointed to the memory modules of a computer [GA03].

In some cases, attackers do not even have to tamper with the hardware; instead, they can simply wait for the errors to happen. Recently, Dinaburg has proposed a new form of attack called *bitsquatting* [Din11]. The attacker registers DNS entries pointing to addresses with potentially dangerous content, as in typosquatting attacks. The registered names differ from well-accessed website names by single flipped bits instead of differing by a typo – *e.g.*, `mic2osoft.com` results from `microsoft.com` by bitflipping the one bit in its fourth character[1]. Dinaburg has collected evidence that such domain names receive a non-negligible number of hits per day, and suggested that the reason for that are hardware errors during the resolution of domain names in one of the many resolvers or even in the local caches at the clients. Nikiforakis *et al.* have recently shown that the number of such domains registered has been increasing continuously [Nik+13], corroborating the claim that such attacks are feasible and successful.

While *undetected* state corruptions are hard to track down beyond single episodes like the one mentioned above due to the need of application-specific information, some work has studied the frequency of *detected* errors at the hardware level in large populations of computers. Hwang *et al.* recently analyzed hardware error detection logs coming from 300 TB/year worth of DRAM memory from Google's servers and other large server systems [HSS12]. They showed a high rate of detected state corruptions in memory of up to 167,066 FIT[2]. Nightingale *et al.* analyzed Windows Error Reporting logs collected over 8 months from nearly one million machines; they observed that the likelihood of a first detected CPU subsystem error occurring in a machine is nearly an order of magnitude higher than of the first detected memory error [NDO11]. These and other studies – *e.g.*, Schroeder *et al.* [SPW09], and Dinaburg [Din11] – confirm the intuition that faults that would be very unlikely in a small cluster become much more likely at scale.

Given that the hardware error rate is expected to increase in the future [Shi+02; Bor05; Fen+10; BC11], such errors might become more frequently at a small scale as well. According to researchers from Intel, the reliability per bit tends to decrease by 8% in every generation of processors [Bor+04; Bor05]. The rate of transient errors for processors is rising and it might reach up to one user-visible failure per day per chip with 16 nm technology [Shi+02; Fen+10]. New processor generations have traditionally achieved higher performance through higher circuit density and lower energy consumption, but this trend has reached physical limits where reliability starts to be negatively affected [Bor05].

2 CONTRIBUTIONS

The dissertation is mainly divided in four chapters. The chapter *Error isolation and hardening* states the problem of this work and defines the theoretical model used by our hardening techniques. Each of following chapters – *Encoded processes*, *Scalable error isolation* and *Hardened state-machine replication* – describe a hardening technique, defines its algorithms, and present a prototypical implementation, which is then evaluated in terms of performance and fault coverage.

[1]In ASCII code, the binary value 1110010 represents the letter *r*, while 1100010 represents the number 2.

[2]Failures in time (FIT) is the number of failures expected in 10^9 device-hours of operation.

We now overview the contributions of each of these chapters.

2.1 ERROR ISOLATION AND HARDENING

Error isolation is a central concept of this dissertation. We define error isolation as the property in which corrupted messages do not propagate errors between processes of the system, *i.e.*, corrupted messages might be sent in the system, but they are always discarded by correct receiver processes. Error isolation can be achieved with traditional techniques such as Byzantine-fault-tolerant algorithms. Nevertheless, we are interested in techniques that achieve error isolation locally, *i.e.*, exchanging no additional messages among processes. We define *hardening* as a transformation of a program p into a program p_h, such that if all processes of the system run such a program p_h, then error isolation is guaranteed locally. We then show that if error isolation holds, crash-tolerant distributed systems can also tolerate arbitrary faults in environments free of malicious attacks and bugs.

To design hardening techniques, we have to first model the effect and extent of arbitrary faults *inside processes*. Therefore, we introduce a new fault model upon which the hardening techniques presented in this work are constructed: the arbitrary state corruption (ASC) fault model. Correia *et al.* first proposed the ASC fault model as the underlying fault model for the PASC algorithm [Cor+12]. In contrast to their ASC fault model, the model recasted here is (1) more precise by using a cleaner notation and structure; (2) more general by having one main assumption, namely, fault diversity, and (3) decoupled from any specific hardening algorithm such as PASC. To show the expressiveness of our ASC fault model, we sketch an ASC-based formalization of the error model used in the AN-encoding literature.

2.2 ENCODED PROCESSES

The first hardening technique presented in this work is an adaptation of AN-encoding [Sch11]. One of the known problems with AN-encoding and other error-detection techniques is a lack of a formalism [Per+07]. We do the first steps in modeling and proving AN-encoding correct in a formal framework for distributed systems. First, we define a simplified AN-encoding technique that only supports a small instruction set. We call the technique Perfect AN-encoding, or PAN-encoding. We then refine the ASC fault model with assumptions specific for PAN-encoding, and prove the technique correct. Next, we estimate the coverage of our assumptions, pinpoint the formalization limitations and difficulties, and discuss existing approaches to increase the assumption coverage.

Despite requiring some strong assumptions, AN-encoding turns out to be very effective in practice. We show that by designing and implementing a framework capable of automatically encoding distributed applications developed on top. Our framework extends the scope of AN-encoding to distributed systems in a systematic way. Moreover, a practical issue not considered in previous work is the excessive network traffic overhead caused by sending encoded messages out. We propose a bandwidth optimization to reduce the network overhead to a constant factor of 8 bytes per message.

To evaluate our approach, we implement two well-known algorithms on top of our framework: the Highly-available Leader Election Service by Fetzer and Cristian [FC99]; and the multi-instance Paxos by Lamport [Lam98], which is widely used in replicated systems [Bol+11]. Our experimental results illustrate the possible trade-off between fault coverage and CPU utilization. Encoded Paxos variants show virtually no network overhead: They reach the same request throughput as the native variant with 5 acceptors at the cost of extra CPU cycles and higher response time. Under a load between 1 k and 5 k.req/s, for example, encoded variants of Paxos provide response times, varying from 20 to 105 ms; an overhead of at least 4 times in relation to the native variant. The encoded variants of the leader election provide excellent election times, while consuming at most 11% more CPU cycles. Finally, our fault injection results suggest that AN-encoding can successfully guarantee error isolation: An encoded Paxos proposer has its probability of undetected errors (given that a failure occurred) decreased two orders of magnitude, from 16% to about 0.34%.

2.3 SCALABLE ERROR ISOLATION

Our second hardening technique is Scalable Error Isolation (SEI), a new hardening technique that wraps a process of a distributed system and executes redundant checks to prevent the process from propagating local errors to other processes. SEI is scalable in three dimensions. For *memory*, SEI can leverage hardware error detection such as ECC to virtually eliminate the memory footprint of redundant information. For *computation*, SEI supports multithreading and, thus, covers complex error propagation patterns among threads sharing the same state. For *development effort*, SEI enables hardening real-world applications with minor developer involvement. Moreover, SEI is designed to formally guarantee error isolation in presence of ASC faults under two main assumptions: fault diversity and a fault frequency assumption, which asserts that at most one ASC fault may occur per traversal.

In this chapter, we start by refining the ASC model and presenting a specification of SEI, assuming single-threaded applications. Next, we prove SEI correct. We then extend the model, algorithms and proofs to support multithreaded applications.

We also present an implementation of our hardening technique called `libsei`. Hardening processes with `libsei` is a *semi-automated* task, requiring the developer to manually mark the regions of the program that represent handlers and messages. Nevertheless, `libsei` allows for hardening existing systems with a small amount of effort. We demonstrate that by hardening two real-world applications: `memcached` and a recursive DNS resolver. We discuss our experience hardening these applications in detail.

We conducted extensive fault injection experiments, both software- and hardware-based, and performance evaluation of our hardened applications. SEI presents an excellent trade-off between fault coverage and performance. It makes the likelihood of error propagation negligible: down by two orders of magnitude from 44% to only 0.15% of the errors using targeted fault injection, and no undetected errors when reducing the CPU voltage. For multithreaded `memcached`, the throughput overhead is almost zero with value sizes of 128 bytes or larger, while the memory overhead is about 30 KiB (the number of bytes modified during event handling).

2.4 HARDENED STATE-MACHINE REPLICATION

State-machine replication is a well-known fault-tolerance technique, applied in industry-level systems such as Chubby, ZooKeeper, and Megastore. All these systems assume a crash-stop failure model, where processes only fail by silently crashing; they cannot withstand arbitrary state corruptions.

Byzantine fault-tolerant (BFT) algorithms can tolerate state corruptions, but typically incur unjustifiable ownership and maintenance costs. In contrast, AN-encoding (encoded processes) and SEI-hardening can harden crash-tolerant systems against state corruptions without requiring the systems to be replicated. These approaches, however, have a space and/or performance cost associated with their generality.

In this chapter, we suggest that if an application already employs replication, one can optimize these hardening approaches to incur a smaller overhead. In particular, we show that, by hardening only part of the underlying atomic broadcast algorithm, benign-fault-tolerant SMR systems and the service running on top can tolerate state corruptions, while incurring little state, throughput and response time overhead.

We propose HardPaxos, a variant of the Paxos, an algorithm widely used to implement SMR-like systems. In HardPaxos, critical functions and state variables are factored out of the algorithm and kept in a module called HardCore. We then describe how to satisfy the Paxos invariants given a *trustworthy* HardCore.

If HardCore is an ordinary software module, it is also subject to hardware errors. HardCore is made trustworthy by hardening with AN-encoding (used in the encoded processes technique) and duplicated execution (a largely simplified version of SEI's approach). If a hardware error affects the state or computation of HardCore, the whole process is aborted with a high probability. Moreover, all messages exchanged in HardPaxos are augmented with end-to-end error detection codes that allow the receiver's HardCore to detect the incorrect execution of a non-silent faulty process.

Finally, we perform extensive fault injection experiments to evaluate the fault coverage of our hardening. The experiments show a decrease of undetected errors from 15% without hardening down to 0.9% with AN-encoding and further down to 0.02% with AN-encoding plus duplicated execution. We also show that the hardening does not hamper performance. Even with the strongest hardening, the throughput loss is at most 4% for small messages (64 bytes) and close to zero for messages of 1 KiB or larger; the increase in the single response time does not exceed 5%.

3 CONCLUSION

Hardware errors may cause state corruptions, which are a threat to distributed systems that can have catastrophic consequences. In this dissertation, we propose hardening as a class of general techniques for protecting distributed systems against state corruptions in an end-to-end fashion, while requiring changes strictly local to each process of the system. Hardening techniques guarantee error isolation, the property in which no faulty process propagates internal errors and contaminates correct processes. With hardening in place, developers can employ existing benign-fault-tolerant algorithms to implement systems tolerant not only to crashes and message loss, but also to a wide range of hardware errors. Our hardening techniques do not require any trusted software or hardware components and are systematic: They are fully- or semi-automatic, requiring minor effort from the developer to employ them.

BIBLIOGRAPHY

[Avi+04] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. “Basic concepts and taxonomy of dependable and secure computing”. In: *Dependable and Secure Computing, IEEE Transactions on* 1.1 (2004), pp. 11 –33. DOI: 10.1109/TDSC.2004.2.

[Bak+11] Jason Baker, Chris Bond, James C. Corbett, JJ Furman, Andrey Khorlin, James Larson, Jean-Michel Leon, Yawei Li, Alexander Lloyd, and Vadim Yushprakh. “Megastore: Providing Scalable, Highly Available Storage for Interactive Services”. In: *Proceedings of the Conference on Innovative Data system Research (CIDR)*. 2011, pp. 223–234.

[BC11] Shekhar Borkar and Andrew A Chien. “The future of microprocessors”. In: *Communications of the ACM* 54.5 (2011), pp. 67–77.

[Bol+11] William J. Bolosky, Dexter Bradshaw, Randolph B. Haagens, Norbert P. Kusters, and Peng Li. “Paxos replicated state machines as the basis of a high-performance data store”. In: *Proceedings of the 8th Conference on Networked Systems Design and Implementation*. NSDI’11. USENIX Association, 2011.

[Bor+04] Shekhar Borkar et al. “Microarchitecture and design challenges for gigascale integration”. In: *MICRO*. Vol. 37. 2004, pp. 3–3.

[Bor05] S. Borkar. “Designing reliable systems from unreliable components: the challenges of transistor variability and degradation”. In: *Micro, IEEE* 25.6 (2005). DOI: 10.1109/MM.2005.110.

[Bur06] Mike Burrows. “The Chubby lock service for loosely-coupled distributed systems”. In: *Proceedings of the 7th symposium on Operating systems design and implementation*. OSDI ’06. Seattle, Washington: USENIX Association, 2006, pp. 335–350.

[CGR07] Tushar D. Chandra, Robert Griesemer, and Joshua Redstone. “Paxos made live: an engineering perspective”. In: *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*. PODC ’07. Portland, Oregon, USA: ACM, 2007, pp. 398–407. DOI: 10.1145/1281100.1281103.

[Cor+12] Miguel Correia, Daniel Gómez Ferro, Flavio P. Junqueira, and Marco Serafini. “Practical Hardening of Crash-tolerant Systems”. In: *Proceedings of the 2012 USENIX Conference on Annual Technical Conference*. USENIX ATC’12. Boston, MA: USENIX Association, 2012, pp. 41–41.

[Das08] Amazon Web Services Service Health Dashboard. *Amazon S3 Availability Event: July 20, 2008*. http://status.aws.amazon.com/s3-20080720.html. July 2008.

[Din11] Artem Dinaburg. “Bitsquatting: DNS Hijacking without Exploitation”. In: *BlackHat Security, Defcon 19*. 2011.

[FC99] Christof Fetzer and Flaviu Cristian. "A Highly Available Local Leader Election Service". In: *IEEE Trans. Softw. Eng.* 25 (5 1999), pp. 603–618. DOI: 10.1109/32.815321.

[Fen+10] Shuguang Feng, Shantanu Gupta, Amin Ansari, and Scott Mahlke. "Shoestring: probabilistic soft error reliability on the cheap". In: *ACM SIGARCH Computer Architecture News*. Vol. 38. 1. ACM. 2010, pp. 385–396.

[GA03] S. Govindavajhala and A.W. Appel. "Using memory errors to attack a virtual machine". In: *Security and Privacy, 2003. Proceedings. 2003 Symposium on*. 2003, pp. 154 –165. DOI: 10.1109/SECPRI.2003.1199334.

[HSS12] Andy A. Hwang, Ioan A. Stefanovici, and Bianca Schroeder. "Cosmic Rays Don't Strike Twice: Understanding the Nature of DRAM Errors and the Implications for System Design". In: *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS XVII. London, England, UK: ACM, 2012, pp. 111–122. DOI: 10.1145/2150976.2150989.

[Hun+10] Patrick Hunt, Mahadev Konar, Flavio P. Junqueira, and Benjamin Reed. "ZooKeeper: wait-free coordination for internet-scale systems". In: *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*. USENIXATC'10. Boston, MA: USENIX Association, 2010, pp. 11–11.

[Kim+14] Yoongu Kim, R. Daly, J. Kim, C. Fallin, Ji Hye Lee, Donghyuk Lee, C. Wilkerson, K. Lai, and O. Mutlu. "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors". In: *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*. June 2014, pp. 361–372. DOI: 10.1109/ISCA.2014.6853210.

[Lam98] Leslie Lamport. "The part-time parliament". In: *ACM Trans. Comput. Syst.* 16 (2 1998), pp. 133–169. DOI: 10.1145/279227.279229.

[Mag09] Wired Magazine. *Ma.gnolia Suffers Major Data Loss, Site Taken Offline*. http://www.wired.com/2009/01/magnolia-suffer/. Jan. 2009.

[NDO11] Edmund B. Nightingale, John R. Douceur, and Vince Orgovan. "Cycles, cells and platters: an empirical analysis of hardware failures on a million consumer PCs". In: *Proceedings of the sixth conference on Computer systems*. EuroSys '11. Salzburg, Austria: ACM, 2011, pp. 343–356. DOI: 10.1145/1966445.1966477.

[Nik+13] Nick Nikiforakis, Steven Van Acker, Wannes Meert, Lieven Desmet, Frank Piessens, and Wouter Joosen. "Bitsquatting: exploiting bit-flips for fun, or profit?" In: *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2013, pp. 989–998.

[Per+07] Frances Perry, Lester Mackey, George A. Reis, Jay Ligatti, David I. August, and David Walker. "Fault-tolerant Typed Assembly Language". In: *Proceedings of the 2007 ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI '07. San Diego, California, USA: ACM, 2007, pp. 42–53. DOI: 10.1145/1250734.1250741.

[Sch11] Ute Schiffel. "Hardware Error Detection Using AN-Codes". PhD thesis. 01062 Dresden, Germany: Technische Universität Dresden, 2011.

[Sch90] Fred B. Schneider. "Implementing fault-tolerant services using the state machine approach: a tutorial". In: *ACM Comput. Surv.* 22 (4 1990), pp. 299–319. DOI: 10.1145/98163.98167.

[Shi+02] Premkishore Shivakumar, Michael Kistler, Stephen W. Keckler, Doug Burger, and Lorenzo Alvisi. “Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic”. In: *Proceedings of the 2002 International Conference on Dependable Systems and Networks*. DSN ’02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 389–398.

[SPW09] Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. “DRAM errors in the wild: a large-scale field study”. In: *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*. SIGMETRICS ’09. Seattle, WA, USA: ACM, 2009, pp. 193–204. DOI: 10.1145/1555349.1555372.

[Ker07] Kernel bug tracker. *Data corruption with Opteron CPUs and Nvidia chipsets*. https://bugzilla.kernel.org/show_bug.cgi?id=7768. Jan. 2007.