Cascading Solution of Data Dependence Constraints with Z3

Eric W. D. Rozier[1] **and Kristin Y. Rozier**[2]
Department of Electrical Engineering and Computing Systems[1,2]
Department of Aerospace Engineering and Engineering Mechanics[2]
University of Cincinnati
2600 Clifton Ave
Cincinnati, Ohio, 45221
Eric.Rozier@uc.edu[1], Kristin.Y.Rozier@uc.edu[2]

Abstract

Large-scale data collection and analysis are revolutionizing modern scientific discovery through the growing fields of Data Science and Engineering, but Big Data also poses serious challenges for infrastructure in cost-constrained environments. NASA's Center for Climate Simulation named data infrastructure the biggest problem facing climate science, with the growth of storage needs outstripping the growth of necessary computing power by 6.67x. To combat this challenge new techniques in Data Engineering are required to help scientists, and other domain specialists, better handle the growing data deluge. In this paper we present a novel technique for efficiently improving the reliability of existing data storage systems without requiring additional hardware. Our technique allows for the intelligent allocation of additional reliability syndromes, in a provably independent manner, by formal analysis of the underlying storage system using Microsoft Research's Z3 Satisfiability Modulo Theories Solver (De Moura and Bjørner 2008; Köksal, Kuncak, and Suter 2011), and an analysis of the dependence of the satisfiability of the solution of successively more reliable systems by characterizing the entropy and resource space of possible system configurations. We provide experimental validation of our techniques, showing its efficiency, and provide an open implementation of our methods.

Introduction

The modern data deluge has presented engineers with a challenge in the form of exponential growth of data (Turner et al. 2014). This challenge is a tremendous obstalce for would-be architects of next-generation storage and data warehousing systems, but also presents a massive opportunity for the development of data-driven disruptive innovations in many scientific fields. For example, today's resource-constrained environments provide a particular challenge for advancements in aerospace as, "the bisectional bandwidth of future avionics platforms could easily exceed Petabytes with video-based applications and possibly reach the capacity of small data centers" (Blumenthal 2011). Unmanned Aerial Systems (UAS) are one of today's hottest new-technology domains yet the storage redundancy required for their safety-critical applications exceeds size, weight, price, and power consumption limits (Sababha, Rawashdeh, and Sa'deh 2012), making the system too complex and costly to fly. Space avionics vitally require fault-tolerant data storage due to radiation and harsh operating environments; combined with their very long mission lifecycles, "strategic storage" is essential to reliably capture the big data produced (Pignol 2010). Indeed, NASA's Solar Observatory produces over 1,600 gigabytes of data each day (Aeronautics and Administration 2009). Collecting large volumes and varieties of data is revolutionizing many other areas of scientific inquiry, including healthcare (Miller 2012), genomics (Howe et al. 2008), urban-planning (Batty 2013), climate-science (Schnase et al. 2014), economics (Chen, Chiang, and Storey 2012), and more.

The challenge is then the counterpoint posed by the difficulty in dealing with the scale of Big Data. According to the NASA Center for Climate Simulation (NCCS), while computing needs have risen by a factor of 300 in the last ten years, storage needs have risen by a factor of 2,000 (Schnase et al. 2014). The NCCS called storage infrastructure one of the largest challenges facing climate scientists, and the case is similar across engineering. Moreover, Big Data centers are increasingly cost-constrained, e.g., by declining federal funding, and must do more with fewer resources. In our previous research we have shown that many systems are often overprovisioned due to the realities of funding decisions in cost-constrained environments, and suggested that those resources could be used for improving the reliability of the underlying infrastructure **with no additional hardware**.

The key to this improvement involves the prediction of changes in user resource needs, demonstrated in (Rozier, Zhou, and Divine 2013) and (Bayram et al. 2015), which allows overprovisioned space to be identified. Our previous work (Bayram, Rozier, and Rozier 2015b) created a new method for efficiently computing *independent reliability syndromes* that utilize over-provisioned disk space to create dependence relationships in the data that can be exploited for reliability. We demonstrated how to practically reason about large-scale systems via encoding dependencies as a specialized variation of the well-known n-queens problem and using an SMT solver to produce strategies for maximizing reliability, and in (Bayram, Rozier, and Rozier 2015a) we detailed the formal constraints necessary to layout these

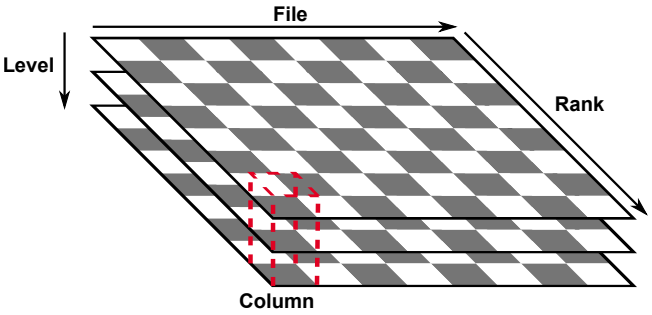


Figure 1: Latin Squares variant N-Queens representation of our problem.

additional reliability syndromes in a provably independent manner. Figure 1 illustrates our problem representation, with each rank representing a pre-existing RAID reliability group, each column representing an individual disk, and each level solving the partially independent sub-problem of forming a new reliability syndrome for blocks located on a given disk.

In this paper we extend these previous results with a novel algorithm for a cascading solver which helps to both maximize our ability to solve complex dependence problems in real-time, by successively solving more difficult, but higher reliability variants for syndrome allocation. Our methods rely on two main innovations for our algorithm. The first is a characterization of possible system configurations using features of available resource levels and entropy as input to machine learning algorithms to predict probability of satisfiability of a given syndrome allocation. Second is a division of the problem's formal constraints given in (Bayram, Rozier, and Rozier 2015a) on the basis of their relation to our cascading problem in order to calculate partial solutions which are universally useful for all subproblems using Z3.

Reliability Syndromes

Storage systems are often made more reliable through the generation of syndromes creating dependence between blocks in a file system through mechanisms such as RAID5 and RAID6, typically through the generation is XOR parity blocks. Data may be made more reliable by creating a new dependence between currently independent data. So given blocks a, b, c, and d stored on separate physical hardware, a new block $z = a \oplus b \oplus c \oplus d$ can ensure that if any block is lost, for example c, it can be easily recreated as $c = a \oplus b \oplus d \oplus z$, adding to the reliability of the underlying file system (Patterson, Gibson, and Katz 1988). Reliability can be further extended through the use of Galois fields (Chen et al. 1994), and in theory with erasure codes (Dimakis et al. 2007), however current patent encumberance has effectively removed performable erasure code algorithms from use (Henderson 2015). In practice this means for any set of initially dependent data, reliability can be increased (given sufficient space) via creation of independent syndromes as shown in 2.

In addition to the traditional RAID geometries used to

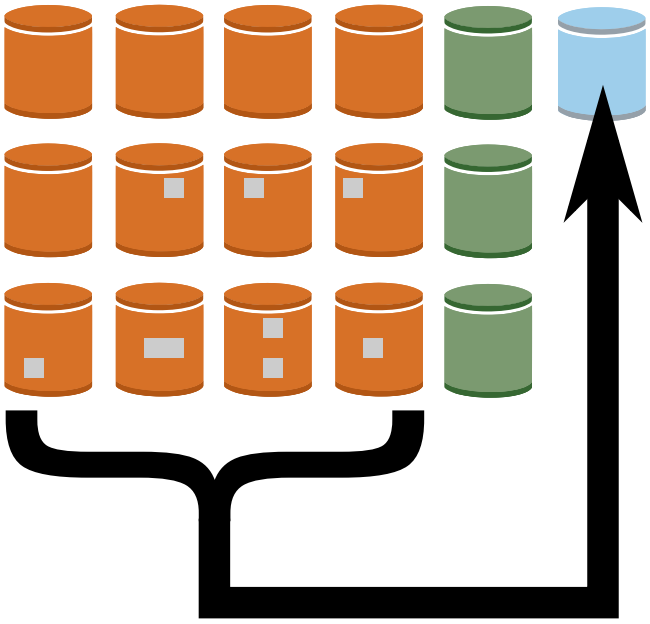


Figure 2: New virtual syndrome being created out of overprovisioned space.

generate reliability syndromes we propose that new reliability syndromes can be allocated using additional blocks, provided they are independent, thus an unused block o on a disk may be used to improve the reliability of some set of blocks a, e, k by storing XOR or Galois field parity information, i.e. $o = a \oplus e \oplus k$. The difficulty inherent in allocating these new syndromes is ensuring independent sets of blocks can be identified, along with independent free space in the storage system. As the storage system becomes fuller over time, the difficulty of this problem increases exponentially, necessitating efficient solution techniques.

Cascading Algorithm

The primary aim of our new contributions is the efficient solution of the problem of finding a large number of additional, independent, reliability syndrome allocation. Ideally we would like to maximize the number of additional reliability syndromes allocated per block in our system, characterized by the protection level q. As was shown in our previous work (Bayram, Rozier, and Rozier 2015b), as q increases the time required to solve the underlying constraints increases, and the probability of satisfiability of the underlying problem decreases. Solving for a small q helps ensure (but does not guarantee) the problem will be satisfiable, providing an answer which can be used to improve the reliability of the underlying system, but limits the additional reliability improvement. Solving for a larger q increases the risk of unsatisfiability, and the runtime, forcing us to risk having to solve for multiple qs in order to build highly reliable systems.

In order to maximize q and minimize the runtime of our solver, we propose a cascading solution algorithm which characterizes the underlying space of possible problems and

uses this characterization to intelligently classify new problems from this space based on prior training data, and selects a range of q to solve for which maximizes the chance of satisfiability, and the resulting reliability improvements, while minimizing run time.

Storage System State Characterization

The first step in our method is to build an understanding of possible file system layouts, and design a machine learning approach that can build an accurate classifier using an easy to derive set of features. For our features we chose the *availability of resources* in a given storage system and the *entropy of those available resources*. The entropy of resource allocation can be interpreted as a diversity metric. High entropy systems are those with more uniform resource distribution. Low entropy systems tend to have resources concentrated on a few disks. Our hypothesis was that systems with higher entropy, and more available resources would, in general, be more satisfiable for higher protection levels.

Given that the underlying feature space is far too large we conducted a random sampling of the feature space to ensure coverage of various resource allocation levels, and entropies. We utilized the Kumaraswamy distribution (Kumaraswamy 1980) due to it's beta-like properties for creating distributions with varying skewness and kertosis, and due to the closed-form nature of it's inverted distribution function (Jones 2009) to generate a thorough exploration of the space of possible distributions for disk resources within a pre-existing RAID stripe. By controlling skewness and kurtosis we were able to produce a number of different resource entropies easily, to ensure even coverage of the underlying space.

We found the underlying space to be highly seperable, leading to a space which is easily used for constructing a basic classifier in the form of a lookup table which can be used to estimate probability that a given protection level would be satisfiable. Excerpts[1] from these tables for per-block protection levels of 2 and 3 are provided in Figure 3.

We observe from Figure 3 that the number of available blocks per stripe for additional syndrome allocation can be used to characterize the underlying system as either **unsatisfiable** for a given protection level q, **satisfiable** for a given protection level q, or **probabilistically satisfiable** for a given protection level q. In the case of probabilistically satisfiable problem instantiations, the probability of satisfiability generally increases with increased entropy of resource distribution. Intuitively this is because the higher diversity leads, more often, to a satisfiable solution when multiple queens must compete for independent resources within a given rank.

Cascading Solution for Successive Level Protections

We utilize these tables to build a strategy for cascading solution of successive protection levels. Given a set of real

$$\forall l \in L, \forall r \in R, \forall f \in F(x[l,r,f] \in \{\lambda, \Delta, \Lambda, \epsilon\}) \quad (1)$$

$$\forall l \in L, \forall r \in R, \forall f \in F((r = |R| - 1) \rightarrow (x[l,r,f] = \Delta)) \quad (2)$$

$$\forall l \in L, \forall r \in R, \forall f \in F((\exists b | b \mathcal{R} l) \rightarrow (x[l,r,f] = \Delta)) \quad (3)$$

$$\forall r \in R, \forall f \in F(d[r,f] \geq \sum_{l} (x[l,r,f] = \Delta)) \quad (4)$$

Figure 4: Z3 assertions with Global Scoping [2]

$$\forall l \in L(\sum_{\forall r, \forall f} (x[l,r,f] = \lambda) \geq q \cdot |F|) \quad (5)$$

$$\forall l \in L(\sum_{\forall r, \forall f} (x[l,r,f] = \Lambda) \geq q) \quad (6)$$

Figure 5: Z3 assertions with Local Scoping

resource availability and entropy, we classify the underlying system using the learned tables from Section , and find some q_p, the largest q for which we predict the system to be probabilistically satisfiable, and q_s the largest q for which we predict the system to be satisfiable, and use these to define a range of protection levels, $[q_p, q_s]$, for which we will apply our cascading solver. Any $q > q_p$ will be unsatisfiable, while any $q < q_s$, while satisfiable, will provide lower reliability. As such, the optimal solution always lies within $[q_p, q_s]$.

Our cascading solver takes advantage of Z3's capability to manage constraints in the form of a stack containing assertions. This stack of assertions may contain nested scopes which can be created and destroyed. We begin by creating a set of assertions which apply globally to our problem regardless of the protection level we wish to solve for, and then create individual scopes for each $q \in [q_p, q_s]$. The global scope contains all assertions required to satisfy our constraints which are independent of q, as shown in Figure 4.

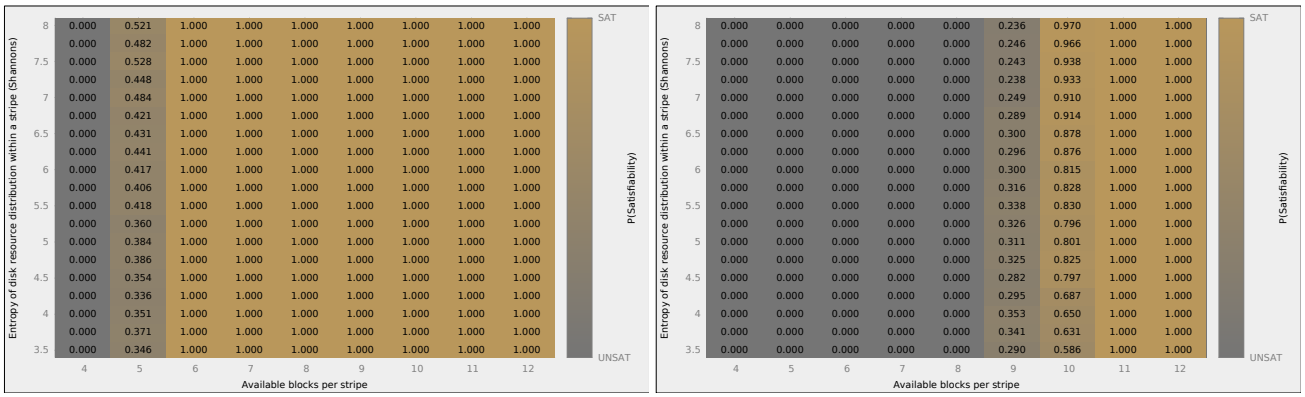
Where $d[r,f]$ is the number of resources available due to overprovisioning for disk $[r,f]$.

Each local scope contains those assertions which are dependent on q, and are shown in Figure 5

We then solve for the global scope using Z3's `SimpleSolver` to establish the initial model of our problem, and use this initial model for our cascading solution for the subsequent problems for $q \in [q_p, q_s]$. The problem corresponding to q_s is known to be satisfiable due to the classification of our current system based on its resources and entropy, and we can prove that some q_s exists experimentally due to the fact that for $q = 1$ all possible systems are

[1] Complete tables can be found at `http://dataengineering.org/research/NQueens`

[2] Where $\mathcal{R}$ is a deduplication dependence relation as defined in (Bayram, Rozier, and Rozier 2015a)



(a) Probability of satisfiability for a random r=5,f=4 subproblem given resource availability and resource entropy for $q = 2$.

(b) Probability of satisfiability for a random r=5,f=4 subproblem given resource availability and resource entropy for $q = 3$.

Figure 3: Satisfiability Probability Table Excerpts for r=5,f=4

satisfiable. This solution for q_s guarantees that we will find some (possibly non-optimal) solution, providing additional reliability. We then solve the problems for the remaining $q \in [q_p, q_s - 1]$ in ascending order (corresponding to higher probability of satisfiability) until we find an unsatisfiable result.

Experimental Results

We evaluated the speedup of using the cascading solver on an Intel Xeon E5-2620 processor, restricting the solver to a single core over a broad spectrum of boards with varying entropy and available resources for systems with initial RAID5 protection of sizes 80TB to 360TB and have summarized the runtime results in Figure 6.

Because of the large overlap in assertions not involving q, and the successive restrictions caused by larger values of q, we find large portions of the model created by Z3 are reusable leading to faster solution times for successively investigated q, translating to speed ups from 1.3x to 2.3x versus naive solution. Further speed up is achieved through using the satisfiability tables through the restriction of q to just the feasible subset $[q_s, q_p]$ with probability greater than zero of satisfiability.

Conclusions and Future Work

Our experiments have shown that the satisfiability of a given system is well characterized by the simple feature space composed of resource availability and entropy, and that using this characterization it is possible to quickly form a strategy for cascading solutions of successively harder variants of our N-Queens problem achieving speed ups from 1.3x to 2.3x. These speed ups make it feasible to solve our independent syndrome allocation problem on inexpensive hardware, while still meeting real-time deadlines for filesystem reconfiguration.

Furthermore, our classification tables can also aid us with the row-shuffling problem first introduced in (Bayram, Rozier, and Rozier 2015a). We have hypothesized in the past that a good way to achieve additional speedups would

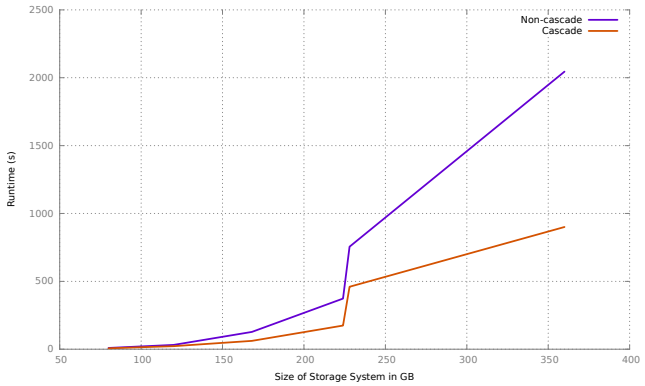


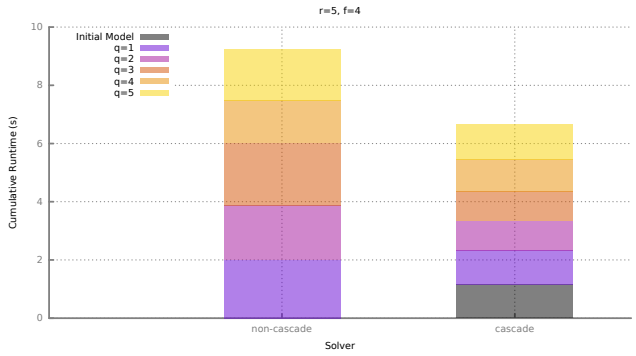
Figure 7: Runtime of all cascading solutions for various storage systme sizes.

be the compositional solution of smaller subsets of the overall filesystem. Given swift growth of runtime for our solver as the size of the underlying storage system increases, as shown in Figure 6, even with the better runtime of the cascade solver, we can achieve even better speedups by limiting the growth of the n-queens representation of our problem through compositional solution.

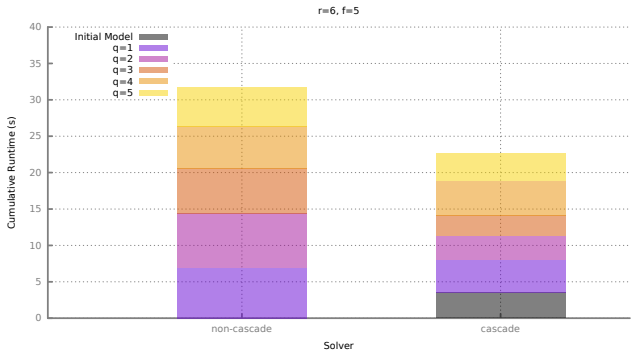
Since the n-queens representation has a size of $r^2 f^2$, it grows as the square of the total number of disks in our underlying storage system. If we partition the underlying system into fixed r, f blocks, however, the growth of the problem will be restricted to linear growth, if we can find a good partitioning. Given the independence of the ranks in our board, the solution which is easiest to generalize would be to slice the board into separate ranks, as shown in Figure 8.

These rows can then be combined into sub-boards for compositional solution, optimizing for the highest average satisfiable q by classifying the boards which can be constructed from these rows, using our classification tables from Section .

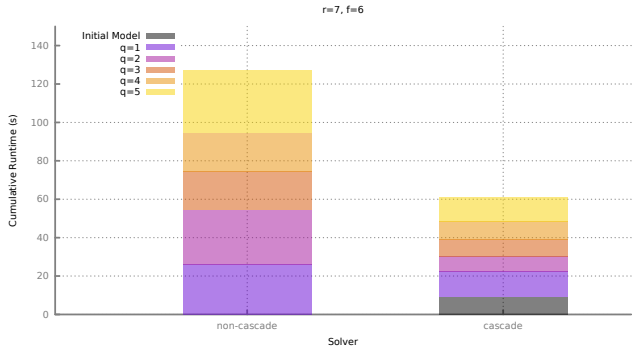
We are working to implement our solution techniques us-



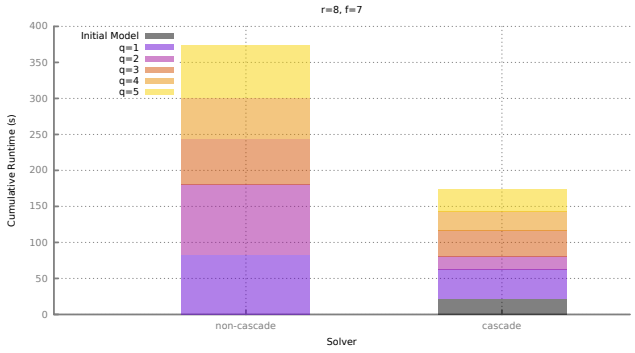
(a) Solution time in seconds for Cascading Solver and Non-Cascading Solver for an 80TB 4+1 RAID subsystem.



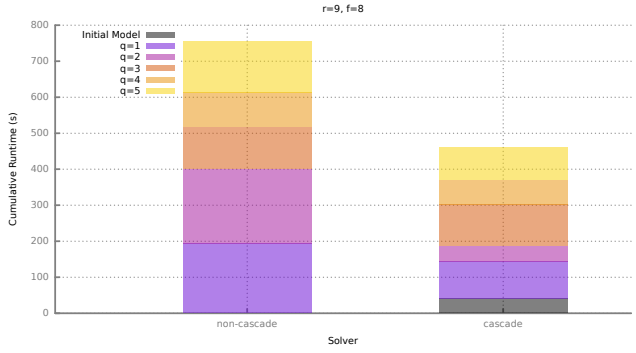
(b) Solution time in seconds for Cascading Solver and Non-Cascading Solver for an 120TB 5+1 RAID subsystem.



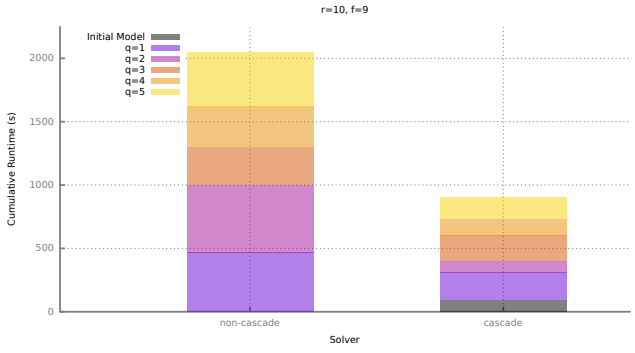
(c) Solution time in seconds for Cascading Solver and Non-Cascading Solver for an 168TB 6+1 RAID subsystem.



(d) Solution time in seconds for Cascading Solver and Non-Cascading Solver for an 224TB 7+1 RAID subsystem.



(e) Solution time in seconds for Cascading Solver and Non-Cascading Solver for an 288TB 8+1 RAID subsystem.



(f) Solution time in seconds for Cascading Solver and Non-Cascading Solver for an 360TB 9+1 RAID subsystem.

Figure 6: Runtime comparison of the cascading and non-cascading solver.

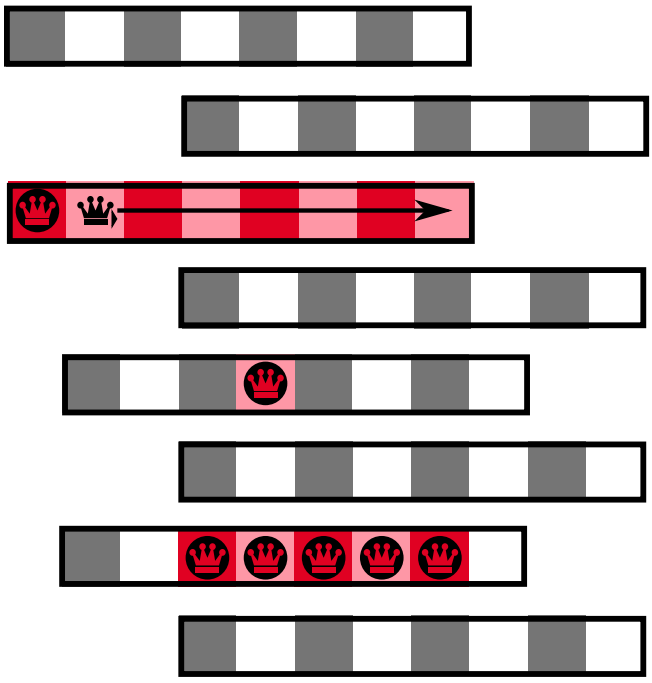


Figure 8: Rankwise slicing of a larger board for compositional solution.

ing a hardware-based middleware controller based on Z3. Given Z3's open-source status, it is a more feasible platform for recompilation on specialized architectures and embedded solutions for real SDDCs. This controller will form the basis for a system with the capabilities required to reshape incoming data traffic in order to build the proposed dynamic allocations of RAID groups in response to predictions for over-provisioning. We envision additional extensions enabling data storage system designers to query our controller regarding hypothetical disk configurations and data dependence constraints as they design a new storage system, allowing for the optimization of designs with respect to both robustness of the underlying systems and the resulting cost trade-offs.

Availability

We have made our implementation, all associated source code, and data available under the terms of the University of Illinois/NCSA Open Source License [3] at our laboratory website `http://trust.dataengineering.org/research/nqueens/`.

Acknowledgments

The authors would like to thank Dr. Nikolaj Bjorner of Microsoft Research for his help with adapting Z3, helpful comments, and proofreading. The authors would like to thank Ulya Bayram for her contributions in table calculation.

[3] http://opensource.org/licenses/NCSA

References

[Aeronautics and Administration 2009] Aeronautics, N., and Administration, S. 2009. Sdo: Solar dynamics observatory. Online: `http://www.nasa.gov/pdf/417176main_SDO_Guide_CMR.pdf`. NP-2009-10-101-GSFC; A Guide to the Mission and Purpose of NASA's Solar Dynamics Observatory.

[Batty 2013] Batty, M. 2013. Big data, smart cities and city planning. *Dialogues in Human Geography* 3(3):274–279.

[Bayram et al. 2015] Bayram, U.; Rozier, E. W.; Zhou, P.; and Divine, D. 2015. Improving reliability with dynamic syndrome allocation in intelligent software defined data centers. In *Dependable Systems & Networks, 2015. DSN'15. IEEE/IFIP International Conference on*. IEEE.

[Bayram, Rozier, and Rozier 2015a] Bayram, U.; Rozier, K. Y.; and Rozier, E. W. D. 2015a. Characterizing data dependence constraints for dynamic reliability using n-queens attack domains. In *Proceedings of the 12th International Conference on Quantitative Evaluation of SysTems (QEST 2015)*, Lecture Notes in Computer Science (LNCS), 211–227. Madrid, Spain: Springer.

[Bayram, Rozier, and Rozier 2015b] Bayram, U.; Rozier, K. Y.; and Rozier, E. W. 2015b. Characterizing data dependence constraints for dynamic reliability using n-queens attack domains. In *Quantitative Evaluation of Systems*, 211–227. Springer.

[Blumenthal 2011] Blumenthal, D. J. 2011. Terabit optical ethernet for avionics. In *Avionics, Fiber-Optics and Photonics Technology Conference (AVFOP), 2011 IEEE*, 61–62. IEEE.

[Chen et al. 1994] Chen, P. M.; Lee, E. K.; Gibson, G. A.; Katz, R. H.; and Patterson, D. A. 1994. Raid: High-performance, reliable secondary storage. *ACM Computing Surveys (CSUR)* 26(2):145–185.

[Chen, Chiang, and Storey 2012] Chen, H.; Chiang, R. H.; and Storey, V. C. 2012. Business intelligence and analytics: From big data to big impact. *MIS quarterly* 36(4):1165–1188.

[De Moura and Bjørner 2008] De Moura, L., and Bjørner, N. 2008. Z3: An efficient smt solver. In *TACAS*. Springer. 337–340.

[Dimakis et al. 2007] Dimakis, R. G.; Godfrey, P. B.; Wu, Y.; Wainwright, M. O.; and Ramch, K. 2007. Network coding for distributed storage systems. In *In Proc. of IEEE INFOCOM*.

[Henderson 2015] Henderson, C. 2015. Usenix association fast 2013 memo. Online: `https://www.usenix.org/system/files/conference/fast13/fast13_memo_021715.pdf`.

[Howe et al. 2008] Howe, D.; Costanzo, M.; Fey, P.; Gojobori, T.; Hannick, L.; Hide, W.; Hill, D. P.; Kania, R.; Schaeffer, M.; St Pierre, S.; et al. 2008. Big data: The future of biocuration. *Nature* 455(7209):47–50.

[Jones 2009] Jones, M. 2009. Kumaraswamys distribution: A beta-type distribution with some tractability advantages. *Statistical Methodology* 6(1):70–81.

[Köksal, Kuncak, and Suter 2011] Köksal, A. S.; Kuncak, V.; and Suter, P. 2011. Scala to the power of z3: Integrating smt and programming. In *Automated Deduction–CADE-23*. Springer. 400–406.

[Kumaraswamy 1980] Kumaraswamy, P. 1980. A generalized probability density function for double-bounded random processes. *Journal of Hydrology* 46(1):79–88.

[Miller 2012] Miller, K. 2012. Big data analytics in biomedical research. *Biomedical Computation Review* Winter:14–21.

[Patterson, Gibson, and Katz 1988] Patterson, D. A.; Gibson, G.; and Katz, R. H. 1988. A case for redundant arrays of inexpensive disks (raid). In *Proc. ACM SIGMOD Conf.*, 109–116. Chicago, IL: ACM Press.

[Pignol 2010] Pignol, M. 2010. COTS-based applications in space avionics. In *DATE*, 1213–1219. European Design and Automation Association.

[Rozier, Zhou, and Divine 2013] Rozier, E. W.; Zhou, P.; and Divine, D. 2013. Building intelligence for software defined data centers: modeling usage patterns. In *SYSTOR*, 20. ACM.

[Sababha, Rawashdeh, and Sa'deh 2012] Sababha, B. H.; Rawashdeh, O. A.; and Sa'deh, W. A. 2012. A real-time gracefully degrading avionics system for unmanned aerial vehicles. In *Aerospace and Electronics Conference (NAECON), 2012 IEEE National*, 171–177. IEEE.

[Schnase et al. 2014] Schnase, J. L.; Duffy, D. Q.; Tamkin, G. S.; Nadeau, D.; Thompson, J. H.; Grieg, C. M.; McInerney, M. A.; and Webster, W. P. 2014. Merra analytic services: Meeting the big data challenges of climate science through cloud-enabled climate analytics-as-a-service. *Computers, Environment and Urban Systems*.

[Turner et al. 2014] Turner, V.; Gantz, J. F.; Reinsel, D.; and Minton, S. 2014. The digital universe of opportunities: Rich data and the increasing value of the internet of things. *International Data Corporation, White Paper, IDC_1672*.