Positioning Antifragility for Software

Achyut Tiwari[a,*], Ravindara Bhatt[b]

[a]*Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat, Solan, 173234, Himachal Pradesh, India*
[b]*Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat, Solan, 173234, Himachal Pradesh, India*

Abstract

Antifragility, a concept pioneered by Nassim Nicholas Taleb, has undergone significant development over the past decades. In his work, Taleb describes antifragility as the opposite of fragility—a system that not only withstands stress and volatility but actually thrives and improves as a result. However, many existing systems, including those in Information Technology (IT) and complex economic models, tend to fail under stress. This paper aims to explore the potential implementation of antifragility principles into software architecture. By embracing the concept of antifragility, software systems could be designed to not only withstand stressors but also harness them to enhance their robustness and performance. The authors recognize that traditional approaches to software architecture often focus on minimizing failure points and ensuring stability. While these approaches are valuable, they often neglect the dynamic nature of real-world systems and fail to adapt to unforeseen challenges. The paper proposes an alternative perspective that considers stress as an opportunity for improvement. By introducing antifragile elements into software architecture, such as decentralized decision-making, self-healing mechanisms, and adaptive resource allocation, the authors argue that software systems can become more resilient, responsive, and capable of capitalizing on stress-induced disruptions. To validate their ideas, the authors present case studies and practical examples of how antifragile software architectures could operate in various domains. They also discuss potential challenges and trade-offs associated with implementing antifragility, such as increased complexity and resource requirements. By shedding light on the possibility of embracing antifragility in software architecture, this paper seeks to inspire further research and innovation in creating more adaptive and robust software systems that thrive in the face of stress and uncertainty.

Keywords: Antifragility, Software

1. Introduction

Nassim Taleb defines the opposite of fragile as antifragile: a system that becomes stronger when stressed [1]. The best examples are found in biological systems. Human Immune System, for example, becomes stronger when stressed through activity and exercise in the same way [2, 3]. Antifragile systems embrace disorder and learn from it rather than avoiding it [1].The concept of antifragility was developed in context of financial risk analysis. The universal mathematical formalism lead to application in various domain.

As we have seen above Cloud Computing facilitates the implementation of the design principle of modularity, weak links, Redundancy & Diversity including fail fast operational principle. In [4] we see how Netflix uses various tools to inject artificial failures into their systems. The Netflix one of the tool chaos monkey randomly shuts down randomly virtual machine to ensure that the streaming service tolerates the failure without the end consumer experiencing any issue. [1]. The best examples are found in biological systems. Human Immune System, for example, becomes stronger when stressed through activity and exercise in the same way [2, 3]. Antifragile systems embrace disorder and learn from it rather than avoiding it [1].

[*]Corresponding Author
Email addresses: `achyuttiwari@acm.org` (Achyut Tiwari), `ravindara.bhatt@juit.ac.in` (Ravindara Bhatt)

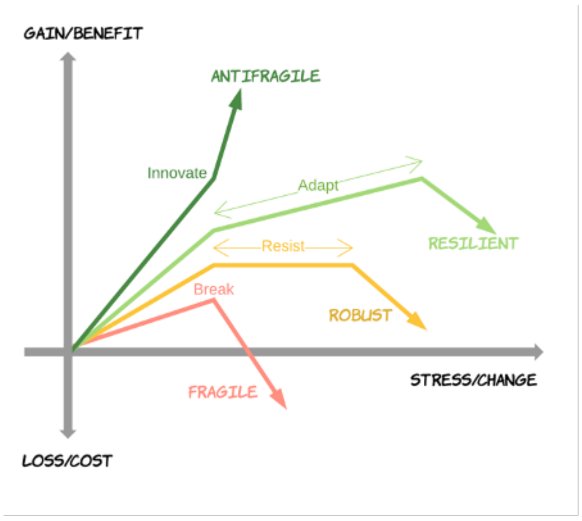

Figure 1: Antifragility with Resilient, Robust and Fragile

In [5], the researchers have proposed an implementation of antifragile concept by defining as payoff function of the complexity system. The researchers then applied antifragile concept to Random Boolean Networks of model of genetic regulatory work.

As part of Netflix effort in order to build a cloud based microservice platform, the company-built software tools that injects artificial failures into its streaming services to detect the problem in a fast paced manner [6]. Netflix's one of the other tool is Chaos Kong which simulates which simulated the outage of an entire region which basically tests the realtime ability to transition service from one region to another [6]. In 2016, Netflix launched the Chaos Automation Platform (ChAP) to run tool-basedtests automatically. ChAP helps compares real time metrics of two groups in order to determine the impact of Failure Injection Testing.

As we can see in [4, 7, 6] that regular and frequent Chaos Engineering detects weakness, determines the robustness of the system, and leads to antifragility in changing system.

In [8] we see the classification of the Taleb's Triad [1] with the perspective of service activities. The researchers have shown how hospitality services, movie theatres, Radio, television, printed press are fragile when it comes to complex systems. Meanwhile Gambling is robust. Construction, repair, maintenance of household, Information services, performing arts, security services & medical services are Antifragile [8].

As embracing disorder is the core of antifragility, the researchers in [9] have implemented the concept of antifragility in Purchasing & Supply Management and denoted it as living supply chain that can gain from disorder. We see how researchers suggested creating redundancies and evolve from a static system to dynamic system of supply chain. Redundancies are created so that when in complex system, the system can take the shock of the unseen. In the same manner how tail event in any contagious disease can have adverse effects [10].

In [11] we the authors have studies fragility and antifragility of defense brain systems in drug dependence. Authors have shown how drugs are not rewarding as the subject continues to consume them. Meanwhile the, unavoidable stimuli such as a natural disaster as an earthquake or a major traffic accident may induce post-traumatic stress disorder (PTSD), As a result of increasing the strength of the defence system.

2. Proposed Solution

A software system with dynamic, adaptive fault tolerance capabilities is antifragile: exposed to faults, it continuously improves [12]. Antifragility is about loving error meanwhile fault tolerance is a great step towards antifragility.

Fault tolerant systems are more about constantly detecting errors in the system and tolerating the same. But they are not error loving in terms of improving. In Taleb's view antifragile systems are those who constantly becomes stronger and better by going through attacks and errors [12].

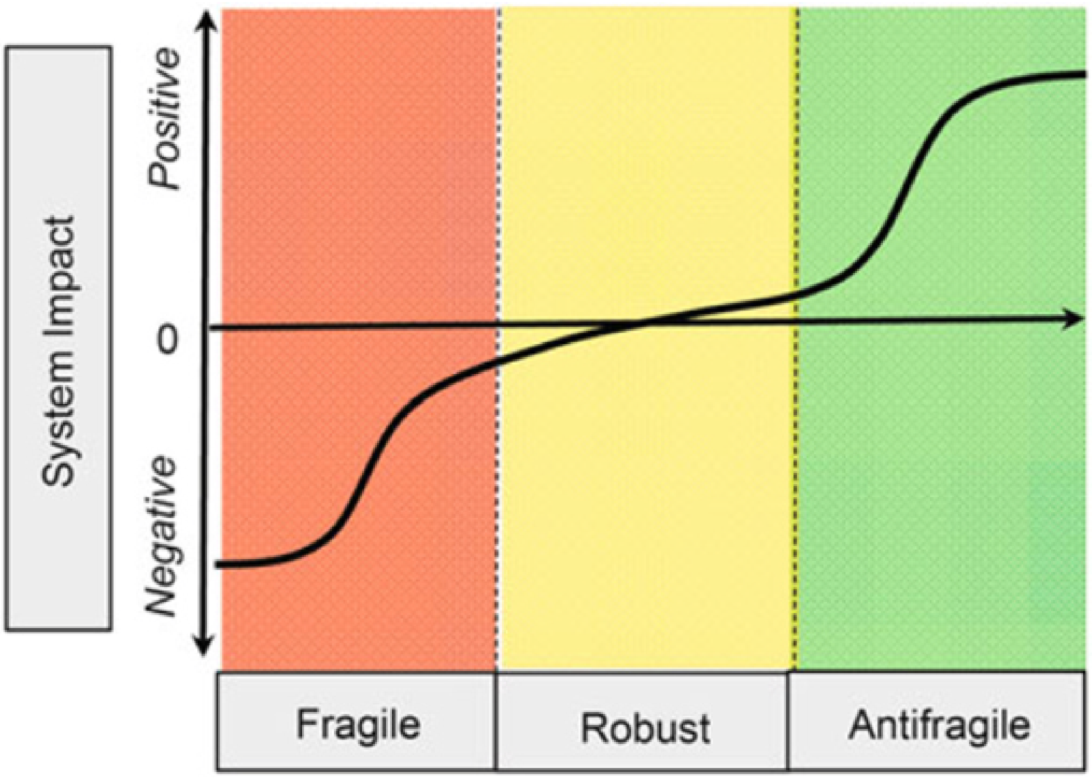

Figure 2: Antifragility with Fragile and Robust

Considering if a fault tolerance system is static as it is in most of the cases. The uncertainty in errors will lead to failure of the fault tolerant system.

"A software system with dynamic, adaptive fault tolerance capabilities is antifragile: exposed to faults, it continuously improves [12]."

Proposed Techniques for achieving Anti fragile system in software:-

1. Fault-tolerance and Antifragility.
2. Automatic Runtime Bug Repair
3. Failure Injection in Production [12].
4. Redundancy in computation power.

ICT Systems are building blocks of the Digital world.In [7] we see the principles which ensures an ICT system is antifragile or otherwise. There are various measures by which we can say a system is antifragile.Some of the Principles which ensure Anti-fragility, Modularity, Weak links, Redundancy, Diversity, Fail Fast & Systemic Failure without Failed Modules [7].

Some of the phenomenon which can make Cloud computing antifragile are Choice of System Realization, Modularity via Microservices, Weak links via Circuit Breakers, Redundancy Provided by the Cloud, Diversity Enabled by the Cloud, Fail Fast Using Software Tools&Top-Down Design and Bottom-up Tinkering [7].

3. Conclusion

In our research paper, we delve into the intriguing realm of achieving antifragility in software systems. Drawing inspiration from Nassim Nicholas Taleb's concept of antifragility, we explore various techniques and principles that can be implemented to make software architectures resilient and adaptive in the face of stress and volatility.

One of the key techniques we propose is fault-tolerance combined with antifragility. Traditional approaches to software architecture often focus on minimizing failure points and ensuring stability. However, we argue that by embracing the concept of antifragility, software systems can not only withstand stress but actually become stronger and more robust when subjected to it. We propose incorporating mechanisms that allow the system to automatically repair bugs at runtime, thus enabling it to adapt and recover swiftly.

To validate our ideas, we emphasize the importance of failure injection in production. By intentionally introducing failures into the system, we can gauge its resilience and identify potential weaknesses. This approach provides valuable insights for enhancing the system's antifragility by addressing vulnerabilities and implementing appropriate mitigation strategies.

Additionally, we advocate for redundancy in computation power. By ensuring redundant resources are available, the system can seamlessly distribute the workload and continue functioning even in the presence of failures or high demand. This redundancy acts as a safety net, bolstering the system's ability to handle stressors and maintain its performance.

Furthermore, we discuss the principles outlined in the research paper by Hole et al. (2019), which serve as a guiding framework for achieving antifragility in ICT systems. These principles include modularity, weak links, redundancy, diversity, fail fast, and systemic failure without failed modules. By incorporating these principles into software design and architecture, we can foster antifragile systems that exhibit resilience, adaptability, and improved overall performance.

In the context of cloud computing, we identify specific phenomena that contribute to making cloud-based systems antifragile. These include the choice of system realization, leveraging modularity through microservices, incorporating weak links via circuit breakers, capitalizing on the redundancy provided by the cloud infrastructure, fostering diversity enabled by the cloud, and employing fail-fast strategies using software tools. We emphasize the significance of top-down design and bottom-up tinkering to drive antifragility in cloud computing environments.

In conclusion, our research sheds light on the potential of implementing antifragile principles and techniques in software architectures. By embracing fault-tolerance, automatic bug repair, failure injection, and redundancy, along with adhering to the principles of antifragility in ICT systems, we can create software systems that thrive in the face of stress and uncertainty. These findings offer valuable insights for practitioners and researchers aiming to develop resilient and adaptive software systems that can withstand the challenges of the ever-evolving digital world.

References

[1] N. N. Taleb, Antifragile: Things that gain from disorder, volume 3, Random House, 2012.

[2] F. S. Dhabhar, A hassle a day may keep the pathogens away: the fight-or-flight stress response and the augmentation of immune function, Integrative and comparative biology 49 (2009) 215–236.

[3] N. N. Taleb, (anti) fragility and convex responses in medicine, in: International Conference on Complex Systems, Springer, 2018, pp. 299–325.

[4] K. J. Hole, Anti-fragile ICT systems, Springer Nature, 2016.

[5] O. K. Pineda, H. Kim, C. Gershenson, A novel antifragility measure based on satisfaction and its application to random and biological boolean networks, Complexity 2019 (2019).

[6] C. Rosenthal, N. Jones, Chaos engineering: system resiliency in practice, O'Reilly Media, 2020.

[7] K. J. Hole, C. Otterstad, Software systems with antifragility to downtime, Computer 52 (2019) 23–31.

[8] M. Dutkowski, Fragile, robust and antifragile services, European Journal of Service Management 28 (2018) 149–155.

[9] E. Nikookar, M. Varsei, A. Wieland, Gaining from disorder: Making the case for antifragility in purchasing and supply chain management, Journal of Purchasing and Supply Management 27 (2021) 100699.

[10] P. Cirillo, N. N. Taleb, Tail risk of contagious diseases, Nature Physics 16 (2020) 606–613.

[11] O. E. Prospero-Garcia, A. E. Ruiz-Contreras, J. Morelos, A. Herrera-Solis, M. Mendez-Díaz, Fragility of reward vs antifragility of defense brain systems in drug dependence, Social neuroscience 16 (2021) 145–152.

[12] M. Monperrus, Principles of antifragile software, in: Companion to the first International Conference on the Art, Science and Engineering of Programming, 2017, pp. 1–4.