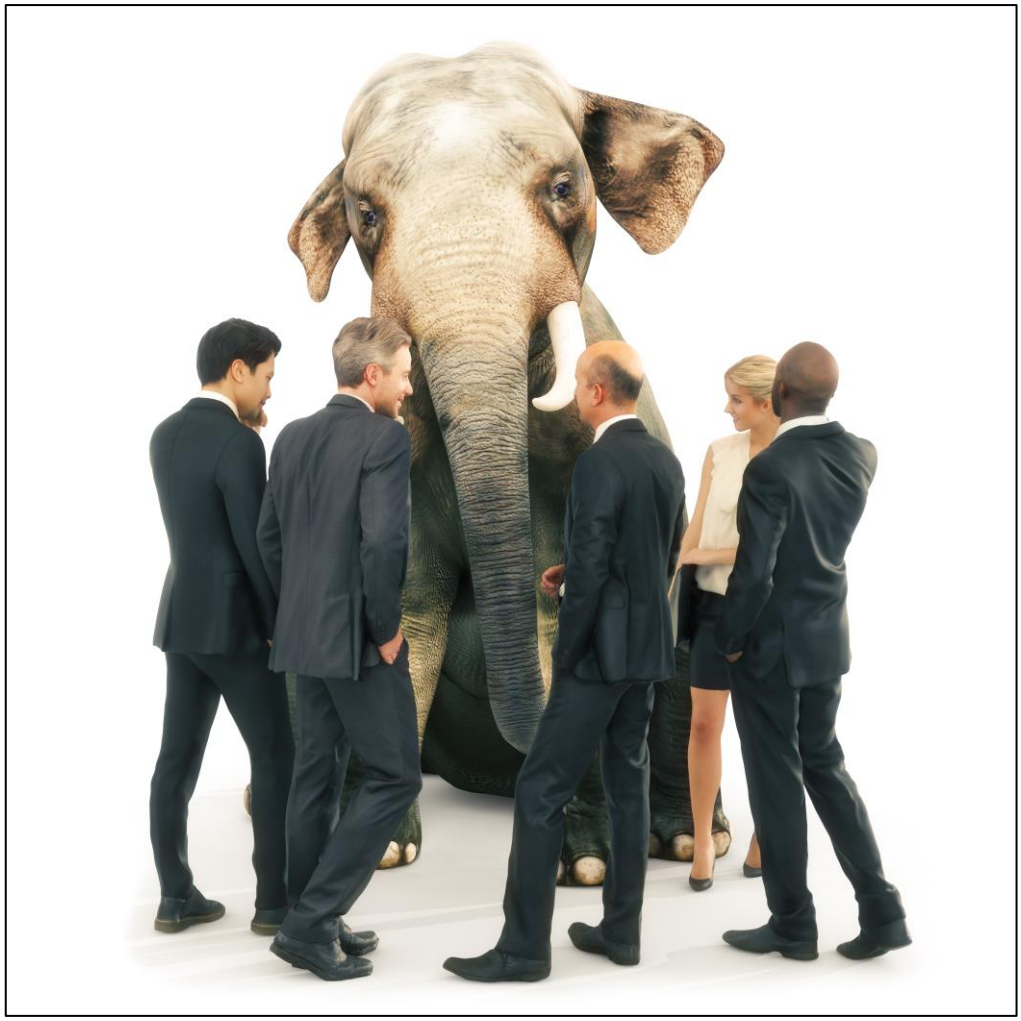
A Modern Look at Reliability, Availability, and Scalability – Part 1

A Gravic, Inc. Article
***The Connection,* May/June 2019**
by Dr. Bruce Holenstein, Paul J. Holenstein, and Dr. Bill Highleyman



The Three Pillars of Mission-Critical Computing

Computer manufacturers often stress performance and availability in their marketing literature. Yet, reliability (R), availability (A), and scalability (S)[1] are the three pillars that support mission-critical applications (Figure 1) and they all need appropriate attention. The elements of RAS are complimentary measures, but they are not the same.

- **Reliability** is a measure of how well a system returns the same correct, consistent, and uncorrupted results each time, and relies on the underlying integrity of the database, application, and system components.
- **Availability** is the *percent of uptime achieved* by the application in servicing users.
- **Scalability** is the *capability to add resources* when needed to handle the application load, and to return those resources when no longer needed.

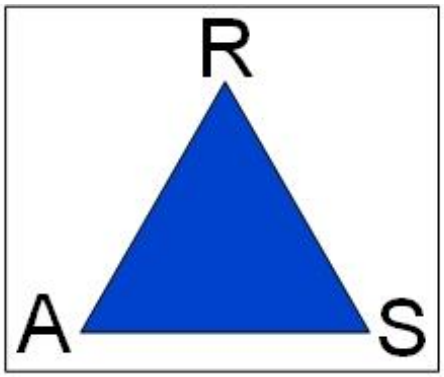


Figure 1 – RAS: The Three Pillars of Mission-Critical Computing

There is a great deal of literature available concerning availability and scalability, but relatively little published on reliability.[2] It is often left out of datacenter planning discussions because it is taken for granted that it will be acceptable. We argue that it is the proverbial "elephant in the room!"

Availability and reliability are complimentary measures, and they each deserve equal consideration for mission-critical applications. We want ***both*** 100% availability and 100% reliability but we can never achieve 100% for either. Availability is the proportion of time that an application or system is running and properly operating, and is measured in "9's." It can be expressed simply as:

Availability = time system is up / total time

Many factors lead to a down system, such as power outages, communication line failures, operator mistakes, and natural disasters.[3] In contrast to availability, reliability issues like data corruption can occur (e.g., malware problems, hardware faults, software bugs), and do not always lead to a down system. In fact, some system operators run their systems for years with known corruption issues, let alone unknown ones!

If the data in a database is corrupt at a level *C*, then the data integrity, *DI as* a measure of its reliability, is:

$$DI = 1 - C$$

What affects data integrity? Disk errors, malware, transaction errors, and bugs in the application programs can compromise application data. However, these are not always detected, nor recognized as important enough to correct.

[1]The authors acknowledge that the "S" in RAS is commonly understood to mean serviceability; however, for the purposes of this paper, the "S" is to mean scalability.

[2]For one exception, please see our article in *The Connection* (March/April, 2017), *Improving Reliability via Redundant Processing*, where we describe various methods, such as the HPE NonStop Logical Synchronization Unit (LSU).

[3]A search of *The Connection* archives will yield many articles on the topic of availability. Also, please reference the *Breaking the Availability Barrier* book series, which covers all aspects of the availability topic for mission-critical systems.

For instance, in a banking application, if there are *N* disk reads (100) in a transaction, and if the silent (undetected) error rate, e_d, of the disks in an array is one in a hundred trillion (10^{14}) reads, then the integrity of the data used by the transaction is:

$$DI = 1 - Ne_d = 1 - 100 \times 10^{-14} = 1 - 10^{-12} = \text{twelve 9s}$$

Are twelve 9s enough? If your system processes 10,000 transactions per second then that is one silently corrupted transaction every three years. You will need to decide on the business impact of that level of corruption.

Consider the case of a problem like a programming bug or malware affecting the results of transactions on a system. If the mean time between problems is *MTBP* and the mean time to detect problems and resolve the issues (*i.e.* repair the database and application) is *MTF*, then the data integrity may be approximated as:

$$DI \leq 1 - MTF/MTBP$$

For instance, consider these parameters in a hospital application modifying each accessed patient record: a malware infection goes undetected for two weeks, and the hospital only averages one issue every hundred years. In this case, DI ≤ 1 – 2/5200weeks = 0.9996 – around three 9's *or less*, because the malware might have actually encrypted all of the hospital's patient records in that two-week period!

Companies already know that they need to specify their datacenters' availability in terms of their:

- *Recovery Point Objective (RPO)* – the amount of data they may lose as the result of a system failure,
- *Recovery Time Objective (RTO)* – the amount of time required to recover from a system failure.

We argue that companies also need to know their:

- *Integrity Point Objective (IPO)* – the amount of corrupted data that the application, company, and users can tolerate,
- *Integrity Time Objective (ITO)* – the amount of time that the application, company, and users can tolerate corrupted data before the problem is resolved.

Figure 2 shows the relationships between these important measures. Businesses with mission-critical applications *must seek to drive RPO, RTO, and IPO, ITO all to zero*.

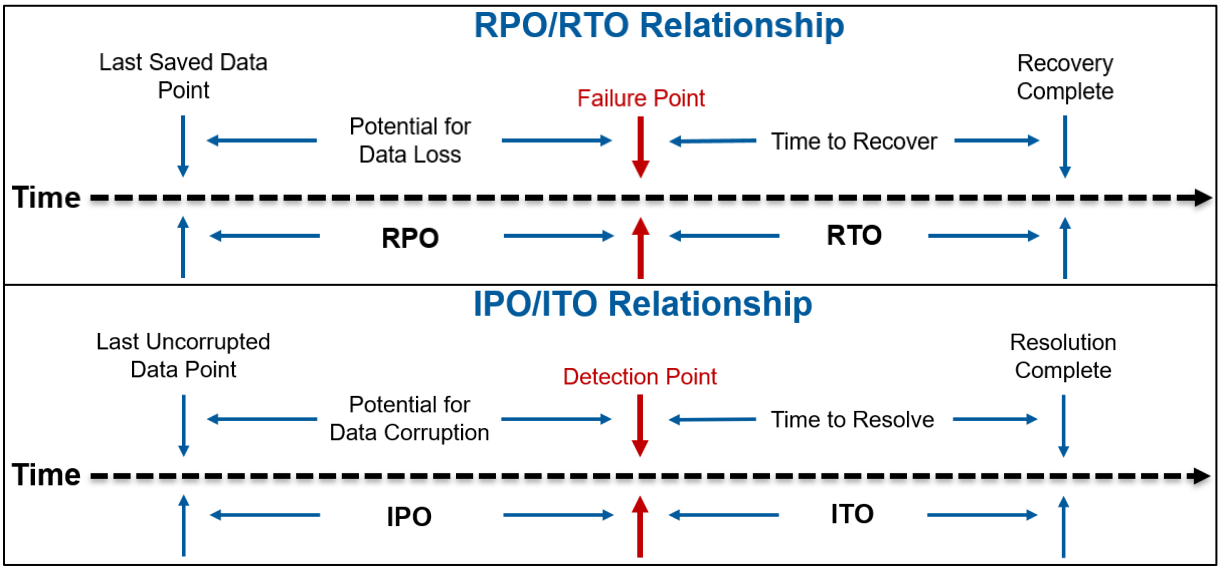


Figure 2 – RPO/RTO and the Similar Relationship that Exists for IPO/ITO

Hardware Is Not the Answer

Although we can work to achieve high reliability and availability in our applications, recent news has revealed an alarming number of chip-level vulnerabilities, which are difficult to identify, manage, or solve, as explained in these three examples:

- *Meltdown* is a hardware vulnerability affecting x86 processors, and allows a rogue process to read all of the chip's memory, even when unauthorized.
- *Row Hammer* is an unintended side-effect in dynamic random-access memory (DRAM), where memory cells leak their charges and interact electrically between themselves, possibly changing the contents of nearby memory rows.
- *Spectre* is a vulnerability that affects modern microprocessors that perform branch prediction, which is the act of assuming the direction of a branch in advance and beginning the data processing in that branch. If the expected branch is not executed, the results of the advanced processing are discarded. However, the data remains in cache, and an attacker may be able to extract that data. Therefore, it is a false assumption that "error correcting" system hardware is working well-enough to prevent data integrity problems. After all, most of this system hardware was designed to detect errors caused by radiation bit-flips and other external triggers, rather than an enemy operating within.

What's Next

In the next parts of this series, we will explore ways to improve reliability, measure IPO and ITO, and explore several new architectures that will yield a superior IPO and ITO for mission-critical systems.

International Partner Information

Global

Hewlett Packard Enterprise
6280 America Center Drive
San Jose, CA 95002
USA
Tel: +1.800.607.3567
www.hpe.com

Japan

High Availability Systems Co. Ltd
MS Shibaura Bldg.
4-13-23 Shibaura
Minato-ku, Tokyo 108-0023
Japan
Tel: +81 3 5730 8870
Fax: +81 3 5730 8629
www.ha-sys.co.jp

Gravic, Inc. Contact Information

17 General Warren Blvd.
Malvern, PA 19355-1245
USA
Tel: +1.610.647.6250
Fax: +1.610.647.7958
www.shadowbasesoftware.com
Email Sales: shadowbase@gravic.com
Email Support: sbsupport@gravic.com

Hewlett Packard Enterprise Business Partner Information

Hewlett Packard Enterprise directly sells and supports Shadowbase Solutions under the name ***HPE Shadowbase***. For more information, please contact your local HPE account team or visit our website.