Improving Reliability via Redundant Processing

Dr. Bruce D. Holenstein >> **President & CEO** >> Gravic, Inc.
Dr. Bill Highleyman >> **Managing Editor** >> Availability Digest
Paul J. Holenstein >> **Executive Vice President** >> Gravic, Inc.

The three pillars of mission-critical systems are summarized by the acronym "RAS." It stands for Reliability, Availability, and Scalability. Reliability is the probability that the system will produce correct outputs. Availability is the probability that the system is operational. Scalability is the ability of the system to handle different size loads in a predictable manner.[1]

What is the difference between reliability and availability? If a system is available but is generating erroneous outputs, it is not reliable. Alternatively, if a system is generating reliable outputs while operational, but it is not always operational, then it is not always available.

Why would a system produce incorrect data? The problem could be a hardware fault, a firmware bug, or a software error. Bad data also can be generated if the system has been infected with malware. Depending upon its design, malware can corrupt any operation within the system. Malware is generally thought to be a software issue. However, malware also can be introduced into the hardware or the firmware of a system during its manufacture.

How can we determine if a system is malfunctioning before damage to the end user or environment takes place? One answer is to use a "Validation Configuration," a redundant system in which two or more sub-systems are running in parallel and are processing the same requests, presumably arriving at the same results. Preferably, the sub-systems are from different manufacturers to ensure they both do not contain the same faults, if any. Their results (or even intermediate processing states) are compared. As long as the results agree, it safely can be assumed that the sub-systems are operating properly. If one (or both) sub-systems have been infected with malware, the results will not agree; and the sub-systems should be taken out of service and checked to determine the problem.

Why Would There Be an Error?

There are several reasons why a sub-system that appears to be operating properly could be delivering erroneous results. Common reasons include a piece of hardware that is producing memory or disk read errors or a sub-system that has been infected with malware of some sort.

It also is possible (though unlikely) that the sub-system hardware or firmware could have been modified during manufacture to provide malicious results. In the most important mission-critical cases involving financial or health decisions or crucial process control sub-systems, care should even be taken that there is no common manufacturing point in the sub-systems, such as the printed-circuit masks being fabricated by the same company.

One challenge with malware is that it may be too late by the time the malware is detected. The system may appear to be operating properly, but the bad data that it is producing due to the malware infection may not be discovered for days, weeks, or even months, if at all.

An interesting example occurred several years ago in a banking application that calculated interest payments for banking customers. Typically, an interest calculation results in fractions of a penny. For instance, 3% of $167.58 is $5.0274. The customer is credited with an interest payment of $5.02. The remainder, $0.0074, is called overage. It is credited to the bank itself.

However, a hacker was able to install malware that took a portion of the overage and credited it to his account. The amount for each transaction was so small that it wasn't noticeable; but over hundreds of thousands of transactions, it amounted to a tidy sum. The malware was not discovered until the next annual audit of the system.

Running the sub-system in a Validation Configuration may expose these types of errors and corruption.

Comparing Results

Several methods exist with which the results of independent sub-systems can be compared. Two of those methods are Logical Synchronization Units (LSUs) and a new scheme that operates at the transaction level.

Logical Synchronization Unit (LSU)

An LSU, such as that used in the HPE Itanium S-series NonStop models, is a hardware device that compares two or more

[1] Originally, IBM called the "S" Serviceability, which is the speed with which a system can be repaired. However, Serviceability is characterized by the mean time to repair and is already incorporated into the term Availability.

values to ensure that they match. An LSU can be used to compare the outputs of two sub-systems (a DMR LSU – dual modular redundancy) or three or more sub-systems (a TMR LSU – triple modular redundancy). If all outputs agree, the result is published by the LSU. See Figure 1.

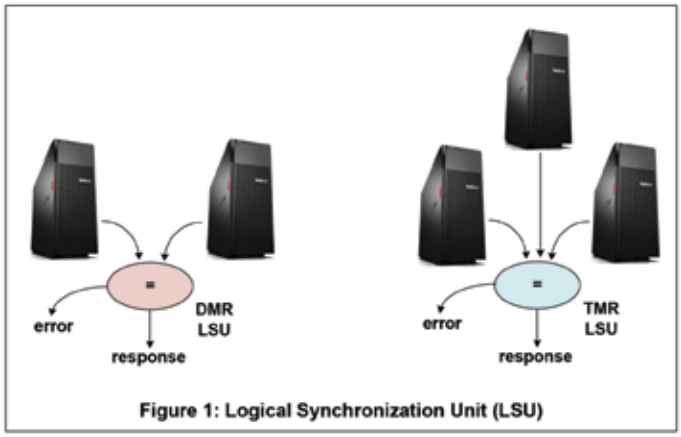

Figure 1: Logical Synchronization Unit (LSU)

In a DMR LSU, if the results do not agree, an error is returned by the LSU. In a TMR LSU, if one of the results is different, that result is ignored; and the majority result is returned. An error is generated identifying the sub-system that produced the erroneous output, and it can automatically be taken out of service.

An LSU was previously used by NASA for the space shuttle to compare the outputs of four independent computers (1, 2, 3, 4), as shown in Figure 2. The computers exchanged interprocess messages over four interprocessor synchronization buses (5, 6, 7, 8). The outputs of all four computers were sent to a voting LSU (9). The LSU outvoted the results of any failed computer and sent the correct output (the one generated by the majority of the computers) to the appropriate actuator in the space shuttle. The astronauts were instructed to turn off a computer that was generating false outputs.

An LSU can face several challenges:

- It represents a single point of failure.
- The LSU itself needs to be validated that it is free of infection.
- It must be simple in order to minimize the probability of failure.
- Therefore, it can vote only on simple inputs.
- If an error is caused by a malicious hardware, firmware, or software implementation, an LSU may not detect it because the same error will exist on all sub-systems.
- A major class of systems is transaction-processing systems, in which the different sub-systems and/or CPUs cannot operate in lock-step, and the outputs are thus not directly comparable.

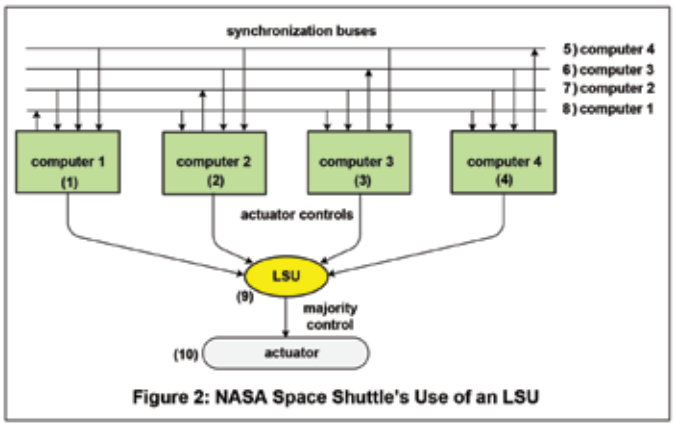

Figure 2: NASA Space Shuttle's Use of an LSU

Transaction Indicia Matching

The authors' ongoing research focuses on improving the data reliability of transaction-processing systems at the transaction level. The premise is that the same request fed into two independent sub-systems running the same application must produce the same transactional changes to the database, or else there is a reliability problem. The method, which we call *"transaction indicia matching* (TIM)," is described below and detects reliability problems, such as those caused by malware infections, operator malfeasance, or hardware or software errors.

The TIM method operates by matching "indicia" generated locally by the Validation Configuration. Indicia are measures of the sub-system state at any particular point in time and are primarily produced at the end of the DML (data manipulation language) operations made to a database. Indicia are calculated by a trusted piece of software or hardware called an *"Indicia Engine"* and may be arbitrarily complex. For instance, in a transaction-processing system, indicia may be the full set of DML changes that were made to the database by a transaction; or they may be a hash sum of the changes that were made to the database.

With two (or more) sub-systems running in parallel, the indicia calculated by each sub-system are compared in a Validation Configuration to ensure all sub-systems are operating correctly. It is optimal if the sub-systems are manufactured to the same specifications by different manufacturers and that no common point of manufacture exists (such as integrated circuit masks). This prevents a design error or a malware error maliciously introduced during manufacture from appearing in both sub-systems. The sub-systems are running the same application, preferably implemented by different software teams.

The indicia calculated by the Indicia Engines in each sub-system are compared. One means of transactional indicia matching is via direct communications between the Indicia Engines, as shown in Figure 3. Two sub-systems (1) and (2) are running versions of the same application (3). Each application version has an Indicia Engine attached or built into it (4).

At specific synchronization points, each application pauses; and each Indicia Engine calculates an indicium representing its application's current state. The Indicia Engines are connected via a communication channel (5, 6). They exchange their calculated indicia (7) and compare their own with that of the other sub-system. If the indicia agree, the Indicia Engines release the applications; and processing continues. If they don't agree, the applications are halted; and an error is posted.

In a DMR system, in the event of an error, both sub-systems can be shut down and tested. In a TMR system, the faulty sub-system is shut down and processing continues.

A Validation Configuration implementation that is designed to fit existing applications is shown in Figure 4. The Validation Configuration comprises two sub-system nodes (1, 2). An Indicia Engine is attached to each application that calculates appropriate indicia at one or more synchronization points within the application. The Indicia Engines communicate via a bidirectional synchronous data replication engine such as the Shadowbase® data replication engine from Gravic, Inc.

An application (3) sends an identical request (4) to both sub-

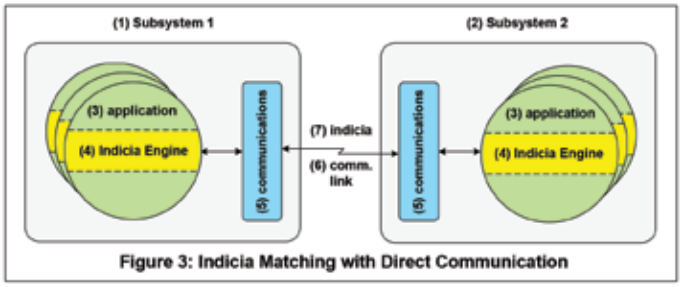

Figure 3: Indicia Matching with Direct Communication

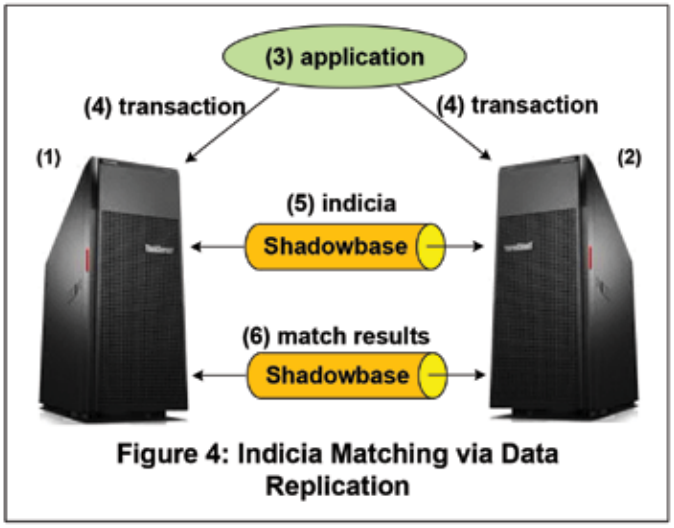

Figure 4: Indicia Matching via Data Replication

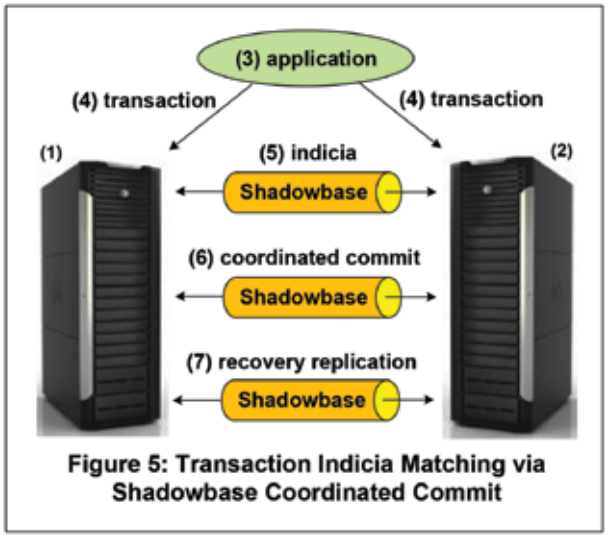

Figure 5: Transaction Indicia Matching via Shadowbase Coordinated Commit

systems. Each sub-system calculates indicia at the one or more synchronization points and sends its calculated indicia to the other sub-system via the bidirectional data replication engine (5). As an example, the indicia can be a hash sum of the changes to be made to the database by a transaction. The hash sum can be calculated by a User Exit[2] in the data replication engine. The indicia received from the remote sub-system are compared to the indicia calculated by the receiving sub-system.

At the end of request processing, each sub-system informs the other sub-system as to the results of its indicia matching (6). If all of the indicia have matched properly in both sub-systems, the result of the request processing is accepted by both sub-systems. In this case, the Indicia Engine at each node instructs the data replication engine to vote in favor of transaction commitment. If there is a mismatch in the calculated indicia by either sub-system, an error is posted and the transactions are aborted. In a DMR system, both sub-systems should be shut down and analyzed. In a TMR system, the sub-system that doesn't match with the other sub-systems is shut down; and processing continues with the operational sub-systems.

High Availability Transaction-Processing Systems

In transaction-processing systems, dual sub-system nodes (1, 2) often are deployed to achieve high availability (see Figure 5). Should one sub-system fail, all transactions can be sent to the surviving sub-system for processing. Advantage can be taken of this system redundancy to construct a Validation Configuration to detect data-reliability problems. The nodes exchange indicia at appropriate synchronization points to ensure that nothing has corrupted processing.

As shown in Figure 5, an application (3) sends a transaction (4) to both nodes in the Validation Configuration. The transaction-processing applications in each node make changes to the database at its node. User exits in the data replication engine

[2] A User Exit is a customized piece of logic that can be added to the Shadowbase data replication engine to perform selected processing on the various transaction components.

serve the purpose of the Indicia Engines to calculate the indicia. The calculated indicia is exchanged between the two nodes (5) via data replication, and each node compares its calculated indicia with that of the other node. If both nodes agree with the indicia calculated at the other node, the transaction is committed. Otherwise, the transaction is aborted.

If the Shadowbase data replication engine is employed, the transaction can be committed (or aborted) via the Shadowbase coordinated-commit[3] facility (6). Coordinated commits work in this case as follows. If a node's indicia have matched properly with the indicia sent by the remote node, each node will send a token to the other node to say that it is ready to commit its transaction. Each node will respond to the token with an indication that it is ready to commit the transaction. When a node receives a confirmation from the other sub-system, it commits the transaction. If either node cannot do so, it instead will send an abort request to the other sub-system. In this case, both nodes will abort their transaction.

Data replication used in this way to achieve both high-availability and data reliability has another important benefit if one of the nodes is taken off-line due a problem. In this case, the data replication tool will queue the changes on the running node and be able to forward them seamlessly to the recovered node once it has been brought back on-line (7).

Encryption

As an option, the indicia being exchanged between the systems can be encrypted. This can prevent a "man-in-the-middle" attack, in which an attacker can modify an indicium that does not match to one that matches in order to mask a malware infection. Alternatively, the attacker can change an indicium that matches to one that does not match to cause a sub-system denial of service outage.

Certifying a New Sub-system

The TIM approach can be used to certify the reliability of a new sub-system. The new sub-system is put into operation along with a known and trusted sub-system. As the sub-systems process requests, they both calculate indicia. The new sub-system sends its indicia to the trusted sub-system, which compares them to its own indicia.

If the indicia should not match, the new sub-system can be taken out of service for further diagnostics. The error could be in the hardware design, the firmware, or the software. Alternatively, the error could be caused by malicious malware that has infected the new sub-system.

Summary

The proper operation of a sub-system can be verified via a Validation Configuration that compares the operation of one sub-system to another sub-system running the same applications. Verification is accomplished by comparing indicia generated by the two sub-systems at specific synchronization points. If the indicia agree, the sub-systems are operating properly. If the indicia do not agree, one of the sub-systems is misbehaving.

Comparing sub-system outputs via TIM is a significant improvement over the use of an LSU for validation purposes in a transaction processing system because of the set of challenges faced by an LSU, not the least of which is that it represents a single point of failure in the system. There is no single point of failure when using transaction indicia matching.

Our research shows that TIM can be implemented with a synchronous data replication engine such as Shadowbase from Gravic, Inc. We would appreciate feedback if you are interested in exploring this concept with us for your real-world application.

[3] Coordinated Commits are a special form of processing that allows the results of two or more transactions to be coordinated to the same conclusion before they complete. Refer to the Breaking the Availability Barrier book series for more information.

Dr. Bruce D. Holenstein, President and CEO. Dr. Holenstein leads all aspects of Gravic, Inc. as President and CEO. He started company operations with his brother, Paul, in 1980, and is presently leading the company through the changes needed to accommodate significant future growth. His technical fields of expertise include algorithms, mathematical modeling, availability architectures, data replication, pattern recognition systems, process control and turnkey software development. Dr. Holenstein is a well-known author of articles and books on high availability systems. He received his BSEE from Bucknell University and his Ph.D. in Astronomy and Astrophysics from the University of Pennsylvania.

Dr. Bill Highleyman is the Managing Editor of The Availability Digest (www.availabilitydigest.com), a monthly, online publication and a resource of information on high- and continuous availability topics. His years of experience in the design and implementation of mission-critical systems have made him a popular seminar speaker and a sought-after technical writer. Dr. Highleyman is a past chairman of ITUG, the former HP NonStop Users' Group, the holder of numerous U.S. patents, the author of Performance Analysis of Transaction Processing Systems, and the co-author of the three-volume series, Breaking the Availability Barrier.

Paul J. Holenstein is Executive Vice President, Gravic, Inc. He has direct responsibility for the Gravic, Inc. Shadowbase Products Group and is a Senior Fellow at Gravic Labs, the company's intellectual property group. He has previously held various positions in technology consulting companies, from software engineer through technical management to business development, beginning his career as a Tandem (HPE NonStop) developer in 1980. His technical areas of expertise include high availability designs and architectures, data replication technologies, heterogeneous application and data integration, and communications and performance analysis. Mr. Holenstein holds many patents in the field of data replication and synchronization, writes extensively on high and continuous availability topics, and co-authored Breaking the Availability Barrier, a three-volume book series. He received his BSCE from Bucknell University, a MSCS from Villanova University, and is an HPE Master Accredited Systems Engineer (MASE). To contact the author, please email: SBProductManagement@gravic.com . Hewlett Packard Enterprise directly sells and supports HPE Shadowbase Solutions (www.ShadowbaseSoftware.com); please contact your local HPE account team.