

Data Integrity in HP NonStop Servers

Alan Wood, Robert Jardine and Wendy Bartlett

Abstract—Data integrity is a key requirement for business-critical computing. HP NonStop servers incorporate specialized hardware and software features to prevent data corruption due to transients and other errors. These features include lockstepped microprocessors (a duplicate and compare technique), memory error correction and background memory scrubbing, ServerNet technology for interprocessor communications and input-output (I/O), and end-to-end checksums on data written to disk drives. The newest generation of NonStop Servers contains a new replicate and compare technology, called the NonStop Advanced Architecture, that allows processors to continue operating in the presence of transient errors with no loss of data integrity.

Index Terms—fault tolerance, data integrity, soft errors

I. INTRODUCTION

HP NonStop® servers¹ are used for applications that are critical to health, safety, and financial security, such as emergency 911 services, healthcare, and stock market transactions. In such applications, it is a fundamental expectation that computer systems will always provide the correct data. Incorrect data could cause an emergency call to be routed to the wrong location. A change of even a single bit of data in a financial transaction can alter its value by millions of dollars or cause the transaction to be recorded in the wrong account. Because of their business-critical application heritage, ensuring data integrity has always been a fundamental requirement for HP NonStop Servers.

Soft, or transient, errors are a one-time change (often called a bit flip) to a data value in a computer system usually caused by alpha particles, cosmic rays, or other sources of electronic noise. Soft errors can potentially cause data corruption in computer systems. This phenomenon is well known for memory devices, and memory error detection and correction is standard in most computers. However, soft errors can also cause data corruption in microprocessors [1], and many other devices have on-chip storage such as buffers

that can be disrupted. Devices will become even more susceptible to data corruption in the future [2] due to the trends toward smaller component feature sizes that require less energy to be disturbed; increased component density in a chip, meaning that high-energy particles are more likely to collide with a component in the chip and cause a disruption; and reduced amounts of voltage used to perform calculations and store data, meaning that electronic noise immunity and transistor breakdown voltage are reduced.

The occurrence of a soft error does not automatically result in an incorrect calculation or an incorrect value in a database. For example, the corruption may change a bit in a memory location that is not used or is overwritten before it is used or may cause an exception or retry or processor halt by branching to an invalid instruction or accessing an invalid address. The computer system may detect the data corruption and correct or mitigate the effects of the incorrect value. However, it is also possible that the soft error can propagate to become an incorrect value in a transaction or data base [3], data corruption that cannot be tolerated by the NonStop Server customer base.

There are two approaches to improving data integrity. The first is to prevent data corruption from occurring. This is the province of integrated circuit manufacturers. Among other techniques, they use special types of materials to make devices that are less sensitive to the types of electronic noise. The second approach is to detect and possibly correct the data corruption. Although some integrated circuit manufacturers provide on-chip error detection and correction, this approach is primarily the province of system vendors. Almost all computer vendors provide some form of error detection and correction on main memory, and most provide some form of parity protection on caches (and, recently, even error correction on large caches). However, as described in this paper, HP NonStop Servers provide data corruption detection and correction throughout the entire system rather than just on the memory devices.

II. NONSTOP SERVER AVAILABILITY AND DATA INTEGRITY FEATURES

Since HP NonStop Servers are used in business-critical applications, availability is an important requirement. HP NonStop Servers are designed to have no single point of failure, using a combination of hardware and software fault

Alan Wood was with Hewlett Packard, Cupertino, CA 95014 USA. He is now with Sun Microsystems, Santa Clara, CA 95054 USA (email: alan.wood@sun.com)

Robert Jardine was with Hewlett Packard, Cupertino, CA, 95014 USA. He is now with Tibion Corporation, Moffett Field, CA 94035 USA (email: robert.jardine@earthlink.net)

Wendy Bartlett is with Hewlett Packard, Cupertino, CA 95014 USA (email: wendy.bartlett@hp.com)

¹ Formerly Tandem NonStop Servers.

tolerance. Each major hardware subsystem (processors, disks, power supplies, controllers, and so forth) is implemented with multiple redundant units. Software fault tolerance is implemented via process pairs using the NonStop Operating System. A process pair consists of a primary and a backup process running in separate processors. If the primary process fails because of a software defect or processor hardware failure, the backup process takes over all the duties of the primary process.

Figure 1 (from [5]) shows a four processor NonStop Server. Each processor contains two microprocessors that duplicate and compare each computation using lockstepping as described in Section II.A. Interprocessor and I/O communication is accomplished using a redundant ServerNet system area network (SAN) described in Section 2.3. As shown in the diagram, the ServerNet SAN, storage adapters, and network adapters are redundant. For more details on the system architecture, see [6].

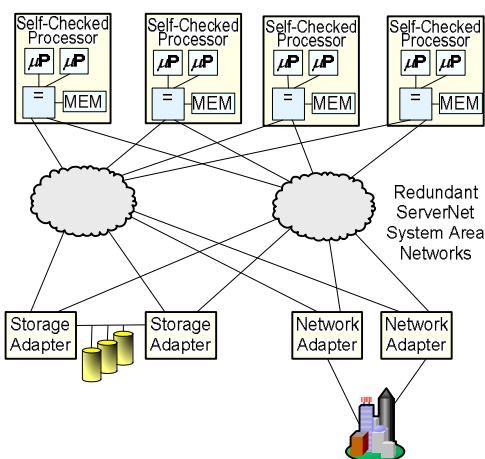


Figure 1: 4 Processor NonStop System with Duplicated and Compared Microprocessors

Because HP NonStop Servers are fault tolerant, it might be expected that all their hardware modules are fault tolerant. On the contrary, the hardware modules are fault *intolerant*, that is, an incorrectly functioning hardware module detects the problem and, if it cannot recover immediately, shuts itself down as quickly as possible to avoid propagating potentially corrupt data. This concept is called *fail-fast* and prevents hardware errors from leading to data corruption. The fail-fast design philosophy means that each hardware and software module is self-checking and immediately ceases operation rather than permit errors to propagate. Although a module may attempt to recover from a fault—for example, ECC on memory—it immediately halts if there is any possibility that data corruption will result from continued operation. This helps avoid data corruption and system outages caused by a single propagated hardware or software fault. The lockstepped microprocessors described in the next section are an example of the fail-fast philosophy.

This fail-fast design approach is important both for data integrity and for serviceability. If either a hardware or software error is allowed to propagate, it is often difficult to determine where the failure occurred because the evidence is either gone or obscured. When a NonStop processor halts, it saves its state. Based on that state, it is almost always possible to determine whether the problem was caused by hardware or software, and often whether the hardware problem is transient or permanent. This allows customers or service personnel to determine whether to replace the processor or to dump and reload it. The processor state information contained in the dump, in conjunction with the fail-fast philosophy that minimizes error propagation, allow HP's hardware and software developers to do a better job of analyzing problems.

The specific features of the NonStop Server architecture that prevent soft errors from causing data corruption are:

- Lockstepped microprocessors
- Memory error correction and background memory scrubbing
- ServerNet technology
- Other hardware error checking
- End-to-end checksums
- System software error checking

A. Lockstepped microprocessors

Modern microprocessors have many features, e.g., cache parity tags and state machine checking, used to detect and possibly recover from soft errors. However, there are still many unchecked areas of the microprocessor vulnerable to undetected data corruption. NonStop Servers detect these errors using lockstepped microprocessors. Each NonStop Server processor contains two microprocessor chips. These microprocessors are lockstepped; that is, they run exactly the same instruction stream. The output from the two microprocessors is compared, and if it should ever differ, the processor output is frozen within a few nanoseconds so that

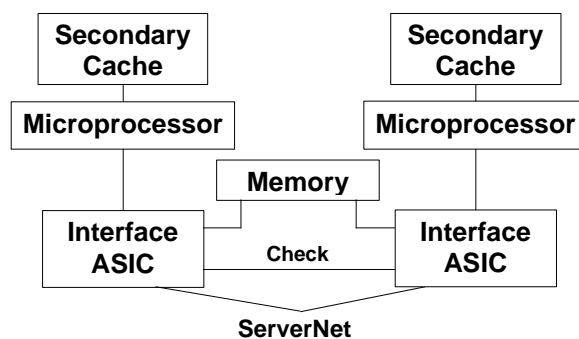


Figure 2. Lockstepped Microprocessors

the corrupted data cannot propagate.

This lockstepped microprocessor architecture is shown in Figure 2. An incoming request (ServerNet packet) is sent to both interface application-specific integrated circuits (ASICs), which translate it and forward it to the microprocessors and to memory. Each microprocessor simultaneously services the request with simultaneous access to secondary cache and memory using controllers and buses that are not shown. The output data from the microprocessors is compared by the interface ASICs during I/O and memory accesses. If there is even a single bit difference, the microprocessor outputs are immediately frozen to prevent a corrupt ServerNet packet from being transmitted. If there is a soft error that causes an undetected data corruption anywhere in the microprocessor, caches, buses or control logic, the lockstepped microprocessor architecture will detect it and prevent it from propagating.

B. Memory error correction and background scrubbing

The most likely location of soft errors is in computer memory. NonStop Servers and other computers use state-of-the-art memory detection and correction to correct single-bit errors, detect double-bit errors, and detect “nibble” errors (three or four bits in a row, which could be caused by complete failure of a single DRAM). However, NonStop Servers go beyond this protection by modifying the vendor’s ECC to include both address and data bits in the ECC code. This helps avoid reading from or writing to the wrong memory location.

In addition, NonStop Servers provide a memory “sniffer/scrubber” that runs in the background and tests the entire memory every few hours. The memory sniffer/scrubber is a background process that progresses through the entire memory, reading data at a rate of about 200 Kbytes per second (this rate changes with memory size and processor model)². If a correctable error is found, the data is written back to the same location and read again (the scrubbing operation). If the correctable error is still present when reread, it is marked as a hard error (e.g., a stuck bit), and an attempt is made by the operating system to remap the data and take the memory location with the hard error out of service. If the correctable error has disappeared when reread, it is logged as a soft error. There are thresholds for soft errors that indicate when a memory module is accumulating too many soft errors and should be replaced. The sniffer/scrubber prevents latent faults in seldom-used areas of memory from accumulating and eventually causing an uncorrectable or undetectable error. It also prevents rapidly deteriorating hardware from accumulating correctable errors that could become uncorrectable or undetectable.

C. ServerNet technology

The ServerNet system area network (SAN) provides the redundant communication network among processors and peripherals in NonStop Servers. ServerNet is a packet-switched, point-to-point, multistage network interconnect technology used for both interprocessor communications and I/O (see [4] and [7] for more details on ServerNet). ServerNet includes extensive features for fault detection and isolation and direct support for alternative communications paths to allow continued operations in the presence of network failures. ServerNet technology includes the following capabilities that help ensure data integrity:

- All command symbols are coded so that single-bit errors create an invalid symbol rather than a different command or data symbol. Data symbols are separated from command symbols by a minimum Hamming distance of 2. Receipt of an invalid symbol triggers a check of the physical connection.
- Each ServerNet packet has a 32-bit cyclic redundancy check (CRC) checksum on a maximum of 512 bytes to detect data or control errors. This is much more robust than simple parity or checksums used in other protocols.
- Routing information and CRC are checked in every ServerNet link. The first ServerNet link that detects a bad packet marks the packet as incorrect and generates an interrupt with the location at which the bad packet was detected. This pinpoints the location of the error.
- The ServerNet routing tables are protected by parity checks. In addition, the ASICs that implement ServerNet technology are self-checking components.
- Link-level flow control is included in the protocol to help alleviate network congestion and route around a device that is “babbling” on the network.
- End-to-end flow control included in the protocol helps prevent devices from injecting more packets into the network than can be handled efficiently, thus avoiding network saturation. This is accomplished by requiring a positive acknowledgment for every packet sent.
- If an acknowledgment is not received for a packet, the protocol automatically checks the end-to-end link and resends the packet if a transient error occurred. This check includes flushing any “stale” packets that could cause spurious errors once the link is again operating normally.

D. Other hardware error checking

After the data leaves the microprocessors, it is subject to soft errors caused by glitches on the buses or defects in other components. In NonStop Servers, data on all the buses is parity protected, and parity errors cause immediate interrupts to trigger error recovery or, if necessary, a processor halt. The microprocessors in I/O controllers are protected by parity checks, packet sequence numbers, and checksums to ensure that the data is not corrupted by bus or component errors. All

² This rate is chosen to be fast enough to sniff all of main memory within 24 hours and slow enough to avoid significant processor overhead.

messages are protected by checksums using a combination of hardware and software.

In a few critical places, such as in the comparison logic in the interface ASIC, totally self-checked logic³ is employed to maintain full error detection coverage.

E. End-to-end checksums

One of the most important NonStop server data integrity features is end-to-end checksums. Because these checksums are added in the microprocessor, they protect against errors in all the buses and components that manage the reading and writing of the data. They also protect against corrupted data values, partial writes, and misplaced or misaligned data. Use of end-to-end checksums also allows the use of less-than-fully-self-checked components in the transfer path – for example, standard SCSI disk controller chips.

The NonStop server disk driver software creates an end-to-end checksum, consisting of a 2-byte checksum appended to a standard 512-byte disk sector before data is written to a disk. For structured data such as SQL files, an additional end-to-end checksum (called a block checksum) encodes data values, physical location of the data, and transaction information. The block checksum is included in the header of a block of data that ranges in size from 512 bytes to 4,096 bytes. When the data is read from disk, the checksum is checked to ensure that the data is correct and that the location from which the data is read is correct. If either the data or location is incorrect, appropriate corrective action, such as reading from the mirror disk, is taken. Separate paths are used to write the data from the processor to each of the mirror disks to ensure that a single path failure does not corrupt the data on both disks.

NonStop Servers now also support standard enterprise storage systems. To conform to the standard 512-byte disk sectors on these disk drives, a new end-to-end checksum has been developed. The new checksum is an enhancement of the former block checksum, and is stored either in the block header or in a separate 512-byte disk sector rather than having 2 bytes appended to the 512-byte disk sector. This checksum is a modified Fletcher checksum (see [9]) with 8 bytes per block, which makes it a stronger checksum than the current checksum, and extends the current block checksum to cover both structured and unstructured data.

Data stored on tape, usually for backup purposes, automatically retains the disk checksums. An additional 2-byte checksum is added to the tape record header to protect against errors such as misaligned data that could occur when transferring the data from disk to tape.

³ Totally self-checked logic is digital logic with embedded redundancy that can detect any single error. See [8] for more discussion and examples.

F. System software error checking

Although it is not possible to prevent all application errors, system vendors must ensure that the system software does not inadvertently change or overwrite correct data. There are many data integrity checks built into the NonStop Operating System. For example, NonStop servers require explicit access validation (source, address, and permissions must all be correct) for all reads and writes to memory. This prevents a processor from incorrectly overwriting a memory location of another processor. The NonStop Operating System also verifies that its data structures and pointers are correct when they are used. For example, it checks pointers on both ends of a link in a doubly linked list to ensure they reference each other, and checks boundary tags when buffers are returned to system pools.

Another protection mechanism is that interrupts are queued rather than automatically given the highest priority. This avoids interrupt flooding caused by a stuck interrupt line on an I/O controller or other hardware that could spoof the protection mechanisms.

The most important function that system software plays in protecting data is proper cleanup following errors or failures. When a computer fails for any reason, in-flight transactions may be partially completed, open files may be corrupted, memory values may be inaccurate, and the database may be left in an incorrect state. In the event of a failure, the NonStop SQL/MP and SQL/MX databases, and NonStop Transaction Management Facility (NonStop TMF) cooperate to ensure that in-flight transactions are aborted and that the database is returned to its last known good state, from which point transactions can be reapplied.

III. NONSTOP ADVANCED ARCHITECTURE

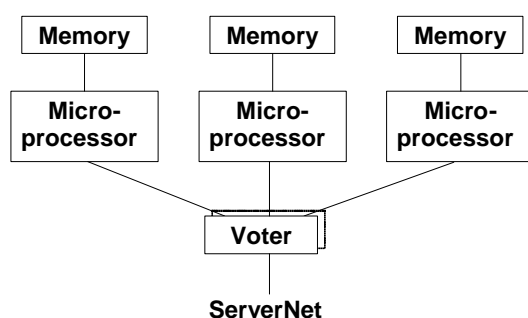
The lockstepped microprocessor architecture described in Section II.A is a variant of a more general “replicate and compare” approach. In the replicate and compare approach, multiple copies of hardware (e.g., the microprocessor and some portions of the supporting chipset for the processor subsystem) are used to perform the same computation, and the results of these computations are compared. For lockstepped microprocessors, there are two replicates of the hardware, and they both execute in a very tightly constrained way – on each cycle, both copies execute the exact same computation, and the chip outputs are compared frequently. In fact, because a single copy of memory is used, every memory operation performed by the two processors must be compared.

The lockstepped microprocessor architecture has the advantages of simplicity and being an almost perfect detection mechanism. Aside from design defects, the odds of both copies of the hardware performing the same erroneous

computation at the same time are vanishingly small. However, there are some disadvantages of this approach. One disadvantage is that the error isolation mechanism is a processor freeze. Although the NonStop server backup processes take over the work of the processes that were running on the frozen processor, the takeover time is on the order of a few seconds, which may cause problems for applications that require near real-time response, and the loss of a processor may cause problems for applications that are not fault-tolerant. In addition, the result of the error recovery is a temporary loss of compute resources, e.g., 25% fewer in a 4 processor system.

A second disadvantage is that lockstepped microprocessors require very strict determinism. Determinism is the property that multiple copies of a design will, when presented with the same input values, produce the same output value(s). The lockstepped microprocessor architecture requires that the same output values be produced at exactly the same time with exactly the same intermediate results after every memory operation, even in the presence of recoverable errors. This requirement is very difficult to meet on current microprocessors with their sophisticated speculation and error recovery techniques, and will be even more difficult to meet on future microprocessors with higher soft-error rates and advanced features such as power management using variable clock frequencies.

The new NonStop server architecture, called the NonStop Advanced Architecture (NSAA), is a loosely-coupled replicate and compare approach that does not require a processor freeze when an error is detected. Figure 3 shows the triple modular redundant (TMR) implementation of NSAA – see [5] for the details of the architecture. Three microprocessors, each with its own memory, are used to perform the same computation. The three outputs are compared by the Voter whenever an I/O output operation is performed rather than whenever a memory operation is performed. If two or three of the three microprocessors agree on the output, the I/O operation is completed. This is a much less frequent comparison than every memory operation, but still provides the same protection against data corruption because all data leaving the microprocessors is compared. Also, if there was a rare memory failure that ECC could not detect, comparing the outputs of the microprocessors would detect the error since each microprocessor has its own memory, an improvement over the single memory of the lockstepped microprocessor architecture.



If one of the microprocessors in the traditional lockstepped microprocessor architecture has a transient error, it is very likely to cause a processor freeze even if the error is recoverable by the microprocessor because the microprocessors will have lost time synchronization. In contrast, if one of the microprocessors in the NSAA TMR configuration has a transient error from which it can recover, the loose synchronization option provides time for the microprocessor to handle the transient error and still sync up with the other two microprocessors. If one of the microprocessors in an NSAA processor has a transient error from which it cannot recover or that causes a data corruption, the output from that microprocessor is voted out, but the processor continues operation since it still has two good copies of the output from the two remaining microprocessors. While the two remaining microprocessors continue operation, the microprocessor that had the unrecoverable transient error can be reloaded. Then, through a process known as reintegration [5], the memory from one of the good microprocessors is copied to the reloaded processor. When the memory copy is complete, the third microprocessor can be re-synchronized with the other two, and the logical processor will once again have a TMR configuration. If the microprocessor had a permanent fault instead of a transient fault, it can be replaced and reintegrated to return to the TMR configuration. The processor continues to operate using the two remaining microprocessors while the third microprocessor is replaced.

After a fault, while a microprocessor is being replaced or reloaded and reintegrated, the processor runs with two microprocessors performing the same computation. This configuration is called dual modular redundant (DMR). The processor is still protected from undetected microprocessor data corruption because the output from two microprocessors is being compared. If one of the microprocessors in the NSAA DMR configuration has a transient error from which it can recover, the loose synchronization provides time for the microprocessor to handle the transient error and still sync up with the other microprocessor. If one of the DMR microprocessors has a transient error that causes an undetected data corruption and miscomparison at the Voter, the processor will halt because the Voter does not know which microprocessor to trust.

If one of the DMR microprocessors detects an unrecoverable transient error and halts or has a permanent error, the processor can operate using a single microprocessor for a small period of time until the other microprocessor is replaced or reloaded and reintegrated. Assuming a one hour time period running with a single microprocessor and using the estimate for undetected microprocessor data corruption from [1] of 3 undetected data corruptions per 1,000 microprocessors per year, the probability of undetected data corruption during this time period is 3×10^{-7} . Note that the probability of this data corruption propagating to output data

is much smaller since the corruption could occur in values that were overwritten before they could be used or errors in the addresses of branches that were not taken [3]. Using NSAA it is possible to avoid even this extremely small risk by halting the processor rather than running for a short time with a single microprocessor, similar to the processor freeze that occurs when a lockstepped microprocessor fails.

The Voter, shown in Figure 3, does not vote its own output, so it is designed with completely self-checked logic to avoid corrupting the voted output data from the NSAA microprocessors. The Voter is designed to either function correctly or self-detect the failure and cease operation. A Voter failure is equivalent to a logical processor failure, but it is possible to avoid the processor failure by configuring redundant Voters as shown by the shadow Voter in Figure 3. However, even if a non-redundant Voter fails, this is equivalent to a lockstepped microprocessor miscompare, and the failure rate of the Voter is much smaller than that of a microprocessor.

IV. SUMMARY

Data integrity features are incorporated throughout NonStop Servers. These features prevent or detect and recover from data corruption anywhere in the processor, memory, I/O, or storage subsystems. HP continues to enhance NonStop Server data integrity. The newest generation of NonStop Servers contains a new replicate and compare technology, called the NonStop Advanced Architecture, which allows processors to continue operating in the presence of transient errors with no loss of data integrity.

REFERENCES

- [1] R. Horst, D. Jewett, and D. Lenoski, "The Risk of Data Corruption in Microprocessor-based Systems," *Proc 23rd International Symposium on Fault-Tolerant Computing*, June 1993.
- [2] J. Borel, "Potential Showstoppers in Technology and EDA Roadmaps", *IEEE Design and Test of Computers*, Vol.16, No. 6, November 2001.
- [3] G. Kanawati, N. Kanawati, and J. Abraham, "FERRARI: A Tool for the Evaluation of System Dependability Properties," *Proc 22nd International Symposium on Fault-Tolerant Computing*, June 1992.
- [4] R. W. Horst, "TNet: A Reliable System Area Network," *IEEE Micro*, Vol. 15, No. 1, February 1995.
- [5] D. Bernick, B. Bruckert, P. Del Vigna, D. Garcia, R. Jardine, J. Klecka, and J. Smullen, "NonStop Advanced Architecture," *Proc International Conference on Dependable Systems and Networks*, June 2005.
- [6] W. Bartlett and L. Spainhower, "Commercial Fault Tolerance: A Tale of Two Systems", *IEEE Trans. On Dependable and Secure Computing*, Vol.1, No. 1, January-March 2004.
- [7] D. Garcia and W. Watson, "ServerNet II", *Proceedings of the Parallel Computer Routing and Communication Workshop*, 1997.
- [8] D. Siewiorek and R. Swarz, "Reliable Computer Systems", 2nd edition, 1992; pp 124-128.
- [9] J. Fletcher, "An Arithmetic Checksum for Serial Transmission", *IEEE Transactions on Communications*, Vol. Com-30, No. 1, January 1982, pp 247-252.