Commercial Fault Tolerance: A Tale of Two Systems

Wendy Bartlett, *Member*, *IEEE Computer Society*, and Lisa Spainhower, *Member*, *IEEE*

Abstract—This paper compares and contrasts the design philosophies and implementations of two computer system families: the IBM S/360 and its evolution to the current zSeries line, and the Tandem (now HP) NonStop® Server. Both systems have a long history; the initial IBM S/360 machines were shipped in 1964, and the Tandem NonStop System was first shipped in 1976. They were aimed at similar markets, what would today be called enterprise-class applications. The requirement for the original S/360 line was for very high availability; the requirement for the NonStop platform was for single fault tolerance against unplanned outages. Since their initial shipments, availability expectations for both platforms have continued to rise and the system designers and developers have been challenged to keep up. There were and still are many similarities in the design philosophies of the two lines, including the use of redundant components and extensive error checking. The primary difference is that the S/360-zSeries focus has been on localized retry and restore to keep processors functioning as long as possible, while the NonStop developers have based systems on a loosely coupled multiprocessor design that supports a "fail-fast" philosophy implemented through a combination of hardware and software, with workload being actively taken over by another resource when one fails.

Index Terms—Computer systems implementation, fault tolerance, high availability.

1 Introduction

Since the 1960s, there have been a number of commercial systems introduced with a specific focus on high availability or fault tolerance. This paper describes the design points and evolution of two of them—IBM's S/360 and its evolution to the zSeries, and Tandem's (now HP's) NonStop systems. Other well-known systems in this space include Stratus and Marathon.

For enterprise customers, definitions of what constitutes "availability" vary and may factor in not only the ability to provide a service but also acceptable response time and whether the results are accurate and consistent. Application availability requirements also vary tremendously, including different applications for the same customer. The need for availability may be "7 x 24 x forever" (e.g., ATM/POS applications and manufacturing floors for three-shift operations), or it may be for 100 percent availability within given timeframes (e.g., stock markets), or it may allow for infrequent, short disruptions (many applications). A framework for describing levels of availability is shown in Fig. 1.

Both system families described in this paper were originally targeted toward the High Availability/Fault Tolerant quadrant and since then have evolved toward Continuous Availability. While the goal has been the same in both cases, their paths toward achieving it have been different. The details appear in later sections, but Table 1 gives a high-level comparison of their approaches.

• W. Bartlett is with Hewlett Packard, 19333 Vallco Parkway MS 4421 Cupertino, CA 95014. E-mail: wendy.bartlett@hp.com.
• L. Spainhower is with IBM, 2455 South Road, Mail Station P314, Poughkeepsie, NY 12601.
E-mail: lisa@us.ibm.com.

Manuscript received 10 June 2004; accepted 20 Aug. 2004.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0084-0604.

2 Initial Target Audiences/Applications for the Systems

Both systems serve commercial business audiences for applications that require very high to continuous availability. From the beginning, IBM mainframes were designed with general-purpose business processing in mind. In the mid-1960s, most real-time business processes were manual and System/360 mainframes were used primarily for background jobs and overnight batch processing. The primary availability goals were to detect failures quickly enough to preserve data integrity and to locate and repair those failures quickly enough to not interfere with getting the tasks done by their deadlines. Over time, mainframes were used for more and more of the real-time work, although it tended to be local with well-defined hours of usage during the day and days during the week. This increased the requirements to lessen the frequency and impact of failures. As businesses became more global and moved to 24 x 7 operations, the demand for continuous operation became common. Limiting planned downtime required for change and upgrade became an increasingly important goal. IBM mainframes responded to each of these evolutionary needs for business computing.

The original Tandem NonStop systems were targeted at a few specific applications that had very high availability requirements, including back-end support for Automated Teller Machines (ATMs) or Point Of Sale (POS) devices and reliable message switching for applications such as credit card authorization. The expectation at the time for ATMs was that the infrastructure might be taken down for service at designated times (typically, 2:00-6:00 a.m. on Sundays) but otherwise was expected to supply 100 percent overall availability, so the emphasis was on fault tolerance rather than continuous operations. Individual machines might fail, but whole sites should not be

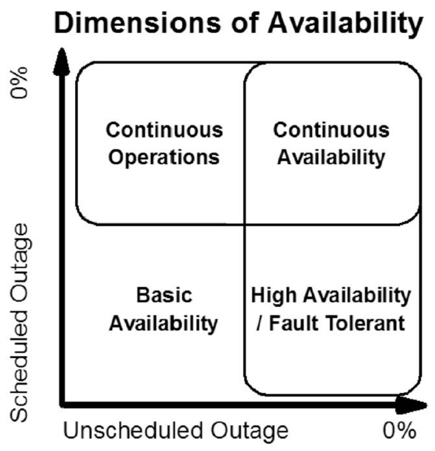

Fig. 1. Availability dimensions.

unavailable. Typically, a single site such as a bank branch would have two ATMs, connected via two different networks to two different systems, or at least to two different processors in the same system. NonStop servers soon moved into other financial markets such as stock exchanges and then into the telecommunications arena—not as switch replacements but for high-availability off-switch functions such as collection of call data records and billing. Use in other markets, including manufacturing shop floors, emergency 911 services, retail, and health care, followed shortly thereafter.

3 Initial Fault Tolerance Philosophies

3.1 zSeries

There was partial, but not complete, overlap in the original design principles for the two systems. Both focused on providing fault tolerance through duplicate components and paths. They also emphasized rapid error detection, as data integrity was and is a key requirement in their markets.

Three of the System/360 founding concepts are specific to dependability:

1. "It must be possible and straightforward to assemble systems with redundant I/O, storages, and CPUs so that the system can operate when modules fail.
2. As CPUs become increasingly reliable, built-in thorough checking against hardware malfunction is imperative for all systems, regardless of application.
3. Since the largest servicing problem is diagnosis of malfunction, built-in hardware fault-locating aids are essential to reduce downtimes. Furthermore, identification of individual malfunctions and of individual invalidities in program syntax would have to be provided" [1].

These concepts have come to be embodied as three prioritized principles on how to react to faults:

1. Ensure system integrity.
2. Provide uninterrupted applications.
3. Repair without disruption.

TABLE 1
System Comparison

	IBM zSeries	HP (Tandem) NonStop
Scope	General purpose server (Supports multiple OSs: zOS , VM, TPF, Linux) Integrated zOS Cluster (Parallel Sysplex)	Integrated system (Server, disk, comm, OS, DB, Txn monitor) Specific classes of target applications Loosely-coupled cluster of systems (ServerNet Clusters)
Fault Model	Two types: transient, permanent	Two types: recoverable, nonrecoverable
FT Operation	Explicit redundancy in support subsystems; implicit redundancy in computational subsystems Parallel Sysplex: no SPOF, distributed applications, data sharing	Multiple hardware components and paths Process pairs, mainly at lower levels of the system Persistent processes plus transactions at application level ServerNet Clusters: Distributed applications
Recovery Techniques	Begin with local retry Recover from transient Spare cache, memory, CPUs Spare or degrade from permanent Parallel Sysplex: Transaction-level restart	Takeover by backup processes in case of processor failure caused by either hardware or software Restart of persistent processes Transaction backout Switch paths
Data Integrity	Redundant CPU logic, Inline checking in I/O subsystem, ECC	Lockstepped microprocessors, ECC, end-to-end disk checksums, CRC

The zSeries, like its System/360-390 standalone mainframe predecessors, is designed to maximize hardware resource utilization during normal, fault-free operation, retry and recover locally when failures are detected, and continue operation without interrupting the ongoing applications. System packaging facilitates online maintenance of hardware components. Firmware and system software are designed to allow updates concurrent with normal operation in most cases. For well-tested software, clusters of zSeries mainframes, Parallel Sysplex, can be configured for no single point of failure in hardware or software. Maintenance, for repair or upgrade, can be performed online as far as the application is concerned.

Ensuring system integrity is paramount, even if application disruption is required in order to preserve it. These design principles apply equally well to the NonStop line, but its implementation philosophy is somewhat different.

3.2 NonStop Server

NonStop systems always have been designed to keep properly designed applications running after a single failure of any kind, to support online repair and reintegration of hardware components, and to preserve data integrity by preventing silent data corruption. For hardware, these goals have been met with a loosely coupled design based on multiple modules with multiple interconnections among them. The modularity helps to delineate fault containment zones and facilitate online repair. Individual modules are designed for ease of maintenance, in order to minimize repair time and reduce opportunities for service-induced failures. Fault detection is built-in to provide "fail-fast" capability to detect errors and isolate the offending hardware from the rest of the system. In contrast to most architectures, the software is an equal partner to the hardware in implementing fault tolerance and preservation of data integrity. The operating system provides its own layer of fault detection and fail-fast response, and supplies the infrastructure that allows applications to survive single processor failures. For a more detailed overview of the NonStop fault tolerance philosophy, see [2].

4 Design Principles

Although they use different system recovery designs, both HP and IBM mainframes rely on near-instantaneous error detection. Whether the aim is to fail fast and rapidly switchover or to transparently retry and recover locally, it is imperative that the scope of any faulty operation be strictly contained.

4.1 zSeries

There has been a steady advancement in the fault tolerant characteristics of IBM mainframes in the 40 years since System/360 was introduced [3]. As previously mentioned, the goal was and is to meet the requirements of general purpose business processing. System/360 was designed with robust error checking and was the first commercial computer to include Error Correction Code (ECC) in memory. Early system requirements were to reduce downtime by improving diagnosis. In the 1960s, computer downtime could be easily masked from customers and clients. Much of the computer workload was overnight batch and businesses depended on manual fall-back methods to continue processing bills, invoices, and service requests during downtime.

In the 1970s, designers improved mainframe fault tolerance with recovery from transient failures and increased the ability to mask hard failures with graceful degradation. For a specialized class of applications, S/370 Extended Recovery Facility was developed to permit one mainframe to monitor another and to assume its workload upon detection of a failure.

In the 1980s and early 1990s, as more and more business applications, particularly online transaction processing (OLTP), started to run 24 x 7, continuous availability grew as a mainframe requirement. To meet the demand, one major change was that online repair capabilities, formerly found only on multiprocessors, were greatly enhanced for all IBM mainframes. Initially, support subsystems—power, cooling, service processor—were either duplicated or designed to be noncritical to the ongoing processing of the computer. Later, those functional elements replicated for higher performance, like CPUs and I/O channels, were designed so that an operational element could take over in a transparent fashion for one that had failed. Increasingly, redundant elements were packaged so that they could be removed, replaced, and restarted without interrupting processing. Another type of online repair, electronic sparing, was also introduced. Mean time to application crash increased to 10 years with these enhancements.

The mid-1990s System/390 semiconductor technology transition from Emitter Coupled Logic (ECL) to Complementary Metal Oxide Semiconductor (CMOS) began a new evolution of fault tolerant design [4]. Fault tolerant enhancements were added with each new generation of CMOS, and today's zSeries has the most extensive detection and recovery in the mainframe history, as well as mean time to hardware-caused application crash in excess of 30 years.

Today, the zSeries memory hierarchy uses store-through (write-through) cache design, spare elements, and Error Correction Codes (ECC) for data redundancy [5]. Pending instruction, results are kept in both a store-through microprocessor cache (L1) and an ECC-protected store buffer and completed results are immediately stored into L2. L1 data is thus always replicated, allowing byte parity to be sufficient within the cache. A transient L1 failure is recovered by CPU instruction retry. For a permanent failure, depending on the scope, a cache line or quarter-cache delete is performed dynamically. A deleted line may be restored with a spare line at the next power on. Shared Level 2 (L2) cache is protected by ECC. Permanent faults in L2 that might result in an uncorrectable data error can be avoided by using a cache delete capability. Faulty locations either in the data array or in the address directory can be dynamically marked as invalid and a spare line can be substituted for the failed one. Each main memory (L3) array chip contributes one bit to an ECC word, so a (72,64) SEC/DED code can protect the system from catastrophic failures due to any single array chip failure. Automatic online repair of faulty DRAMs is done using built-in spare

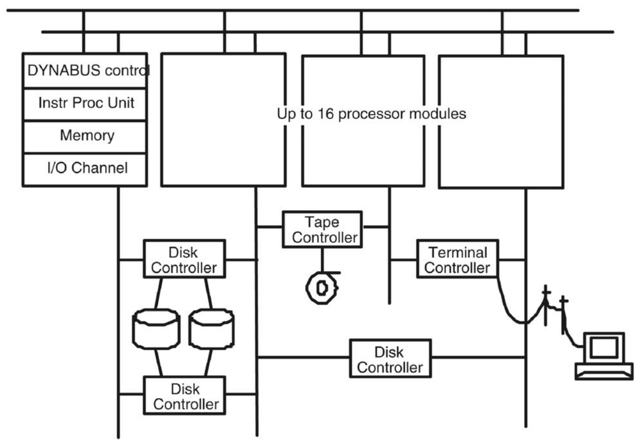

Fig. 2. Original tandem system architecture (1976).

chips. Counts of correctable errors are maintained on a per chip basis. When a threshold is exceeded, the data from the overthreshold chip is dynamically transferred to an error-free spare chip.

The I/O subsystem exploits the redundant paths between all devices and main memory to minimize the scope of any failures [6]. All paths are normally active and are used to enhance performance as well as to provide backup for one another if one should fail. The I/O channel adapters perform direct memory access with robust memory protection on behalf of I/O devices such as disk and tape storage, network communications, and server-to-server cluster (Parallel Sysplex) services. They prevent I/O devices from unauthorized memory read operations and from memory write operations into arbitrary memory locations. I/O channel adapters also provide error isolation by preventing the propagation of interface errors into the system. The bulk of the I/O subsystem hardware is packaged into various circuit cards that provide many different I/O interfaces and most of these cards can be concurrently replaced. Each I/O card position has an interface that controls the soft switch on the card. Before an I/O card is removed, the soft switch is deactivated causing the power on the card to drop. After a card is inserted, the soft switch is activated and the power to the card is turned on. To reduce human errors during concurrent card replacement, a group of LED indicators, one for each card position, is used to positively identify the card to be replaced.

To meet mainframe availability requirements, the power and cooling subsystem is designed with no single points of failure and concurrent repair capability of virtually all components.

4.2 NonStop Server

The Tandem NonStop system shown in Fig. 2, introduced in 1976, consisted of two to 16 independent processors connected by a pair of interprocessor buses collectively referred to as the Dynabus. Each processor had its own memory and ran its own copy of the operating system. Each processor also had an I/O bus. Each dual-ported I/O controller was connected to two processors' I/O buses, and had internal logic that selected which port was currently the primary path. If a processor or its I/O bus were to fail, the controllers whose primary paths were currently configured to use that I/O bus would switch ownership to their backup paths. Controller configuration was flexible enough that the workload of a failing processor could be spread over multiple surviving processors rather than all being taken over by a single processor.

The general design principle was that there be at least two of everything, including power supplies and fans as well as the more obvious processors, controllers, and peripherals. Dual-ported controllers and dual-ported peripherals provided four paths to each device. For disks, the use of host-based RAID-1 mirrored pairs of drives provided eight paths to the data and offered improved data integrity.

There were several special requirements on the hardware design. The first was that the Dynabus interfaces had to protect against a single processor failing in a way that disabled both buses or induced errors on both buses when power was removed from the processor module. The second was that power be managed in a way that allowed a power supply failure to be tolerated. The third was that hot insertion of boards into powered slots be supported, which enabled both online repair and online expansion. Battery backup, which preserved memory contents and supported power-fail restart, was standard [7]. Online repair was designed in for all hardware components, including power supplies and fans.

Beyond those requirements, there was a general need for extensive error checking using appropriate techniques such as parity, parity prediction, ECC, self-checking, and checksums. Controllers had their own fail-fast requirements to not corrupt data and, if possible, make available diagnostic information on the failure. End-to-end checksums were generated at the host and written to disk along with the data, with the checksum being recalculated and compared when the data was read back. The hardware was, in a sense, fault-intolerant—it was designed to fail quickly and cleanly at the first sign of a nonrecoverable error.

The system was architected to maximize useful, active modules and minimize the existence of "redundant," inactive modules, both for cost reasons and to prevent the accumulation of undetected, latent errors. This maximization was primarily the responsibility of the operating system software. As examples, processes were run in all processors—there were no "standby" processors; the message system used both paths of the Dynabus; and the disk process issued reads against both halves of the mirror (which also improved read performance).

Both the hardware and the software were optimized to support the message-based operating system design, including microcode-level support for sending and receiving messages. The choice of design yielded a system that is both highly distributed and fault tolerant. In addition to being easy to repair, the system could be easily expanded online.

Software fault tolerance was built into the operating system from the beginning [8], [9]. The system was structured as a set of processes communicating via interprocess messages, influenced by Dijkstra's "THE" system [10] and Brinch Hansen's implementation of a message-based operating system nucleus [11]. Each processor was preconfigured with a small set of daemon-type processes that were responsible for specific tasks within that processor such as process control and memory management, along with a supporting set of library routines, interrupt handlers, etc. The key software abstraction, process pairs, allowed two communicating processes running in different processors to collectively represent a named resource such as a disk volume, communications line, or group of terminals, with one process functioning as the primary at any point in time and checkpointing necessary state to its backup. If a primary process or its processor were to fail, the backup process would take over and the operating system would transparently redirect requests to it without the application seeing an error. Operating system infrastructure was put in place to reliably maintain the named resource table that provides the redirection transparency.

Process pairs are an effective design for recovering from transient problems ("Heisenbugs" [12]) such as race conditions, where the identical conditions are unlikely to be in effect when the backup process takes over. They are vulnerable to deterministic defects ("Bohrbugs"), but those are relatively rare in a well-tested system. See [13] for an empirical study on this topic.

The initial focus of the system was to provide tolerance against single faults. Over time, the direction has been to add more support for continuous operations in order to serve the needs of customers who have been under increasing pressure to provide true continuous availability.

5 Advanced Design

5.1 Chip Technology

The designers of both system lines have had to continuously redesign their error checking logic in order to deal with both evolving technologies and the increasing rate of transient errors that is the inevitable side effect of the decreasing silicon feature size associated with newer, faster microprocessors and other components [14].

5.1.1 *zSeries*

From System/360 through System/390 ECL mainframes, increasingly complete and effective inline error checking was built into the circuitry. Within the CPU, parity and ECC were used extensively for all arrays, data paths, and control paths. Control logic, such as state machines and ALUs, used a variety of mechanisms such as parity prediction or illegal state detection.

For CMOS, a totally new design approach was needed [15]. The requirement for dynamic recovery, coupled with other design constraints (e.g., power, performance, second-level packaging) pointed to a highly checked single-chip microprocessor. However, inline checking of control and arithmetic logic was prohibited by performance penalties. As shown in Fig. 3, the decision was made to duplicate I-unit and E-unit and compare outputs. With a carefully laid out floorplan, there are no cross-chip performance-limiting paths. The cycle time is the same as if the same design were implemented on a smaller chip with only one unchecked I-unit and E-unit. The compare and detect cycle is

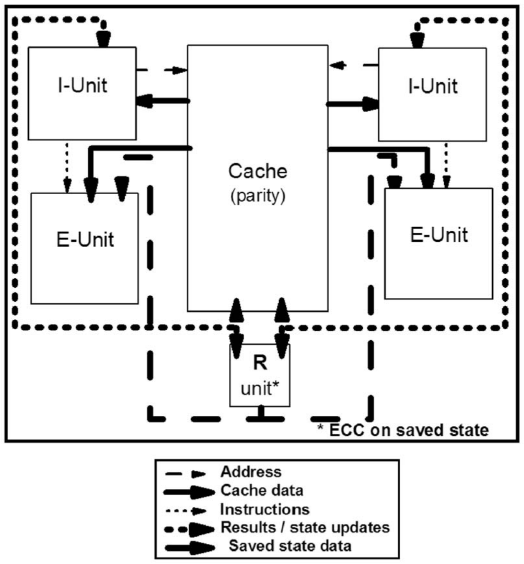

Fig. 3. Logical layout of the zSeries microprocessor chip.

completely overlapped in the instruction execution pipeline, so cycles per instruction (CPI) is not increased.

Most of the microprocessor area is devoted to the cache where data is primarily moved around unchanged. Updates to memory are maintained in an ECC-protected store buffer during instruction execution and transferred to an ECC-protected L2 cache when instruction execution completes. Thus, for the on-chip cache, low-overhead parity checking is sufficient.

A key element of the microprocessor's R-unit is an ECC-protected register file, the checkpoint array. The checkpoint array keeps track of the entire state of the CPU including register contents and instruction addresses. Completed instructions are known to be error-free and the checkpoint array and the store buffer accurately describe the state of the machine at successful instruction execution.

The R-unit permits recovery to be completely controlled by hardware and identical for all instructions. The R-unit manages instruction retry when an error is detected as follows: Except for the R-unit itself, the CPU is reset and store buffer contents are sent to L2. The state of the CPU is returned to what it was at the last instruction completion. Serialized instruction processing starts up, retrying the instruction that caused the error. If it is successful, the failure was transient and the CPU resumes pipelined instruction processing. If not successful, then the error is permanent and the CPU will be stopped. For permanent failures, the service element moves the checkpoint to another, typically spare, microprocessor and instruction execution resumes with no apparent interruption.

Total circuit overhead for cache parity, Register Unit ECC and duplicate I-unit and E-unit is about 30 percent. Design, verification, and test are greatly simplified compared to inline checking.

5.1.2 *NonStop Server*

The original NonStop system was designed in 1974-1975. The hardware logic was based on 7400 series TTL integrated circuits, with the most complex being the 4-bit ALU. As gate density within an IC package increased and gate arrays became available, the designers were able to increase the self-checking logic. Because processor design was under Tandem's control, it was possible to build in as much self-checking logic as the developers deemed necessary, using techniques such as parity on buses and branch prediction. This approach continued to be used for more than 10 years. In the late 1980s, Tandem moved to ECL logic for its Cyclone processor [16]. By then, it was becoming evident that the economics of building its own processors from the ground up were no longer attractive, and the Cyclone was the last internally designed processor in the line.

In 1991, Tandem moved to building its processors around commercial microprocessors, choosing the MIPS 3000 for its initial CMOS system. Because the amount of error checking within the microprocessor was not within Tandem's control and was considered to be inadequate for its target markets, the design changed to a scheme based on tight lockstepping of a pair of MIPS chips. That approach has continued to be used up until now, and a variant is being developed for future Intel Itanium®-based platforms. The new approach enables use of a looser, but still effective, level of synchronization between microprocessors. This design allows individual microprocessors to temporarily diverge in their instruction streams, e.g., to perform transient error recovery, without their associated logical processors being declared down. It also supports a triplex configuration for customers who wish to retain processor-level hardware single fault tolerance and data protection during processor repair actions.

5.2 Other Changes

5.2.1 zSeries

The next step in the evolution of availability beyond a single operating system and mainframe, whether uniprocessor or SMP, was the tightly coupled multiprocessor. Such mainframes were comprised of two mainframes, which could be run as one operating system image or as two isolated images. When run as two, there was no common hardware, thus no single point of failure. A very common operational mode was to run a tightly coupled multiprocessor as one operating system image and, when needed for maintenance or when a failure occurred, degrade to a single mainframe. This is in keeping with the general design philosophy of delivering all available capacity during normal operation and only using capacity for availability in unusual conditions.

Logical partitioning, (LPAR), further enhanced the ability to design a single mainframe for higher availability. LPAR allows multiple operating system instances to concurrently reside on one mainframe. Each is isolated from the other, test and production can be run at the same time, and new releases run with old. With the advent of LPAR came Extended Recovery Facility (XRF), which allowed a backup partition to be created within the same or a different mainframe. The partition needed very few resources during normal operation as it monitored the primary operating partition. It did require a high priority such that, upon failure, it could take over the necessary resources to keep the primary workload running on the backup. XRF in one mainframe had no software single point of failure (SPOF) and running across two mainframes, no software or hardware SPOF.

A major new design paradigm came about with the introduction of Parallel Sysplex, a data-sharing cluster design [17]. Multiple mainframes (up to 32) can be connected together via a coupling facility, or for the most robust installations, two coupling facilities. All mainframes, or at the least, all mainframes in the Sysplex sharing data, have equal access to shared disks as well as to the coupling facilities which manage the sharing. The basic hardware structure of the Sysplex is therefore to have no SPOF in the hardware.

To get the same results in the software, it is necessary to spread identical workloads across each (or a subset) of the mainframes in the Sysplex, shown in Fig. 4. At first, the notion of putting all critical software on each of the mainframes is somewhat counter intuitive if a primary guideline has always been to isolate workloads from one another on different hardware. With Parallel Sysplex, by contrast, the ideal high availability configuration is one in which each mainframe is a clone of the others, from a software perspective. Transactions are distributed to the mainframes via a Workload Manager (WLM) that determines which mainframe is least busy for that particular type of work at the moment. In the event of a failure anywhere within the Sysplex, there is no formal switch-over—merely a cessation of assigning work to a no-longer-available application, operating system, or mainframe. The design makes it unnecessary to preassign backups and, during normal operation, all available capacity can be used by the ongoing workload.

For those customers without the stringent availability needs of Parallel Sysplex but for whom the software reliability of a standalone mainframe is insufficient, it is possible to use LPAR to construct a partition (or partitions) that will serve as coupling facilities and to build a "Sysplex in a box." Given the highly robust mainframe hardware, with a mean time to unplanned customer application loss in excess of 30 years, this middle ground is adequate for many applications that are effectively mission-critical.

5.2.2 NonStop Server

As the engineers gained experience after the initial shipment, they adjusted algorithms to reflect what was seen in the field. Data integrity was and is the key system requirement, and several refinements were made to strengthen end-to-end data checks. Changes included adding end-to-end checksums on messages and sequence numbers to individual packets (originally, messages were simply checksummed at the packet level), and including the disk unit number in sector checksums to enable detection of the right data being written to the wrong disk. A less obviously availability-related feature was the addition of instrumentation to help debug performance problems; this was important because a poorly performing application effectively is unavailable.

The primary hardware change for NonStop servers (aside from the move to CMOS technology) has been to

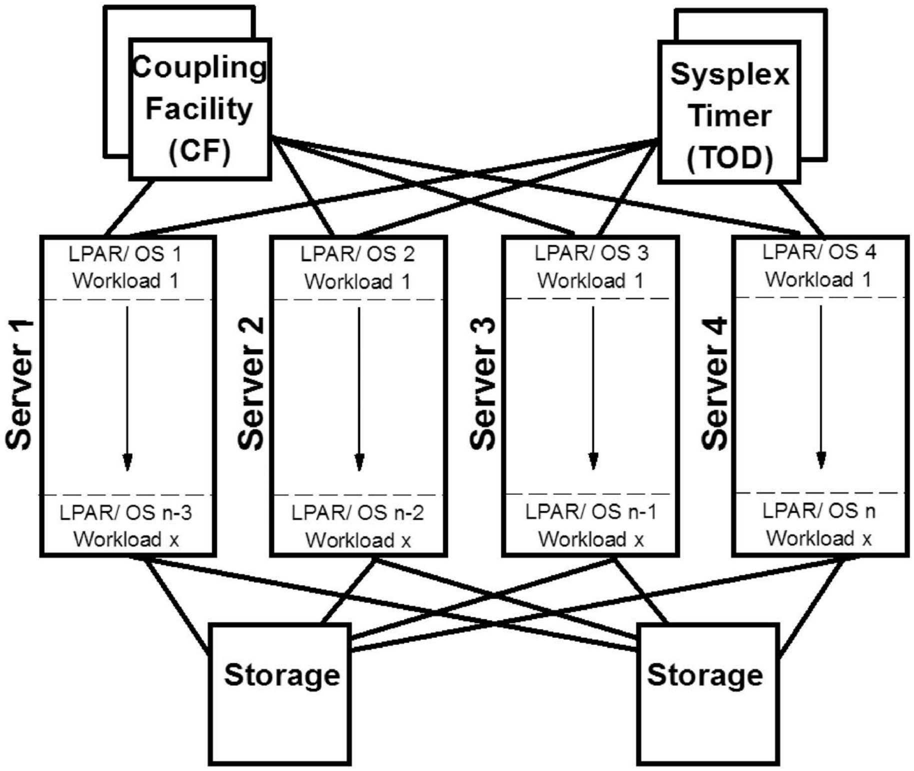

Fig. 4. zSeries parallel sysplex architecture.

rearchitect the system interconnect. As was shown earlier, the original NonStop architecture used a combination of I/O channels and interprocessor buses for its interconnect. Both were designed in-house, once again offering the opportunity to build in an appropriate level of checking. However, the Dynabus had to be greatly overdesigned in terms of capacity in order to support up to 16 processors and allow for future online upgrades to faster processors. In 1997, the NonStop S-series systems were introduced with a completely redesigned interconnect architecture. This architecture, shown in Fig. 5, takes a more network-oriented approach, which by its nature includes the capability to scale up bandwidth in all dimensions as the system grows.

The heart of the design is ServerNet, which carries both I/O and interprocessor traffic. The new interconnect had requirements to improve system performance, both through higher bandwidth and through offloading of part of the software communication stack, while meeting all of the expected fault tolerance requirements [18]. At the time, there were no commercially available interconnects that met all of the requirements, so ServerNet was designed internally. The original design was a 6-port router with 300MB/sec aggregate bandwidth. Wormhole routing was and is utilized to minimize both latency and the need for buffering at the router. DMA transfers, with appropriate protection, remove the need for host-level execution at the remote node. Single-bit command signal errors map to invalid commands. Each packet is protected with a cyclic redundancy check (CRC), and each router recalculates the CRC and appends either a “This Packet Good” or “This Packet Bad” symbol to the packet, making it easy to isolate link failures. ServerNet has subsequently been redesigned to provide even better bandwidth and diagnosability [19].

NonStop servers are inherently clustered due to their loosely coupled design. In the late 1980s, that design was

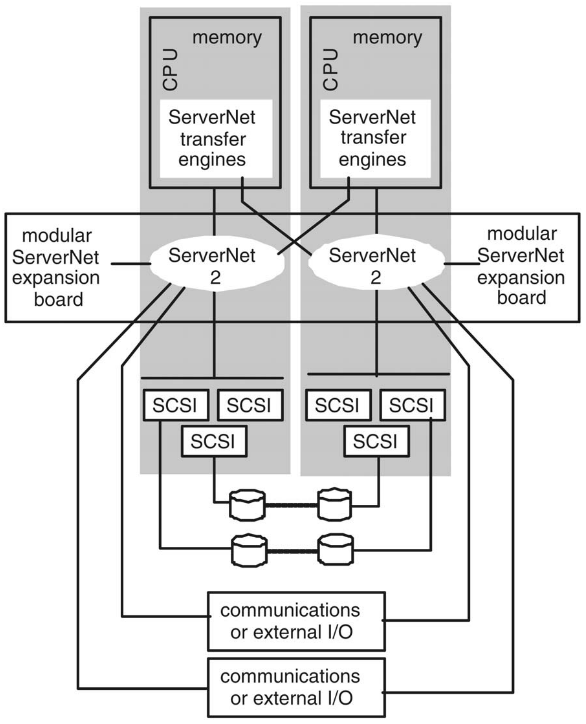
memory
CPU
ServerNet transfer engines
memory
CPU
ServerNet transfer engines
modular ServerNet expansion board
ServerNet 2
ServerNet 2
modular ServerNet expansion board
SCSI
SCSI
SCSI
SCSI
SCSI
SCSI
communications or external I/O
communications or external I/O

Fig. 5. Tandem NonStop Server S-series architecture (two processors shown).

extended to build the EXPAND networking product supporting homogeneous "clusters of clusters," allowing multiple interconnected systems to function in many ways as a single system image [20]. The original interconnect was via X.25 and other WAN protocols, with support added later for SNA, LAN, and IP. The process-and-message architecture provided a base that could be very straightforwardly extended to support multiple systems. Customers also began to require the ability to group together multiple systems in a data-center environment using high-speed, low-latency connections. A fiber-optic interconnect, FOX, was developed in 1983 to connect systems at subkilometer distances, and EXPAND was optimized to take advantage of the high reliability and speed of the new interconnect. For the S series, the data-center interconnect is ServerNet, which can cover distances of up to 15 KM.

The original software design assumed that programmers would work "close to the metal" and use a programming language, TAL, whose functionality was similar to C, with some customers possibly even supplying their own file systems. However, COBOL support also was available from almost the beginning. It soon became evident that most customers wanted their programmers to be able to focus on the business problem at hand without having to do any special coding to achieve fault tolerance and scalability. The PATHWAY transaction processing monitor [21] was developed to meet this need by handling the distributed and fault-tolerant aspects of the work on the programmer's behalf, with a high degree of transparency as far as the application is concerned. The programmer uses normal business logic and indicates transaction boundaries, and the system takes care of aborting incomplete transactions, restarting server instances, and redirecting requests when an application server process fails.

The other need that soon became evident was for data consistency as well as data integrity—that is to say, a database that is transactionally consistent. The Transaction Management Facility (TMF) was developed in the early 1980s to coordinate data accesses and updates in a manner that guaranteed the ACID properties of atomicity, consistency, isolation, and durability. In order to take advantage of TMF, an application programmer need only determine the appropriate locations in a program to begin, commit, and, if necessary, abort transactions. The Remote Database Facility (RDF) allows a TMF-protected database to be replicated at one or multiple additional sites at any geographic distance.

6 Design Trade Offs

Trade offs in computing system design are generally made among price, performance, and reliability. Most commercial servers are optimized for price, performance, or price/performance, and reliability enhancements which detract from the optimization point are closely scrutinized. Over time, reliability features once found only on servers optimized for reliability have become standard on virtually all servers. For instance, memory ECC and redundant power are now common. Reasons for the evolution include changing economics (e.g., when spare parts and warranty costs are included in the equation, certain reliability features may save money), technology failure rates which are unacceptably high for a broad range of servers, and increased customer expectation for better system availability at almost every price point.

For Tandem and zSeries systems, from the beginning, the target markets had extremely high data integrity requirements and the systems had to be designed accordingly. As described above, both systems were constructed using proprietary processors, which allowed for extensive self checking to be incorporated. Microprocessors have a component to fetch and decode instructions (I-unit), one or more instruction execution elements (E-unit), and a cache. Modern microprocessors usually include parity for cache data and data paths. Checking the control, arithmetic, and logical functions in the I-unit and E-unit is considered to be difficult, time-consuming, and to introduce performance penalties. As a result, other than zSeries, today's microprocessors leave these components unchecked. The Tandem design also checked those components in early systems, but since 1981 the approach has been to use off-chip logic to create perfectly checked microprocessor pairs.

In the industry, accepted wisdom is that single chip microprocessors are sufficiently reliable, that failures are rare and that other mechanisms—timeouts and software detection, for instance—will discover the few problems that do occur. This philosophy depends on continual reliability improvements; in reality, predicting chip technology failure mechanisms is problematic. Future CMOS technologies are expected to be less reliable; latches and dynamic logic will become more susceptible to alpha particles and other cosmic radiation [14]. Latent manufacturing defects and design errors that manifest themselves as transient problems also are sources of customer-visible errors.

Even today there is an exposure to malfunctions in the hardware resulting in incorrect data being written to memory. The model proposed by Horst et al. [22] projects a pessimistic bound of one in 75 unchecked microprocessors causing a data corruption error each year. Horst et al.'s prediction is based on an optimistic evaluation of software detection. It is true that, in some cases, timeouts and software detection will prevent data integrity problems; however, software does not reliably fail fast when errors are injected, resulting in corrupted data [23]. Even when the detection is successful, a system with no recovery will usually hang, causing application downtime, unacceptable for systems designed for high availability. zSeries and NonStop servers are the most well-known commercial systems designed to prevent data corruption by including extensive error checking in all microprocessor functional elements—combinatorial logic as well as arrays.

There are subtle advantages of building in a high level of integrity checking and retry/recovery logic to handle errors. For example, the resulting system is relatively resilient to design flaws. It may be possible to ship a system with one or more remaining occasional transient design errors if the occurrence rate is relatively low and the cost of recovery is not visible at the application level. Another advantage is that test and bringup time can be significantly reduced. The ability to fail-fast often makes it much easier to determine failure sources and uncover design and other problems.

Another important design trade off is "build versus buy." Particularly for subsystems which are compliant with industry standards, like I/O subsystems, there are notable advantages in using high volume components. Commodity components are used wherever they provide a cost or time-to-market advantage, if the developers are able to provide the expected level of data integrity. For instance, most I/O adapters do not meet zSeries or NonStop server needs for memory protection; that is, inadequate checking is provided to assure that data does not get written to or read from an incorrect location. Some level of customization, through techniques such as lockstepping and end-to-end checksums, is added to incorporate commodity components into the two systems described herein.

6.1 zSeries

zSeries runs multiple operating systems—VM, Linux, and TPF as well as its flagship high-availability operating system, z/OS, the successor of MVS. Similarly to the hardware design principles, z/OS recognizes the inevitability of errors, resulting in a comprehensive approach of error isolation, identification, and recovery. z/OS provides many software reliability and availability features, quite a few of which can also be used by applications which run on z/OS.

Much of the z/OS kernel operating system is dedicated to providing advanced reliability, availability, and serviceability capabilities. All z/OS platform code is written to a set of guidelines to assure that errors can be detected, isolated, and documented. For example, "all code must be protected by a recovery routine including the code of recovery routines themselves; all control areas and queues must be verified before continuing; recovery and retry must be attempted if there is hope of success; all failures which cannot be transparently recovered must be isolated to the smallest possible unit, e.g., the current request, a single task (thread), or single address space (process); diagnostic data must be provided which, as an objective, will allow the problem to be identified and fixed after a single occurrence. The diagnostic data is provided even when retry is attempted and successful" [24].

Approximately 30 percent of the z/OS kernel operating system code is written to adhere to these guidelines, in addition to component code supporting the guidelines or code specifically designed to handle processor, storage, or I/O error recovery. Overall, about half of the z/OS kernel code is devoted to error handling. In addition to adhering to the guidelines, z/OS system software is committed to ensuring system integrity. Data and system functions are protected from unauthorized access, whether accidental or deliberate.

6.2 NonStop Server

Percentages of recovery code are not available for the NonStop server kernel, but the intent is similar to that of z/OS: to identify and contain the scope of faults in order to minimize application-level impact while preserving system and data integrity. Software consistency checking was built into the Tandem operating system from the beginning. As an example, the process ready list is kept as a doubly linked list, with the immediate links checked on every insertion and deletion. Data being written to disk is checkpointed to the backup disk process, but data being written to comm devices is not checkpointed as the performance penalty has been deemed to be too high and comm protocols have built-in recovery mechanisms. Data being written to disk includes host-generated sector checksums, which introduces a small I/O performance penalty and, until recently, required the drives to be specially formatted using 514-byte sectors.

7 The People Factor

In 1985, Gray [12] analyzed a set of Tandem failure data and published his results. His conclusion was that the key to high availability is tolerating operations and software faults—even then, fault-tolerant systems were very effective at masking hardware faults. Almost 20 years later, his conclusion still is valid. Recent evaluation of three very large Internet sites finds operator error to be the number one contributor to system downtime [25]. If anything, increased system complexity has offered even greater opportunities for fatal operational mistakes.

The best tool for reducing operator errors is automation, both to prevent operator errors and to better handle fault analysis and recovery. Automation can help the latter in three ways: It can analyze a long series of events more quickly and thoroughly than a human can, it can speed up recovery by initiating corrective action, and it can reduce the chance of an already perilous situation turning into a full-blown outage due to "fat-finger" mistakes. In cases where an automated event analysis subsystem cannot pinpoint the faulty component, it can at least narrow the options that the human needs to consider. Ideally, a fault-tolerant system also is a self-tending system, monitoring its own health, making corrections as necessary, and alerting the outside world where appropriate.

7.1 zSeries

In 2001, IBM introduced its Autonomic Computing initiative, a longterm strategy to invent, design and deploy computing systems, which operate to achieve business-level policies [26]. The realization of Autonomic Computing will be systems that demonstrate self-CHOP capabilities, that is, self-configuring, healing, optimizing, and protecting. Certain characteristics of zSeries servers exemplify what is meant by each of these.

Dynamic CPU Sparing, a process by which a failed CPU can be transparently replaced by a spare, was earlier described and is a good example of self-healing. Spare CPUs can also be brought online to meet increased customer resource requirements, a process known as Capacity Upgrade on Demand, and an example of self-configuring. LPAR, earlier mentioned as a mechanism for running multiple concurrent operating systems on one mainframe and keeping them isolated from one another, is also a self-optimizing feature. LPAR will maximize resource utilization in the server based on user established priorities. If a high priority partition is not using its allowed resources, LPAR will make them available to lower priority partitions. It will attempt to keep the mainframe as busy as possible all the time. A partition can also be capped, meaning that, even if all other operating system partitions are in a wait state, the capped partition can have only a certain amount of resources. This is useful for service providers who are "renting" capacity to others, but can also serve as a self-protecting mechanism to prevent denial of service attacks from taking over the entire mainframe.

7.2 NonStop Server

Automation is easier to construct if there is a unified logging subsystem to provide access to an integrated stream of information about what is going on in the system. In the mid 1980s, Tandem built a Distributed System Management subsystem (DSM) [27] to meet this need. DSM provides a consolidated log and logging infrastructure that are available to partners and end user applications as well as to the system provider, and includes a call-home capability. DSM events are tokenized, and can be viewed in filtered form to meet specific needs for programmatic response, Web viewing, PDA viewing, hardware incident analyzers, and so forth. However, the logging infrastructure alone is not sufficient; subsystems need to be intelligently instrumented to use it, and incident analysis software needs to be kept updated as new patterns are seen in the field. NonStop servers have long had a high degree of self-tending built into the system, both within the operating system and individual subsystems and, especially for hardware, in the form of incident analyzers.

8 Conclusion

Fault tolerance and rapid fault detection for both hardware and software are key building blocks for constructing continuously available applications, but are not in themselves sufficient to accomplish the task. Similarly, built-in error recovery is important for handling the increasing rate of transient hardware and software errors. But, for builders of highly available systems, much of the current challenge lies in addressing the "second 90 percent" of the problem—the network, the environment, cyber attacks and, most importantly, the people and operational procedures responsible for keeping applications and systems running on an ongoing basis. The move toward self-tending systems by the producers of today's highly available commercial systems is a recognition of that need, and a trend that the authors expect to see continue.

Acknowledgments

Wendy Bartlett would like to thank Joel Bartlett, Bob Taylor, and Kate Wilson. Lisa Spainhower would like to thank Bob Abrams.

References

[1] G.M. Amdahl, G.A. Blaauw, and F.B. Brooks, "Architecture of the IBM System/360," *IBM J. Research and Development,* vol. 8, pp. 87-101, 1964.

[2] W.B. Bartlett and B. Ball, "Tandem's Approach to Fault Tolerance," *Tandem Systems Rev.,* vol. 8, pp. 84-95, Feb. 1988.

[3] L. Spainhower, "System/360 to zSeries: Dependability in IBM Mainframes," *Dependable Computing Systems: Paradigms, Performance Issues and Applications,* 2004.

[4] L. Spainhower and T.A. Gregg, "IBM S/390 Parallel Enterprise Server G5 Fault Tolerance: A Historical Perspective," *IBM J. Research and Development,* vol. 43, nos. 5/6, pp. 863-874, 1999.

[5] P.R. Turgeon, P. Mak, M.A. Blake, M.F. Fee, C.B. Ford III, P.J. Meaney, R. Seigler, and W.W. Shen, "The G5/G6 Binodal Cache," *IBM J. Research and Development,* vol. 43, nos. 5/6, pp. 661-670, 1999.

[6] T.A. Gregg, "S/390 CMOS Server I/O: The Continuing Evolution," *IBM J. Research and Development,* vol. 41, nos. 4/5, pp. 449-462, 1997.

[7] J.A. Katzman, "System Architecture for NonStop Computing," *Proc. IEEE CS Int'l Conf. Technologies for the Information Superhighway,* pp. 77-80, 1977.

[8] J.F. Bartlett, "A 'NonStop' Operating System," *Proc. Hawaii Int'l Conf. System Sciences,* pp. 103-119, 1978.

[9] J.F. Bartlett, "A NonStop Kernel," *Proc. Eighth SIGOPS European Workshop,* pp. 22-29, 1981.

[10] E.W. Dijkstra, "The Structure of the 'THE' Multiprogramming System," *Comm. ACM,* vol. 11, pp 341-346, 1968.

[11] P.B. Hansen, "The Nucleus of a Multi-Programming System," *Comm. ACM,* vol. 13, pp. 238-241, Apr. 1970.

[12] J. Gray, "Why Do Computers Stop and What Can Be Done About It?" Technical Report TR85.7, Tandem Computers, Cupertino, Calif., June 1985.

[13] I. Lee, "Software Dependability in the Tandem Guardian System," *IEEE Trans. Software Eng.,* vol. 8, pp 455-467, May 1995.

[14] C. Constantinescu, "Impact of Deep Submicron Technology on Dependability of VLSI Circuits," *Proc. 2002 Int'l Conf. Dependable Systems and Networks,* pp. 205-209, 2002.

[15] C.F. Webb and J.S. Liptay, "A High-Frequency Custom CMOS S/390 Microprocessor," *IBM J. Research and Development,* vol. 41, nos. 4/5, pp. 463-473, 1997.

[16] D. Siewiorek and R. Swarz, *Reliable Computer Systems: Design and Evaluation.* Burlington, Mass.: Digital Press, pp. 586-648, 1992.

[17] J.M. Nick, B.B. Moore, J.Y. Chung, and N.S. Bowen, "S/390 Cluster Technology: Parallel Sysplex," *IBM Systems J.,* vol. 36, no. 2, pp. 172-201, 1997.

[18] B. Horst, "TNet: A Reliable System Area Network," *IEEE Micro,* vol. 15, no. 1, pp. 37-45, Feb. 1994.

[19] D. Garcia and W. Watson, "ServerNet II," *Proc. Parallel Computing, Routing, and Comm. Workshop,* 1997.

[20] Tandem Computers, *EXPAND Reference Manual.* Cupertino, Calif., 1986.

[21] Tandem Computers, *Introduction to Pathway.* Cupertino, Calif., 1985.

[22] R. Horst, D. Jewett, and D. Lenoski, "The Risk of Data Corruption in Microprocessor-Based Systems," *Proc. Fault-Tolerant Computing Symp.,* pp. 576-585, 1993.

[23] S. Chandra and P.M. Chen, "How Fail-Stop Are Faulty Programs?" *Proc. 28th Int'l Symp. Fault Tolerant Computing,* pp. 240-249, 1998.

[24] S. Turner, C. Henry, G. Horvath, and J. Kibble, "Selecting a Server—Value of S/390," *IBM Redbook SG24-4812-01,* 1999.

[25] D.A. Patterson and D. Oppenheimer, "Architecture and Dependability of Large-Scale Internet Services," *IEEE Internet Computing,* pp. 41-49, Sept.-Oct. 2002.

[26] J.O. Kephart and D.M. Chess, "The Vision of Autonomic Computing," *Computer,* vol. 36, no. 1, pp. 41-50, Jan. 2003.

[27] P. Homan, B. Malizia, and E. Reisner, "Overview of DSM," *Tandem Systems Rev.,* vol. 4, no. 3, Oct. 1988.

Wendy Bartlett received the BS degree in statistics and the MS degree in computer science: computer engineering, both from Stanford University. She is a distinguished technologist in Hewlett Packard's NonStop Enterprise Server Division. She was employed at Four Phase from 1976-1978, and has been employed by Tandem Computers, Compaq, and HP since then. She currently heads the indestructible scalable computing initiative for the NonStop server. She is a member of the IEEE Computer Society.

Lisa Spainhower is a graduate of the University of Michigan. She is a distinguished engineer in the Systems Technology and Architecture organization in IBM's Systems and Technology Group in Poughkeepsie, New York. She is a member of the IBM Academy of Technology, the IEEE, and the IEEE Computer Society.