Tandem TR 86.2

FAULT TOLERANCE IN TANDEM COMPUTER SYSTEMS

Joel Bartlett

Jim Gray

Bob Horst

March 1986

ABSTRACT

Tandem builds single-fault-tolerant computer systems. At the hardware level, the system is designed as a loosely coupled multi-processor with fail-fast modules connected via dual paths. It is designed for online diagnosis and maintenance. A range of CPUs may be inter-connected via a hierarchical fault-tolerant local network. A variety of peripherals needed for online transaction processing are attached via dual ported controllers. A novel disc subsystem allows a choice between low cost-per-megabyte and low cost-per-access. System software provides processes and messages as the basic structuring mechanism. Processes provide software modularity and fault isolation. Process pairs tolerate hardware and transient software failures. Applications are structured as requesting processes making remote procedure calls to server processes. Process server classes utilize multi-processors. The resulting process abstractions provide a distributed system which can utilize thousands of processors. High-level networking protocols such as SNA, OSI, and a proprietary network are built atop this base. A relational database provides distributed data and distributed transactions. An application generator allows users to develop fault-tolerant applications as though the system were a conventional computer. The resulting system has price/performance competitive with conventional systems.

TABLE OF CONTENTS

INTRODUCT ION

Conventional well-managed transaction processing systems fail about once every two weeks for about an hour [Mourad], [Burman]. This translates to 99.6% availability. These systems tolerate some faults, but fail in case of a serious hardware, software or operations error.

When the sources of faults are examined in detail, a surprising picture emerges: Faults come from hardware, software, operations, maintenance and environment in about equal measure. Hardware may go for two months without giving problems, and software may be equally reliable. The result is a one-month MTBF. When one adds in operator errors, errors during maintenance, and power failures, the MTBF sinks below two weeks.

By contrast, it is possible to design systems which are single-fault-tolerant -- parts of the system may fail but the rest of the system tolerates the failures and continues delivering service. This paper reports on the structure and success of such a system -- the Tandem NonStop system. It has MTBF measured in years -- more than two orders of magnitude better than conventional designs.

DESIGN PRINCIPLES FOR FAULT TOLERANT SYSTEMS

The key design principles of Tandem systems are:

- Modularity: Both hardware and software are decomposed into fine- granularity modules which are units of service, failure, diagnosis, repair and growth.

- Fail-Fast: Each module is self-checking. When it detects a fault, it stops.

- Single Fault Tolerance: When a hardware or software module fails, its function is immediately taken over by another module -- giving a mean time to repair measured in milliseconds. For processors or processes this means a second processor or process exists. For storage modules, it means the storage and paths to it are duplexed.

- On Line Maintenance: Hardware and software can be diagnosed and repaired while the rest of the system continues to deliver service. When the hardware, programs or data are repaired, they are reintegrated without interrupting service.

- Simplified User Interfaces: Complex programming and operations interfaces can be a major source of system failures. Every attempt is made to simplify or automate interfaces to the system.

This paper presents Tandem systems viewed from this perspective.

HARDWARE

Principles

Hardware fault tolerance requires multiple modules in order to tolerate module failures. From a fault tolerance standpoint, two modules of a certain type are generally sufficient since the probability of a second independent failure during the repair interval of the first is extremely low. For instance, if a processor has a mean time between failures of 10,000 Hours (about a year) and a repair time of 4 hours, the MTBF of a dual-path system increases to about 10 million hours (about 1000 years). Added gains in reliability by adding more than two processors are minimal due to the much higher probability of software or system operations related system failures.

Modularity is important to fault-tolerant systems because individual modules must be replaceable online. Keeping modules independent also makes it less likely that a failure of one module will affect the operation of another module. Having a way to increase performance by adding modules allows the capacity of critical systems to be expanded without requiring major outages to upgrade equipment.

Fail-fast logic, defined as logic which either works properly, or stops, is required to prevent corruption of data in the event of a failure. Hardware checks including parity, coding, and self-checking, as well as firmware and software consistency checks provide fail-fast operation.

Price and price performance are frequently overlooked requirements for commercial fault-tolerant systems -- they must be competitive with non-fault-tolerant systems. Customers have evolved ad-hoc methods for coping with unreliable computers. For instance, financial applications usually have a paper-based failback system in case the computer is down. As a result, most customers are not willing to pay double or triple for a system just because it is fault-tolerant. Commercial fault-tolerant vendors have the difficult task of designing systems which keep up with the state of the art in all aspects of traditional computer architecture and design, as well as solving the problems of fault tolerance, and incurring the extra costs of dual pathing and storage.

Tandem Architecture

The Tandem NonStop I was introduced in 1976 as the first commercial fault-tolerant computer system. Figure 1 is a diagram of its basic architecture. The system consists of 2 to 16 processors connected via dual 13 mbyte/sec busses (the “Dynabus”). Each

processor has its own memory in which its own copy of the operating system resides. All processor to processor communication is done by passing messages over the Dynabus.

Each processor has its own I/O bus. Controllers are dual ported and connect to I/O busses from two different CPUs. An ownership bit in each controller selects which of its ports is currently the "primary" path. When a CPU or I/O bus failure occurs, all controllers that were primaried on that I/O bus switch to the backup. The controller configuration can be arranged so that in an N-processor system, the failure of a CPU causes the I/O workload of the failed CPU to be spread out over the remaining N-1 CPUs (see Figure 1.)

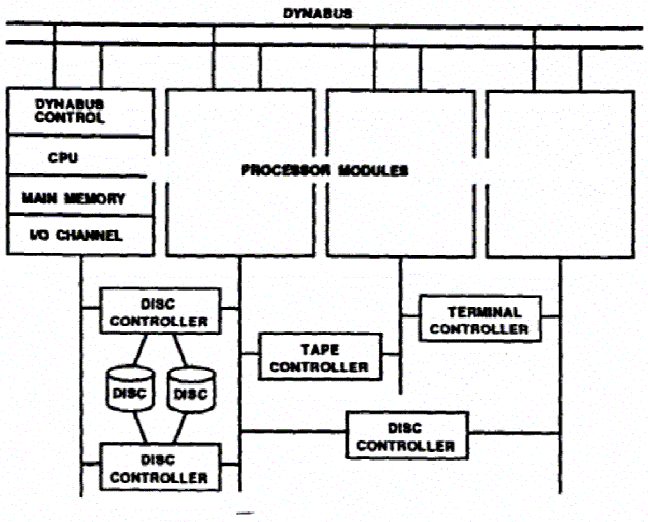

Figure 1. The original Tandem architecture. Up to 16 CPUs are connected via the dual 13 Mbyte Dynabus. Each processor has its own main memory and copy of the distributed operating system. The system can continue operation despite the loss of any single component.

CPUs

In the Tandem architecture, the design of the CPU is not much different than any traditional processor. Each processor operates independently and asynchronously from the rest of the processors. The novel requirement is that the Dynabus interfaces must be engineered to prevent a single CPU failure from disabling both busses. This requirement boils down to the proper selection of a single part type -- the buffer which drives the bus.

This buffer must be “well behaved” when power is removed from the CPU to prevent glitches from being induced on both busses.

The power, packaging and cabling must also be carefully thought through. Parts of the system are redundantly powered through diode ORing of two different power supplies. In this way, I/O controllers and Dynabus controllers tolerate a power supply failure.

Table 2 gives a summary of the evolution of Tandem CPUs.

	NonStop I	NonStop II	TXP	VLX
Year Introduced	1976	1981	1983	1986
MIPs	.7	.8	2.0	3.0
Cycle Time	100ns	100ns	83.3ns	83.3ns
Gates	20k	30k	58k	86k
CPU Boards	2	3	4	2
Integration	MSI	MSI	PALs	Gate Arrays
Virtual Mem Addressing	512KB	1GB	1GB	1 GB
Physical Mem Addressing	2MB	16MB	16MB	256MB
Memory per board	64-384KB	512KB-2MB	2-8MB	8MB

Table 2: A summary of the evolution of Tandem CPUs.

The original Dynabus connected from 2 to 16 processors. This bus was “over-designed” to allow for future improvements in CPU performance without redesign of the bus. The same bus was used on the NonStop II CPU, introduced in 1980, and the NonStop TXP, introduced in 1983. The II and the TXP can even plug into the same backplane as part of a single mixed system. A full 16 processor TXP system does not drive the bus near saturation. A new Dynabus has been introduced on the VLX. This bus provides peak throughput of 40 MB/sec, relaxes the length constraints of the bus, and has a reduced manufacturing cost due to improvements in its clock distribution. It has again been over-designed to accommodate the higher processing rates of future CPUs.

A fiber optic bus extension (Fox) was introduced in 1983 to extend the number of processors which could be applied to a single application. FOX allows up to 14 systems of 16 processors (224 processors total) to be linked in a ring structure. The distance between adjacent nodes was 1 Km on the original FOX, and is 4 Km with FOX II, which was introduced on the VLX. A single FOX ring may mix Nonstop II, TXP and VLX processors.

Fox is actually four independent rings. This design can tolerate the failure of any Dynabus or any node and still connect all the remaining nodes with high bandwidth and low latency.

Transaction processing benchmarks have shown that the bandwidth of FOX is sufficient to allow linear performance growth in large multi-node systems [Horst 85].

In order to make processors fail-fast, extensive error checking is incorporated in the design. Error detection in data paths is typically done by parity checking and parity prediction, while checking of control paths is done with parity, illegal state detection, and self-checking.

Loosely coupling the processors relaxes the constraints on the error detection latency. A processor is only required to stop itself in time to avoid sending incorrect data over the I/O bus or Dynabus. In some cases, in order to avoid lengthening the processor cycle time, error detection is pipelined and does not stop the processor until several clocks after the error occurred. Several clocks of latency is not a problem in the Tandem architecture, but could not be tolerated in systems with lockstepped processors or systems where several processors share a common memory.

Traditional mainframe computers have error detection hardware as well as hardware to allow instructions to be retried after a failure. This hardware is used both to improve availability and to reduce service costs. The Tandem architecture does not require instruction retry for availability. The VLX processor is the first to incorporate a kind of retry hardware, primarily to reduce service costs.

In the VLX, most of the data path and control circuitry is in high density gate arrays, which are extremely reliable. This leaves the high speed static RAMs in the cache and control store as the major contributors to processor unreliability. Both cache and control store are designed to retry intermittent errors, and both have spare RAMs which may be switched in to continue operating despite a hard RAM failure.

Since the cache is store-through, there is always a valid copy of cache data in main memory; a cache parity error just forces a cache miss, and the correct data is re-fetched from memory. The microcode keeps track of the parity error rate, and when it exceeds a threshold, switches in the spare.

The VLX control store has two identical copies to allow a two cycles access of each control store starting on alternate cycles. The second copy of control store is also used to retry an access in case of an intermittent failure in the first. Again, the microcode switches in a spare RAM online once the error threshold is reached.

Traditional instruction retry was not included due to its high cost and complexity relative to the small system MTBF it would yield.

Fault tolerant processors are viable only if their price-performance is competitive. Both the architecture and technology of Tandem processors have evolved to keep pace with trends in the computer industry. Architecture improvements include the expansion to 1 Gbyte of virtual memory (Nonstop II), incorporation of cache memory (TXP), and expansion of physical memory addressability to 256 Mbyte (VLX). Technology improvements include the evolution from core memory to 4k, 16K, 64K and 256K dynamic RAMs, and the evolution from Shottky TTL (Nonstop I, II) to Programmable Array Logic (TXP) to bipolar gate arrays (VLX) [Horst 84, Electronics].

The Tandem multiprocessor architecture allows a single processor design to cover a wide range of processing power. Having processors of varying power adds another dimension to this flexibility. For instance, for approximately the same processing power, a customer may choose a four processor VLX, a six processor TXP, or a 16 processor Nonstop II. The VLX has optimal price/performance, the TXP can provide better performance under failure conditions (losing 1/6 of the system instead of 1/4), and the NonStop II may be the best solution for customers who wish to upgrade an existing smaller system. In addition, having a range of processors extends the range of applications from those sensitive to low entry price, to those with extremely high volume processing needs.

Peripherals

In building a fault-tolerant system, the entire system, not just the CPU, must have the basic fault-tolerant properties of dual paths, modularity, fail-fast design, and good price/performance. Many improvements in all of these areas have been made in peripherals and in the system maintenance system.

The basic architecture provides the ability to configure the I/O system to allow multiple paths to each I/O device. With dual port controllers and dual port peripherals, there are actually four paths to each device. When discs are mirrored, there are eight paths which can be used to read or write data.

The original architecture did not provide as rich an interconnection scheme for communications and terminals. The first asynchronous terminal controller was dual ported, and connected to 32 terminals. Since the terminals themselves are not dual ported, it was not possible to configure the system in a way to withstand a terminal controller failure without losing a large number of terminals. The solution for critical applications was to have two terminals nearby which were connected to different terminal controllers.

In 1982, Tandem introduced the 6100 communications subsystem which helped reduce the impact of a failure in the communications subsystem. The 6100 consists of two dual

ported communications interface units (CIUs) which talk to I/O busses from two different processors. Individual Line Interface Units (LIUs) connect to both CIUs, and to the communication line or terminal line. With this arrangement, CIU failures are completely transparent, and LIU failures result in the loss only of the attached line(s). An added advantage is that each LIU may be downloaded with a different protocol in order to support different communications environments and to offload protocol interpretation from the main processors.

Dual pathing has also evolved in the support of system initialization and maintenance. NonStop I systems had only a set of lights and switches per processor for communicating error information and for resetting and loading processors. NonStop II and TXP systems added an Operations and Service Processor (OSP) to aid in system operation and repair. The OSP is a Z80 microcomputer system which communicates with all processors and a maintenance console. It can be used to remotely reset and load processors, and to display error information. The OSP is not fault-tolerant, but is not required to operate in order for the system to operate. Critical OSP functions, such as processor reset, reload and memory dump, can also be performed by the front panel switches.

In the VLX system, dual pathing and fault tolerance was also extended to a new maintenance system. This new system, called CHECK, consists of two 68000 based processors which communicate with each other and other subsystems via dual bit-serial maintenance busses. The CPUs, FOX II controllers, and power supply monitors all connect to the maintenance busses. Any unexpected event, such as a hardware failure, fan failure or power supply failure, is logged by CHECK. CHECK communicates with an expert system based program running in the main CPUs which later analyzes the event log to determine what corrective action should take place. The system also has dial-out and dial-in capability for notification of service personnel. Having a fault-tolerant maintenance system means that it can be always counted on to be functional, and critical operations can be done solely by the CHECK system. The front panel lights and switches were eliminated, and more functionality was incorporated into the CHECK system.

Modularity is standard in peripherals -- it is common to mix different types of peripherals to match the intended application. In online transaction processing in it is desirable to independently select increments of disc capacity and of disc performance. OLTP applications often require more disc arms per megabyte than is provided by traditional 14 inch discs. This may result in customers buying more megabytes of disc than they need in order to avoid queuing at the disc arm.

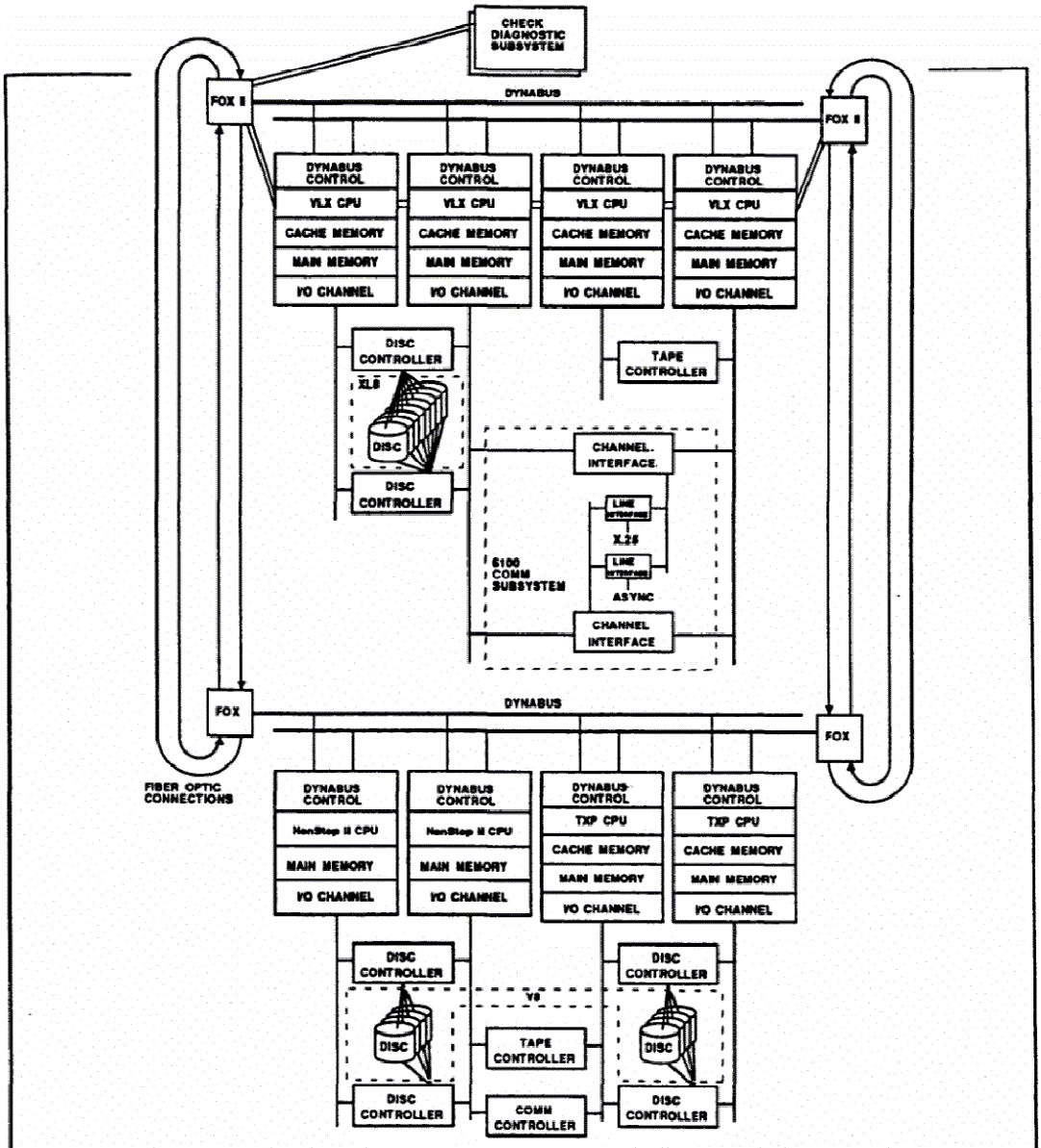

Figure 3. The 1986 Tandem architecture. Up to 14 systems of 16 CPUs (224 processors) are connected at distances of up to 4Km in a fault-tolerant fiber-optic ring network. The network can include three different processor types - the .8 MIPs NonStop II, the 2 MIPs TXP and the 3 MIPs VLX. New architectures for communications, disc drives, and maintenance have also been introduced.

In 1984, Tandem departed from traditional disc architectures by introducing the V8 disc drive. The V8 is a single cabinet which contains up to eight 168 Mbyte eight-inch Winchester disc drives in six square feet of floor space. Using multiple eight-inch drives instead of a single 14-inch drive gives more access paths and less wasted capacity. The modular design is more serviceable, since individual drives may be removed and replaced online. In a mirrored configuration, system software automatically brings the replaced disc up to date while new transactions are underway.

Once a system is single fault-tolerant, the second order effects begin to become important in system failure rates. One category of compound faults is the combination of a hardware failure and a human fault during the consequent human activity of diagnosis and repair. The V8 made a contribution to reducing system failure rates by simplifying servicing and eliminating preventative maintenance which combine to reduce the likelihood of such compound hardware-human failures.

Peripheral controllers have fail-fast requirements similar to processors. They must not corrupt data on both their I/O busses when they fail. If possible, they must return error information to the processor when they fail.

Tandem's contribution in peripheral fail-fast design has been to put added emphasis on error detection within the peripheral controllers. An example is Tandem's first VLSI tape controller. This controller uses dual lockstepped 68000 processors with compare circuits to detect errors. It also contains totally self-checked logic and self-checking checkers to detect errors in the random logic portion of the controller.

Above this, the system software uses "end-to-end" checksums generated by the high-level software. These checksums are stored with the data and recomputed and rechecked when, the data is reread.

In fault-tolerant systems design, keeping down the price of peripherals is even more important than in traditional systems. Some parts of the peripheral subsystem must be duplicated, yet they provide little or no added performance.

For disc mirroring, the two disc arms give better read performance than two single discs because the seeks are shorter and because the read work is spread evenly over the two servers. Writes on the other hand do cost twice as much channel and controller time. And mirroring does double the cost per megabyte stored. In order to reduce the price per megabyte of storage, Tandem introduced the XL8 disc drive in 1986. The XL8 has eight nine-inch Winchester discs in a single cabinet and has a total capacity of 4.2 Gbytes. As in the V8 drive, discs within the same cabinet may be mirrored, saving the cost of added

cabinetry and floor space. Also like the V8, the reliable sealed media and modular replacement keep down maintenance costs.

Other efforts to reduce peripheral prices include the use of VLSI gate arrays in controllers to reduce part counts and improve reliability, and using VLSI to integrate the stand alone 6100 communications subsystem into a series of single board controllers.

Year	Product	Contribution
1976	NonStop I	Dual ported controllers, single fault tolerant I/O system
1977	NonStop I	Mirrored and dual ported discs
1982	InfoSat	Fault tolerant satellite communications
1983	6100	Fault tolerant communications subsystem
1983	FOX	Fault tolerant high speed fiber optic LAN
1984	V8 Disc Drive	Eight drive fault-tolerant disc subsystem
1985	3207 Tape Ctrl	Totally selfchecked VLSI tape controller
1985	XL8 Disc Drive	Eight drive high-capacity / low-cost disc
1986	CHECK	Fault tolerant maintenance system

Table 4. Tandem contributions to peripheral fault tolerance.

SYSTEMS SOFTWARE

Processes and Messages

Processes are the software analog of processors. They are the units of software modularity, service, failure and repair. The operating system kernel running in each processor provides multiple processes, each with a one gigabyte virtual address space. Processes in a processor may share memory, but for fault containment, and to allow processes to migrate to other processors for load balancing, sharing of data among processes is frowned upon. Rather, processes communicate via messages. They only share code segments.

The kernel of the system performs the basic chores of priority dispatching, message transmission among processes, and reconfiguration in case of hardware failure.

The kernel sends messages among processes in a processor and also to kernels of other processors which in turn send the message to the destination process. The path taken by the message, memory-to-memory or across a local or long-haul network, is transparent to the sender and receiver. Only the kernel is concerned with the routing of the message. This is the software analog of multiple data paths. If a physical path fails, the message is retransmitted along another path.

The kernel is able to hide some hardware failures. For example, if a read-only page gets an uncorrectable memory error, then the page can be refreshed from disc. Similarly, a power failure is handled by storing the processor state to memory, quiescing the system and waiting for power restoration. Batteries carry the memory for several hours. Beyond that an uninterruptible power supply is required. When power is restored, the system resumes operation. Fail-fast requires that some hardware and software faults cause the kernel to fail the processor. Most such faults are induced by software bugs or operator errors -- the hardware is comparatively reliable [Gray 85].

When a processor stops, the other processors sense the failure by noticing that it has not sent an “I’m Alive” message lately (the latency is about 2 seconds). The remaining processors go through a “regroup” algorithm to decide who is up and who is down. This logic is fairly simple if the failure is simple, but can be complex in the presence of faulty processors or marginal power causing frequent failures. It has not been necessary to adopt the Byzantine Generals model of this problem, rather the regroup algorithm assumes that each processor is dead, slow, or healthy, and based on that assumption, the processors exchange messages to decide who is up. After two broadcast rounds, the decision is made and processors begin to act on it.

Process Pairs

When a process fails because of a transient software bug or a processor fault, single-fault tolerance requires that the application continue functioning. Process Pairs are one approach to this. A process can have a "backup" process in another CPU. The backup process has the same program, logical address space and sessions as the primary. Together these two processes comprise a process pair [Bartlett].

While the primary process executes, the backup is largely passive. At critical points, the primary process sends the backup "checkpoint" messages. These checkpoint messages can take many forms: they can be a "new state image" which the backup copies into its address space, a "delta" which the backup applies to its state, or even a function which the backup applies to its state. Gradually, Tandem is evolving to the delta and function approaches because they transmit less data and because errors in the primary process state are less likely to contaminate the backup's state [Borr 84].

When the primary process fails for some reason, the backup becomes the primary of the process pair. The kernels direct all current and future messages to the backup. Sequence numbers are used to regenerate duplicate responses to already-processed requests during the takeover. During normal operation, these sequence numbers are used for duplicate elimination and detection of lost messages in case of transmission error [Bartlett].

Process pairs give single-fault-tolerant program execution. They tolerate any single hardware fault and some transient software faults. We believe most faults in production software are transients (Heisenbugs). Process pairs allow fail-fast programs to continue execution in the backup when the software bug is transient [Gray 85].

Process Server Classes

To obtain software modularity, computations are broken into several processes. For example, a transaction arriving from a terminal passes through a line-handler process (say x.25), a protocol process (e.g. SNA), a presentation services process to do screen handling, an application process which has the database logic, and several disc processes which manage discs, disc buffer pools, locks and audit trails. This breaks the application into many small modules. These modules are units of service and of failure. If one fails, its computation switches to its backup process.

If a process performs a particular service, for example acting as a name server or managing a particular database, then as the system grows, traffic against this server is likely to grow. Gradually, the load on such a process will increase until it becomes a

bottleneck. Such bottlenecks can be an impediment to linear growth in performance as processors are added. The concept of process server class is introduced to circumvent this problem. A server class is a collection of processes all of which perform the same function. These processes are typically spread over several processors. Requests are sent to the class rather than to individual members of the class. As the load increases, members are added to the class. If a member fails or if one of the processors fails, the server class migrates into the remaining processors. As the load decreases, the server class shrinks. Hence process server classes are a mechanism for fault tolerance and for load balancing in a distributed system {Tandem Pathway].

Files

The data management system supports unstructured and structured (entry sequenced, relative, and key sequenced) files. Structured files may have multiple secondary indices. Files may be partitioned among discs distributed throughout the network. In addition, each file partition is mirrored on two discs. A class of process pairs manages each disc. Reads of a partition go to the primary class which maintains a cache of recently accessed disc pages. If the page is not in the disc cache, then the disc process reads it from the disc which is idle and which offers the shortest seek time. Because duplexed discs offer shorter seeks, they support higher read rates than two ordinary discs. Writes are a different matter. When a file is updated, it is updated on both of the mirrored discs -- so writes are twice as expensive. In addition, the disc process maintains file and record locks to avoid inconsistencies due to concurrent updates.

Files may optionally be protected with a transaction audit trail of undo and redo records along with file-granularity and record-granularity locks to prevent concurrency anomalies.

In the event of a hardware or software failure of the primary disc process server class, the backup server class in the other cpu assumes responsibility for that mirrored disc and continues service without interruption or loss of data integrity.

Transact ions

The work of a computation can be packaged as a unit by using the Transaction Monitoring Facility (TMF). TMF allows the application to obtain a transaction identifier (transid) for a particular job. All work done for that job is “tagged” with the transid. Locks are correlated to the transid. Undo and redo audit trail records are tagged by the transid. If the transaction commits, then all its effects are made durable, if it aborts, then all its effects are undone. For many applications, this is simpler than coding process

pairs. In fact, most Tandem customers now use this transaction mechanism in lieu of process pairs to get application fault tolerance. [Borr 81].

Device drivers and kernel software continue to need process pairs, because they are "below" the TMF interface. Indeed, process pairs are used to implement a non-blocking commit protocol in TMF and other basic systems features.

A process begins a transaction by invoking the BeginTransaction verb. This verb allocates a network-unique transaction identifier which will tag all messages and all database updates sent by this requestor and by servers working on the transaction. Locks acquired by the transaction are tagged by the transaction identifier. In addition, disc processes generate log records (audit trail records) which allow the transaction to be undone in case it aborts or redone in case it commits and there is a later failure. When the requestor is satisfied with the outcome, it can call CommitTransaction to commit all the work of the transaction. On the other hand, any process participating in the transaction can unilaterally abort it. This is implemented by the classic two-phase-locking and two-phase-commit protocols. They coordinate all the work done by the transaction at all the nodes of the local and long-haul network.

The transaction log, combined with an archive copy of the database, allows the system to tolerate dual disc media failures as well as software and operations failures which damage both media of a pair. Such failures may result in temporary data unavailability, but the data is not lost or corrupted.

This multi-fault tolerance has paid off for several customers and is becoming standard -- although multiple faults are rare, the consequent cost is very high. The transaction mechanism costs about 10% more and gives the customer considerably more peace of mind.

The most exciting new fault tolerance issue is disaster protection. Conventional disaster recovery schemes have mean time to repair of hours or days. Customers are increasingly interested in distributing applications and data to multiple sites so that one site can take over for another in a matter of seconds or minutes with little or no lost transactions or lost data. The transaction log applied to a remote copy of the data can keep the database up-to-date. By having a symmetric network design and application design, customers can have two or more sites back one another up in case of disaster.

Networking

The process-message based kernel naturally generalizes to a network operating system. By installing line handlers for a proprietary network, called Expand, Tandem was able to evolve the 16-processor design to a 4096 processor network. Expand uses a packet-switched hop-by-hop routing scheme to move messages among nodes, in essence it is a gateway among the individual 16-processor nodes. It is now widely used as a backbone for corporate networks, or as an intelligent network, acting as a gateway among other nets. The fault tolerance and modularity of the architecture make it a natural for these applications.

Increasingly, the system software is supporting standards such as SNA, OSI, MAP, SWIFT, etc. These protocols run on top of the kernel message system and appear to extend it [Tandem Expand].

APPLICATION DEVELOPMENT SOFTWARE

Application software provides a high-level interface for developing on line transaction processing applications to run on the low-level process-message-network system described above.

The basic principle is that the simpler the system, the less likely the user is to make mistakes.

For data communications, high-level interfaces are provided to "paint" screens for presentation services and a high-level interface is provided to SNA to simplify the applications programming task.

For data base, the relational data model is adopted and a relational query language integrated with a report writer allows quick development of ad hoc reports.

Systems programs are written in a Pascal-like language. Most commercial applications are written in Cobol or use the application generators. In addition, the system supports Fortran, Pascal, C, Basic, Mumps and other specialized languages. A binder allows modules from different languages to be combined into a single application, and a symbolic debugger allows the user to debug in the source programming language. The goal is to eliminate such low-level programming.

A menu-oriented application development system guides developers through the process of developing and maintaining applications. Where possible, it generates the application code for requestors and servers based on the contents of an integrated system dictionary.

Applications are structured as requestor processes which read input from terminals, make one or more requests of various server processes, and then reply to the terminals. The transaction mechanism coordinates these multiple operations making them atomic, consistent, integral, and durable.

The application generator builds most requestors from the menu-oriented interface, although the user may tailor the requestor by adding Cobol. The template for the servers is also automatically generated, but customers must add the semantics of the application -- generally using Cobol. Servers access the relational database either via Cobol record-at-a-time verbs or via set-oriented relational operators.

Using automatically generated requestors and the transaction mechanism, customers can build fault-tolerant distributed applications with no special programming.

As explained earlier, customers demand good price performance of fault-tolerant systems. Each VLX processor can process about ten standard transactions per second. Benchmarks have demonstrated that 32 processors have 16 times the transaction throughput of two processors: that is throughput grows linearly with the number of processors and the price per transaction declines slightly. We believe a 100 processor VLX system is capable of 1000 transactions per second. The price per transaction for a small system compares favorably with other full-function systems. This demonstrates that single-fault tolerance need not be an expensive proposition.

OPERATIONS AND MAINTENANCE

Operations errors are a major source of faults. Computer operators are often asked to make difficult decisions based on insufficient data or training. The Tandem system attempts to minimize operator actions and, where they are required, directs the operator to perform tasks and then checks his action for correctness.

Nevertheless, the operator is in charge, the computer must follow his orders. This poses a dilemma to the system designer -- how to limit the actions of the operator. First, all routine operations are handled by the system. For example, the system automatically reconfigures itself in case of a single fault. The operator is left with exception situations. Single-fault tolerance reduces the urgency of dealing with failures of single components. The operator can be leisurely about dealing with most single failures.

Increasingly, operators are given a simple and uniform high-level model of the system's behavior which deals in physical "real-world" entities such as discs, tapes, lines, terminals, applications, and so on, rather than control blocks. The interface is organized in terms of actions and exception reports. The operator is prompted through diagnostic steps to localize and repair a failed component.

Maintenance problems are very similar to operations. Ideally, there would be no maintenance. Single fault tolerance allows hardware repair to be done on a scheduled basis rather than "as soon as possible", since the system continues to operate even if a module fails. This reduces the cost and stress of conventional maintenance.

In practice, maintenance consists of diagnosing a fault and replacing a field replaceable unit (FRU). The failing FRU is sent back for remanufacture. Increasingly, the remote online maintenance system is used to diagnose and track the history of each FRU.

The hardware and software is extensively instrumented to generate exception reports. A rule-based system analyzes these reports and forms hypothesis on the cause of the fault. In some cases, it can predict a hard fault based on reports of transient faults prior to the hard fault. When a hard fault occurs or is predicted, a remote diagnostics center is contacted by the system. This center analyzes the situation and dispatches a service person along with the hardware or software fix.

The areas of single-fault-tolerant operations and single-fault-tolerant maintenance are major topics of research at Tandem.

SUMMARY AND CONCLUSIONS

Single-fault tolerance is a good engineering tradeoff for commercial systems. For example, single discs are rated at a MTBF of 3 years. Duplexed discs, recording data on two mirrored discs and connecting them to dual controllers and dual CPUs, raises the MTBF to 5000 years (theoretical) and 1500 years (measured). Triplexed discs would have theoretical MTBF of over one million years, but because operations and software errors dominate, the measured MTBF would probably be similar to that of duplexed discs.

Single-fault tolerance through the use of fail-fast modules and reconfiguration must be applied to both software and hardware.

Processes and messages are the key to structuring software into modules with good fault isolation. A side benefit of this design is that it can utilize multiple processors and lends itself to a distributed system design.

Modular growth of software and hardware is a side effect of fault tolerance. If the system can tolerate repair and reintegration of modules, then it can tolerate the addition of brand new modules.

In addition, systems must tolerate operations and environmental faults. Tolerating operations faults is our greatest challenge.

REFERENCES

[Bartlett] Bartlett, J.,"A NonStop Kernel," Proceedings of the Eighth Symposium on Operating System Principles, pp. 22-29, Dec. 1981.

[Borr 81] Borr, A., "Transaction Monitoring in ENCOMPASS," Proc. 7Th VLDB, September 1981. Also Tandem Computers TR 81.2.

[Borr 84] Borr, A., "Robustness to Crash in a Distributed Database: A Non Shared-Memory Multi-processor Approach," Proc. 9th VLDB, Sept. 1984. Also Tandem Computers TR 84.2.

[Burman] Burman, H. "Aspects of a High Volume Production Online Banking System", Proc. Int. Workshop on High Performance Transaction Systems, Asilomar, Sept. 1985.

[Electronics] Anon., "Tandem Makes a Good Thing Better", Electronics, pp. 34-38, April 14, 1986.

[Gray] Gray, J., "Why Do Computers Stop and What Can We Do About It?", Tandem Technical Report TR85.7, 1985, Cupertino, CA.

[Horst 84] Horst, R. and Metz, S., "New System Manages Hundreds of Transactions/Second." Electronics, pp. 147-151, April 19, 1984. Also Tandem Computers TR 84.1

[Horst 85] Horst, R., Chou, T., "The Hardware Architecture and Linear Expansion of Tandem NonStop Systems" Proceedings of 12th Intl. Symposium on Computer Architecture, June 1985, or Tandem Technical Report 85.3

[Mourad] Mourad, S. and Andrews, D., "The Reliability of the IBM/XA Operating System", Digest of 15th Annual Int. Symp. on Fault-Tolerant Computing, June 1985. IEEE Computer Society Press.

[Tandem] "Introduction to Tandem Computer Systems", Tandem Part No. 82503, March 1985, Cupertino, CA.

"System Description Manual", Tandem Part No. 82507, Cupertino, CA.

"Expand(tm) Reference Manual", Tandem Part No. 82370, Cupertino, CA.

"Introduction to Pathway", Tandem Computers Inc., Part No: 82339-A00, Cupertino, CA.