



the hp NonStop Tuxedo software advantage

An architectural comparison of the BEA Tuxedo product on Sun, IBM AIX, and other platforms with HP NonStop Tuxedo software on NonStop servers

contents

- The BEA Tuxedo architecture** **2**
 - Shared memory 2
 - Interprocess communication 3
 - Transaction management 4
 - Process management 4
 - Load balancing 5
 - Client distribution 5
- HP NonStop Tuxedo architecture** **6**
 - HP NonStop servers 6
 - Cluster and transaction services software infrastructure 7
- Optimizing NonStop Tuxedo software for NonStop servers** **13**
 - Application availability 13
 - Performance and scalability 16
 - NonStop software integration and interoperability 18
 - Simple administration 18
- Appendix a: glossary** **19**
- Appendix b: a summary of NonStop Tuxedo software characteristics** **23**

BEA Tuxedo is an industry-leading enterprise transaction processing monitor that provides a premier development platform for building and deploying business-critical applications. HP NonStop Tuxedo Release 8.0 software is the HP implementation of the BEA Tuxedo product on HP NonStop systems. This paper explains the cluster and transaction services for the HP platform and the modifications made to the BEA Tuxedo product to create the HP NonStop Tuxedo product.

Because NonStop Tuxedo external interfaces are identical to those of BEA Tuxedo, customers can use the same application program interfaces (APIs) and the same development, management, and administrative tools in creating applications for either product.

The BEA Tuxedo product is architected for networks of heterogeneous systems. It is ported to more than 70 hardware platforms and several operating systems. BEA Tuxedo assumes the use of external resource managers. By comparison, resource managers are tightly integrated with NonStop Tuxedo software. This product can participate fully and transparently in a network of heterogeneous platforms running BEA Tuxedo applications. It is optimized for clustered systems via use of cluster services (components of the HP NonStop Kernel operating system) and transaction services (provided by HP NonStop Transaction Management Facility and NonStop Transaction Services/MP software).

the BEA Tuxedo architecture

The result of these different architectures (clusters versus networks of machines) is that NonStop Tuxedo software for NonStop servers offers very high levels of application scalability and availability not possible on other platforms.

Note that the description of the NonStop Tuxedo product assumes it is being used with the HP NonStop SQL database, or with /Q queue files or both, unless otherwise stated.

BEA Tuxedo runs on symmetric multiprocessing (SMP) machines using a UNIX® or Windows NT® Server operating system. These general-purpose operating systems do not provide services that support transaction processing applications. BEA Tuxedo internally incorporates these services, thus providing an online transaction processing (OLTP) environment.

UNIX and Windows NT Server operating systems are general purpose and do not provide the services required for support of transaction processing applications (for example, transaction management, client concentration, load balancing, and process restart). Tuxedo incorporates these services internally, thereby providing an OLTP environment for applications running on UNIX and Windows NT Server operating systems.

Figure 1 depicts key elements of the BEA Tuxedo system architecture. Significant aspects of the implementation include

- Use of shared memory
- Interprocess communication
- Transaction management
- Process management
- Load balancing
- Client distribution

shared memory

SMP machines provide a single physical memory space that is shared by all processors in the machine. Memory is shared regardless of the processor on which a process is running. This facet of SMP architecture is exploited by BEA Tuxedo, which uses shared memory extensively for data structures such as these:

- *Bulletin board.* This data structure contains current configuration and processing state information. Some information is global and is replicated on every machine in the BEA Tuxedo application (for example, the names and locations of all servers offering a particular service). Other information is local and is visible only within the SMP machine on which the bulletin board exists (for example, the actual number and type of client requests currently waiting on a local server request queue). The bulletin board is accessed by clients, servers, and the bulletin board liaison process, which is responsible for updating the global information. Every machine in the application has a bulletin board liaison and a single master, or distinguished bulletin board liaison (DBBL) process, that is responsible for keeping all of the replicated bulletin boards synchronized.
- *Global transaction table.* This data structure contains current transaction state information and is accessed by transaction management server processes.

Concurrent access to shared memory is controlled via the use of semaphores.

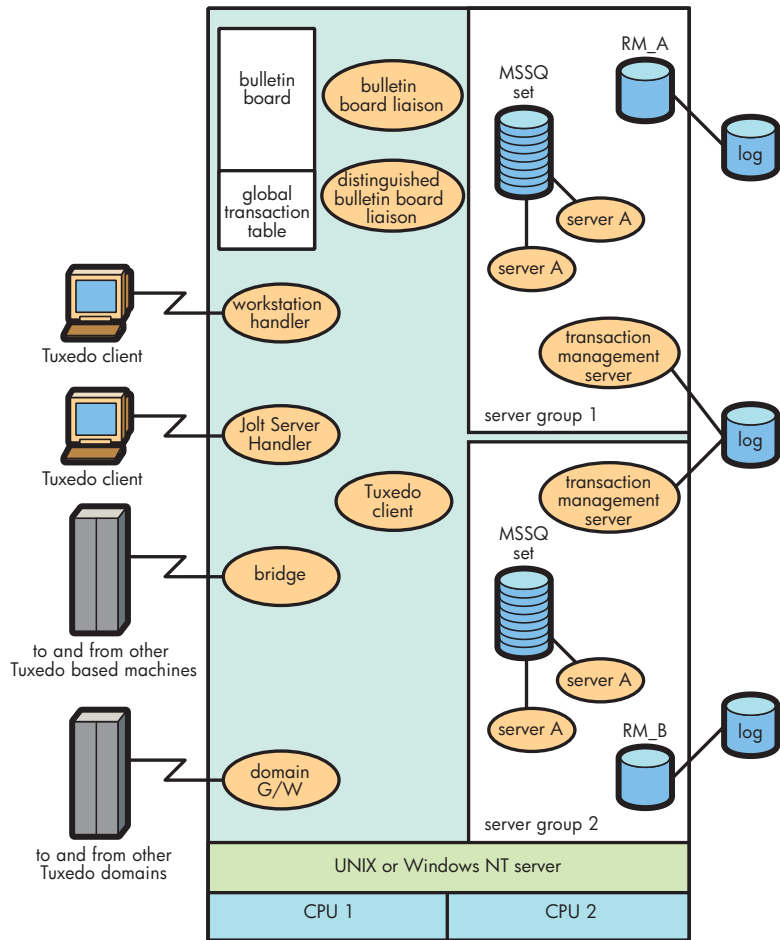


Figure 1. The BEA Tuxedo SMP architecture.

interprocess communication

In the BEA Tuxedo product, client/server communication between processes on the same machine is effected through the use of operating system local procedure call services. Because these services do not span machines, communication between processes on different SMPs is via network transport mechanisms (for example, TCP/IP). A BEA Tuxedo system process (the bridge process) exists on every machine in the application. The function of bridges is to route requests from local clients to remote machines, and to route requests from remote machines to local servers. Bridges communicate with each other across the network.

All of the server processes that read from a given queue are known collectively as a multiserver, single-queue (MSSQ) set. Because the scope of a message queue is local to a single SMP machine, all of the server processes in an MSSQ set are local to that machine.

transaction management

BEA Tuxedo uses external resource managers for application data access (for example, access to relational databases). Together, the resource managers and Tuxedo perform global transaction coordination using the X/Open XA interface.

BEA Tuxedo servers may access one or more resource managers as part of a single global transaction. The software is responsible for coordinating the work of all resource managers so they all commit or all roll back their branches of the transaction. This all-or-nothing characteristic of the transaction is known as *transaction atomicity*. Atomicity is achieved using the transaction management server process, which is provided for each server group defined in the application.¹ Each transaction management server manages that portion of the transaction branch executed by servers in its group. One transaction management server acts as the transaction coordinator (that is, the root). It identifies the other server groups involved in the transaction and broadcasts two-phase commit messages for those groups. The subordinate transaction management servers coordinate with their resource managers via the XA interface and return the reply to the coordinating transaction management server.

The coordinating transaction management server in each Tuxedo domain, as well as each resource manager involved in the transaction, writes a synchronous transaction log record during transaction commitment. There is one BEA Tuxedo physical log file per machine, plus one physical log file for each resource manager instance.

process management

One of the key functions of a transaction processing monitor is to provide fast, reliable business application access to many clients simultaneously. In a BEA Tuxedo application, this characteristic is achieved by providing multiple server processes that execute client requests. To ensure server availability, a periodic polling mechanism checks whether all server processes are up and running. (Polling periodicity is configurable.) Each server instance is polled by its local bulletin board liaison process. If a server process fails, the Tuxedo system cleans up and restarts the failed server process. If an entire machine fails, the servers on that machine are unavailable until either the machine is recovered manually or the servers are migrated administratively to another machine.

¹ At least one server group exists for every resource manager configured per application. Servers within a server group may access a single resource manager—for example, an Oracle® database or an Informix database. Servers in different server groups may access the same or different resource managers.

load balancing

Fast response time to clients is achieved by starting several identical server processes at once, with client requests distributed evenly among them. This approach enables a few server processes to handle many clients.

If these server instances are configured as an MSSQ set, then load balancing is achieved between the servers in the MSSQ set.² That is because only an idle server will read a request from the queue.

If only a single MSSQ set (or a single server) is configured to offer a particular service, then no further load balancing is required. If, however, more than one MSSQ set (or single server) offers a particular service, then requests must be distributed evenly between the server queues. Two methods of load balancing are practiced:

- *Within a machine.* The client and all servers offering the service are located on the same machine.
- *Across machines.* The client and servers offering the service are located on different machines.

In both cases, the target server is selected at the client. Within a machine, the bulletin board contains current statistics for the load on each server queue. Thus, the least-loaded server queue can be selected with precision. This is not the case for servers on remote machines, because the client site cannot reasonably determine the load on remote servers; there is no global view. For remote servers, a simple, weighted, round-robin algorithm toggles between servers on different machines.³

client distribution

Clients external to the Tuxedo application connect by using the /WS mechanism.⁴ Each client connection is assigned to a workstation handler process. An application can include many such processes, but these processes must all be running on the single machine serving the IP address used by the client to connect. Different clients can connect to different machines by using different IP addresses.

² In an MSSQ set, several identical server processes read from a single request queue.

³ The weighting is user configurable and allows for differences in the capacity of each machine—for example, the relative network bandwidth between the client machine and each possible server machine, or the power of each server machine (workstation versus mainframe).

⁴ External Java™ clients may also connect via Jolt. The mechanics of Jolt are like those of the /WS function. However, it has a Jolt Server Handler instead of a workstation handler.

hp NonStop Tuxedo architecture

NonStop Tuxedo software is architected for NonStop servers and for cluster and transaction services software from HP.

Although it has the same external features and functions as the BEA Tuxedo product, NonStop Tuxedo software is modified to adapt to and exploit the attributes of the underlying infrastructure of the NonStop Kernel operating system and the cluster and transaction services.

hp NonStop servers

A NonStop server consists of a set of loosely coupled, shared-nothing processing elements that act as a single logical machine (see figure 2). Each processing element is a uniprocessor together with its associated memory. A server comprises up to 16 uniprocessors, which are connected by HP ServerNet technology, a high-speed, any-to-any switching fabric. As many as 24 servers can also be networked together using HP ServerNet technology, to form an HP NonStop ServerNet Cluster (or "supercluster"). All critical components and data paths are replicated. In other words, there is no single point of failure.

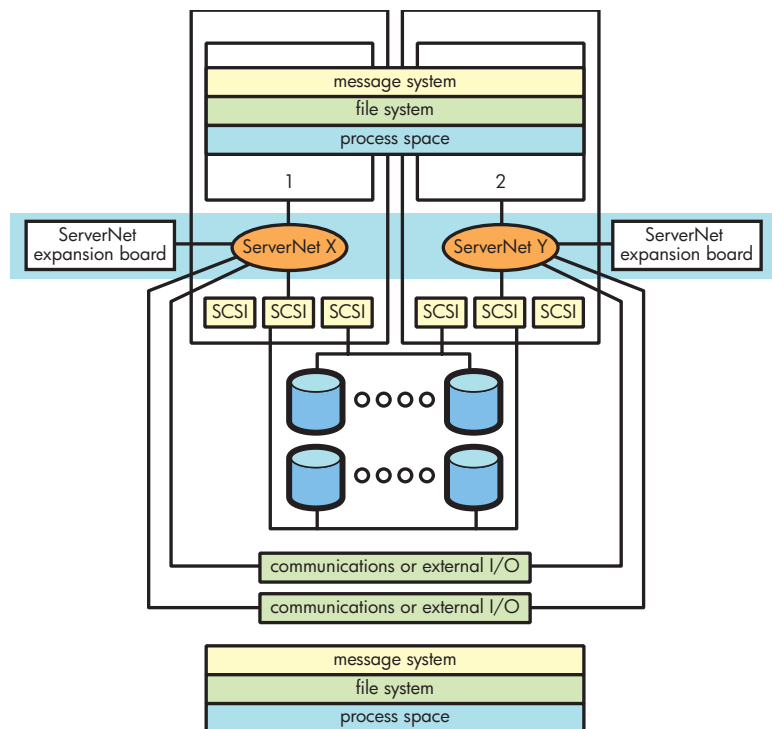


Figure 2. HP NonStop server hardware.

Each shared disk is accessible from two uniprocessors. Only processes in the same processing element, however, can share physical memory.

The server is homogeneous. Any data is accessible from any processing element, and any function can be executed on any processing element. Any single processing element can fail without causing a system failure. Processes send messages to each other via cluster services, and any process in the server can communicate with any other process in the server. The location of the target process is transparent to the caller.

The server has a single system image (SSI) and is administered as a unit. The fact that a server comprises many processing elements is transparent to users and application processes.

cluster and transaction services software infrastructure

NonStop Tuxedo software is built on a set of services that is designed explicitly to support loosely coupled systems and the needs of OLTP applications. These underlying services provide many OLTP-specific functions. Whereas the BEA Tuxedo product must incorporate OLTP functions to compensate for the generalities of the Windows NT Server and UNIX operating systems, a NonStop Tuxedo software implementation can simply rely on the underlying services of the NonStop Kernel operating system and transaction services provided by add-on software.

The same underlying infrastructure supports resource managers such as NonStop SQL database and /Q queue files. It includes cluster services and a set of transaction services (see figure 3).

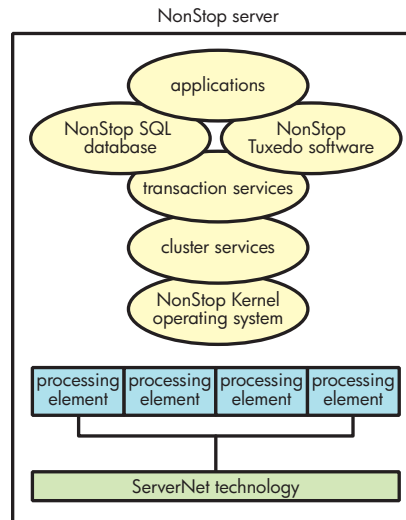


Figure 3. HP NonStop system software.

cluster services

Cluster services, which are integral to the operating system software, create a single system from a set of loosely coupled processors that are physically connected using a high-speed bus. These services include the following:

- A serverwide message system, which is used to send interprocess communication messages between processes on any processing element in the server. The processing element of the target process is transparent to the sender.
- A serverwide process space, so processes can be created on any processing element in the server, no matter where the process creator resides.
- Server membership and regroup services, which manage the processing elements and resources that make up a particular server and keep the server members synchronized in the event of any failures.
- Server failover services, which ensure that resources owned by a particular processing element (for example, IP addresses or disks), are made available on another processing element in the event of a failure.

transaction services

Transaction services are a set of common services that optimize the operating system for OLTP applications. NonStop Tuxedo software, NonStop SQL database, and other NonStop software products use these transaction services, which include the following components:

- NonStop Transaction Management Facility (NonStop TMF) software, which associates requests made to resource managers with the appropriate transaction and coordinates completion (commit/rollback) of the transaction for all resource managers involved. This component performs transaction recovery processing in the event of failure and provides transaction and data logging services for resource managers and for itself. This service is tightly integrated with the resource managers and with the cluster services message system (for transparent transaction propagation). It also includes facilities that allow transactions to be exported to and imported from other heterogeneous transaction managers. For example, it supports the Transaction Internet Protocol (TIP).
- NonStop Transaction Services/MP (NonStop TS/MP) software, which performs server process management the services necessary to ensure that application server processes are always available to clients. This software detects server process and processing element failures and automatically restarts server processes (on other processing elements, if necessary). It ensures that server processes are utilized efficiently by balancing the load of client requests among them: server processes are distributed across processing elements in the cluster, whereas client requests are routed to those that are least busy.

To adapt to and exploit this unique, loosely coupled system architecture, NonStop Tuxedo software is modified to handle the following key functions differently than BEA Tuxedo: shared memory, interprocess messages, transaction management, process management, load balancing, and client distribution. Figure 4 shows key elements of the NonStop Tuxedo system architecture.

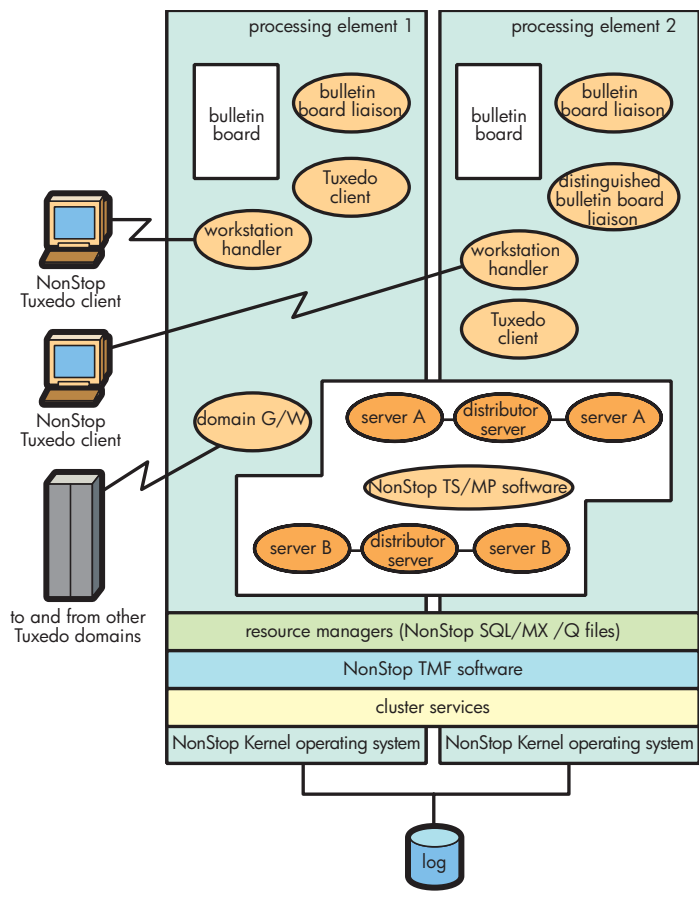


Figure 4. System architecture of NonStop Tuxedo software for NonStop servers.

shared memory

In a NonStop server, memory can only be shared between processes in the same processing element. If the BEA Tuxedo architecture were maintained in the NonStop Tuxedo product, execution of all client, server, and system processes would take place on a single processing element (because all of these processes require access to the shared-memory data structures, such as the bulletin board and global transaction table). Obviously, this design would not exploit the server at all, so a different architecture is used.

The simplest solution is to have each processing element represent a Tuxedo machine and to replicate every shared-memory data structure on every processing element. But such an implementation would have serious shortcomings. It would be difficult to configure and manage because every processing element would require an entry in the UBBCONFIG *MACHINES section, eliminating the SSI advantage. Also, the implementation would offer poor performance, due to the need to keep the replicated data structures synchronized.

The NonStop Tuxedo product resolves the issue in the following way:

- The bulletin board data structure and bulletin board liaison process are replicated across every processing element, and services are invoked via messages (as opposed to the server process reading from a shared queue). This enables clients and servers to be located on any processing element. During normal operation, there is little change to the global part of the bulletin board, so the overhead of propagating such changes to multiple copies of the bulletin board is not significant.
- The software does not use the global transaction table.

The architecture maintains a key BEA Tuxedo attribute: a single server appears equivalent to a single Tuxedo machine, thereby preserving the BEA Tuxedo externals with regard to configuration and runtime administration.

interprocess messages

All interprocess communication uses the cluster services message system, which seamlessly spans all processing elements in a server. The location of the destination process—whether on the same processing element as the sending process or on another processing element—is transparent to the caller and recipient. This approach contrasts with that of BEA's local procedure call implementation. The message communication takes place only within a single processing element. Therefore, BEA Tuxedo provides bridge processes that logically extend the local procedure call mechanism across processing elements. The NonStop Tuxedo product does not use bridge processes.

Because the cluster services message system spans processing elements, NonStop Tuxedo software has no constraints on server instance collocation with its message queue. Hence, server processes in a server class (equivalent to an MSSQ set in a BEA Tuxedo application) may be distributed across processing elements.

transaction management

NonStop TMF software provides the functionality of the global transaction table data structure and the transaction management server process of BEA Tuxedo. Therefore, these elements are not included in the NonStop Tuxedo product.

NonStop TMF software is cluster enabled; that is, transaction states are shared across processing elements. Any process in the server can do work on the same transaction branch. (Resource manager locks are shared.) Also, transaction recovery can be performed by any processing element in the server.

Functionality is also tightly integrated with the cluster services message system; transactions are propagated automatically between application processes. Resource managers (NonStop SQL and /Q queue files) are also cluster enabled; any resource manager data can be accessed from any processing element in the server. (There is logically a single resource manager instance per server.) A NonStop Tuxedo server group can therefore be distributed across all of the processing elements in the server. By contrast, BEA Tuxedo limits a server group to a single machine processing element.

NonStop TMF software initiates a transaction. When the client sends a request to a service, the cluster services message system propagates the transaction to the server process. All work done in the service is then associated with that transaction. When the client commits the transaction, NonStop TMF software broadcasts the two-phase commit sequence to all processing elements and then interacts with the local resource managers (actually, the local disk process for the processing elements) to prepare and commit—or rollback—any changes made on that processing element on behalf of the transaction.

A key difference in the interaction between NonStop TMF software and the resource manager, as compared with the compatible interaction for BEA Tuxedo, is that NonStop TMF software simultaneously acts both as a global transaction coordinator and a local transaction coordinator for the resource manager. Logically, this software performs two functions as defined by the XA interface: It performs the role of the resource manager and the role of the transaction manager. NonStop TMF software provides a single common transaction log per server, which saves all of the necessary transaction recovery information for both itself and the resource manager. A single synchronous log write per server occurs at transaction commitment, irrespective of the number of server groups, resource managers, or processing elements involved in the transaction. For example, if a single transaction calls for the software to update NonStop SQL and /Q queue files on every processing element in the server, all of the information necessary for recovery is logged with a single synchronous write.

process management

NonStop Tuxedo process management is performed by NonStop TS/MP software. (This fact is not externally noticeable.) NonStop Tuxedo server processes are implemented as NonStop TS/MP server classes. A server class—the equivalent of a BEA Tuxedo MSSQ set—is a set of identical server processes, meaning that the same set of services is offered by each server instance. Server processes in a server class may be distributed across processing elements. A server class may also consist of a single server process. A NonStop Tuxedo server group comprises one or more server classes. Administration commands such as `tmboot` and `tmshutdown` are transformed internally into NonStop TS/MP commands to start and stop server classes.

NonStop TS/MP software is responsible for server process failure detection and restart. It also provides the server process surveillance functionality of BEA Tuxedo. This mechanism is considerably different from the BEA mechanism because of the OLTP optimization provided by the cluster services of the operating system. Any process can request notification from cluster services of specific events—for example, the abnormal termination of another process, or a processing element failure. NonStop TS/MP software (actually, the supervisor monitor) receives notification when any server process in a server class fails, and automatically restarts the failed server. If it receives notification that a processing element has failed, it can, if necessary, automatically restart all server processes from the failed processing element on another processing element. In addition, once a failed processing element is restored, server processes may be automatically moved back to the restored processing element in order to maintain balance.

This software also manages the following system processes: workstation listener, Jolt Server Listener, /TDomain gateway, bulletin board liaison, distinguished bulletin board liaison, event server, /Q server, /Q forwarding server, NOREPLY server, and ADMIN SVR. All of these system processes are restarted automatically in the event of failure.

load balancing

In a NonStop Tuxedo application, the name of a requested service is mapped in the bulletin board to the name of the server class or classes that offer that service.⁵ The server class name is thus obtained from the bulletin board and is used to address the message. In the case of a service offered only by a single-server server class, client requests are sent directly to the server process (where queuing occurs if necessary). No load balancing is required.

In the case of a service offered by a multiple-server server class, NonStop TS/MP software performs load balancing between servers in the server class via a distributor server process. (Each server class has a separate distributor server process.) Logically, this process represents the single queue in an MSSQ set to which all client requests are sent. The server processes in the server class are single servers—each with its own queue—to which the distributor server process forwards the request. When the NonStop Tuxedo service returns, the reply is routed via the distributor server process back to the client. Because it handles all client requests and replies, this process has a real-time global view of the current state of all server processes in the server class (busy or idle), across all processing elements in the NonStop server. The distributor server process can therefore perform accurate load balancing when forwarding the client request. Queuing occurs if necessary (for example, if all of the server processes in the server class are busy).

In the event that a service is offered by more than one server class, NonStop Tuxedo software selects the server class to receive the request. It does so by using the same round-robin algorithm that BEA Tuxedo uses when the same service is offered by server processes on different machines. NonStop TS/MP software then selects a server process instance within that server class, as described.

client distribution

NonStop Tuxedo clients typically connect to NonStop servers by using the /WS mechanism (or by using Jolt for NonStop Tuxedo software). However, NonStop Tuxedo software distributes workstation handler processes and Jolt Handler processes across the processing elements of the server and automatically load balances client connections across them. Clients are therefore distributed across the processing elements in the server.

Clients use a single IP address to connect to the server. In the event of a processing element failure, cluster services create an automatic failover of the IP address to another processing element in the server. (The software also creates a failover, when necessary, for any application server or system processes to another processing element.) Workstation clients can then reconnect to the application by using the same IP address as before.

This mechanism has been enhanced in NonStop Tuxedo Release 8.0 software to take advantage of the new NonStop Parallel TCP/IP feature. This is described more in the text that follows.

⁵ The UBBCONFIG *SERVERS RQADDR parameter specifies the server class name.

optimizing NonStop Tuxedo software for NonStop servers

NonStop Tuxedo software is optimized to take advantage of specific features of the NonStop Kernel operating system. The resulting implementation exhibits unique and beneficial characteristics when running on NonStop servers.

Beneficial characteristics of NonStop Tuxedo software on NonStop servers include

- Application availability
- Performance and scalability
- Simple administration

application availability

NonStop Tuxedo software provides high levels of application availability because all application services remain available despite the failure of any of the following:

- Application server processes
- NonStop Tuxedo system processes
- NonStop SQL or /Q queue file system processes
- Transaction services system processes
- Cluster services system processes
- NonStop Kernel operating system
- Hardware (memory, processor, disks, and so on)

All data remains accessible from any processing element, despite any single failure. This high level of application availability is the result of several factors.

continuously available hardware

The components in the underlying hardware of a NonStop Tuxedo application are replicated and isolated. (For example, the hardware has separate memory per processing element and dual-ported, mirrored disks.) As a result, no single point of failure renders an entire NonStop server unavailable. Failure of a processing element or its memory affects only the processes running on it. No single hardware error can occur that makes any NonStop Tuxedo service or data on a server unavailable.

For an SMP machine, though, failure of memory or a single processor can bring down the entire machine along with all the Tuxedo servers and databases running on it.

service availability despite machine failure

Loss of a processing element does not compromise the availability of any NonStop Tuxedo resources. In fact, there is virtually no chance of service disruption, for the following reasons:

- Server processes are distributed across processing elements. If an element fails, server processes in another processing element (one that offers the same services) continue to process client requests uninterrupted.
- NonStop TS/MP software detects the processing element failure and, if necessary, restarts (migrates) the server processes automatically to another processing element. This includes critical NonStop Tuxedo system server processes as well as application server processes, so the system itself remains up and running.
- All physical devices are dual ported across two processing elements; so if one fails, the devices are still available to all other processing elements and to the server processes running on them. NonStop SQL data and /Q queue files on disks connected to the failed processing element remain accessible from all other processing elements in the NonStop server. Workstation clients connected to the failed processing element simply reconnect using the same IP address.
- A NonStop Tuxedo application may be configured to run across two or more NonStop servers (a supercluster). In the event that an entire NonStop server fails, the application still remains available on the other NonStop servers in the supercluster. In addition, if the failed NonStop server was the master node, the DBBL process may be migrated to another node in the supercluster, enabling the NonStop Tuxedo application to remain fully functional. This capability of DBBL migration also facilitates the maintenance of application availability during planned application or system reconfiguration involving the master node.

automatic transaction recovery

Transaction recovery can be performed from any processing element and is automatic. The NonStop TMF transaction log is accessible from any processing element. Because NonStop TMF software also acts as the recovery manager of the resource manager, the log is also available from any processing element. When an element fails, the software automatically performs recovery of in-doubt transactions on another processing element.

Using the audit log generated by NonStop TMF software, changes to primary databases are instantaneously replicated to backup databases on one or more target systems using HP NonStop Remote Database Facility (NonStop RDF) software, no matter how many transactions per second your application generates. If a primary database becomes inaccessible for any reason, application processing can continue using the backup database with minimal service disruption or data loss. This facility enables NonStop Tuxedo services to remain available even in the event of failure of an entire NonStop server node.

Application processes that cannot tolerate any data loss whatsoever can request that transactions be lockstepped between the primary and backup databases. A simple system call ensures that the process cannot continue until the changed data is safely stored on the target backup system. The application decides which transactions should or should not be lockstepped for maximum flexibility. Active throughput is approximately 3 megabytes per second per audit trail, and if the two systems somehow become disconnected, NonStop RDF software can catch up at more than double that rate. Figure 5 shows how NonStop RDF software can minimize the effects of regional disasters.

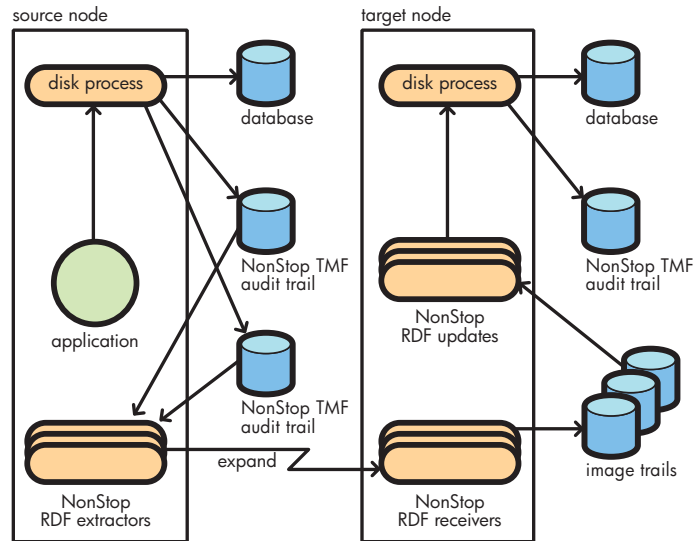


Figure 5. NonStop RDF software minimizes the effects of regional disasters by efficiently sending NonStop TMF audit trail information to one or more target systems.

fault-tolerant system software

Not only are the NonStop Tuxedo server processes fault tolerant, but so are other components of the cluster infrastructure. Key system processes use process-pair technology: a primary process has a backup process on another processing element, and state information is passed periodically between the primary and backup processes to keep the two synchronized. When a primary process fails, the failure is detected by the backup process, which assumes the role of the primary and performs any necessary recovery actions. NonStop Tuxedo applications benefit from at least three examples of process-pair technology.

- The data access manager, a component of NonStop SQL software, provides access to resource manager data (NonStop SQL or /Q queue files) located on disks attached to the processing element on which a given data access manager is running. One such process manages the physical disks. When a processing element fails, backup data access managers on other processing elements take over and the data remains accessible. (All physical disks are dual ported across two processing elements. The backup data access managers run on the same processing element as the backup device port. In addition, disks may be mirrored in order to maintain data availability in the event of a disk failure).
- NonStop TMF software performs transaction completion and recovery. It works closely with the data access managers. If a processing element fails, a backup on another processing element takes over and performs transaction recovery for any in-doubt transactions.
- NonStop TS/MP software monitors server processes (and restarts them if there are any failures). If the processing element on which the supervisor process is running fails, the backup supervisor process takes over and restarts any other server processes that were running on the failed processing element.

immediate failure detection

Server process failure detection occurs immediately when the failure happens. This event-driven mechanism makes failed servers quickly available.

access quickly restored

Access to NonStop Tuxedo services via /WS, Jolt, and /TDomain is quickly restored. The handler and listener processes for /WS and Jolt are automatically restarted if a process or processing element fails. (The workstation listener restarts workstation handlers, and the Jolt Server Listener restarts Jolt Server Handlers. NonStop TS/MP software manages workstation listeners and Jolt Server Listeners, and is responsible for restarting them.) When a client program reconnects, it is connected with a restarted handler. Thus, remote clients can quickly resume work in the event of a processing element failure.

NonStop Tuxedo Release 8.0 software takes advantage of the new Parallel TCP/IP (PTCPIP) feature. This feature enables multiple /WS and Jolt listener processes on different NonStop server processing elements to listen on the same IP port address.⁶ This conveys the following benefits:

- *Improved scalability.* When clients request a connection, the PTCPIP mechanism automatically distributes the connections across listener processes across processing elements in order to balance the load.
- *Improved availability.* In the event of a processing element failure, the client may immediately retry the connection that will be routed to an alternate listener process; there is no need to wait for the failed listener process to be restarted on another processing element.
- *Simplified client programming and administration.* Previously, in order to avoid the brief outage or processing element failure before the listener process is restarted, and/or to explicitly partition clients across multiple listener and handler process groups, clients had to be configured with a list of listener addresses (one for each of these groups). Using this list and some programmatic algorithm, clients could distribute themselves across listeners. If the listener first selected is not available, then the next listener in the list could be tried, and so on. This is an inelegant and inefficient mechanism, which is obviated by PTCPIP. With PTCPIP the client only needs to know one IP address, connection requests are partitioned across listeners automatically, and connection requests will never be sent to an unavailable listener. This removes the need for providing a listener address list in the client.

/TDomain gateways are also implemented as server processes managed by NonStop TS/MP software and are restarted automatically after a failure on another processing element, if necessary.

performance and scalability

NonStop Tuxedo software offers high levels of performance and scalability on NonStop servers.

minimal contention for shared resources

NonStop Tuxedo software runs with minimal contention for shared resources; it can scale to high levels before it becomes necessary to add extra machines to the configuration. For SMP architectures, increasing contention for shared resources (for example, shared memory) affects scalability as more CPUs are added to the machine.

⁶ Note that /WS and Jolt listener processes must use different port numbers.

Because NonStop Tuxedo server processes are distributed across processing elements (and, thus, across memory), adding server processes only marginally increases contention for shared resources and continues to increase real throughput until the processing elements are utilized completely. At that point, additional processing elements may be added to the server; these will also deliver 100 percent useful work.

efficient load balancing

The system achieves optimal load balancing, regardless of how many processing elements a NonStop Tuxedo server class is distributed across, because a single distributor server process (queue) remains for the entire server class. There is no performance penalty if the distributor server process is not local to the client (that is, on the same processing element).

minimal system overhead per transaction

NonStop Tuxedo applications require little processing at transaction completion and few message exchanges during the two-phase commit process. For high-volume OLTP, the amount of synchronous I/O per transaction—at transaction completion—is a critical throttle on throughput. NonStop TMF software reduces this overhead significantly by executing just one synchronous log write per transaction. It executes no additional log writes, no matter how many resource managers are involved in the transaction.

A two-phase commit process reduces message overhead. That is because a single, logical transaction management server per NonStop server is involved in the message exchange. By contrast, other processes involve one transaction management server per server group.

event-driven server failure detection

The event-driven, server failure detection mechanism of NonStop Tuxedo software does not consume processor cycles unless a failure actually occurs. Therefore, protection against failure does not affect performance during normal processing.

efficient /TDomain gateway mechanism

The /TDomain gateway mechanism is optimized for NonStop Tuxedo software in the following ways:

- The system merges the separate BEA Tuxedo domain gateway and administrative server processes into a single process, thereby consuming minimal system resources.
- The domain gateway never preempts while waiting for work to complete (for example, while waiting for the administrative server to perform logging). The domain gateway does not waste processor cycles; if it needs to wait, the gateway gives up control until a completion event occurs.⁷
- The domain gateway uses an efficient work scheduling algorithm. By using a single queue for requests and replies, a queue is never “starved” and the domain gateway processes work in an optimal fashion.⁸

absence of bridge processes

The absence of bridge processes in NonStop Tuxedo software reduces the overhead of sending messages between machines.

⁷ In preemption, the code loops while waiting for a completion event; it does not give up control to allow the processor to execute other work in the meantime.

⁸ Starvation occurs when work queued is not processed for some reason (for example, if it is preempted by work on a higher-priority queue).

clients distributed across processing elements

In NonStop Tuxedo software, workstation and Jolt client connections are balanced across multiple handler processes (that is, Jolt Server Handler and workstation handler processes), which may in turn be distributed across multiple processing elements.

NonStop software integration and interoperability

NonStop Tuxedo is well integrated and interoperable with other NonStop system software applications:

- NonStop Tuxedo services may invoke NonStop CORBA object methods with full transactional interoperability. This support is bidirectional.
- NonStop Tuxedo services may be invoked as Enterprise JavaBeans (EJB) methods from BEA WebLogic Server components with full transactional interoperability. This is achieved via the WebLogic Tuxedo Connector service, which is supported on the NonStop server. The BEA WebLogic Server components themselves may be running on a NonStop server, or on some other remote machine type.
- Because these NonStop system software components (NonStop Tuxedo, NonStop CORBA, BEA WebLogic Server) all utilize NonStop TMF software for transaction management when a business application process spans these environments, only a single NonStop TMF transaction context is created, which is shared across the environments. This improves scalability because there is only a single transaction log record written when the transaction is committed. In other systems, each of these application infrastructures, as well as any databases that are accessed, will each have their own transaction log writes.

simple administration

NonStop Tuxedo software offers simple administration for the following reasons:

- Only one physical transaction log needs to be configured and managed per NonStop server.
- A NonStop server is configured and managed as a single logical machine, regardless of the number of processing elements in the server.
- Identical application server processes (for example, the members of an MSSQ set) are managed as a single entity (a server class), rather than as individual processes.

appendix a: glossary

/TDomain

A communications mechanism that enables clients in one Tuxedo domain to access services in another Tuxedo domain, as if the services were local.

/WS or /Workstation

A Tuxedo subsystem in which clients running on desktop machines such as PCs can access Tuxedo services on remote server (host) machines without having to install the full Tuxedo software on the workstation. Clients use the full Application Transaction Monitor Interface (ATMI) API.

application

In the context of Tuxedo, a term that is synonymous with domain. (See domain.)

application server process

An operating system process that hosts NonStop Tuxedo application service code; a member of a server class.

cluster services

Software services that group individual NonStop server processing elements into a clustered SSI. Cluster services are components of the NonStop Kernel operating system.

distinguished bulletin board liaison (DBBL)

The distinguished bulletin board liaison process. A NonStop Tuxedo system process that ensures consistent application configuration state across a NonStop Tuxedo system domain. There is one DBBL process per domain. If the DBBL process becomes unavailable for some reason, no changes may be made to the NonStop Tuxedo system configuration that is running.

distributor server

A system server process (one per server class), managed by NonStop TS/MP software, that balances the client request load between server instances in the server class. The distributor server process is equivalent to the single queue in an MSSQ set configuration in BEA Tuxedo.

domain

The collection of machines and the server processes running on those machines, as defined by a single Tuxedo UBBCONFIG configuration file. A domain is administered as a unit.

in-doubt transaction

A transaction that the application program has requested to be committed and that has successfully completed the first phase of the two-phase commit process. At this point, the transaction outcome is known only to the transaction coordinator and is said to be "in-doubt" at subordinate nodes. Transactions in the in-doubt state are the ones that must be recovered in the event of system failure.

Jolt

A BEA product analogous to /WS that enables a client written in Java language, possibly also running as an applet in a Web browser, to access remote Tuxedo services.

Jolt for NonStop Tuxedo software

The Jolt software adapted to be compatible with NonStop Tuxedo applications.

loosely coupled, massively parallel (MP)

A shared-nothing hardware architecture in which each processing element executes against its own physical memory and a private copy of the operating system. Processing elements communicate by exchanging messages across a high-speed bus.

machine

A computer (uniprocessor or SMP). BEA Tuxedo server processes are configured to run on a per-machine basis. A machine is equivalent to a processing element in a NonStop server.

master node

The NonStop server running the DBBL process.

multiserver, single-queue (MSSQ) set

A Tuxedo construct in which several identical server processes read from a single client request queue. Such a configuration provides optimal load balancing. This concept parallels the distributor server process and server class mechanism of NonStop Tuxedo software.

node

A NonStop server (comprising two to 16 processing elements).

NonStop server

An HP server machine that is configured typically in a loosely coupled, massively parallel cluster running the NonStop Kernel operating system.

NonStop Kernel operating system

The microkernel operating system for hardware. NonStop Kernel operating system supports Guardian and Open System Services (OSS) application-level personalities.

NonStop SQL database

A SQL relational database management system (RDBMS) from HP.

NonStop Transaction Management Facility (NonStop TMF) software

The HP product that provides transaction management services for transaction processing monitors and resource managers on the NonStop server platform. NonStop Tuxedo software uses NonStop TMF software for transaction management.

NonStop Transaction Services/MP (NonStop TS/MP) software

The HP product that provides common services for transaction processing monitor environments on the NonStop server platform. This software offers services such as link management, process management, and load balancing to provide high availability and scalability. NonStop Tuxedo software uses NonStop TS/MP services.

NonStop Tuxedo software

HP's implementation of BEA Tuxedo System /T (Release 8.0) for the NonStop server platform.

online transaction processing (OLTP)

Also known as transaction processing, a style of data processing whereby user requests are processed in real time. Results of the processing are reflected in business data in real time. OLTP applications often represent the business itself (for example, an airline reservations system). Transaction processing monitors are often used to simplify end-user programming and provide facilities to meet the demanding requirements of OLTP applications (especially in terms of data integrity, availability, and performance [throughput and response time]).

processing element

The hardware "atom" in a NonStop server. It comprises the processor, memory, and operating system instance. Up to 16 processing elements can be configured in a single NonStop server. A processing element is equivalent to a "machine" in BEA Tuxedo.

server class

A NonStop TS/MP construct that describes a group of identical server processes, for which a distributor server process performs load balancing of client requests evenly among the server processes in the group. Server classes are the server process units of NonStop Tuxedo configuration and administration (boot, shutdown, and so on). A server class is equivalent to a BEA Tuxedo MSSQ set.

ServerNet

A core technology for server architectures that allows data to flow within the server and between server nodes without first having to flow through a processor. Data can travel efficiently from device to device, from device to processor, or from processor to processor (an any-to-any connection architecture).

shared nothing

Refers to a type of hardware and software architecture in which each processor has its own physical memory and copy of the operating system. An example is the NonStop server. See also loosely coupled, massively parallel.

single server process

A Tuxedo application server process configured with its own individual queue (that is, not a member of an MSSQ set). In NonStop TS/MP terms, it results in a server class with a membership of one server process instance and no distributor server process.

supercluster

Up to 24 NonStop server nodes connected together via a ServerNet system area network. Also known as a NonStop ServerNet Cluster.

symmetric multiprocessor (SMP)

A hardware architecture whereby a single copy of physical memory and a single copy of the operating system is shared among multiple processors.

transaction processing monitor

A software subsystem that provides an optimized OLTP application execution environment. Examples include NonStop Tuxedo software and the Pathway environment.

transaction services

Common system software services used by transaction processing monitors and resource managers on NonStop systems. These services include NonStop TMF and NonStop TS/MP software.

two-phase commit

The protocol used by transaction managers and resource managers to coordinate transaction updates in such a way that changes are either all committed or all rolled back.

workstation client

A Tuxedo client process accessing Tuxedo services via the /WS mechanism.

workstation handler

A Tuxedo system process that acts as a surrogate client for workstation clients on the host Tuxedo system. There may be multiple workstation handler processes. A workstation client is allocated to a specific workstation handler when it joins the Tuxedo application. ATMI calls made by the workstation client are sent to the workstation handler, which reissues the calls and returns the reply. The Jolt Server Handler process is analogous for Jolt clients.

workstation listener

A Tuxedo system process that receives initial connection requests from workstation clients and allocates the client to a specific workstation handler. The Jolt Server Listener process is analogous for Jolt clients.

XA specification from X/Open

A subsystem interface implemented by transaction managers and resource managers; a transaction manager coordinates commitment with a resource manager. Via this interface, a transaction manager can control the coordination of transactions involving updates to multiple resource managers.

appendix b: a summary of NonStop Tuxedo software characteristics

high availability

- Processing element failure will not result in a denial of service.
(There is virtually no chance of service disruption.)
 - Application and system server processes remain available.
 - NonStop SQL data and /Q queue files remain accessible.
 - Transaction recovery is performed automatically.
 - Client access through /WS, Jolt, and /TDomain is quickly restored.
- Continuous availability of underlying hardware
- Timely and efficient server failure detection
- Fault-tolerant system software

high performance and scalability

- Minimal contention for shared resources
- Few I/Os and messages per transaction
- Efficient load balancing
- Efficient /TDomain gateway mechanism
- Clients distributed across processing elements

simplified administration

- One physical transaction log per cluster
- One logical machine to be managed per cluster
- Relatively few server processes to be managed per application

For more information, go to www.hp.com/go/nonstop.

December 2002, first published 1999. Windows NT is a U.S. registered trademark of Microsoft Corporation. UNIX is a registered trademark of The Open Group. Oracle is a registered U.S. trademark of Oracle Corporation, Redwood City, California. Java is a U.S. trademark of Sun Microsystems, Inc. All other product names mentioned herein may be trademarks of their respective companies. HP shall not be liable for technical or editorial errors or omissions contained herein. The information is subject to change without notice. The warranties for HP products are set forth in the express limited warranty statements accompanying such products. Nothing herein should be construed as constituting an additional warranty.

5981-5090EN

©2002 Hewlett-Packard Company, L.P.

