Technical white paper

NonStop Open System Services

The UNIX-like operating environment for MPP architecture

Table of contents

HP NonStop Open System Services

For nearly four decades, HP Integrity HP NonStop servers have been the IT backbone for enterprises around the world, spanning across vertical markets such as finance, telecom, transportation, and many others. The HP NonStop UNIX-like programming and runtime environment will help the large ecosystem of UNIX/Linux solutions easily migrate over to HP NonStop.

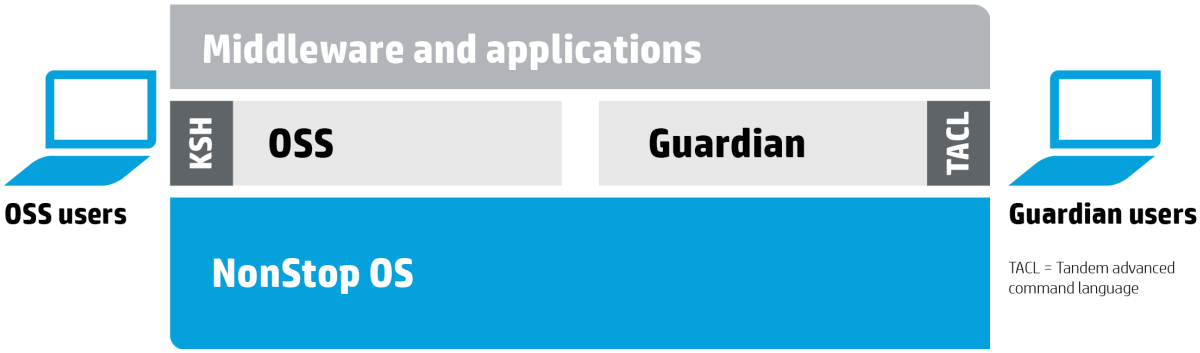
Introduction to Open System Services

HP Integrity NonStop servers provide a near continuously available and near linearly scalable platform for mission-critical applications. They achieve these characteristics through a massively parallel processing (MPP) architecture built on a platform with reliability, availability, and serviceability (RAS) capabilities and hardware and software redundancy that enable them to survive almost all single point failures.

The base operating system (OS) of NonStop, called the NonStop OS (NSOS), provides two different environments, or set of services and application programming interfaces (APIs), for users and applications. One is the HP proprietary Guardian environment, supported from the inception of NonStop computing. In the 1990s, HP embarked on a journey to support a UNIX®-like (*NIX) environment on NonStop, resulting in the development of the Open System Services (OSS) environment. The NSOS underpins both the Guardian and OSS environments; they co-exist—and even interoperate—on NonStop. As the following diagram illustrates, applications access the same NonStop OS features and services regardless of whether they use OSS or Guardian interfaces.

Implementation of OSS is integrated deep inside the NonStop Operating System, rather than being built as a layer on top of the OS, in order to achieve a low performance overhead. The OSS environment has matured into a strong and stable operating environment on NonStop servers, and existing and new customers use it for developing newer applications or migrating existing *NIX applications over to NonStop. Through its rich UNIX/Linux environment, it enables easy migration of applications and solutions from the *NIX world into NSOS.

Figure 1. OSS and Guardian Environments of NonStop

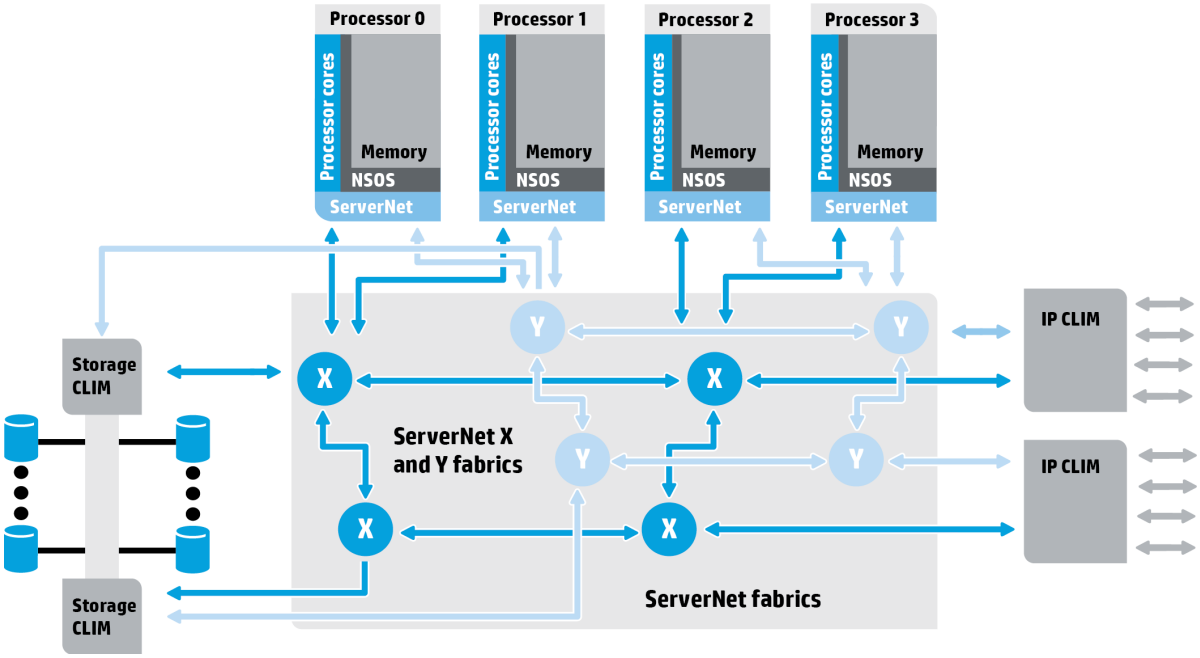


MPP architecture of NonStop

A single NonStop system is a cluster of two to 16 independent computing units, each consisting of a processor with one or more cores and its own memory, which is generally referred to as a logical processor. Note that, throughout the rest of the document, the terms 'processor' and 'logical processor' are used interchangeably. Each logical processor runs its own copy of the NonStop OS, including both the Guardian and OSS environments. All the logical processors in a system are interconnected via a high-speed switching fabric, called ServerNet, which is a forerunner of InfiniBand, which incorporated a number of its technical innovations. All input/output (I/O) peripherals (e.g., storage and communications) are also connected to ServerNet via I/O controllers called Cluster I/O Modules (CLIMs), and are accessible from all processors in a system.

Up to 255 independent NonStop systems can be interconnected via a proprietary layered networking protocol, called Expand, that can run over a variety of interconnects such as ServerNet or wide-area IP networks. In effect, a NonStop-based implementation can scale out to 4,080 (16*255) independent logical processors—a massive number that can handle most of today's demanding computing requirements.

Figure 2. MPP architecture of NonStop



Apart from being massively scalable, a NonStop system is also extremely fault tolerant. NonStop hardware is designed to survive any single-point failure. For example, ServerNet is not one fabric, but two, called 'X' and 'Y'. Both are used for bandwidth purposes, but a failure of one will still allow communication to continue. All CLIMs are attached to ServerNet via two connections, again allowing one to fail. Disks are configured as RAID1 mirrored pairs—so that there are two copies of all data—and are actually connected to two CLIMs. And so on.

Software

The program memory layout in NonStop is similar to that of most other operating systems. A process in NonStop represents the running instance of a program executable, called the "image." Multiple process instances can be run using the same image, but all those instances running on the same processor share the same copy of the code text. Each process will have separate data segments, but these are shareable voluntarily between processes running on the same processor. NonStop provides useful memory management utilities for runtime diagnosis and consistency checking.

The fundamentals of NonStop system design also form an excellent architecture for system security by providing a clear separation of functions, starting at the lowest levels of the system. The code segments in NonStop are read-only (i.e., cannot be changed) and no code can be executed out of data segments. Non-privileged (user) processes cannot run any code from data space, corrupt statically-compiled code space, or corrupt dynamically-generated code in a way that

escalates their privileges to kernel mode. These user processes also cannot access their privileged stacks, nor can they corrupt the data space of another process other than explicitly-shared segments.

A fundamental principle of the NonStop architecture is that the processes communicate via messages rather than through shared memory. This paradigm provides both fault isolation, preventing one process from corrupting another's state, and scalability.

Contrast this model to the traditional UNIX system, which uses the Symmetric Multi-Processing (SMP) architecture, where a server consists of multiple processors or cores all connected to the same memory and devices, and running a single instance of the operating system. Applications typically scale through the addition of more threads within a single process, which communicate via shared memory. Processes can also communicate via other IPC mechanisms such as sockets, queues, pipes, etc. However, a single misbehaving thread can cause an entire application to fail, or worse, corrupt data.

Note

NonStop servers running on today's multi-core systems do have SMP aspects, with multiple processing units (cores) and associated memory sharing a single OS instance. However, the SMP scale-up (current maximum of four cores) is modest compared to the scale-out across processors. Processes running in the same processor typically continue to communicate via (streamlined) IPC, but also can communicate via shared memory if they so choose.

The IPC mechanism in NSOS and the application programming interface (API) used to implement it, is identical whether the processes are in the same or different logical processors, thus allowing processes to run anywhere on a NonStop system. If a server process is named, the requester does not need to know the server's CPU or process ID. The message system also is used for communication over Expand between processes running in two different NonStop nodes, allowing a single application to scale out across multiple NonStop systems. (In an Expand network, each NonStop system connected to it is sometimes referred to as a "node.")

The NonStop system implements many critical services that must always be available, such as disk volume and file management, through a program designed to run as a NonStop "process pair" consisting of a primary and backup process. The entire pair of processes is assigned a unique logical name to abstract out process addressing information such as Process ID (PID), processor number, etc. A named process is always reachable via Inter Process Communication (IPC) through the NonStop message system, regardless of its actual location, and thereby provides a virtualized environment for the services it provides. Service requests are directed to the primary process, which checkpoints request state as needed to a backup process running on a different logical processor. In the event of a failure of the primary process—either directly or because of a processor failure—the backup will take over from the state last received in a checkpoint. Service requests will automatically be redirected to the backup process by the NonStop message system, typically without the requester even knowing this has occurred. This mechanism is extensively used by the NSOS itself to deliver fault tolerant OS services such as I/O, database, and transaction management. You may choose to write application-level process pairs as well, but that is not the normal paradigm.

All I/O devices on the NonStop server (e.g., storage, communication, etc.) are configured with their own assigned device handlers, which are also NSOS processes. As noted above, all communication to I/O devices goes via ServerNet to a CLIM and then to the physical device. This allows I/O processes to be spread out across the system and run in any of the logical processors to even out CPU activity. Because I/O devices are also represented by processes, these too can be accessed as if they were local, allowing files or even a SQL database to span multiple nodes that can act as one large NonStop system.

In line with this architecture, every disk on the system has its own named I/O process called the "disk process" (informally known as DP2). Each DP2 process will receive all file IO or database requests for its disk from all processes in the system or network. This design serializes disk IO to a disk volume across the cluster without needing cluster-wide locking mechanisms used in typical SMP systems. It also permits DP2 to provide special handling for mixed online transaction processing (OLTP) and batch/query workloads, as it has visibility into all outstanding requests for the volume.

Services that can be interrupted briefly can be implemented as OS-managed restartable (persistent) processes, which are restarted immediately by the system upon detection of a failure. This form of persistent process support is usually used for management-type processes. The persistence manager can manage both single processes and process pairs, and can manage individual instances of application-level processes if desired.

In order to enable applications to scale out across multiple processors in a node and/or across nodes while still providing a single interface to their clients, NonStop provides a middleware solution called "Transaction Services/Massively Parallel" (TS/MP). Using TS/MP you can configure an application program as a "server class," and run multiple instances of it on a

single processor, multiple processors, or multiple NonStop nodes connected by Expand. In essence, this results in a service-oriented architecture (SOA) implemented using a proprietary messaging protocol that is a highly-optimized IPC mechanism provided by the NSOS. TS/MP provides an infrastructure for scalability ideal for applications supporting extremely high levels of transaction processing and near continuous availability, and also provides server persistence by monitoring its server class members and restarting processes as needed.

The applications themselves can be largely platform-agnostic. The only NonStop-specific fault-tolerance knowledge that you might require as an application developer is delineation of transaction boundaries through a simple begin/commit/abort transaction API. If you write your servers in this fashion and an instance should die, the Transaction Management Facility (TMF) automatically backs out the transaction that was in progress and TS/MP takes care of the service persistence aspects by starting another instance as needed.

With this brief introduction to MPP, let us now take a closer look at OSS.

What is Open System Services?

Open System Services (OSS) is the development and runtime environment on NonStop that complies with X/Open Common Application Environment (CAE) specifications, incorporating the POSIX.1, POSIX.2, and some XPG4 standards. POSIX is an acronym for Portable Operating System Interface, which defines the API, shell commands, and utilities that UNIX computing systems should provide. The X/Open consortium (now a part of The Open Group) defined the X/Open Portability Guide (XPG) to enhance portability across UNIX operating systems. Refer to the POSIX—Austin Joint Working Group for more information.

Through its support for these standards, OSS provides a UNIX-like environment on NonStop and thereby provides developers with an open and standards-based environment to develop or port their applications onto the NonStop platform. OSS also provides applications with access to NonStop fundamentals such as fault tolerance, parallel processing, data integrity and reliability, distributed data, linear expandability, and security in order to truly exploit the power of NonStop.

The following sections provide a brief overview of important components of OSS. Please note that OSS is being continuously enhanced, along with other NonStop features, as a part of Release Version Updates (RVUs—NonStop terminology for periodic OS update releases). For the latest set of features and command sets supported in OSS, please see the "Open System Services Programmer's Guide" and "Open System Services User's Guide" at hp.com/go/nonstop-docs.

Command shell

OSS provides the "Korn shell" (ksh) or "Born again shell" (bash) as its command line interface. You can use all the built-in commands and utilities provided by the ksh or bash (e.g., `ls`, `cd`, `pwd`, etc.). These commands have the same syntax and semantics as in other UNIX systems. You can enter multiple commands on a single command line, link outputs and inputs of commands using pipes and filters, group commands and so on. You can set your default environment through .profile/.bashrc file, create command aliases, or write shell scripts. You can use POSIX utilities such as `date`, `compress`, `find`, `eval`, etc. and prepare command procedures using shell scripts such as `awk` or `perl` to automate common tasks. You can refer to main pages to learn more about commands, utilities, system, and library calls supported in OSS.

In short, you will find working with OSS to be very similar to working in a UNIX environment. However, the functions of some OSS commands have been extended from ordinary UNIX usage to allow interoperability with the NonStop operating system or to provide greater flexibility. For example, some commands have a cpu parameter through which you can specify the target processor for the outcome of the command (e.g., running a process).

File system

The OSS file system has a very similar set of features to UNIX file systems. Each user gets a home directory as the default work area for his or her account. You can create files, assign names, and organize them hierarchically into directories and subdirectories. Files and directories have UNIX-like naming rules, path specifications, and also provide similar file system services and APIs. Directories are grouped together for storage purposes and each group of directories is administered as an entity called the "fileset." While the OSS file content is stored natively as Enscribe (unstructured) files, the metadata management is done through a component called the "OSS name server." The OSS filesystem supports regular disk files and special files, such as directories, pipes, FIFOs, sockets, and terminal character files. OSS also provides a Network File System (NFS) server to enable access to files from remote clients.

For access control, OSS files have permission bits similar to a UNIX file system. However, you can provide a higher degree of control on access to OSS files through Access Control Lists (ACLs). ACLs allow a process whose effective user ID matches the file owner, SUPER ID ("root" equivalent in NSOS), or a member of the security administrator group to permit or deny access to a file to a list of specific users and groups. All OSS system calls that include pathnames are subject to ACLs that are present on any directory or file in the path. ACLs are supported as a superset of the UNIX operating system discretionary access control (DAC) for directories, regular files, first-in, first-out (FIFO) files, and AF_UNIX sockets but not for IPC objects. ACL-based restrictions are also applicable for file access from remote NFS clients. Additional security capabilities, configured on a fileset basis, make it possible to restrict root access to files or directories.

System services and application programming interfaces

OSS provides UNIX-like system call APIs for OS services in accordance with the POSIX.1 and POSIX.2 specifications. Some optional features and those that are outside of these specifications, but found in other commercial UNIX operating systems, may not be available in OSS. OSS does not support streams or memory mapped files. NonStop supports user-level threading through POSIX User Threads (the PUT library) but does not support kernel level threads. For more information on the threading model supported on OSS, please refer to the "Open System Services Programmer's Guide" at hp.com/go/nonstop-docs.

You can create, control, and terminate OSS processes as you would in a standard UNIX environment. You can fork a process, query its attributes such as its process ID, change its scheduling priority, and send/receive signals to and from processes. While creating a process, you can specify the target CPU in which to run the process. You can run multiple instances of the process either in the same CPU or separate ones using the same binary image.

OSS supports IPC mechanisms such as pipes, message queues, semaphores, shared memory, signals, and sockets. However, due to the 'shared nothing' architecture of NonStop, there are some important usage aspects of these IPCs that should be noted. For example, shared memory and semaphores can only be used between processes in the same processor, since there is no common memory address space across processors and each processor runs a separate OS instance. FIFOs and sockets are the only IPC mechanisms that can be used between processes running on different processors in the same node or on different nodes connected over Expand. All these services are available through APIs similar in syntax and semantics to the ones supported in UNIX.

System management (or 'system administration' in UNIX)

NonStop has a Guardian-based user management model that provides features such as user names, groups, password rules, file access permissions, etc. A product called Safeguard, which is bundled with NSOS, extends this model and provides services such as user aliases, advanced password rules, auditing, etc. OSS leverages this infrastructure and provides user management and security services that are similar to UNIX systems and, in some cases, more advanced. Hence OSS user and group administration does not use files or directories such as /etc/groups, /etc/passwd, etc. which you will find in a typical UNIX system.

OSS system administration has many UNIX-like features such as user accounts, aliases, file sharing groups, process access privileges, ACLs, and auditing. A user in NonStop is identified by a combination of user name and group name as *user.group*. Using Safeguard you can assign aliases to these user names that look similar to UNIX-style user names. The root super user of UNIX maps onto the Guardian *super.super* user. Users also have user IDs and group IDs similar to the ones in UNIX systems. For example, the user ID of root user is 65535, The super group, with an ID of 255, is analogous to the UNIX group name "wheel" with a UNIX group ID of 0. OSS supports most user management commands with a similar look and feel as UNIX. For example, you can use name aliases, as mentioned earlier, and use commands such as `useradd`, `userdel`, etc.

OSS supports file access permissions similar to UNIX file permissions, with a few NonStop-specific extensions to enable features such as enhanced data integrity. Three access modes are supported: read, write, and execute (rwx). Execute also has options for setting user and group IDs on execution. OSS supports ACLs to allow a more flexible and granular control of file access based on the POSIX 1003.1e draft standard.

Using Safeguard, OSS supports very granular controls over audit generation, including user and operation type. OSS also supports an authentication "Security Event Exit Procedure" (SEEP) to enable extensibility of user access management through external (or third-party) plug-ins. There are other differences between the two security models; refer to "Safeguard User Guide" and "Open System Services Management and Operations Guide" at hp.com/go/nonstop-docs for further information on this topic.

Programming languages and development tools

OSS supports the C, C++, Java, and COBOL programming languages and associated runtime libraries compliant to relevant standards.[1] In addition, these NonStop language compilers also provide features and routines to exploit the power of NonStop. The language compilers for NonStop can run on a Microsoft® Windows® PC (cross compilers) or on a NonStop server (host-based compilers).

The NonStop Server for Java (NSJ) product provides Java language support on NonStop servers based on Java Standard Edition (Java SE) version 7.0 headless Java.[2]

There is a lot of similarity in the set of development tools available on OSS and UNIX systems. You can use the vim editor to write code, create search paths for include files, use makefiles to define your build configuration and target executable, display the symbol table, run the program, debug using a GDB-like debugger, called Native Inspect, and many others.

As a part of the philosophy of providing modern, industry standard tools for NonStop developers, HP provides NSDEE, which is a PC-based integrated development environment (IDE) for NonStop. This runs as a plug-in into Eclipse , which is a widely popular IDE in the industry today. It provides several advantages, such as availability of a wide pool of developers with Eclipse skills, using a single IDE for multi-platform application development, availability of a large number of open source or third-party plug-ins, and so on.

Using NSDEE you can create and edit projects, build the project using cross compilers, and debug your code by deploying and launching your software onto a remote NonStop host over a secure connection, all from a Windows desktop. If you prefer to build your projects on remote NonStop hosts using your own makefile, you can do so from your NSDEE environment. Thus NSDEE gives you the capability to develop all your NonStop software from a commodity PC and choose to build your code either locally or remotely all through a rich graphical user interface (GUI)-based IDE.

Open source tools and utilities

One of the important aspects of OSS is its support for tools and utilities from the open source world, particularly the ones from GNU. Tools such as samba, bash, vim, make, and many others are now included in OSS. Developers with a Linux background will find NonStop development to be familiar because of these tools.

Newer open source packages have been added into the NonStop RVUs. The list of open source tools on NonStop is regularly expanded in order to help customers benefit from the work of the open source community, just as they are in other UNIX systems. Please refer to the "Open System Services User's Guide" at hp.com/go/nonstop-docs for the latest set of open source commands and utilities available on NonStop.

Apart from these, the NonStop community that is a part of the HP Connect forum has ported a large number of open source packages onto NonStop. They are shared for free download, on an as-is basis, in the ITUG library under the HP Connect portal to enable NonStop users to benefit from a large pool of popular open source software through a community process. At the time of this writing there are about 280 open source packages in the ITUG library. Visit connect-community.org for more information.

OSTU includes a package called the Freeware Look for OSS (FLOSS), which is a set of command-line utilities and libraries that aid in porting open source packages to NonStop. FLOSS provides the wrappers for commands and libraries commonly used in open source packages to map them to their NonStop equivalents, where required. FLOSS also provides helper

[1] The unified C and C++ compiler conforms to the C99 (ISO/IEC 9899:1999) and C++98 (ISO/IEC 14882:1998) standards. The COBOL compiler supports COBOL 85 (ANSI COBOL85 X3.23-1985) and select amendments until 1994. It also supports a few select features from COBOL 2002 (ISO/IEC 1989:2002).

[2] Note that Java on NonStop does not support Swing or Java FX.

utilities to help you verify the correctness of configuration and builds when these packages are built on NonStop. Hence if you would like to bring in your preferred open source tools onto NonStop, FLOSS makes it easier. For more information on FLOSS please refer to the "Floss on NonStop User Manual" at hp.com/go/nonstop-docs.

In summary, NonStop OSS provides open, modern, familiar, and industry standard development tools to aid software development on NonStop.

Database

The only supported database products on the NonStop server are NonStop SQL/MX and SQL/MP; there is no support for Oracle or other third-party databases. SQL/MX is the modern, ANSI SQL compliant database designed for a clustered, massively-parallel system architecture, NonStop SQL/MX is tightly integrated with the NonStop operating system.

The NonStop SQL/MX database can be accessed using Open Database Connectivity (ODBC) 3.x and Java Database Connectivity (JDBC) Type 4 interfaces from Microsoft Windows, Linux, HP-UX, and Solaris. In addition, the Microsoft ODBC .NET data provider enables .NET applications to access the NonStop SQL/MX database software. NonStop SQL/MX also comes with on-platform OSS ODBC/MX and JDBC Type 2 drivers.

Unlike a typical UNIX deployment, which separates application and database onto separate servers in a tiered architecture, SQL/MX and applications typically run on the same NonStop servers. NonStop applications use embedded SQL/MX statements and directives to access the database, which can be distributed and is accessible from any processor in the system, or across systems.

Java applications running on the NonStop platform can access the database through the standard Java Database Connectivity (JDBC) API using a Type-2 driver that runs in the JVM and uses the same NonStop IPC mechanism described earlier to communicate with DP2 and the database.

NonStop also supports the Hibernate ORM framework, which maps Java classes to SQL database tables, converts database calls to SQL queries, and handles the result sets. A dialect file that understands the object mapping to SQL/MX syntax is required, and can be downloaded from the HP NonStop eServices portal at h22204.www2.hp.com. The NonStop COBOL compiler conforms to the ANSI Database—Embedded SQL Standard (ANSI X3.168-1989), which helps you to develop COBOL applications that efficiently access the NonStop SQL Database.

Middleware

Modern applications are commonly developed using one or more middleware solutions for productivity gains and robust solution implementation. There is a rich set of such middleware frameworks that run on NonStop OSS.

As described earlier, the HP TS/MP product is one of the basic building blocks for applications written on NonStop. It allows a single application service to be instantiated as multiple instances of the same process, spread across single or multiple NonStop systems, and represented and invoked as a single logical name, providing automatic load balancing of requests. With these services, it delivers continuous availability, near linear scalability, and load distribution for mission-critical applications. The applications themselves are written as single-threaded servers and use the Transaction Management Facility (TMF) for database protection.

TMF, which is tightly integrated into the NonStop file system and SQL/MX, underpins the transactional applications and provides data integrity to ensure that all database updates for any given transaction are committed to the database, or rolled back in the event of a failure. TMF can also protect transactions distributed across multiple NonStop nodes connected through Expand using a two-phase commit capability, and in some circumstances can provide XA capabilities to protect a transaction distributed across other platforms.

For building a SOA-based solution around Web technologies, NonStop provides the HP iTP Secure Web Server and NonStop SOAP 4 products, both of which run in the OSS environment. The iTP Secure WebServer provides support for the HTTP(S) protocol and CGI-based interface for Web applications. NonStop SOAP 4 software implements the popular SOAP protocol for interconnecting disparate systems using Web services standards. Both iTP Secure Web Server and NonStop SOAP 4 are built on the NonStop fundamentals of availability, scalability, and data integrity. Therefore the applications built on top of them inherit these qualities out of the box.

NonStop also provides deep ports of popular Web application infrastructure products. NonStop Servlets for Java Server Pages (NSJSP) is the implementation of open source Apache Tomcat on NonStop. NSJSP implements both the servlets and

Java Server Pages specifications. NonStop Application Server for Java (NSASJ) brings the very popular open source JEE Application Server JBoss AS to NonStop. NSASJ provides the NonStop TMF and SQL/MX-based services through Java Transaction APIs and Java Persistence APIs, respectively. NonStop Message Queue is a port of Apache ActiveMQ onto NonStop and provides a messaging system that enables applications to communicate with one another.

In addition to the above, Java middleware and other open source Java solutions such as Hibernate ORM, Apache Axis2 framework, Spring framework, and Apache MyFaces are certified and supported on NonStop. These open source solutions have been widely adopted by Java developers and, in many cases, have become the *de facto* technology for developing enterprise Java applications.

Axis2 and Java development frameworks such as Spring and MyFaces are also available on NonStop. All of these products are integrated with iTP Secure WebServer for out-of-the-box scalability and availability.

HP also has a strong ecosystem of independent software vendor (ISV) partners who build their solutions on the NonStop platform. Solutions are available for various industry verticals such as finance, telecommunications, manufacturing, healthcare, retail, etc. Please visit the "Partners" tab at hp.com/go/nonstop to learn more.

Porting considerations

The earlier sections describe how NonStop OSS provides a UNIX-like environment on an MPP architecture. If you would like to port an application written for UNIX to NonStop, OSS greatly reduces the effort needed to port the code as long as it is written using the standard set of program constructs and system APIs. Third-party tool such as CodeCheck or Open Systems Portability Checker (OSPC) can be used to verify program source code for compliance with ISO/ANSI C, POSIX.1, and POSIX.2 standards and to estimate the extent of porting needed for NonStop. You may also want to look for any platform-specific dependencies in your application code such as data type sizes, data alignment, structure padding, byte ordering (NonStop uses big endian), and floating-point representations (NonStop supports IEEE float).

A more important consideration is that, as described earlier, NonStop is architecturally different from a typical UNIX system that is based on the SMP architecture. If you have a UNIX application that you would like to port to NonStop, you must first evaluate how its design maps to the NonStop's MPP architecture so that it can be made linearly scalable and highly available. If your application is monolithic and designed for high transaction load on a "scale-up" system, you should consider breaking down the application functionality into smaller subunits suitable for single process execution but replicated as multiple instances on different processors. Essential considerations are how discrete components communicate with each other and whether that communication can be decoupled across different processors, and whether state or context is maintained across client requests. Such a redesign helps make the best use of NonStop's MPP design, giving your application the benefit of the NonStop fundamentals of linear scalability and fault tolerance.

Because of the way in which some of the UNIX features are implemented in the OSS environment, there may be some performance implications to consider. For example, the cost (time and resources) to create a process is higher in the OSS environment than in most UNIX environments, and file creations and opens are more expensive. Applications should be designed to open all resources during initialization and keep them open. Keep in mind that applications on NonStop should be designed to be run for months or years with no downtime, either planned or unplanned.

Porting the database is another aspect that needs to be considered while creating your porting plan. The physical database design aspects of tables, indices, partitioning, etc. will differ in SQL/MX compared to databases such as Oracle or IBM DB2. Data types will often need to be changed and some SQL DML statements may also need changing. In addition, any use of stored procedures is going to involve rework, though HP has tools that can assist in this and other parts of the conversion. Refer to NonStop SQL Fundamentals and A Comparison of NonStop SQL/MX and Oracle 11g RAC as a start.

Conclusion

HP NonStop is the enterprise computing solution of choice for workloads that demand extremely high availability and near linear scalability. OSS provides an open, standards-based development and execution environment on NonStop. It is complemented by a rich set of solutions and services from HP and ISV partners, resulting in a platform that is ideal for software vendors for developing or migrating their applications for deployment into mission-critical environments.

Learn more at
hp.com/go/nonstop

Sign up for updates
hp.com/go/getupdated

Share with colleagues

Rate this document

4AA5-4239ENW, August 2014