Migrating From Tru64 UNIX to the Solaris™ Operating System

Ken Pepple, Brian Down, David Levy

Sun BluePrints™ OnLine—March 2005

Please
Recycle

Adobe PostScript

Migrating From Tru64 UNIX to the Solaris™ Operating System

This book presents an established methodology for transitioning the people, processes, and technologies in IT environments to the Solaris Operating System. It steps you through the various phases of the migration process, using detailed case studies to illustrate the benefits, costs, and requirements associated with a migration project. While this book focuses on UNIX server migrations, the methodology and best practices presented apply for most migrations to the Solaris environment. They can be used for projects ranging from the smallest data conversion to the largest legacy migration.

Using a fictional case study, this article illustrates the methodology, tools, and best practices used to migrate a Tru64 environment to the Solaris environment. In this study, we examine a simple, custom-written application that uses a Sybase database to store information about the company's inventory, as well as client-specific data. This application is converted to run under the Solaris Operating System (Solaris OS) and is integrated with directory services. Additionally, the database vendor is changed from Sybase to Oracle.

This article contains the following sections:

Overview of Tru64

Tru64 was the first commercially available UNIX environment that supported a 64-bit data model and computing environment. Originally released to support the Alpha hardware platform from the Digital Equipment Corporation (DEC), this operating environment and hardware combination formed a powerful computing environment that overcame the barriers associated with 32-bit computing.

The Tru64 kernel architecture is based on Carnegie-Mellon University's Mach V2.5 kernel design, with components from Berkeley Software Distribution (BSD) 4.3 and 4.4, UNIX System V, and other sources. Tru64 implements the Open Software Foundation (OSF) OSF/1 R1.0, R1.1, and R1.2 technology. The Tru64 UNIX operating system complies with numerous other standards and industry specifications, including the X/Open XPG4 and XTI, POSIX, and the System V Interface Definition (SVID). Additionally, Tru64 UNIX is compatible with Berkeley 4.3 and System V programming interfaces and conforms to the OSF application environment specifications (AES), the last that specifies an interface for developing portable applications.

Tru64 and the Solaris OS share a common ancestry. In addition, the development of standards and the willingness of vendors to faithfully implement them ensures that porting an application from Tru64 to the Solaris OS should not prove to be too daunting.

In the following sections, we examine the benefits and drawbacks of 64-bit computing and explore the data models supported by Tru64 and the Solaris OS. Then, we provide a little background on workarounds that were in place before the advent of 64-bit operating environments that permitted us to use files larger than 2 gigabytes.

64-Bit Computing

As the complexity and functionality of applications increase, their data sets and address space requirements increase as well. Many applications (databases, web caches, simulation and modeling software, and the like) run more effectively if they are not subject to the 4-gigabyte address space limitation imposed by a 32-bit architecture.

The ability to support larger amounts of primary memory allows the 64-bit architecture to afford performance benefits to a broad class of applications, including the following:

- A greater proportion of a database can be held in primary memory.
- Larger CAD/CAE models and simulations can fit in primary memory.
- Larger scientific computing problems can fit in primary memory.
- Web caches hold more data in memory, reducing latency.

The following are also compelling reasons for creating 64-bit applications:

- To improve performance by performing several computations on 64-bit integer quantities, using the wider data paths of a 64-bit processor

- To improve efficiency with arithmetic and logical operations on 64-bit quantities

- To enable operations to use full-register widths, the full-register set, and new instructions

- To improve efficiency with the improved parameter passing of 64-bit quantities

Not all applications are well suited to a 64-bit data model. If your application does not use or require any of the features listed above, it should probably be left as a 32-bit application. 32-bit applications usually run without problem in a 64-bit environment. 64-bit applications are usually larger in size. Additionally, the use of longer pointers in the 64-bit version could degrade performance because of cache misses. Doubling the size of pointers means that the cache cannot hold as many entries as om the 32-bit version of the application.

Understanding Differences Between 32-Bit and 64-Bit Data Models

Like Tru64, the Solaris OS supports the LP64 data model. In this convention, longs and pointers are 64 bits in length (hence the acronym LP64) and an integer is 32 bits in length. This is in contrast to the data model used in 32-bit environments, referred to as an ILP32, in which integers, longs, and pointers are all 32 bits in length (hence the acronym ILP32). The following table highlights the differences between these two models:

TABLE 1   Differences Between 32-Bit and 64-Bit Data Models

C Data Type	ILP32	LP64
char	8	unchanged
short	16	unchanged
int	32	unchanged
long	32	64
long long	64	unchanged
pointer	32	64
enum	32	unchanged
float	32	unchanged
double	64	unchanged
long double	128	unchanged

Most of the problems associated with migrating an application to 64 bits arise from the following differences:

- Long values (long) are 64 bits in length, not 32.
- Pointers are 64 bits in length, not 32.
- Integers (int) are not the same size as longs and pointers in a 64-bit environment.

Many of these size issues can be mitigated if the developer uses derived types. These definitions and others can be found in /usr/include/sys/types.h and /usr/include/sys/inttypes.h in both the Solaris and Tru64 environments. A derived type can specify the size of the attribute (for example, int32_t) or its intended use (for example, blksize_t), a capability that increases program clarity. The derived types themselves are safe for both ILP32 and LP64 environments, making them 32-bit and 64-bit safe.

All releases of the Solaris OS after—and including—the Solaris 7 OE support both 32-bit and 64-bit data models. Under most circumstances, binaries that were created under the 32-bit environment can be run in the 64-bit environment.

Using Large Files to Overcome 32-Bit Limitations

DEC was one of the first computer manufacturers to understand the limitations of a 32-bit environment as it related to file size. In the past, under a 32-bit environment, files could not exceed 2 gigabytes in length. Files are typically accessed relative to a pointer that points to a location within the file. Programmers can move this pointer by specifying an offset, but this offset is a signed quantity because the pointer can be moved forward and backward within the file. Consequently, the maximum offset that could be specified was 31 bits, limiting the file size to less than 2 gigabytes ($2{**}31 = 2$ gigabytes).

Vendors created a "large file" option that allows the offset variable to be a long quantity. This enables the file to grow well beyond the 2 or 4 gigabyte limit, because the offset variable used to move the pointer is much larger ($2{**}63$). This technology can be found in many legacy applications but is not necessary in a 64-bit computing environment like Solaris or Tru64.

—————

Clustering

In today's business environment, time really is money. In the financial markets, applications can be responsible for moving literally trillions of dollars a day. Any significant unavailability of such an application—or even an outage on a smaller system that is responsible for only billions or possibly millions of dollars of daily revenue—can have a significant impact on a company's bottom line. Consequently, highly available (HA) environments are required for certain applications. One strategy for creating an HA environment might include a cluster of computing devices and related storage.

In the following sections, we provide an overview of clustering technology and focus on TruCluster and Sun Cluster 3.0 software. Although our example does not involve the migration from one clustering technology to another, we discuss the technology here to provide background information that will be helpful if you encounter an opportunity to migrate from using TruCluster to using Sun Cluster software.

Overview

A *cluster* is a group of two or more computers (nodes) that share a common storage device and are connected in a way that allows them to operate as a single, continuously available system. Should an application on one of the computers, or the computer itself, fail, a companion machine in the cluster takes over to provide the same functionality as the failing computer. Whereas fault-tolerant hardware can provide near-continuous uptime by providing specialized proprietary hardware sharing the same memory, clustering technology provides highly available applications through the use of redundancy (redundant servers and redundant interconnects, networking, and storage, even redundant adapters and controllers). All of this redundancy allows work to continue if a hardware or software failure occurs, by transparently switching to a working component.

Clusters provide an enterprise a cost-effective and flexible method for deploying technology. Machines can be added or removed from a cluster as business demands vary. As newer technology becomes available, it can be added incrementally to the cluster, thereby reducing the need to perform a "forklift" upgrade. Clustering provides the following benefits:

- High availability
- Scalability in several directions
- Ease of use and administration
- Cost-effective, incremental growth path

TruCluster software was a pioneering version of cluster technology. Simple to configure and highly reliable, this framework allowed the deployment of campus clusters as well as machines located over great distances.

Sun Cluster 3.0 software is a scalable and flexible solution that is equally suited for a small local cluster or larger extended clusters.

Cluster Agents—TruCluster and Sun Cluster 3.0 Software

The ability to detect when an application or resource is no longer operating as it was designed to is an integral component of any cluster. When the cluster detects these types of changes, the application can be restarted or moved to a different node.

In the TruCluster environment, this functionality is provided by the Cluster Application Availability (CAA) subsystem. This facility provides a way for the environment applications to determine whether they are operating properly, and it allows administrators to specify what actions should be taken if problems are detected.

Sun Cluster 3.0 software supports a similar framework that enables IT staff to develop a customized agent that can be used to monitor the health of the clustered application.

Although these subsystems provide similar functionality, they have significantly different implementations. Porting an application from one clustered environment to another requires you to not only transform the application source code to adhere to the new OS APIs, but also that you integrate the application into the high availability framework of that cluster so that application failure can be detected. The framework will also need to be programmed to specify the actions that should take place if an application fails.

Justifying the Migration

The first stage in any migration project involves justifying the migration. In this particular case study, a manufacturing company has a number of systems deployed. Although the company has numerous Sun platforms in place, its inventory application and the associated database run under the Tru64 OS. The custom-written application was written in the C programming language and uses a Sybase database to store inventory data. This database uses third-party tools to manage the database and produce reports.

The following figure shows a simplified overview of the application and the supporting environment.
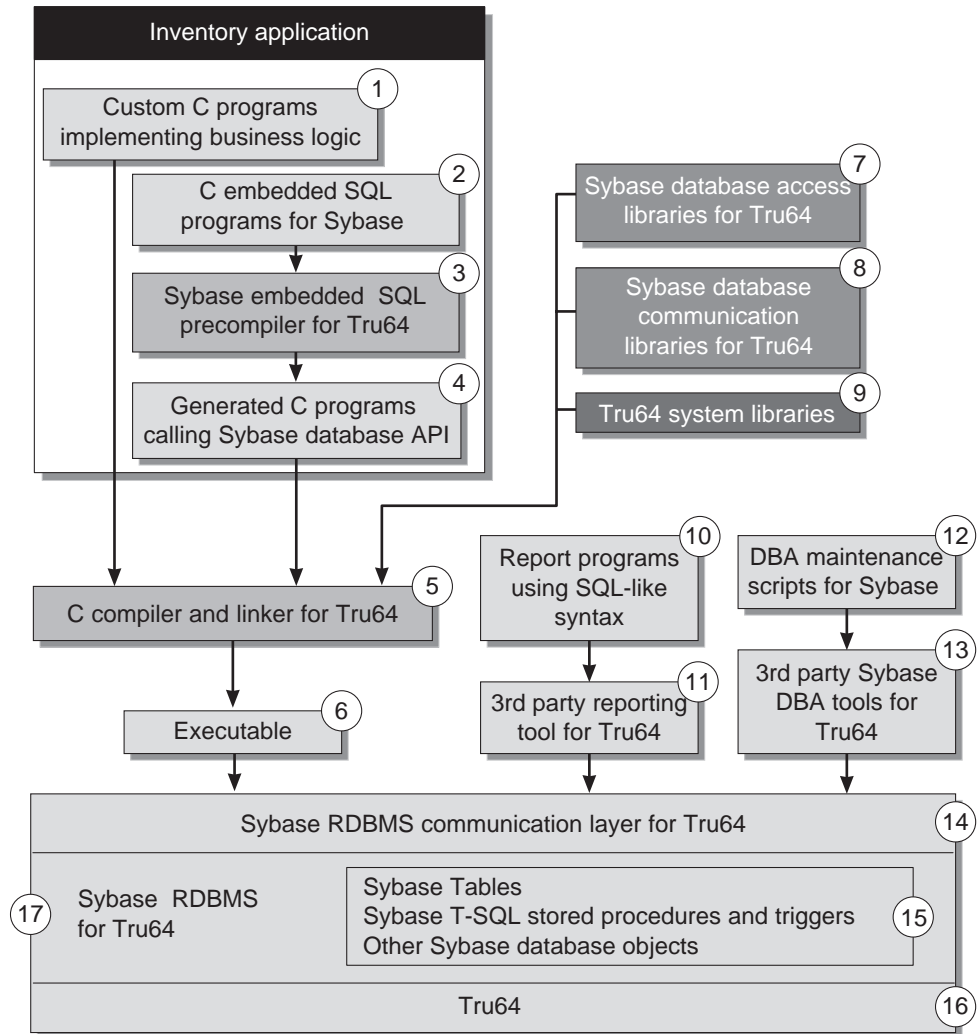
FIGURE 1 Overview of Application and Supporting Environment

Identifying Migration Motivators

The platform supporting this application is running out of capacity. The enterprise must decide whether it wants to purchase another Alpha server to provide the required capability, or to migrate the application to another vendor's platform that has the capacity to support the enterprise requirements. Two issues suggest that migration to a different platform would be the preferred choice:

- **End of life (EOL) of the Alpha processor.** Industry consolidation has led to the acquisition of DEC by Compaq Computer Corporation. This consolidation resulted in the announcement of the EOL of the Alpha processor after its manufacturing was turned over to Intel.

- **Changes to the Tru64 roadmap.** After Compaq acquired DEC, they were in turn acquired by Hewlett Packard (HP). HP already has its own version of the UNIX OS, HP/UX. The new HP/COMPAQ entity has stated that they will be phasing out Tru64 and consolidating on the HP/UX version of UNIX. To compound the problem, HP has also elected to use the Itanium processor as the basis for its new platform. If the client migrated to HP/UX, they would need to migrate again when the Itanium platform is introduced.

If the enterprise chooses to migrate its inventory application to Sun's Solaris environment, it can leverage its existing Oracle licensing agreement and can reduce expenses by switching their database from Sybase to Oracle.

You will explore these issues during a one-day meeting with all stakeholders from the enterprise. By performing that due diligence, you will gain a better understanding of the drivers and end goals of the migration.

Identifying Migration Strategies

In this case, the benefits of the migration are well understood. The application provides the business functionality that the enterprise requires. It does not want to move to a COTS application. Because the enterprise's problems relate to IT effectiveness, the system has run out of cycles, and the platform/environment product line they are using has a limited life expectancy, the recommended solution is to rehost the application. The benefits of the migration will be that the IT effectiveness of the platform will be improved and the required capacity will be achieved.

In other situations, the drivers might not be as obvious as they are in this case study. If poor total cost of ownership (TCO) or return on investment (ROI) was the driver, and the goal was to improve TCO or ROI, a more detailed investigation might need to be completed to determine whether rehosting is the correct migration solution.

The result of the meeting is an agreement to proceed with a more detailed assessment of the application environment to determine the associated migration costs. The executive responsible for this initiative then releases a mission statement and the detailed assessment begins.

Architecting the Migration

The first phase in the SunTone migration methodology involves architecting the solution. At this stage, you assess the existing environment and design a first-cut architecture.

Assessing the Current Environment

The next step in the migration is the assessment of the existing application and the associated environment. This will allow you to create a risk list that can be used to identify any areas of the project that might require a proof of concept to ensure that the project can be completed. The outcome of the assessment is a risk list (where appropriate) and a work breakdown structure that details the amount of effort required to migrate the application and the associated environment. This work breakdown structure is then used to create a plan and schedule various activities, overlapping independent subtasks, where appropriate.

For custom-written applications, provide the migration team with a snapshot of the application source and associated infrastructure to serve as a baseline for the migration activity. When possible, you should also acquire a build log for the application. This log will provide the following information:

■ Tools used
■ Options provided to these tools
■ Source that is compiled
■ Libraries that are linked
■ Order in which symbols are resolved

Although development documentation is welcome, a simple build log can serve as a guide to the "facts on the ground." It will show how the application is actually built.

In the following sections, we explore the assessment process.

Assessing the Application Infrastructure

Scripts provide an easy way for IT staff or administrators to create tools to administer an application, analyze or modify data, and provide functional support for an application. Scripts can leverage utilities that exist elsewhere in the operating environment to perform various administrative tasks. In addition, scripts will identify which utilities are used, as well as the options that are specified.

When the application is migrated, the associated scripts must be migrated to the new environment as well. Although the script tool (for example, Ksh, bash, sh, csh, PERL, or Python) might support the same syntax in the new environment, the location of the programs or files used by the script might be different in the new environment. Additionally, the options of the programs called by the scripts might also require modification.

Ensure that a version of the script tool is available in the new environment.

Analyze Scripts

The Perl utility is becoming popular as a scripting tool because of its power and flexibility. However, the venerable shell is still the script tool of choice for most developers, primarily because of its availability across a variety of platforms and environments.

When assessing shell scripts, check each command for the following conditions:

- Command is unavailable on the Solaris OS.
- Command is in a different location and the location is not in the user's path.
- Command uses a flag that does not exist on the Solaris OS.
- Command uses a flag that has different functionality on the Solaris OS.
- Output of a command is different and is redirected.

This check can be done manually or through the use of the `scriptran` tool.

The following sample presents the analysis of the issues associated with the shell scripting used with the Tru64 example.

```
alias         |       27
ar            |      365
cc            |       86
colrm         |       1
df            |       14
du            |       1
e             |       2
ed            |       1
egrep         |       68
expr          |       11
fold          |       1
get           |       1
iostat        |       1
ipcs          |       45
ld            |       13
lex           |       4
ln            |      177
lpr           |       2
make          |       1
mcopy         |       1
more          |       12
mt            |       3
netstat       |       5
printenv      |       5
sleep         |       94
stty          |       1
style         |      219
tail          |       61
tset          |       1
vmstat        |       26
w             |       5
wait          |       14
whoami        |       2
xconsole      |       3
xhost         |       4
xlsclients    |       1
xset          |       14
xsetroot      |       4
xterm         |       1
yacc          |       4
Total: 40     |      1301
```

Analyze Build Tools

When working with a custom application, you also need to migrate the tools used to build the application executable. These usually include a compiler, a source code management system, and the build environment used to create the executable. Additionally, any third-party products that were used to build the application must be migrated.

Obtaining a build log created when the application was last built is the best way to ensure that the build process and the tools involved in that process are identified. Be certain that you understand the semantics of the options that were specified when the application was built. Although tools in the new environment will most likely support the required functionality, different options might need to be specified to invoke the desired behavior. For example, static linking, position-independent code, extended symbol table information, and the like might require the use of new and different options.

In this example, the assessment reveals that a number of development tools are currently available on another Sun platform within the enterprise. Although this development environment has not been used to create the Tru64 version of the application that you want to port, you can leverage some of the existing tools that are available (for example, compilers and debuggers). Assume that you have determined that this platform can be used for the migration exercise.

Determine Third-Party Products Usage

While all applications depend on support from the operating environment and associated utilities, many applications are also designed to work with the functionality provided by third-party products that are integrated into the execution architecture. When the application is ported, this supporting software must be ported as well, as part of the application infrastructure. In the example, the most significant piece of third-party software is the Sybase database that is implemented in the Tru64 environment. However, additional third-party software is used to generate reports and administrate the database.

When assessing third-party products, you must ensure that these or similar products are available for both the new OS and the new database.

This migration case study involves the conversion of a Sybase database implemented on the Tru64 platform to an Oracle database running on the Solaris platform. FIGURE 1 on page 8 provides an overview of the Sybase implementation.

When attempting to assess the database component of the application, be sure to assess the deployment of database technology, not just the database itself. Databases have evolved to become much more than simple repositories for data. Complex logic can be programmed into the database. Database vendors encourage developers and database administrators (DBAs) to store database-related (or data-intensive) logic inside the database. The program units that are locally stored in databases are often called stored procedures and triggers.

The practice of storing program logic in the database aids in the assessment because the majority of the database-related logic is centralized in a single location, although some interaction with the database will be specified in the programs themselves. For DBAs who are concerned about database performance, storing program logic in the database is encouraged as well, because logic that is locally stored in the database has many positive performance implications. These stored program units are written in a language that is commonly known as the Structured Query Language (SQL).

Regrettably, although there is an SQL standard, the degree of compliance with this standard varies greatly from one database vendor to another. Different database vendors might develop their own extensions to the SQL language to make it more powerful and easier to use and, in some cases, to address specific database performance issues through optimization.

The assessment of the database technology must address the stored procedures as well as database object behavior. Among the different databases, database objects (box 16 in FIGURE 1 on page 8) that have the same name behave differently. For example, database objects such as stored procedures, triggers, and temporary tables are supported in both Sybase and Oracle. However, there are no standards for the behavior of these objects. Consequently, procedures, triggers, and temporary tables stored in Sybase behave differently from those stored in Oracle. These differences must be well understood before you can accurately assess the amount of change and effort that will be required in a migration.

Take extra care when migrating application logic from one version of SQL to another. In this example, translating a full-blown Sybase T-SQL application to Oracle's PL/SQL could result in an extensive modification or a total rewrite. You must carefully identify the use of language features that might require the reimplementation of logic on the new deployment, because the SQL extensions and their underlying functionality might not be available. For this reason, the conversion of the Sybase implementation will be considered a reengineering or rearchitecture effort.

When assessing the database technology integration with the application, be aware that each database vendor has its own version of SQL and that these versions can vary considerably. Understanding the differences in SQL implementations will help you understand the nature and amount of work that is needed for a project of this nature.

In addition to the Sybase database technology, our example makes use of third-party reporting tools (box 11), and DBA tools (box 13). If the tool vendor supports both source and target databases and platforms, these can most likely be replaced. If a tool cannot be replaced for any reason, then all of the components that use it will most likely need to be rewritten. To keep the example simple, assume that you can replace all the third-party tools and libraries.

In the example, all the components that use SQL will be affected in the same manner. These components are:

- **Stored procedures and triggers (box 15).** These are pure native SQL and are discussed above.

- **C programs that use embedded SQL (box 2).** Embedded SQL allows developers to directly use SQL statements inside a programming language with which they are familiar. In our example, the SQL statements are embedded inside C programs. These embedded SQL programs are then passed to a precompiler (box 3). The precompiler converts the embedded SQL to statements that directly call the native database API (box 4). The output is a generated C program that is then passed to the C compiler and linker.

- **Report programs that use third-party reporting tools (box 11).** Note that, for this scenario, it is not enough to replace the reporting tool. Report programs that use third-party tools usually issue SQL or SQL-like syntax (possibly allowing database vendor SQL language extensions), so they will need to be modified or rewritten.

- **DBA maintenance scripts (box 13).** The database engine (box 17) stores data in objects called tables (box 16). The type of data that is going to be stored is defined at the table level by data types that are native to the database engine being used. When changing database engines, one of the first tasks is to determine whether all of the data types used by the source database can be successfully mapped to data types in the target database.

Problems arise when the data types that are used in the source database cannot be mapped to the target database. If a data type cannot be mapped, you must find a way to mimic its functionality in the target database. This simple data type issue could potentially trigger a chain reaction of changes that need to be made to all components that reference the table. The extent of modifications will depend on the nature of the data type in question and how extensively it is used by all the components that are using the database.

In our example, all data types map from the Sybase implementation to the Oracle implementation without difficulty.

Assess the Application

As detailed previously, you must acquire the code for the application. That code will help you estimate how much effort will be required for the migration. There are two issues to consider when assessing an application:

- **Understanding the composition of the code used by the application.** Many legacy applications have significant size (for example, millions of lines of code). Simply trying to understand the layout of the source tree and the types of files can be a complex task.

- **Understanding which files within the source distribution are actually used to build the application.** As an application evolves, business functionality might no longer be required and new functionality can be added. Although this can be reflected when the application is built, developers seldom remove the old, unused code from the source code directory. Avoid transforming code that isn't being used.

The following appsurvey output represents the composition of the files under the source code repository of the `inventory` application.

```
Module | FileType   | # Lines | # of Files | # API issues

invtry | .4         | 127     | 1          | 0
invtry | .C         | 429661  | 605        | 44
invtry | .H         | 24570   | 216        | 9
invtry | .Make_files| 20174   | 126        | 0
invtry | .Msg       | 6572    | 24         | 0
invtry | .acf       | 1916    | 86         | 0
invtry | .bak       | 430     | 8          | 0
invtry | .bld       | 1914    | 6          | 0
invtry | .c         | 656575  | 415        | 14
invtry | .cat       | 25      | 1          | 0
invtry | .cfg       | 131     | 11         | 0
invtry | .cl        | 5070    | 34         | 0
invtry | .cpp       | 6017    | 2          | 0
invtry | .ctl       | 27908   | 54         | 0
invtry | .dat       | 20684   | 11         | 0
invtry | .def       | 81      | 1          | 0
invtry | .h         | 116618  | 356        | 1
invtry | .sh        | 2790    | 6          | 0
invtry | .sql       | 301904  | 699        | 0
invtry | .test      | 133     | 1          | 0
invtry | .tidl      | 4780    | 51         | 0
invtry | .tmp       | 453     | 1          | 0
invtry | .tpl       | 8169    | 36         | 0
invtry | .wpm       | 162     | 1          | 0
invtry | .zip       | 146     | 2          | 0
TOTAL  |            | 2672376 | 4066       |68
```

Remember, it is possible that not all of these files will be used to create the application. An analysis of the build log will reveal which files are used when the application is created.

In this example, you are considering a custom application written in the C programming language. When implementing this sort of migration, focus on the differences between the APIs provided by the Tru64 environment and those provided by the Solaris OS. The following sample breaks down the APIs differences.

```
Total Files: 3717        LinesOfCode: 1185289        Statements: 388262
Issues:
accept          4        Weight: 5
acosd           2        Weight: 5
asind           4        Weight: 5
atand           5        Weight: 5
bind            33       Weight: 5
bind_to_cpu     1        Weight: 25
connect         5        Weight: 5
cosd            15       Weight: 5
endhostent      2        Weight: 5
exp             1        Weight: 5
fork            28       Weight: 3
freopen         6        Weight: 5
fseek           28       Weight: 5
gethostbyaddr   4        Weight: 5
gethostent      1        Weight: 25
getsockname     2        Weight: 5
getsockopt      15       Weight: 25
getsysinfo      6        Weight: 200
gettimeofday    90       Weight: 5
getuid          1        Weight: 3
htonl           44       Weight: 5
htons           63       Weight: 5
inet_addr       18       Weight: 3
inet_lnaof      1        Weight: 5
inet_netof      1        Weight: 5
inet_network    1        Weight: 3
inet_ntoa       14       Weight: 3
ioctl           104      Weight: 25
kill            26       Weight: 5
listen          4        Weight: 5
log             2        Weight: 5
min             9        Weight: 25
mq_setattr      1        Weight: 5
msgctl          2        Weight: 5
msgrcv          31       Weight: 5
munmap          3        Weight: 5
nint            9        Weight: 5
nintf           4        Weight: 5
ntohl           14       Weight: 25

(continued on next page)
```

(continued from preceding page)

```
ntohs                    15       Weight: 25
open                     6        Weight: 25
opendir                  13       Weight: 5
pfopen                   1        Weight: 200
pow                      22       Weight: 5
pthread_cleanup_pop      3        Weight: 5
pthread_cleanup_push     3        Weight: 5
pthread_delay_np         21       Weight: 25
pthread_get_expiration_np 13      Weight: 25
pthread_lock_global_np   125      Weight: 25
pthread_unlock_global_np 128      Weight: 25
recv                     14       Weight: 5
recvfrom                 12       Weight: 5
remainder                1        Weight: 5
sched_getscheduler       1        Weight: 5
semctl                   4        Weight: 5
semget                   1        Weight: 3
semop                    8        Weight: 3
send                     11       Weight: 5
sendto                   8        Weight: 5
sethostent               1        Weight: 5
setsid                   3        Weight: 3
setsockopt               20       Weight: 25
setsysinfo               1        Weight: 200
settimeofday             4        Weight: 5
shmat                    7        Weight: 3
shmctl                   10       Weight: 3
shmdt                    5        Weight: 3
shmget                   7        Weight: 3
sigaction                5        Weight: 25
sigwait                  5        Weight: 25
sind                     13       Weight: 5
socket                   55       Weight: 5
sqrt                     118      Weight: 5
statvfs                  2        Weight: 3
strftime                 49       Weight: 5
system                   2        Weight: 5
table                    3        Weight: 200
tand                     3        Weight: 5
template                 1        Weight: 3
times                    2        Weight: 3
ulimit                   8        Weight: 25
uswitch                  2        Weight: 200
wait                     40       Weight: 3
waitpid                  2        Weight: 3
write                    2        Weight: 5
```

Assess the Compute and Storage Platform

In the example, the capacity of the existing hardware platform is determined. Based on this information, a replacement platform is chosen from the Sun product line that will provide the required performance, reliability, scalability, and manageability. The details of hardware sizing are outside the scope of this document.

Assess the Network Infrastructure

Next, examine the networking facilities inside the enterprise's data center to determine whether they can support the required future capacity and load generated by the migrated environment. Where appropriate, additional capacity might need to be acquired (10BASE-T to 100BASE-T). All aspects of the network must be considered, from the transport technology (FDDI, Token Ring, Ethernet, and the like) to the number of ports that are available on the switch or hub that will be used to cable the Network Interface Card (NIC).

Once you determine the networking technology, you can order the correct NIC for the hardware described above.

In the example, the 100-megabyte network has sufficient capacity, and a port is available on the switch serving the data center. A 100BASE-T NIC is required for the platform, as well as a 10-meter cable to make the connection.

Assess Facilities

During the next part of the assessment, you assess the facilities and any changes that will be required to support the migrated solution. During this assessment, consider power, space, network connections, door frame size, and similar requirements.

In the example, the new platform is roughly the same in size as the older platform. As a result, it can fit through all the doorways. However, it will need to be installed in a previously unused corner of the data center because the old machine will not be retired for some time.

The newer Sun hardware in this example requires more power but produces less heat than the older platform. However, a new electrical receptacle will be required for compatibility with the new hardware. In this case, the client decides to re-route a cable run for cabling efficiencies with existing machines and to bring power to the new location.

Assess Management Tools

Next, you assess the existing management tools and determine how they can be moved to the target platform. In this case, the client uses BMC Patrol to monitor the old Tru64 environment. This product is also available for the Solaris environment and has already been deployed on other Sun platforms within the data center. Additional ad hoc system monitoring is performed using the `cron` utility, to schedule scripts that use conventional UNIX utilities such as `iostat`, `vmstat`, `df`, and the like.

Assess People and Process

The skills of the organization must be assessed to determine whether any gaps exist. A curriculum is then developed to address any shortfalls. In our example, the IT staff already supports a number of Sun/Solaris/Oracle environments, which means that no additional training should be required.

Understanding Threading Models

Applications use threads to implement fine-grained parallelism. Thread libraries have been created for most modern operating environments. The most common threading implementations are POSIX threads and Solaris threads, which have similar semantics. The Solaris OS supports both threading models.

DEC's implementation of threads differs slightly from these implementations. In the example, you would use a compatibility library to replace threading APIs that are found in the Tru64 environment, but not found in the Solaris environment.

Implementing the Migration to the Solaris Environment

During the previous phase of the project, you produced a work breakdown structure that was used to develop a plan for this phase of the project. Implementation activities can be broken into tasks that relate to hardware and tasks that relate to software. Where appropriate, independent tasks can be overlapped in the schedule, depending on the required completion date, budget, and skill sets available. In the following sections, we examine the tasks involved with implementing the solution in a Solaris environment.

Modifying the Facilities

Typically, the facilities must be modified before hardware is installed. Depending on the modifications required, considerable lead time might be required. In the example, you need to reroute a power cable, install the appropriate receptacle, and clear space for the new hardware. These activities are coordinated to reduce their impact on the existing environment.

Creating the Networking Infrastructure

This task involves preparing the network for the addition of the new platform. Here, you make decisions about IP addresses, routing, network masks, and so on. If warranted, new load balancers, switches, hubs, cable drops, and the like will be deployed and tested. Care must be taken to minimize the impact of these activities on the existing environment.

In the example, the networking infrastructure requires minimal change because a cable must be routed from the switch to the machine location. However, if routers or load balancers were required, significantly more time and effort would go into this task.

Deploying the Compute and Storage Platforms

Because this activity requires specialized skills, it is typically performed by the service organization of hardware vendors (compute and storage). Before installing platforms, the supporting infrastructure (for example, facilities and networking) should be in place. If the compute platform and storage platforms are provided by different vendors, their activities must be coordinated.

In the example, the compute hardware and storage platforms are provided by the same vendor. Once the power and networking are in place, you can arrange for the equipment to be delivered and installed.

Implementing the Application Infrastructure

When the development environment is in place, you can begin transforming the application source code and any of the third-party scripts that support the application.

With the production hardware platform in place, you can begin creating the application infrastructure. The activities in this task include:

- Installing third-party products used at runtime. These can include the database and any tools or scripts used to administer the database or to produce reports.
- Installing the modified scripts that manage the application.
- Configuring the platform to support the application.

Implement the Build Environment

The analysis of the build log identified the tools and utilities that were used to create the application executable. Where possible, you should acquire and install the same tools. In certain cases (most likely, the compiler), you might need to acquire a different product with similar functionality.

When installing these build tools, examine the old build log to determine where the tools were located in the old environment. Putting your tools in the same location will minimize the changes that need to be made to the make files.

Modify the make files to use the new tools, utilities, and libraries. Translate the tool options, when required, such that they provide the same functionality. Our usual and preferred methodology is to port and redesign the entire build environment before the application source is modified. If the build environment is well designed, only a few key make files and setup scripts need to be changed. The general approach is as follows:

1. Understand the key files that affect the whole build system, and port those first.

2. Do a global search for hard-coded values in make files, and change them so they benefit from the new design if applicable.

3. Port the disconnected hard-coded instances, if any are left.

4. Release the build environment for code porting work.

These steps might need to be performed on a per-module basis because of the project schedule, code availability, and resources.

Distributing this work across a development team can create efficiencies in the project schedule, but if a large number of code-porting specialists make changes to the make files in the module they are porting, then the make files can become inconsistent with hard-coded values that might conflict with each other.

Translate Scripts

During the assessment process, a number of issues were identified that made the shell scripts that were originally developed for a Tru64 environment incompatible with the Solaris environment. These differences were usually related to the location or options that are used by the shell script. The following example presents the types of changes that will need to be made.

```
#! /bin/ksh
echo "Generated on `date`" >> $longreport
echo "Hostname : ", `hostname` >> $longreport

# __sun: change tr 'a-z' 'A-Z' to tr '[a-z]' '[A-Z]'
# NEW_TARGET=`echo ${TARGET_NAME} | tr 'a-z' 'A-Z'`
NEW_TARGET=`echo ${TARGET_NAME} | tr '[a-z]' '[A-Z]'`

echo "New target: ${NEW_TARGET}" >> $longreport

echo "Current environment" >> $longreport
# __sun: add path
#printenv >> $longreport
/usr/ucb/printenv >> $longreport

echo "Print who's logged in"  >> $longreport

#__sun: 'f' is 'finger' on Solaris
#f \@`hostname`  >> $longreport
finger \@`hostname`  >> $longreport

echo "Check if $filename is a link?" >> $longreport

#__sun: change -h to -L
#if [ -h $filename ]; then
if [ -L $filename ]; then
echo "$filename is a link" >> $longreport
fi

echo "Extract all names from $filename " >> $longreport

#__sun: add path because of option -F
grep -F "^NAME^" $filename >> $longreport
/usr/xpg4/bin/grep -F "^NAME^" $filename >> $longreport
echo "Extract all tasks from $filename " >> $longreport
grep "TASK:" $filename >> $longreport

echo "Extract all TOTALs from $filename " >> $longreport
grep "^TOTAL:" $filename >> $longreport

#__sun: change -w to -m. Send mail on completion.
#lp -w -d ${laser_printer} $longreport
lp -m -d ${laser_printer} $longreport
mv $longreport $LOG/${longreport}.old
exit
```

Go through each shell script and make the appropriate changes. As you can see from the preceding example, comments should be inserted to provide history and context.

Integrate Databases

As with shell scripts, the application might also depend on third-party products for support. In the example, the only third-party products involved with the migration were related to the database technology, so you will be replacing a Sybase implementation with Oracle technology, as shown in the following figure.
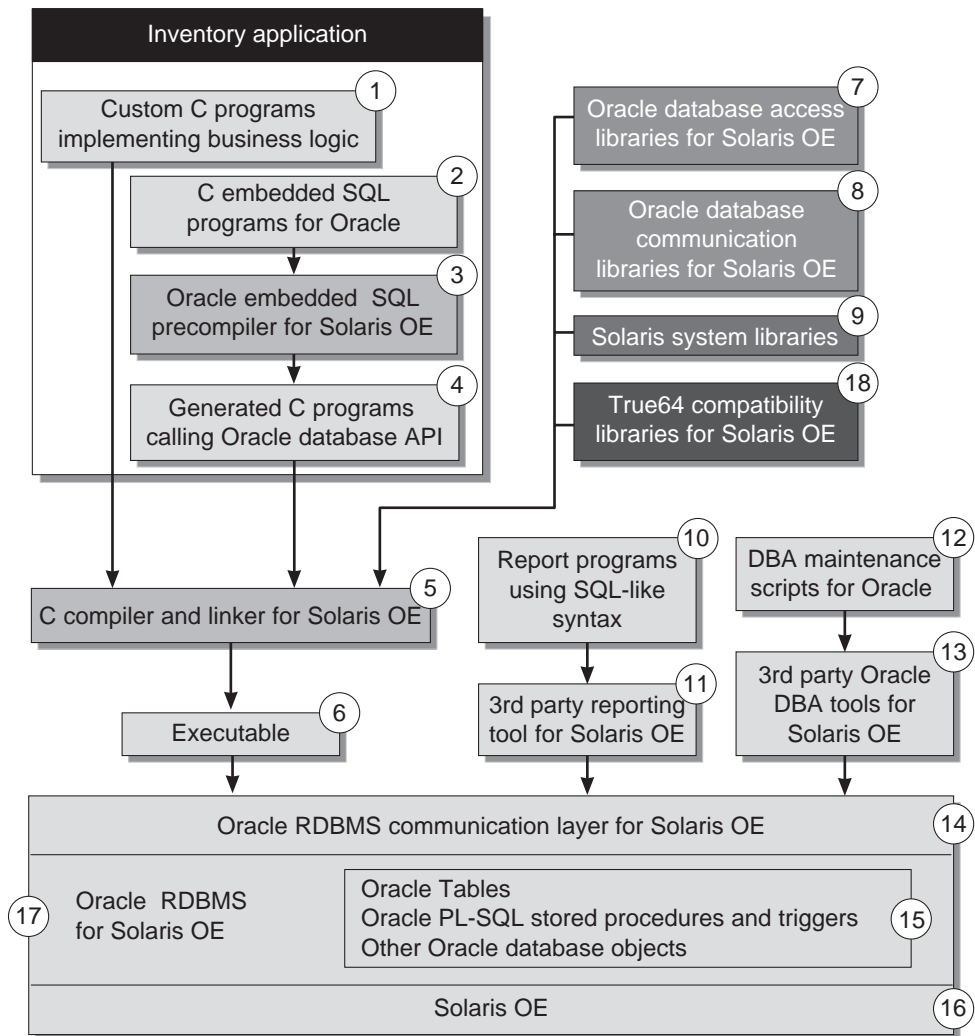

FIGURE 2   Replacing Sybase With Oracle

This assessment indicated that you should address issues associated with the following components:

- Stored procedures and triggers (box 15).
- C programs that use embedded SQL (box 2).
- Report programs that use third-party reporting tools (box 11).
- DBA maintenance scripts (box 12).

As illustrated in the preceding figure, the replacement environment is almost a one-to-one mapping of component technology. When implementing the Oracle environment on the Sun platform, you must acquire and install the appropriate products with their respective licenses as follows:

- The database communication layer (box 14). This is supplied by the database vendor and usually consists of database client and server libraries.

- The embedded SQL precompiler (box 3). This is supplied by the database vendor, in this case, Oracle.

- The C compiler and linker (box 5). This is usually supplied by the hardware vendor, in this case, Sun.

- The database engine (box 17). This is supplied by the database vendor, Oracle.

Other third-party products you should acquire and install include database reporting tools (box 11) and database management tools (box 13). Versions of these tools exist for the Solaris platform, reducing the amount of change that must be introduced. However, changes will most likely need to be made to the products if they issue SQL statements. These products might need to be rewritten altogether. Additional configuration of these products might also need to be introduced, reflecting different environment variables and path names.

Any API changes introduced in the database technology will need to be reflected in the source code of the applications. At this stage, you are only addressing issues with the database components of the application. The outcome of this stage of the process is new embedded SQL programs that will be compiled with the application source after it has been modified to conform to the new operating environment. These changes are described later in this article.

Data Extraction, Transformation, and Loading (ETL)

After you install the supporting database environment, you can create the database objects to accept the data. You can then extract the data from the old Sybase system by using vendor-provided utilities. Finally, you can load the data into the Oracle database, using utilities that are provided by the vendor.

There are times when database vendor-provided tools can be used. This is usually the case when the database structure is simple and data types can be mapped one-to-one. In such cases, the bulk copy (`bcp`) utility from Sybase can be used to extract data from Sybase. The output of this command can be an ASCII delimited file. You can then feed this file to an Oracle utility called SQL*Loader.

For more complicated scenarios, the use of third-party extraction, transformation, and loading (ETL) tools—such as Hummingbird's ETL or Embarcadero's DT/Studio—might be appropriate. Of course, you can also write scripts to perform these tasks.

Although our example requires no data translation, data types that existed in the Sybase implementation that cannot be reproduced in the new Oracle implementation might require that the data be transformed or translated to conform the new data type. Depending on the data type in question and the extent to which it is used throughout the application, this simple change could create significant complexity when attempting to modify the application source code. All references to that data type might require change. In certain cases, depending on the change to the data type, the application logic might also need to change.

Transforming Source Code

In the assessment process, you identified a number of APIs that were incompatible with the target Solaris environment. Rather than modifying the source code in-line to effect these changes, you should create a compatibility library to implement any changes that need to be implemented to rectify incompatibility issues. You can limit source code modifications to conditional compilation directives that are used to ensure backward compatibility.

The following example shows how to use conditional compilation to ensure backward compatibility.

```
#ifdef _sun
    SunVersion();
#else
    OriginalVersion();
#endif
}
```

The function `SunVersion()` that will emulate Tru64 functionality will exist in a compatibility library.

In the example, you create a compatibility library (box 18) that is linked in when the application executable (box 6) is created. You then modify the source code for the application (box 1) to conform to the application infrastructure and to use the functions provided by the compatibility library, using conditional compilation as discussed above.

Managing the New Solaris Environment

The management tools and utilities used to monitor the environment should integrate with the existing management policies and procedures. Management processes, such as trouble ticket reporting, should most likely remain independent of the new vendor platform. However, certain processes, such as change management, could be affected by the introduction of operational platforms that support dynamic reconfiguration or hot swapping.

Once the production environment is in place, you can install any tools or agents required to integrate the new environment with the existing management architecture.

In our example, BMC Patrol is installed and configured on the new platform.

System administrators must be able to manage the new environment. You identified areas in which training was required when you assessed the people and process of the IT organization. Enrolment in—and delivery of—training may be scheduled at any time during the migration effort, but should be structured to minimize the impact on existing IT operations. For example, all the operators cannot take training at the same time, because a certain number of them must provide operational support to the production environment.

In our example, no training was required because the operators were already supporting similar technologies. However, training may be required in areas such as these:

- Storage configuration
- Volume management
- Cluster operation
- Dynamic reconfiguration
- Resource management
- Database configuration and administration

Related Resources

This article is an excerpt from the Sun BluePrints book *Migrating to the Solaris Operating System*. Refer to the book for more information about the topics presented in this article.

About the Authors

Ken Pepple

At the time of creation of this article, Ken Pepple was an IT Architect in the Sun Professional Services (SunPS) Asia Pacific practice. In this role, he assisted clients with enterprise computing architectures, concentrating on advanced data center projects. Ken is now currently the Chief Technology Officer (CTO) of Sun's Desktop and Mobility Client Solutions Practice. In addition to these activities, Ken recently co-authored, with David Hornby, the Sun BluePrints™ book, *Consolidation in the Data Center: Simplifying IT Environments to Reduce Total Cost of Ownership*.

Before moving to his current position, Ken managed the SunPS high-end platform services program, and focused on complex performance issues for the IT Consulting and Operations practice. While there, he co-authored and taught the Sun Education seminar "Solaris Performance and Tuning Secrets."

Brian Down

Brian Down is the Chief Technology Officer (CTO) of the Enterprise Migration and Applications team within Sun's Data Center Client Solutions Practice. Prior to this role, Brian was a Senior Staff Engineer, most recently in SunPS, where he held the position of Chief Architect for Enterprise Migration for the Americas. For several years, Brian has focused on developing Sun's migration methodology and solution strategy, helping to develop and identify the methodologies associated with such implementations. Prior to joining SunPS, Brian focused on performance and custom engineering initiatives related to strategic server installations for the GSO.

With over 25 years of industry experience, Brian has held various engineering positions, ranging from senior engineer with a computer security company to Research Associate for the Department of Computer Science at the University of Toronto, where he worked for over 10 years.

David Levy

David Levy is a Principal Engineer in the Sun Data Center Practice United Kingdom (UK) organization. He is currently leading the program for the UK's Consolidation and Migration team, concentrating on data center architectures and economics. Prior to this role, Dave led the UK's financial services consulting team based in London. Dave has successfully completed numerous consolidation and migration projects for banking and media customers.

Before working for Sun, Dave worked for a number of financial services, IT manufacturing, and government organizations, primarily as a database architect and engineer. Dave has authored presentations for the Oracle and Sybase user groups, and is a member of the British Computer Society and Chartered Institute of Management. Dave is an Honors graduate of the University of Exeter, where he majored in Economics.

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is `http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at: `http://www.sun.com/blueprints/online.html`