SUN BLUEPRINTS APPROVED

[illegible]

Migrating From HP UNIX to the Solaris™ Operating System

Ken Pepple, Brian Down, David Levy

Sun BluePrints™ OnLine—March 2005

Sun microsystems

`http://www.sun.com/blueprints`

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 U.S.A.
650 960-1300

Part No. 819-2275-10
Edition: March 2005

Migrating From HP UNIX to the Solaris™ Operating System

Editor's Note - This Sun BluePrint™ article is the complete eleventh chapter of the Sun BluePrints™ book, *Migrating to the Solaris Operating System: The Discipline of UNIX-to-UNIX Migrations* by Ken Pepple, Brian Down and David Levy (November, 2003, 272 pages, ISBN 0-13-150263-8).

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.

This Sun BluePrint article contains the following sections:

About the Case Study in this Article

The case study presented in this article draws on several cases in which Sun Professional Services migrated customers from HP/UX platforms to the Solaris Operating System (Solaris OS).

The most significant of these projects involved the migration of one of the United Kingdom's (UK's) leading personal-line insurance companies. This customer was a typical UK-based health care insurance provider in that they primarily sold to corporate benefits managers, but dealt directly with claimants. This organization deployed a commercial-off-the-shelf (COTS) integrated- accounts solution, and enhanced it to support their risk-underwriting and claims-processing business functions.

The primary goals of the project were as follows:

- To provide a hardware platform that supported business growth over the proposed lifetime of the platform
- To support the business functionality inherent within the COTS package
- To transition to the target production platform without losing time from the business day

Additionally, the customer was in the middle of a one-year transformation from a legacy application, driving renewals from the legacy system and renewing the customers in the system to be migrated. The customer decided to change the hardware platform to extend the life of the software investment decision.
The migration solution we provided needed to be designed in the light of these business goals.

Justifying the Migration

Before authorizing the project, the customer undertook a due diligence phase to ensure that the company's business goals could be achieved by moving their platforms from HP/UX to the Solaris OS. The engagement and justification tasks required for this migration project were undertaken collaboratively by the customer and Sun. This joint process enabled the discovery of the customer's key requirements and current state, as documented in the following sections.

Business Background

In this case study, the organization's business strategies involved excellence around process, people, and business tools. The applications the organization used enabled it to run business operations and measure their efficiency and profitability.

The primary reasons for pursuing a migration effort involved meeting the system-capacity forecasts for new business and legacy replacement. The organization was moving customer data files from a legacy mainframe to its client-server environment, which created a demand for an additional disk. The IT department forecasted that they would run out of disk and CPU capacity, and hence the server environment needed to be upgraded. The Sun solution was selected for a number of reasons, including the requirement of the two systems and their associated storage for minimal floor space and the availability of vertical scalability strategies.

The organization expected Sun to migrate the applications and data from the incumbent HP systems to the new Sun servers. The transition needed to be seamless, with minimal impact on business operations, application development schedules, and the legacy migration schedule.

Enabling the legacy migrations became a key design constraint; the customer did not want to pay for a second migration infrastructure, and the benefits case required the retirement of the HP systems. The organization had already developed software and processes to move data from its mainframe to the client-server architecture, and they required the move from HP to the Solaris OS be undertaken as a deployment. The company had deployed the enhanced ledger package to meet the CEO's vision of employing good people, creating and enabling industry-leading processes, and supplying clients with the best technology. The COTS package had created an integrated risks, claims, and ledger solution, and it was in the process of taking on the current customer base a month at a time from disparate legacy mainframe systems. The target system had to be configured to act as the target for the legacy migration software. In addition, it had to seamlessly support a migration of the underwriting, claims administration, and finance department staff. (The insurance example presented earlier in this article illustrates how an insurance company can buy time for migrations by leveraging natural business-processing cycles.) The migration from the legacy mainframes utilized the annual cycle of insurance underwriting, but the migration from HP to the Solaris OS was unable to take advantage of this cycle; it was necessary to switch off the HP system as soon as possible.

Technology Background

In this case study, the goal was to replace a number of HP/UX server systems with a lesser number of Solaris Operating Systems. The key services hosted by the HP systems were database services, print management services, and job management services. These services served the risk management, claims administration and reserving systems, and the accounting and ledger systems. In addition, the systems also acted as hosts for the database client programs that undertook journaling services and end-user reports. In accounting systems, much of the referential integrity between ledgers occurs overnight (or on another periodic basis).

The other driver for the migration effort that was contrary to “work load consolidation” in the source estate design was the need to support different management environments, including development, training, quality assurance, production, and management information instances of the organization’s software solution.

The COTS package the customer used was GEAC’s SmartStream, which was written as a two-task client-server package developed using Powersoft’s PowerBuilder and Sybase Software’s SQL Server. At the time of the case study, Sybase had completed its takeover of Powersoft and was the product author and vendor of PowerBuilder. GEAC SmartStream’s natural architecture consisted of Wintel PCs hosting the client application and UNIX servers hosting the database.

At the customer organization, a PC file-sharing LAN was implemented. File servers were used to locate the client binaries, and user PCs acquired client runtime images with Microsoft’s LAN networking protocols. The Sybase SQL servers were deployed on HP/UX servers that also acted as job hosts for the accounting overnight-batch programs. (These were GEAC-provided binaries.) One of the management environments, known as the Quality Assurance (QA) environment, acted as the interface between the legacy systems and the current system. Files with the next renewals were loaded into the QA environment database, transformed with Sybase-stored procedures, and then copied across to the production environment for renewal notice production. The customer also had a development environment for the bespoke risks and claims system, a training environment for training new and existing staff in new functionality, a production environment, and an MIS environment that held an image of the production database as of the close of the previous month. This last environment was used to run reports by the finance department and corporate management. These reports were of data warehousing or online analytical processing (OLAP) profiles, with large reads and small outputs. A separate instance was configured to reduce database contention between these users and business processing case workers. The following figure illustrates the client's application components model.

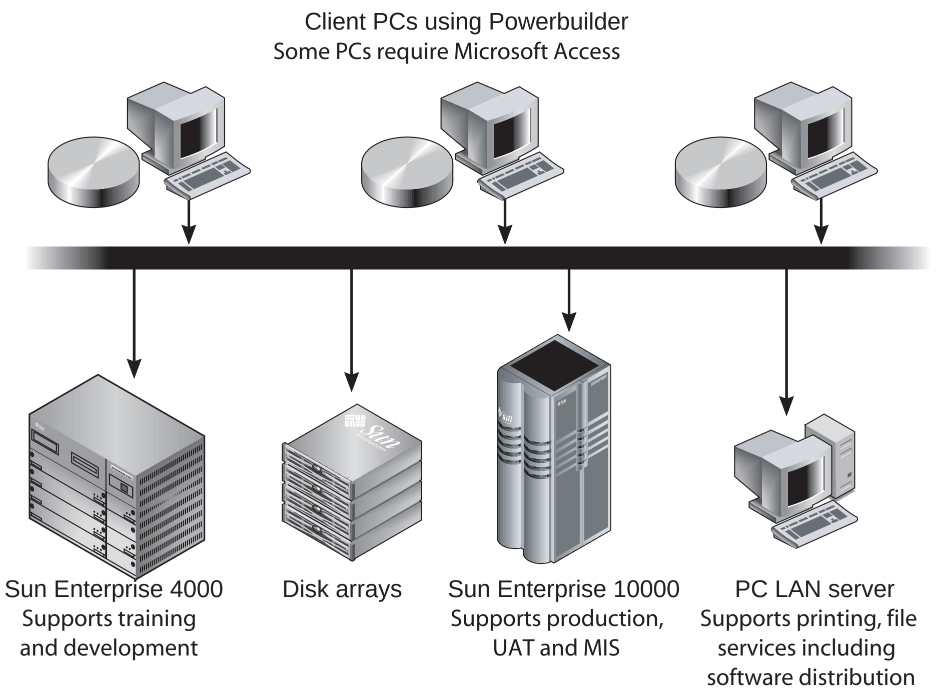
Client PCs using Powerbuilder
Some PCs require Microsoft Access
Sun Enterprise 4000
Supports training and development
Disk arrays
Sun Enterprise 10000
Supports production, UAT and MIS
PC LAN server
Supports printing, file services including software distribution

FIGURE 1 Target Deployment Platform

The company ported its database and some of the application infrastructure to the Solaris OS as part of this process, to ensure a degree of confidence that a platform migration was possible within reasonable budgets and that the proposed Sun platform would meet business growth and performance expectations.

Architecting the Migration Solution

The first task in Sun's Architect, Implement, and Manage (AIM) methodology is to determine which of the key migration techniques should be applied to each of the source components. This determination requires a component model specific to the purpose of migration planning—a component/technique map. The second output from such a component model is the definition of the scope of the migration. If a software component or function is not a member of the agreed-on component model, it is considered to be out of scope.

The key migration techniques used at the organization in our case study included a rehosting approach, supplemented by replacement and reverse engineering. The key rehosting techniques involved reusing GEAC and Sybase's platform independence and was supplemented with source code porting techniques or some of Sybase objects and HP/UX shell scripts. In this case, the component model included the database servers and their hosted business logic, print management, journaling services, and job management. The key technique driving the migration project was rehosting, using a new installation of a Solaris instance of the COTS package and the runtime software infrastructure. The reason for this is that the independent software vendors (ISVs) support their applications on multiple operating systems and basically support a common applications programming interface (API) for their products. These practices ensure that code or infrastructure changes are minimized.

The boundary of the migration problem is defined by exclusively analyzing components that are currently located on the platform to be retired. Any components located on other systems will communicate with the migrated componentry through the ISV's API. Some testing or research is required to validate that the ISVs have a common API across platforms, but this is a smaller task than attempting to migrate the calling components. The migration process can be focused on the customer's data and proprietary code base.

Defining the Scope and Approach

The architectural study's two outcomes are a definition of scope and a technique/component map. The definition of scope means that there are two sets of objects for which no technique will be applied—those determined to be out of scope and those to which a retirement/replacement technique is to be applied. The development of the technique/component map is iterative.

This section describes how to use an architectural approach to define the scope of the project. This approach involves the discovery of business constraints, the application of migration technique to identified components and strategies, and the identification of any external batch or real-time feeds. A version of the component/technique map is presented in TABLE 1 on page 13.

The following figure illustrates the scope of the project. Components within the shaded box are in scope for migration and an examination of the arrows crossing the shaded box show the communication protocols used by the migrated components to receive their input and output. Transactional Data Stream (TDS) is Sybase's client-server protocol, which encapsulates their implementation of SQL, Transact-SQL. These protocols are all stable UNIX-guaranteed or ISV-guaranteed protocols, reinforcing the decision to use rehosting as the strategy.

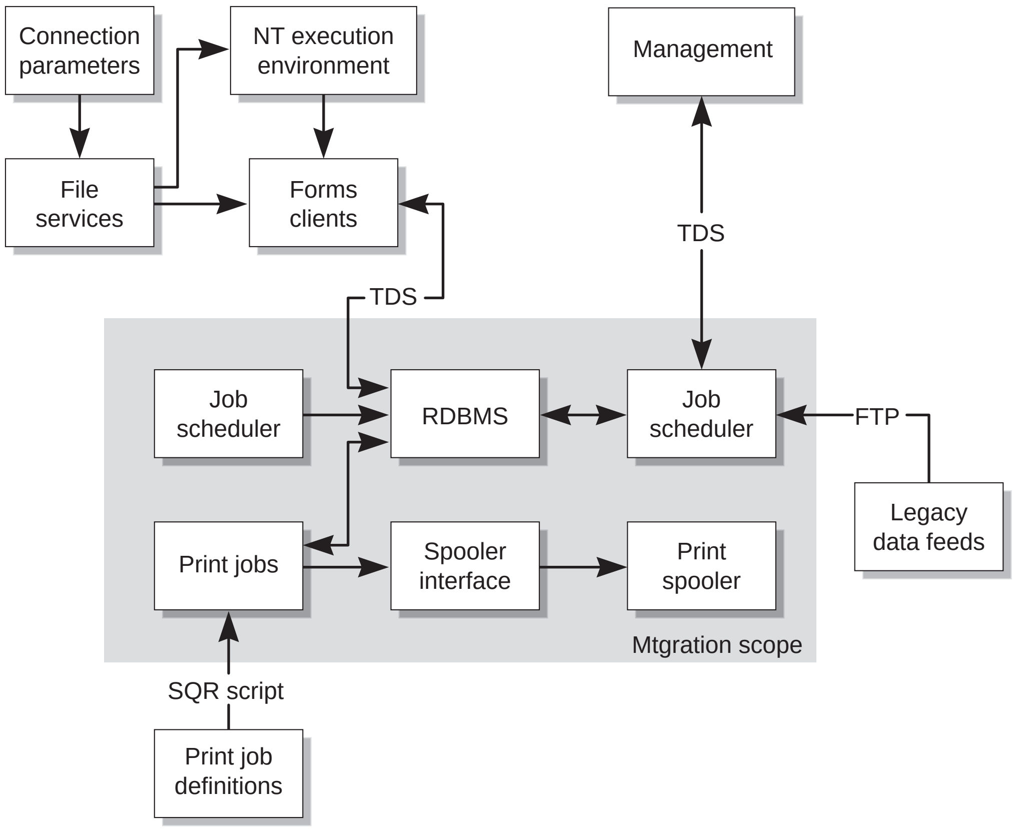

FIGURE 2 Applications Component Model

Creating a Transition Plan

The creation and design of the transition plan is a separate task within the migration project. It requires the application of project planning skills and might involve prototyping and the application of technical design skills.

Discover Business Constraints

The migration team consulted with the business and IT departments to discover any business and operational constraints that existed. Fortunately, this was primarily a front-office system for a call center and it was required only during an extended working day. The overnight batch process typically took most of the night during the week, but data load prototyping showed that an overnight run and the copy process could be undertaken on a weekend without impacting business hours.

Design a Plan

A key feature of the transition plan in this case study was to build in a regression path and a final user acceptance.

The basic plan was to close the source system at the close of business on Friday, copy the outstanding data from source to target, and then run the overnight batch on both systems. This meant that both systems should be at “start of day Monday” state. This would permit the two systems to be compared and either system to operate as the production host on Monday morning. The plan met the goals of testing for success and regression in case final testing exposed catastrophic failure conditions. It also provided a test that the business logic in the overnight batch run was identical on both systems. The inputs to both overnight runs would be designed to be identical, and if the outputs were not the same, the test would fail.

Additional features in the plan included check-pointing the process and inserting test points so that tasks between test points could be repeated should the intermediate tests fail or alternative remedial action be undertaken.

Prototyping both timings and the development of metadata discovery tools led to changes in the strategy/component map. This was reinforced when the downtime window was finally established. The change was to leverage the installation processes of GEAC and Sybase. This meant that metadata and configuration data were prepopulated and other, less-volatile, objects were also precopied. These included the table definitions, views, triggers, and procedures. The precopying of the procedures meant that the project team had to amend the development change control process to ensure that any changes to procedures already copied were applied to both source and target system.

This change of approach leveraged the principle of precopying to minimize the work to be undertaken on the weekend when the transition was to occur. In this case, we reached a point where only data table contents (and hence their indexes) were to be transferred on the transition weekend.

Note – Indexes are a key to page map and this must reflect the UNIX volume implementation and hence the RDBMSs intermediate structures. This generally mandates the rebuilding of indexes on the target system. Depending on the implementation details of the RDBMS, this can be done before or after the data copy transaction.

Design Test Plans

The key benefit from leveraging both rehosting and reverse engineering is that runtime testing of the new environment is minimized. The underlying assumption that the ISV implementations have a common and stable API requires testing, and the testing process needs to be sufficiently broad to ensure that the assumption is considered safe. This means that testing the input interface for semantic meaning is not required. The basic purposes of testing in this case study were to prove the following:

- The copy process was comprehensive.
- The target system represented the business accurately (or at least as accurately as the source system).
- The required service improvement goals had been met.

A further way of reducing the testing required is to utilize prototyping as a technique. In this particular case, the less critical systems (in revenue-earning terms) were migrated before the production systems, and any bugs in the transition process were discovered and rectified. The copy completion checks were developed and improved during the prototyping process.

Note – The copy completion checks were based on checking that all rows of a subset of the tables were copied. Additionally, we ran the check sum script against the contents of certain columns.

Testing tools consisted of five sets of tests:

- Copy integrity—Precopy
- Copy integrity—Transition phase
- Semantic integrity
- Business acceptance
- Performance acceptance

The copy integrity suites both involved writing programs that read the source and target systems to compare them. These browsed the database catalog tables, and since the query language used was SQL, only one language was used. In addition, critical columns were check-summed. These columns were the critical item level financial columns. One error was discovered at this phase based on a bug in the application. This error was corrected in the production code lines, and a fix was applied to erroneous data. This illustrates a principle of fixing a problem at its root cause, rather than, as in this case, writing a data transformation and fixing the problem during migration.

Semantic integrity tests were limited in this case study. The key area they were applied to was the shell-script-based print management solution. The key question to be answered in this test suite is, does the code behave the same on both systems?

We supplemented semantic tests by co-opting members of the business unit's training department and their training scripts to test the unchanged client layer against a Sun-hosted migrated environment. To conduct these tests, we undertook the first full-scale migration test on the training instance of the application.

The performance tests had been specified in the contract prenegotiations and pretests undertaken in Sun's Global Benchmarking center. These tests were repeated on site on an appropriately sized instance of the database before the migrated solution was placed into production.

Specify the Business Acceptance Tests

The business acceptance tests were specified to meet the following goals:

- To prove the target system accurately represents the business
- To prove the target system meets the performance-based system improvement goals

The axiom of the project was no change in business logic, so the basic acceptance test was running the test chart of accounts, which was a report option within the package. The view taken was that, if the target system represented the business accurately, then it was suitable. This view was based on one of the fundamental theories of software development: the primacy of design is in the data model and if the data model implementation is accurate, then the process implementations can change to support changes in process. This view had the advantage that the run times of the test were low and the verification time was also low. Using this test mandated constraints on the transition strategy.

It was decided jointly by the customer and the team that the best way to ensure that the system accurately represented the business was to run a test chart of accounts. To simplify the comparison between the two reports, we ran both of them at the same logical time. These activities leveraged the decision to ensure that both the source and target system were available, because they were at the start-of-day post transition.

Given that the business acceptance test involved running a program that relied on denormalized tables, a risk was identified that the acceptance test be sufficiently comprehensive to test the data migration, so additional instrumentation of the migration process was developed. As stated above, not only was a target container created that held all the non-business/transactional data instances (rows) so that the migration was constrained to business/volatile data only, test programs were also written to ensure that source and target container properties were identical. Test programs were written to ensure that all the contents of business data tables were copied accurately in terms of both number of entries and summing critical columns.

Implementing the Migration to the Solaris Operating System

The next phase of a migration project involves implementation tasks. In this section, we examine the tasks involved with implementing an HP/UX solution to the Solaris OS.

Applying Migration Techniques

Migration involves the movement of business logic (for example, code), business data, configuration data, and metadata from the source environment to the target environment.

In this case study, the various objects to be migrated were categorized and a strategy for moving them was developed. Because we were certain that the source system worked satisfactorily, the initial strategy was to copy all the objects from the source environment to the target environment. However, during prototyping, we discovered that the database volume map needed to be recreated and that it would be difficult to port certain configurations and metadata. Because it was going to be more difficult to adopt a “copy everything” strategy than we’d expected, we decided to use the installation scripts provided by GEAC and Sybase to create a container in which the remaining data and executables were installed.

The decision to implement a new disk map was made for the following reasons:

- The target system had a different disk architecture from that of the source system.
- The source system disk map had only two virtual disks, which makes implementing database recovery difficult.

Note – Database recovery requires that database files, the write-ahead log, and the offline images of these objects be held on different disks. Databases with split journals and before-image logs (such as Oracle) might require an additional disk. Database recovery is designed to protect work against a lost disk, so the write-ahead log must be on a different disk from the database or it will be unavailable if the database disks become unavailable. Offline copies should be held on other disks so that they are available if the write-ahead log becomes unavailable.

The target system had significantly more disk volumes than were available on the source system. The disk map redesign also enabled a simplification of the database's internal object placement design.

Both Sun and the customer agreed that the rehosting strategy would be applied to GEAC, Sybase, and the Sybase Client reporting tool used. The data migration process would be a logical copy. It was not possible to undertake a physical copy of the source system—Sybase implements certain features of its system differently on the Solaris OS and HP/UX. The methods used to extract the data for the procurement due diligence were based on logical copy technology for this reason.

Note – Print-report logic and the output formats were defined in another third-party product, originally sold by Sybase, which remained available. The customer obtained a Solaris license for this product. This product and an ISV portability guarantee permitted the interface between the `Print Job Definitions` and `Print Jobs` commands to be defined as a third-party protocol and obviated the need to rewrite the report programs.

Solutions for the individual data item types needed to be developed.

The team undertook a volatility analysis of the data objects. The database schema definitions were very stable. The majority of the databases were part of the ISV package (SmartStream); therefore, the schemas were very stable, typically only changing when software updates were applied. At the other end of the spectrum, the online transaction processing (OLTP) data changed minute by minute. Between both ends of this spectrum were the customer proprietary schemas and user identity data.

Sybase data servers organize their catalog tables either in a configuration database called `master` or in the application's databases. `master` holds server wide data, whereas the application databases hold database-specific configuration data as subsets of the data. Each database has its own local catalog tables, including private lists of user objects such as tables, indexes, views, and users. Each database has its own write-ahead log and is therefore a unit of recovery. Identifying Sybase RDBMS metadata is relatively simple. However, GEAC originally architected its distributed-systems solution to hold significant amounts of metadata either in specific application databases that were solely responsible for holding this data, or in tables within application databases that had business-functional purposes.

An audit of the available technology to copy data from the source to the target was undertaken. The procurement due diligence resulted in a suite of programs to extract critical business data and metadata from the customer's systems. On the general principal of trying to copy everything, we initially decided to extend the copy and upload scripts from the initial subset to all data objects. We planned to use Sybase's bulk copy program (`bcp`) to transfer the data (and certain other objects) and to use Sybase's `defncopy` utility program to migrate the triggers, views, and procedures. We decided to maximize the advantage of using the Sybase and GEAC capability of running on both the source and target system. A copy strategy also allowed the migration team to minimize their understanding of the GEAC schemas as if they were the same on both systems.

However, Sybase's instance-to-instance copy facilities were limited and needed to be augmented. Sybase had logical copy tools such as `bcp` or `defncopy` that could be used for tables or procedural objects. The database was the only larger object that could be made the argument of a copy program: the `dump` and `load` commands could be used on it. However, dump and load activities need certain common configuration parameters for the source and target servers. The default collating sequence for a Sybase server is different for HP/UX and Solaris. The team chose to implement the default on the target (Solaris) system for maintainability reasons, so the `dump` and `load` commands were not available. An object-by-object copy policy needed to be defined.

The customer had access to two schema extractors. These extractors were part of the two computer-aided software engineering (CASE) tools and could act as reverse engineering tools. Despite relying on the rehosting strategy, we realized that weaknesses in the environment's capability meant that the rehosting strategy had to be supplemented. Reverse engineering techniques and source code porting techniques were used to supplement the strategy.

We chose a schema extractor primarily due to the availability of certain skills and because of current licensing commitments (the customer had two CASE tools licensed and preferred one over the other).

The transfer strategies are summarized in the following table.

TABLE 1 Application of Migration Techniques to Objects

Object Type	Transfer Tool	Transfer Technique
Executables	Reinstall	Rehost
Table definition	Schema extractor	Reverse engineer
Table data	Sybase BCP utility	Rehost
Index definition	Schema extractor	Reverse engineer
`syslogin` table data	Sybase BCP utility	Rehost

TABLE 1 Application of Migration Techniques to Objects *(Continued)*

Object Type	Transfer Tool	Transfer Technique
`sysusers` table data	Schema extractor	Reverse engineer
User datatypes, rules and defaults	Schema extractor, supplemented with bespoke DDL	Reverse engineer, supplemented with source code port
Views	Sybase `defncopy` utility	Rehost
Procedures	Sybase `defncopy` utility, GEAC source files or bespoke DDL files	Rehost, supplemented with source code port
Permissions	Schema extractor	Reverse engineer

Although table definitions and contents are two separate objects in the component map, an index definition contains a description in SQL that is defined in text and held in a catalog table. In the case of Sybase, the index definition possesses either a clustered structure or a B-Tree structure. The definition can be run at any time, but running it involves building the index contents, which, in the case of a clustered index, involves sorting the table. For this reason, a `create index` statement can have significant runtime implications, but it obviates the need to copy index contents.

The disk map redesign made obvious the need to copy the `sysdevices` table that maps the RDBMS's name for a disk volume to the OS name. It also revealed the need to port any named segments. The role of a segment is to provide a location name to which a table or index can be bound. This allows DBAs to manage the location of objects, permitting, for example, a B-Tree index or a log object to be located on specific disks different from their bound table or database. The creation of the `sysdevices` table and the minimum necessary segments was undertaken when the RDBMS and their component databases were installed.

The stored procedures were copied by one of two methods. Procedures that were also implemented as text objects in the catalog tables were copied with `defncopy`. In cases in which there were unresolved external references, the original source code files for the data definition language (DDL) were inspected and rerun unchanged or were massaged. The original author was either the COTS vendor or the customer.

The print management solution was the outstanding piece of code that had yet to be ported. The print queues were reallocated to the file servers, but the print management component had been implemented in UNIX shell script. There was a very limited amount of this code, and rather than install Sun's standard tools and test harnesses, we ported the code manually by inspection and iterative testing. This means that the replacement strategy was applied to the queue management function, and that a source code porting technique was applied to the runtime management functions.

TABLE 1 on page 13 shows how the basic strategy of utilizing the COTS vendor's guarantee of platform independence and a consistent API across platforms was supplemented. The initial strategy of copying everything was amended for the following reasons:

- Manifest quality improvement was achieved by a redesign of the disk map.
- The cost of isolating the physical changes that were required and applying separate strategies was too high.
- The time required to transition with this strategy demonstrated a need for a more incremental approach.

Let us consider further the ease of undertaking the creation of a component/ technique map. In the real circumstances of the case study, the axiomatic properties of an RDBMS ease the mapping of object instance to object classes. The descoping of all the client logic also eased the migration task significantly. In many cases, allocating an object to a class of data is not easy. This problem is eased by the facts that a technique can be reused and that object types can have multiple techniques applied to them, as illustrated TABLE 1 on page 13.

With RDBMS systems, it can be very difficult to define which objects sit in which category. Note that it should be easier for database objects than where a significant amount of 3GL code exists, because the types of objects available within 3GLs are far more restricted, and active dictionaries in which metadata are held are less frequently implemented. In the RDBMS case above, the metadata and many runtime objects are held in the data dictionary, alternatively called the database catalog. 3GL systems more frequently build and integrate their metadata within the executables. In this case study, one of the executable database objects was the Sybase stored procedures, which were easy to isolate, although some required significant massage to port because they had unresolved external references. If the stored procedure used a temporary object, which means one that is created and destroyed by the procedure itself, despite the fact that the creation commands exist within the stored procedure, unless the object exists at the time the `CREATE PROCEDURE` command is issued, the `CREATE PROCEDURE` command will fail.

An example of a tightly integrated business logic and execution logic is described in the following paragraph. For instance, Cobol 88-level definitions are business logic objects, and are embedded into an executable, or at least into the source code lines.

```
77  BOOL-TEST-1                       PIC(X)VALUE 0.
88      TEST-1-TRUE REDEFINES BOOL-TEST-1   VALUE 1.
88      TEST-1-FALSE REDEFINES BOOL-TEST-1  VALUE 0.

    IF < complex condition true>
        MOVE 1 TO BOOL-TEST-1.
    IF < complex condition false >
        MOVE 0 TO BOOL-TEST-1.

    IF TEST-1-TRUE
        PERFORM TRUE-CONDITIONS
    ELSE
        PERFORM FALSE-CONDITIONS.
```

In this example, data division entries state that a binary test condition exists (for example, it can be true or false), and that the variable acts as a flag. The next two procedural statements evaluate the condition and encapsulate a business rule. The final statement executes the business transaction logic. To simplify the example, we've used the PERFORM statement to invoke a section that undertakes this work. The business rules, business logic, and business data types are all distributed throughout the source code lines. Extracting these three object classes as individual items from a source code file is hard. There are no clear rules for distinguishing among these object classes, and identifying the elements is equally difficult. Fortunately, the richer data types available in an RDBMS solution allow application designers to isolate business rules and transaction logic from implementation details.

This case study shows how iteration and prototyping were applied to various objects, object types, and classes. For example, an index can contain both business logic (a uniqueness constraint) and implementation factors (such as a fill factor). The value of, or the necessity for, iteration and prototyping might depend on physical design and implementation details of the RDBMS. Other examples of implementation details that are encapsulated in the index include the location where the index was built and the sort order of a clustered index.

In the case study documented here, the RDBMS was implemented with a cost-based query parser and predated the implementation of query hints. For this reason, the index is used as an example, but where rule-based analyzers are implemented, performance-critical transactions need to be tested to ensure that the query plan resolution remains optimal. Query plans are usually calculated at runtime, so prototyping and preimplementation testing might be required because rule-based optimizers can require code changes to permit the analyzer to choose the optimal query plan. This is particularly important when RDBMS version upgrades are undertaken.

TABLE 1 on page 13 shows the application of migration strategies and techniques to specific database and execution objects. These strategies and techniques were developed during the architecture stage and were refined during the implementation phase while software to implement the migration was developed.

With an RDBMS, business logic can be held in database objects (such as views, indexes, constraints, triggers, and procedures) or in client-side objects. The definition of scope is critical in defining the techniques used to migrate business logic. As described above, business-object logic can be in scope or out of scope. The scope status of business logic can depend on several factors. The business logic can be redundant, as is the case when the retirement/replacement technique is used. It can be encapsulated in part of the environment that exists in both the source and target environments. In the case of this study, client-side PowerBuilder procedures were executed in both the source and target environments; therefore, they remained out of scope. For this reason, these business-logic entities remained the same in both source and target environments. Given the techniques applied to the migration, the business and presentation logic encapsulated in these procedures and objects was defined as out of scope.

The `Print Job` component was defined as being within the scope of the project because print queue management had been undertaken by the HP/UX jobs. Actual queue management was moved into the LAN, and the organization's LAN servers were configured to hold the print queues. If the job were undertaken today, it is likely that the printers would manage their own queues, depending on the output management requirements—such as restarting and reprinting. Scheduling print jobs against the databases was managed by a series of shell scripts that ran the third-party report generators. The report logic was held in ASCII files holding SQL script definitions that were invoked by the shell scripts. The following example illustrates shell syntax that allows the script to run on either an HP/UX system or a Solaris system, and thus supports backward compatibility. Furthermore, it has the advantage of indirectly referencing the UNIX utility in the example (for example, `cpio`). In this case study, indirection was implemented but backward compatibility was not.

```
#!/bin/ksh
OS=`uname | ${cutpath}/cut -f3 -d' '`
case $OS in
HPUX)    OS_PATH_LIST=${HPUX_PATH_LIST};;
SunOS)   OS_PATH_LIST=${SOLARIS_PATH_LIST};;
*)       exit 1;;
esac
#Original Line
#PATH=${HPUX_PATH_LIST}
PATH=${OS_PATH_LIST}
CPIO=`whence cpio`
.
.
$CPIO ${CPIO_FLAGS}
```

Metadata is data that describes data. In the case study, this was absolutely critical because the key migration object was an RDBMS that possesses an active data dictionary. Three strategies were applied to the metadata:

- Utilizing the installation processes provided by the COTS and DBMS vendors
- Copying metadata objects from the source environment to the target environment with tools based on the appropriate migration technique
- Applying reverse engineering techniques

Financial considerations were the primary influence over the decision of which strategy to apply. There are several factors in calculating strategy costs. These include the following:

- The cost of identifying objects. In the case study, a number of objects could not be identified and the installation processes were utilized.
- The cost of applying the strategies. SQL-BackTrack was rejected because of cost.
- The runtime cost implications of the strategy.

Not all metadata is held in obvious metadata objects. In the case of the COTS product under consideration, metadata was held in the RDBMS catalog tables, user tables defined by the COTS vendor, and index definitions. One additional piece of metadata included the representation of the system namespace within objects that are available to the application. GEAC SmartStream used multiple databases within a database server and used Sybase remote procedure calls to implement inter-database transactions. This permitted the deployment of a SmartStream implementation across any number of server instances. One of the advantages of this implementation feature is that different application components can be deployed in separate servers on separate hosts. The development of blade technology gives this architecture a new lease on life. This technology requires each server and stored procedure to know about the location of the database within a server. Sybase implements a name service based on flat files, mapping a server name to a TCP/IP address/port location. In addition, when a remote-stored procedure semantic is implemented, this name service must be placed within a security model implemented in the catalog tables. In the case study, most of the COTS metadata was applied to the target the installation scripts were run.

The examination of security data in the context of application name services brought us to security data itself. Within Sybase, both authentication and privilege management functionality is implemented. Privilege management is part of the SQL standard. The permissions row in TABLE 1 on page 13 represents the implementation of each object's execution, read, insert, update, delete, create, and destroy privileges. The mapping of a user's identity to a privilege set is a business issue based on roles within the business. At this customer site, the authentication data was treated as data, not as metadata; therefore, it was copied across. This is represented by the `syslogins` row in TABLE 1 on page 13. Sybase also implements an alias for each login within each database and, at the time, it presented the team with a referential integrity issue between the database user alias and login. The aliases were migrated with reverse engineering techniques, with manual inspection and adjustment as the remediation techniques used when reverse engineering failed.

Namespace Migration

In the case study, there are three namespace problems as follows:

- Database object namespace. Objects within the data servers (except `databases`). This includes a need to migrate or transition the server names and address maps
- Applications component namespace.
- System component namespace.

Different techniques were used to manage the namespace implementations to the target environments.

We utilize the application's installation procedures to preserve the database server's internal namespace. This meant that the proprietary extensions to GEAC SmartStream deployed by the customer also needed to be ported and the object namespace preserved. The mechanism used to preserve the object namespace is documented below. It utilized the file system by writing the object definitions to files with the object name in the file system name.

The target data servers were given new names. This was required because the servers had separate TCP/IP addresses and both needed to be on the network at the same time. This policy conformed to the strategies adopted and aided transition because the customer had a good server name management-distribution policy. The server name and address file, `${SYBASE}/interfaces`, was held on a file server and read by each of the user client's systems. This system also allowed the default data server to be configured by the LAN administrator.

The application's component namespace was managed as defined by GEAC, and existing documentation explained how to transition the namespace from HP/UX to the Solaris OS. This transition involved manually updating rows in three tables. The customer had previously moved systems and had scripts we could use to update the system names. Only one of the rows involved specifying the target OS. The system namespace was implemented in `bind`.

Data Migration

The use of the supplementary techniques is mainly constrained to nondata objects. In the case study, data was defined as only the content of database tables that contained business data. Previous sections discussed the techniques used to identify metadata, configuration data, and security data; what's left is the business data. We had two choices for copying the data:

- Logical copies
- Physical copies

The need to apply data transformation to the source data is one of the primary influences on the decision about which technique to use for copying data. In this case study, copying data from the legacy mainframe required the application of transformation techniques. With one exception, moving the business data from the source environment to the target environment did not require transformational work. This means that a logical copy was simple and that a physical copy was possible. At this site, a physical copy was not possible because of implementation differences in the RDBMS on HP/UX and Solaris systems. Therefore, a logical copy was the only option. In the case of Sybase, this suggests the `bcp` program; in the case of Oracle, it would imply the use of the `export` and `import` commands.

In all cases, object namespace preservation and mapping is required. This means that, because we were using different techniques—in the case of Sybase—to copy the table definitions and table contents, the planners needed to map the target DDL file, table name, and table contents file. (This would not be the case with Oracle's import/export, but would be if SQL/ODL were used.) This issue was resolved by use of the UNIX file system to preserve the table namespace between systems, as shown in the following example.

```
mkdir ${database_name}; cd ${database_name}
for table_name in ${table_name_list}
do
   mkdir ${table_name};cd ${table_name}
   extract_ddl ${table_name}> ${table_name}.ddl
   bcp ${bcp_flags} out ${table_name} \
   > ${table_name}.data
done
```

In this case, `extract_ddl` is a script or function that performs the table DDL extraction so that `${table_name}.ddl` contains the table DDL code. The queried object might be the database, or it might be a flat file that contains the complete RDBMS-instance DDL, prepared by the selected schema extractor. The following example code can also be used to preserve objects transferred by `defncopy`.

```
mkdir ${database_name}; cd ${database_name}
mkdir views ;cd views
for view_name in ${view_name_list}
do
    defncopy ${defncopy_flags} ${view_name} \
    > ${view_name}.ddl
done
```

In both cases, input scripts can be driven by parsing the directories for `*.ddl` files.

In the case study site, migration harnesses were built to parse the database catalogs to extract the `ddl` and data files, and the input scripts parsed the UNIX file system to drive the database inputs. The input scripts also used symmetrization techniques to leverage the power of the SMP platforms proposed for the target implementation. Each job stream uploaded a quarter of the database bound to a single CPU, and the jobs ran concurrently.

Specify the Implementation Platform

The procurement due diligence exercise led Sun and the customer to specify the hardware platforms. It was proposed that a system with three domains would support the production, QA, and MIS environments, and a second system would support development and training and act as a business continuity system if the production machine became unavailable. This meant that the customer wanted both a physical consolidation and workload sharing consolidation benefits. These decisions allowed the customer to recover significant floor space through the consolidation of three environments onto a single system. The shared solution also delivered significant floorspace savings.

One of the aims of this project was to reduce the number of system hosts at the customer site. The current estate consisted of five HP/UX systems, and the goal was to reduce this number to two Sun systems. However, because the number of management environments was five, separate instances of the OS were required to allow differing and separate management policies to be implemented and enforced. At the time, an instance of the OS could have only one security model and the business necessity of ring fencing nonoperational users from production systems was—and still is—almost universal. The target platform design established during the customer's due diligence phase consisted of two Sun servers, with only one being capable of hosting multiple OS instances. Both systems were SMP systems, and the smaller was designated to become the development and training system host. This involved the implementation of two application instances within a single instance of the Solaris OS and used an aggregation design pattern. The remaining instances of the application (production, QA, and MIS) were planned to be hosted within a domain in a multidomain system.

Specify the OE Tune State

We initiated a requirements-capture exercise. This exercise primarily involved collecting the constraints that the superstructure products such as the RDBMS placed on the `/etc/system` file tunables. The following were the two key tunables for the RDBMS:

- **`SHMMAX`**. Maximum size of a contiguous shared memory segment. With the versions of Sybase proposed, a limit of 2 gigabytes was the maximum. More recent and current versions support Very Large Memory (VLM) addressing, so a more appropriate setting is to set `SHMMAX` to high values.

  If explicit values are set for `SHMMAX`, the system will require rebooting if the database administrator decides to increase the database buffer cache beyond the `SHMMAX` limit. Restarting the database server process will cause a service outage to their users. In a shared infrastructure solution, rebooting a system is undesirable because other customers might take a service outage for no benefit.
- **`ISM`**. The Solaris default is intimate shared memory on which is the advantageous performance configuration. This configuration option had implications for defining the swap partition size.

Prototyping during the test loads of the development and training instances was undertaken to determine whether the available processor-management tools were necessary or desirable to manage service level provisioning for the multiple communities proposed to use the shared second system. These management tools allowed the system administrator to provide rules to the dispatcher. It was discovered that the Solaris affinity algorithms did not need the help of the process management tools, and so the final production configuration for this system did not use them.

Build a Migration Harness

The copy programs were encapsulated into a harness so that the migration team could undertake relevant jobs of work. These included “extract an instance,” “load an index,” and “rebuild indexes.” These were supplemented by jobs to copy various objects that were planned to be precopied. These latter programs could take an instance, database, or object as arguments so that they could be copied incrementally. They were all driven by lists that were created by the developer team or developed by browsing the database catalogs. By creating programs to undertake this work, not only was human productivity enhanced, but the programs could be tested and trusted. This minimized the requirement for testing the processes activity; if the jobs reported success, then the prior testing of the programs enhanced the confidence that the job had been performed accurately. It made the process testable.

The transition process was principally tested by migrating the training and QA instances before the production instance of the application. This permitted both real timings for the data extraction and target index builds to occur. It also meant that the user training began on the target Solaris system several weeks before the migrated solution was to be placed into production. This allowed the training team, as well as the trainees, to comprehensively test the migrated application. This was advantageous because it ensured that trainees were introduced to every aspect of the system, and it had the added benefit of thoroughly testing the client-server interfaces.

Utilize Management Environments to Enhance Testing

The transition plan for this project included testing plans for testing outputs and regression testing. The migration process was pretested, and checkpoint tests were inserted. In addition, checkpoints were designed into the plan to use backup solutions.

The migration team utilized nonproduction environments as part of the enterprise transition plan. The training and QA environments were ported in advance of the production instance, which improved the confidence the team had in the transition harness and the application of basic strategies. The migration of the training department allowed enhanced, comprehensive testing of the client APIs. The migration of the QA instance delivered confidence that the production performance tests would be achieved.

The development and MIS environments were ported after the production transition. The development environment was created by copying the training environment and then applying the developers' subsequent changes to the new development environment. This is a process that the customer had frequently undertaken and was satisfied with.

The MIS environment was created with the production mechanism, which was to use Sybase's block-level online dump and load. This gave us the advantage of testing that this process/program worked in the new environment. (The technology had been tested before the production transition).

Managing the New Solaris Environment

The case study described in this article was a project that was identified to end when the final user acceptance test was successfully completed. Both backup and job management tasks are discussed in this section of the case study, but the key work performed by the migration team was the design and implementation of the solution.

Operations management remained the responsibility of the customer's computer operations department. One of the reasons for this was that, while the HP/UX systems were to be retired, the PC LAN and legacy mainframe remained part of the production environment. Therefore, the management problem was a heterogeneous one.

The two key management requirements were to provide off-line recovery capability and to allow business superusers to start and manage application jobs. Both of these functional areas were subjected to a business requirements capture, design, and test life cycle before handover.

Backup

The move to the Solaris OS gave the customer the opportunity to take advantage of the predecessor product, Sun StorEdge™ Enterprise NetBackup. The source system was also bundled with a backup solution, and the Sun team captured the policies for occurrence, strategy, and tape maintenance and reimplemented them in the new technology. This involved implementing the "No worse than before" strategy. One of the solution-design constraints was the customer's tape pool size policy.
The company also dumped the production database directly to tape, using Sybase's online backup utility.

The change in technology mandated a change in the backup technology. Fact-finding was undertaken, user policy constraints discovered, and a suitable backup solution implemented. This involved configuring a small robotic tape device on the production domain of the multidomain system and the second system. Network backups were used for the QA domain. Note that the MIS environment was not backed up because it changed only once a month.

Job Management

The key problem introduced by the proposed platform architecture was related to job management. The COTS solution had an integrated job manager for which certain jobs (for example, accounts journaling jobs) had to be organized. System tasks could be organized by traditional Solaris/UNIX solutions, but application-related tasks had to be organized by the application's encapsulated job manager. In addition, a solution to the consolidation/workload sharing design needed to be found.

Another problem was that the software had not been designed to run in a work-sharing environment. It was network aware—for example, it used TCP/IP for its interprocess communication—but two instances of the job scheduler could not be distinguished in the UNIX process table. System managers could not distinguish between the development and training instances of the job management daemon when using the UNIX utilities. While these daemons were correctly manipulated by an applications component running on remote PC systems, the system's managers felt uneasy about this new feature. A feature in the software was discovered that permitted this problem to be overcome.

Results

The migration was successfully undertaken. Sun used its architecture methodology for migration to move the customer's business-critical financial and insurance applications from HP/UX to the Solaris OS using a rehosting strategy supplemented by reverse engineering, source code porting, and retirement/replacement techniques. This was undertaken within an acceptable system down-time window with zero business downtime.

The project life cycle articulated in Sun's AIM methodology was a key enabler to the success of the project. The case study presented in this article show how the application of methodology makes migration projects simpler and less risky.

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^{SM} 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`